KERNFORSCHUNGSZENTRUM

KARLSRUHE

Januar 1973

KFK 1594
EUR 4730 e

Institut für Reaktorentwicklung
Projekt Schneller Brüter

SEDAP
An Integrated System for Experimental Data Processing

M. Audoux, F. W. Katz, W. Olbrich, E. G. Schlechtendahl

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.
KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 1594
EUR 4730e

Institut für Reaktorentwicklung

Projekt Schneller Brüter

S E D A P

An Integrated System for Experimental Data Processing

M. Audoux *)
F. W. Katz
W. Olbrich
E. G. Schlechtendahl

Gesellschaft für Kernforschung mbH, Karlsruhe

*) delegated from EURATOM

SEDAP - Ein integriertes System zur Meßwertverarbeitung

Zusammenfassung

SEDAP (System for Experimental Data Processing) ist ein vielfältig einsetzbares Programmsystem zur Verarbeitung und Reduktion experimentell gewonnener Daten. SEDAP wurde in FORTRAN IV programmiert und auf den Datenverarbeitungsanlagen vom Typ IBM 360 und 370 der Gesellschaft für Kernforschung, Karlsruhe, implementiert.

Das System erlaubt die schrittweise Verarbeitung von Meßdatendateien, sogenannter "Experimental Records", mit der Möglichkeit der freizügigen Kombination von Standard-Operationen (wie Integration, Erstellen von Diagrammen usw.) Eine der Grundideen in SEDAP war, dem Experimentator die Möglichkeit zu geben, den Auswerteprozeß seiner Meßdaten mit Hilfe einer, in ihrer Struktur sehr einfachen aber trotzdem mächtigen und speziell auf seine Anliegen zugeschnittenen Sprache, selbst programmieren zu können, ohne durch EDV-Probleme von seinen Versuchsproblemen abgelenkt zu werden.

Der Bericht beschreibt das gewählte Verfahren zur Lösung der Probleme des Datenauswertungsprozesses und erläutert den Begriff: "Experimental Record". Detaillierte Angaben zum Programmsystem beschreiben die Datenspeicherverwaltung, das Steuerprogramm, die verschiedenen Operatoren, die Subsysteme (Transfer, Input/Output, Service) und die Fehlerinterpretation. Der nächste Abschnitt enthält eine ausführliche Liste aller Sprachelemente (Kommandos) zusammen mit verschiedenen Beispielen, die eine rasche Einarbeitung in die Benutzung von SEDAP erlauben.

Schließlich folgen noch einige Hinweise über mögliche Weiterentwicklungen.

SEDAP - An Integrated System for Experimental Data Processing

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 IV and implemented on the IBM 360 and 370 installations of the Karlsruhe Nuclear Research Center. The system insures the modular processing of so-called "experimental records" and provides a straightforward way to use standard operators (integration, conversion, plot ... etc. ..). The leading principle in designing SEDAP was to allow experimenters who are not familiar with a programming language to conduct their own data reduction with the help of a very simple processing language. The report explains the approach which was selected to solve the problem of the experimental data processing and introduces the concept of "experimental record". The detailed description of the system includes the storage management, the main program, the various operators, the different subsystems (transfer, input-output and service) and the interpretation of errors. The different elements of the language are listed with different examples which allow any prospective user to become rapidly familiar with all the features of SEDAP. The last part of the report gives a tentative evaluation of the system together with some guidelines for further developments.

SEDAP - Système de programmes intègres pour le traitement de mesures expérimentales

Résumé

SEDAP (System for experimental data Processing) permet à un experimentateur de réduire un ensemble de données numériques acquises pendant le déroulement d'une expérience scientifique. SEDAP est un système de software qui utilise exclusivement le FORTRAN IV et qui a été mis au point sur les ordinateur systèmes IBM 360 et 370 du Centre Nucléaire de Karlsruhe. Le système permet le traitement modulaire de "records expérimentaux" et fournit la possibilité de combiner librement la plupart des opérations usuelles (intégration, différentiation, conversion, filtrage, transformation de Fourier, plot, etc. ...). L'idée maîtresse de SEDAP est de permettre à l'expérimentateur d'effectuer lui-même la réduction de ses données numériques grâce à un langage approprié. La simplicité de ce langage est telle qu'elle ne nécessite aucune connaissance prálable dans le domaine de l'Informatique. Le rapport explique la méthode suivie pour aborder le problème de la réduction des données numériques et introduit la notion de "record expérimental". Il fournit une description détaillée du système: gestion des ensembles-mémoires, programme principal, structures d'appui (Entrées-Sorties, Transfert, service), routines de calcul et interprétation des erreurs. Les éléments constitutifs du langage SEDAP sont expliqués en liaison étroite avec de nombreux exemples qui permettent à l'utilisateur éventuel de se familiariser rapidement avec le maniement du système. La dernière partie du rapport est consacrée à l'évaluation sommaire du système et indique certaines des améliorations susceptibles d'être apportées au système.

Contents

Contents

Figures

Page

Tables

1. THE SEDAP APPROACH

1.1 Basic principles of the experimental data processing

Because of their reliability and their accuracy the digital data acquisition systems are more and more widely used to record the different phases of modern technological experiments. The rational organization of such systems calls for a two-sided approach which involves the two following closely related fields:

1) The data acquisition
The acquisition of the data requires a signal amplification with appropriate filtering as well as a multiplexed analog to digital conversion. Many of the off-line systems record the converted data on magnetic tapes which are processed during a further task performed by a large size computer. Initially the adaptation between the analog to digital converter and the tape requiring a block segmented transfer was provided by a buffer memory and its associated circuitry. The recent evolution of the relatively low priced small size computers has radically changed the situation. The minicomputers provide the interface with the tape units, a buffer memory and a programmable operating mode which gives such a versatility to the modern data acquisition systems that they can be adapted to different types of experiment by only typing in a few instructions on a keyboard.

2) The data reduction
An efficient data acquisition system must rely upon a good software package in order to obtain the best possible information from the values stored on the magnetic tape. It is quite unfortunate that one often does not take advantage of the great flexibility achieved by the modern data acquisition systems because of the rigid structure of the software

---

Zum Druck eingereicht am 21.12.72

support. It has been often observed that an undesirable gap exists between the planning and the recording of an experiment on one side and the data reduction and the interpretation of the results on the other side. Many times the existing subroutines have to be modified to take into account the latest changes in an experimental set-up and the situation can be worsened by a lack of communication between the experimenter and the scientist in charge of the software.

All these considerations have led the Institut für Reaktorentwicklung to develop an experiment oriented software package called SEDAP (System for Experimental Data Processing).

1.2 Guidelines for the development of SEDAP

SEDAP was developed according to the following guidelines:

a) SEDAP should provide a rationalization of the data reduction. In other words a better efficiency should be obtained from the available resources in different domains.
   For the user the system should be easy to learn and should offer real advantages concerning service and comfort.
   For the programmer, a careful planning of the task and the deliberate attempt to implement a modular system should save many of the manhours which would be necessary to perform the modifications of small programs.
   For the installation, the integration of the system should save some of the computing time, not only by the optimization of the programs but also by the reduction of aborted runs which are avoided by a good documentation of the errors

b) SEDAP should be large enough to satisfy most of the wishes of a users group which were invited to influence the specifications of the options to be implemented. The frame of the system should not be restricted to a special branch of engineering sciences but should however fullfill the basic requirements of the IRE program (thermo-

dynamics, vibration analysis, sodium boiling etc. ...) and similar experiments and should be relatively easy to extend.

c) SEDAP should provide a computer assisted data reduction without impeding the scientific aspects of the task. A good interface between man and machine requires a basic study of the assisted activity in order to offer a more comfortable and more efficient solution without radically changing the methods which are applied in the current state of art.

A basic study of the data reduction in our present context has shown that the data acquisition system was used as a kind of recording center for a wide range of technological experiments and that the number of channels, the range of frequencies as well as the recording time could considerably vary from one type of experiment to the other. Furthermore, it was determined that the data reduction was a part of the research work and could be considered as an iterative process. This process receives the sampled values as input and should deliver, after suitable treatment, numerical values or curves which are suitable for interpretation and adequate to document the scientific aspects of the experiment. To direct such a process, nobody is more qualified than the scientist in charge of the experiment. The flexibility should be provided by a set of commands which he can use in many different combinations to perform his own data reduction. The system should be conceived as a tool which enables the scientist to addapt the data processing to the task he performs. The shape of a cross-correlation curve for instance can lead to new investigations which are easy to perform if a sufficient modularity has been implemented in the system.

As a consequence of the very general guidelines of SEDAP, the system is not restricted to experiments of a specific technical or scientific area (e. g. mechanical engineering).

However, the system should be very suitable for the investigation of all experiments, which may be characterized by the following attributes

- the expected information from the experiment is contained (maybe hidden) in the evolution of a number of measured signals over a period of time
- the algorithms which are suitable to make the important experimental information evident, are not completely known beforehand, but must rather be developed or selected iteratively during the data analysis and interpretation process
- the number of signals should not exceed 64
- the total number of sampled data should not exceed a few millions.

1.3 The concept of "Experimental Records"

Once a technological experiment has been brought into a desired initial state the active part of the experiment can begin and a number of state variables are recorded by an appropriate instrumentation over a certain period of time (from milliseconds to hours). These sampled signals are assigned to channels and contain the basic information related to the experiment. Within the frameword of SEDAP, the digital representation of such a signal is called an "experimental record". The experimental records are generally a sequence of equispaced numerical values sampled at constant frequency and which are the basic quantities of data considered for the process.

The following parameters are associated with the definition of an experimental record:

a) the name is any combination of four valid alphanumeric characters used to address the experimental record.
b) the length is the number of points covered by the experimental record. The length is not stored as a formal parameter but can be calculated from the three following para-

meters.

c) the first pointer is the absolute address of the first storage block in the storage file.

d) the last pointer is the absolute address of the last storage block in the storage file.

e) the filling factor takes into account the fact that the last block may be incompletely filled.

f) the frequency is the sampling frequency which determines the time interval between two consecutive points.

g) the date (day, month, year).

h) the time (seconds).

Since SEDAP was conceived as a processor of experimental records, the concept of experimental record is fundamental to understand the organization of the system.

2. PROGRAM SYSTEM DESCRIPTION

2.1 Storage-management

The vast amount of data which can be processed by SEDAP has required the use of a large storage capability and the storage management is one of the most important features of the system

2.1.1 The warehouse

The main storage area is called the LAGER or the WAREHOUSE. The warehouse is a direct access file created by a special subroutine call during the system initialization. The statement defining the file is of the following type:

```
DEFINE FILE 40 (5000,512,U,IA)
```

which calls for the following remarks.

1) The index assigned to the file is 40 and requires a concording DD card with the proper space allocation.
2) The number of blocks or physical records is 5000. Since this argument cannot be represented by a variable integer like ISIZE in the described system and since it is reasonable to allow the user to specify the dimension of his storage file, different defining calls are provided within SEDAP. During the initialization of SEDAP the smallest of the seven options which can satisfy the storage needs specified by the user is selected after some straightforward computation.
3) The size of the physical records or blocks is 512 words. This choice has been motivated by two considerations:
   - 0.5 K is a good compromise for the medium-sized experimental records and is compatible with the output of the presently used data acquisition system which blocks the recorded data into 1 K physical records (1024 points) on the magnetic tape.
   - A storage using records comprised of 2^N is especially suitable for the use of fast Fourier transforms algorithm.
4) U indicates that READ and WRITE operations are performed

without format control. This mode achieves a faster transfer speed and implies that the words will be moved or copied back and forth without any transformation or interpretation.

5) IA is the integer variable also called associated variable and points to the IA-th block when accessing the file.

6) The expression "experimental record" is derived from the fact that we are concerned with data recorded in performing an experiment and was greatly influenced by the concept of "records" described by Hoare /1/. A possible confusion exists when one refers to the physical records used in the storage file. To avoid any confusion in the following pages, the word record will be reserved for the experimental records while the physical records will be exclusively called blocks. The reader should be aware that this practice is in contradiction with a current convention which consists of grouping logical records into blocks.

2.1.2 The catalog

The management of the warehouse requires some elementary bookkeeping which gives an exact account of its content. This is achieved by a catalog located in the COMMON storage area and divided to provide a two level information:

a) The warehouse level

The warehouse level is comprised of three parameters:

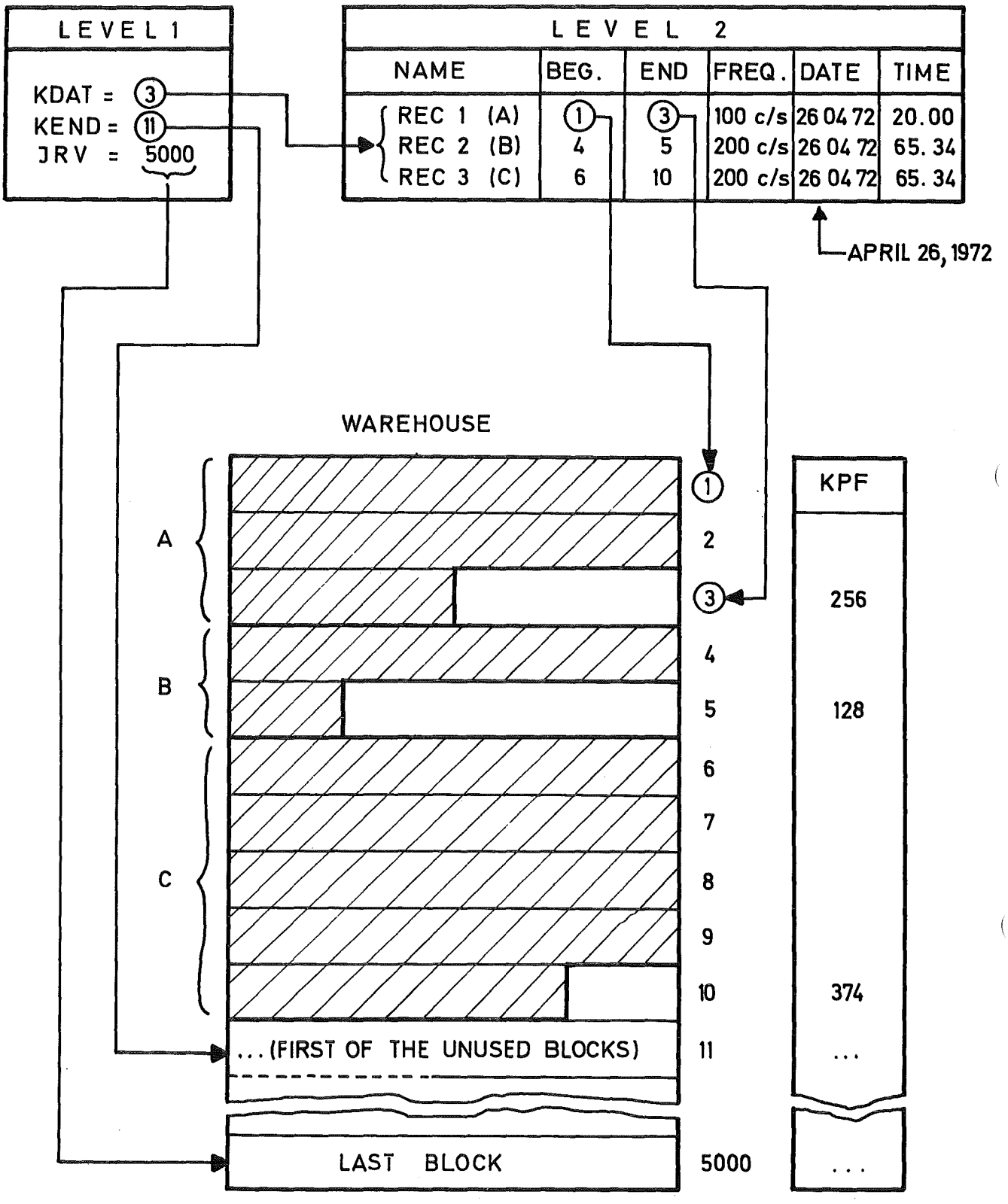
KDAT which indicates the number of records contained in the warehouse

KEND which represents the value of the associated variable pointing to the next unused block (i. e. the warehouse contains (KEND - 1) blocks)

JRV carries the maximum number of blocks which can be stored in the warehouse according to the specification given by the user (limit = 5000)

b) The record level

All the records contained in the warehouse are tracked by the following catalog parameters:



PRINCIPLE OF THE CATALOG

BENAM(K): contains the name of the record K ($1 \leq K \leq KDAT$)
NANF(K): points to the first block of the record K
NEND(K): points to the last block of the record K
WFREQ(K): stores the sampling frequency of the record K
ADAT(K): stores the coded expression of the date (260472 = 26 th day of April 1972) of the record K
BZEIT(K): stores the time corresponding to the first value of the record K. The time is computed in seconds and the time origin (0.0) corresponds to the first value recorded on a tape.
KPF(K): is the filling factor of the last block of a record where any value from 1 to 512 can be expected.

2.1.3 The dumping file

The warehouse is a direct access file which can only exist during the execution of a job and which is destroyed after the completion of the computer run. The user has the possibility to dump a part or the totality of the warehouse on a magnetic tape and to restore the records in a subsequent job. This feature will be described in the following chapters but should be mentioned here as an extension of the storage.

2.2 The main program

The most important functions of the SEDAP process are performed by the main program which

- initializes the system
- receives the commands
- checks the validity of the requests
- formulates the resulting tasks
- supervises their execution
- acknowledges their completion

and orderly closes the system when the process is terminated or when a severe error has been detected.

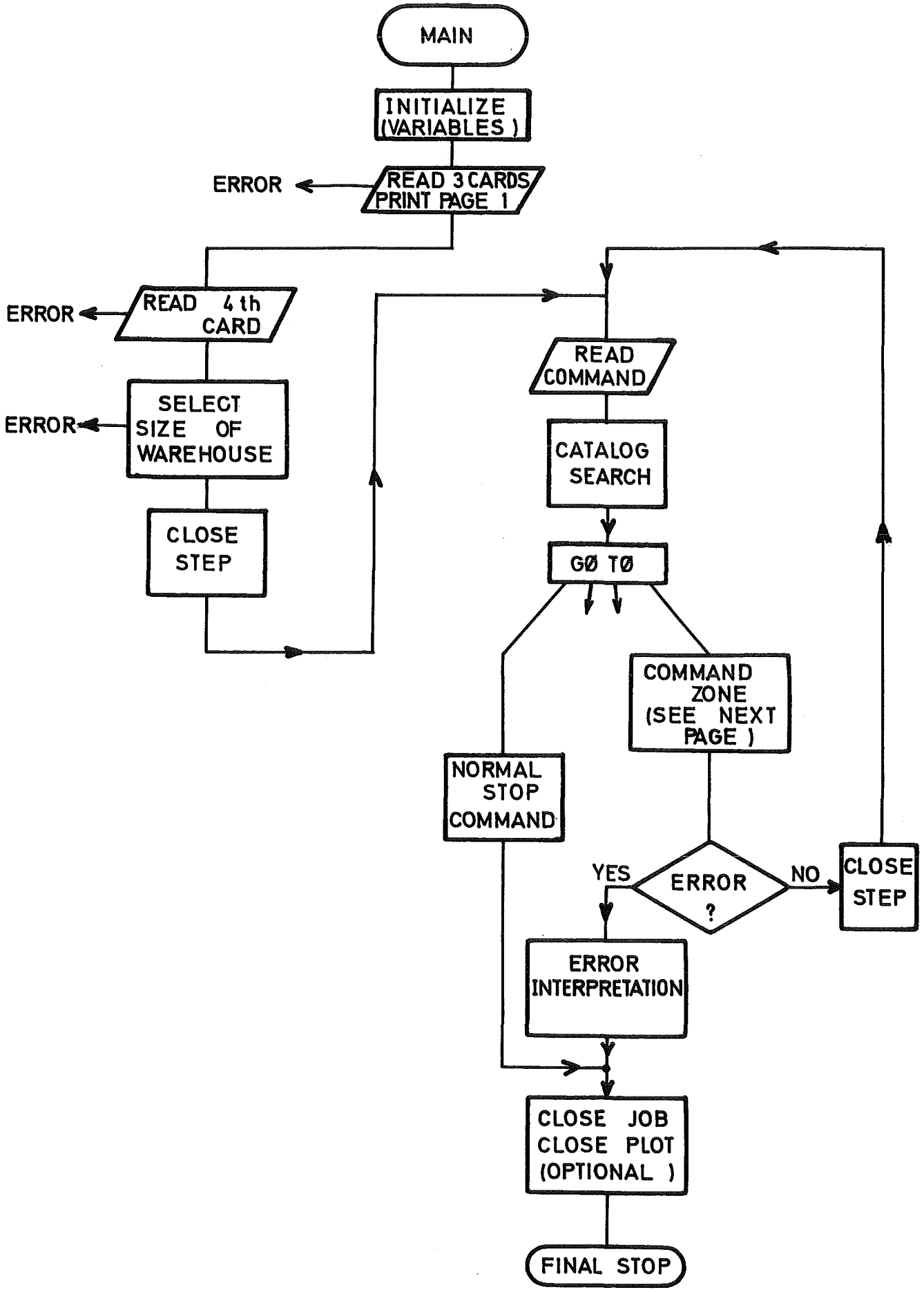
2.2.1 Initialization of SEDAP

The program initializes the service variables, sets the job timer to zero, and reads the first card which contains the system identification (Name "SEDAP") and a eight character title stored in ZNAM. If the identifier is not correct, the initialization is stopped and the job is terminated with an error code IERR = 11. The two next cards are read and the 160 characters reserved for the user's comments are stored. A full page is printed with the system heading and the user's title by calling the special subroutine A8FORM /2/. The two lines of comments are added at the bottom of the page.

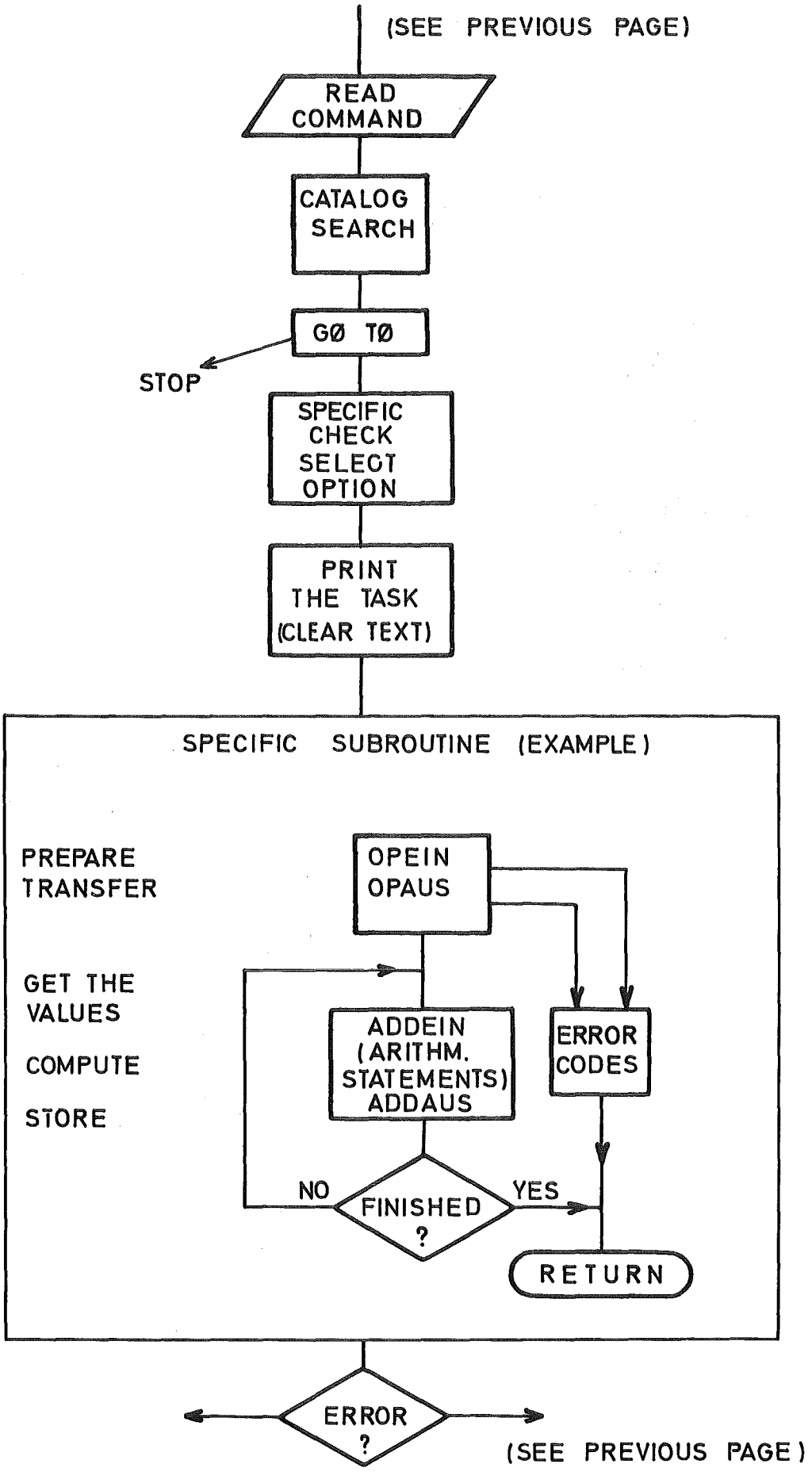
The initialization is almost terminated but the fourth card which is handled by the normal command interpreter (see 2.2.2) belongs to the initialization. This card must be a SEDA card with the parameters which are required to specify the size of the warehouse and the possible options. If the first command card (it is the fourth of the deck) does not begin with SEDA, the job will be terminated with an error code IERR = 11.

The system sets the two options indicators KSTOP and KDUMP to zero. KSTOP will be changed to +1 if the error test option has been specified (this option is used by the system's programmer for testing purposes when programmed errors justify a restart of the system after the error interpretation). If the user has specified the automatic dump option, KDUMP will be stored as +1 and the dump file number passed by INT(3) will be stored by the integer KFILE for later use.

The size of the warehouse can be selected between seven standard sizes comprised between 100 and 5000 blocks. This requires seven similar subroutines where a corresponding DEFINE FILE statement opens the file 40. The smallest size which satisfies the number of blocks passed by INT(1) is called and the real size of the warehouse is stored by IRV which is used to detect a possible warehouse overflow (OPAUS). If the user



MAIN (GENERAL STRUCTURE)



MAIN (COMMAND ZONE)

has requested more than 5000 blocks, the job is terminated with an error code IERR = 17.

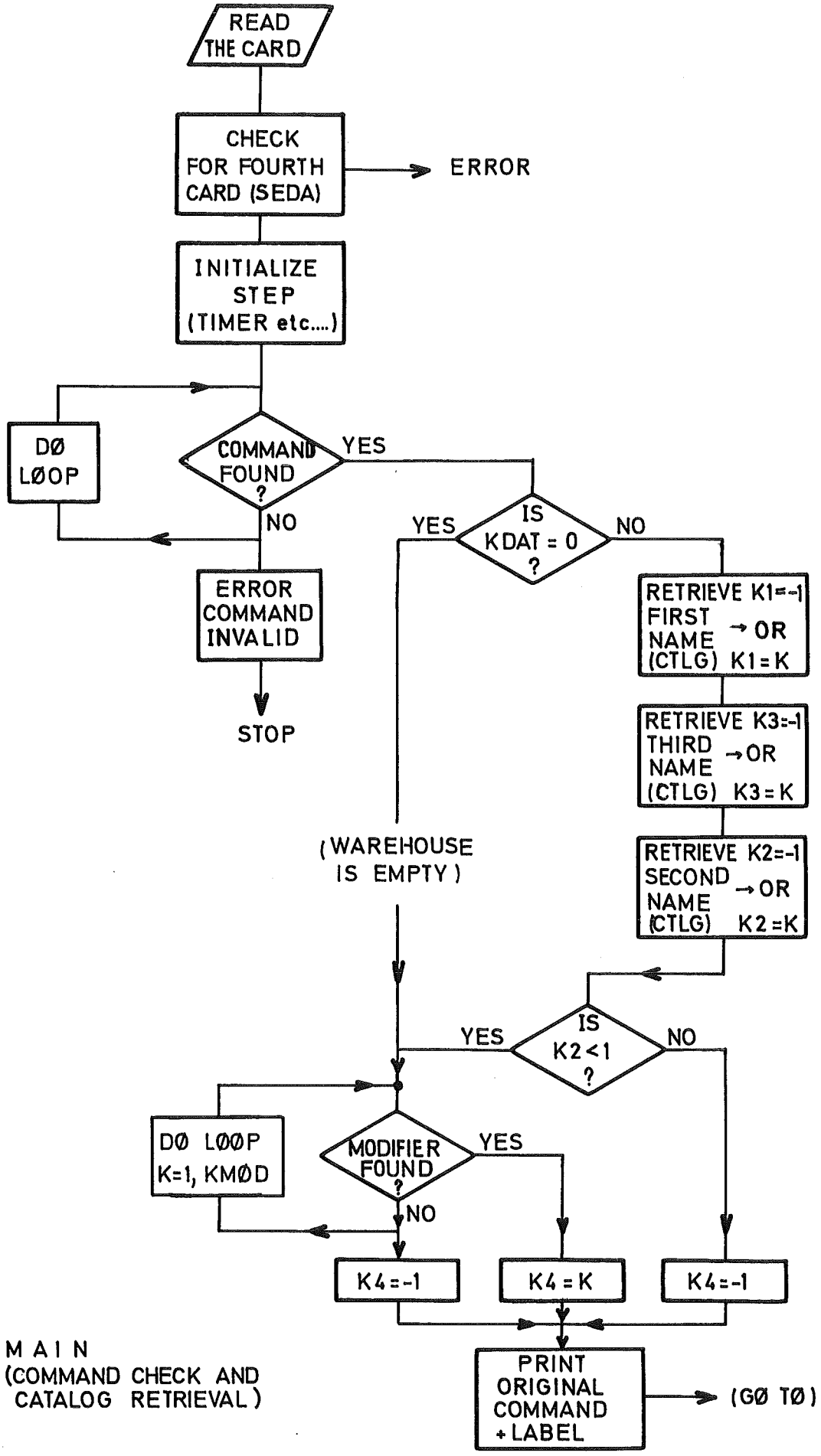
If a second SEDA command is received during any further phase of the job, the card will be normally processed but a second access to any of the DEFINE FILE subroutine will be protected by an IF statement which verifies if the card index NZAE is equal to one. This card will then only change the DUMP or RESTART options and can allow an ON/OFF switching of the two features during the execution of a job.

2.2.2 Command interpretation and execution

a) Preparation of the task

When the system has been initialized, the main program is ready for the processing of the different tasks specified by commands. This operation is organized according to a general scheme. The command card is read and the task timer is reset to zero. The validity of the command is checked by matching the first word against the keywords of the commands list. An invalid command causes an interruption and the whole processing is stopped. A successful retrieval determines the index of the command and the resulting KTYP parameter will be later used to branch to the appropriate specific zone of the main program. The card is then printed in his original punching format with a differentiated underlining pattern which provides a clear contrast in the case of a shift due to a punching error.

The second operation consists of systematically searching the catalog to see if the three experimental record names which can be associated to a command name match with names contained in the warehouse catalog. If a search has been unsuccessful, the K index remains equal to -1 but if the name is known, the K index will be replaced by the value corresponding to the position of the name in the catalog. The search is performed by the ENTRY CTLG for the first name with the index K1, for the third name with K3 and for the second name with K2. A special case is involved since the second name can



MAIN
(COMMAND CHECK AND CATALOG RETRIEVAL)

be specified as a modifier and if K2 remains equal to -1, the matching of the second name will be extented to the modifiers list with the resulting index K4 remaining -1 or being replaced by the position of the name in the modifiers list. This points to one of the system limitations: the keywords used as modifiers should never be used as experimental record names. If the warehouse is empty all these tests are bypassed with the exception of the determination of the index K4 which is not bound to the contents of the warehouse.

b) Specific processing of a task
The value of KTYP which has been previously determined is used to transfer the control to a region of the main program which has been specifically designed to handle a given type of command.

According to the type of commands, some preliminary checks can be performed to select a given option or to insure that the command has been formulated in a valid context. The system has now to print a clear text interpretation of the command which must transform the coded parameters into an easily understandable statement. Different elements of formatted sentences can be concatenated in a modular way (within the limits of FORTRAN IV) to provide a storage saving reduction of the text.

The control is then passed to a specific subroutine which will handle the task (The main program performs the execution of some simple tasks without external support for the simpler cases like renaming a record or clearing the warehouse). This subroutine can eventually complement the previous task formulation and initiates the transfer operations. The transfer is described in details in the following pages (2.3) and it is sufficient to explain that the input/output requests specified by the command will be checked to see if they are compatible with the situation of the warehouse. This involves the examination of the parameters K1 to K4 and the test can be extended to the other arguments like the sorting factor, the file numbers or other numerical values

which are described in relation with the specific subroutines. If the request is valid, the task is performed and the control is passed back to the main program. In the mean time the control could have been transferred if any severe error has been detected, the minor errors cause only the printing of a warning.

c) End of the task
As it will be explained in the chapter concerning the error handling, the error situation is immediately checked upon the return into the main program. If an error has been detected, the control is shifted to the error zone where a detailed interpretation of the error is provided. If no error code has been issued, the end of the task is acknowledged by the main program which prints the value stored in the task timer and the system is ready to process the next command.

2.2.3 End of the job

The normal termination of a SEDAP job is issued when processing the final command which is called STOP and which causes the total time needed for the job to be printed. The control can also be passed to the so-called STOP zone if a severe error has been detected. In that case the FEHLER subroutine is called to provide an interpretation of the error (see error handling). Before jumping to the final STOP statement, the MAIN program checks if an automatic dump of the warehouse contents has been specified. This will cause a call to the DUMP subroutine to secure the back-up copy of the warehouse. The termination of the job due to an error will be delayed if the user has given a special password with the initial system's call. In that case the system's programmer has the intention to test the error system and the next card will be read after the error code IERR has been reset to zero. Due to the peculiarity of the Plot package used by SEDAP a call ENDPLT (end of the plot) is required before the execution of the last STOP to orderly close the PLOT file.

CROSS REFERENCE LIST OF SUBROUTINE AND FUNCTION CALLS

```
AL100
AL250
AL500
AL750
AL1000
AL2500
AL5000
A8FORM
BEFA        OPEIN ADDEIN OPAUS ADDAUS SQRT ATAN2
CTLG
DAGEN       OPEIN ADDEIN OPAUS ADDAUS SIN COS RANDU
DAKA
DATUM
DIFINT      OPEIN ADDEIN OPAUS ADDAUS TRAP SIMP DIF3
JUMP        OPEIN ADDEIN
ERAKON      OPAUS ADDAUS
EXTSED
FEHLER
FILTER      OPEIN ADDEIN OPAUS ADDAUS FIL13 FIL15 FIL35 FILVAR FILHAN
FOUR        OPEIN ADDEIN OPAUS ADDAUS FOUR1 SQRT
GRAPH           (ENTRY GRAPH1)
GRAPH1      OPEIN ADDEIN PLOTA AMIN1 AMAX1 MINO MAXO MOD
HOLE        OPAUS ADDAUS CLTG
LAGER           (ENTRY CTLG)
MEPODE      OPEIN ADDEIN OPAUS ADDAUS FOUR1 POT2 HAGL MIWESU MIWIBU HYPER
OPERA       OPEIN ADDEIN OPAUS ADDAUS TNICR2 CDVD# CMPY#
PAPTAP      OPAUS ADDAUS PCHCK RECO PDUMP MOD
PLOTC
PRINT       OPEIN ADDEIN
SORTIK      OPEIN ADDEIN OPAUS ADDAUS CTLG MOD
STATUT
WERT        OPEIN ADDEIN OPAUS ADDAUS
ZEIT
```

2.3 Transfer-subsystem

The handling of the experimental records by the different parts of the program requires a continual transfer of data back and forth between the core and the warehouse. The large size of some types of experimental records and the modest dimensions of the computing arrays make it necessary to split the experimental records into working segments (or pages) which correspond to the computing arrays of the different subroutines. This operation is controlled by the TRANSFER subsystem which can be divided into two zones according to the direction of the transfer.

2.3.1 Computing arrays

The transfer subsystem involves different computing arrays which are stored in the COMMON area. The computing arrays are basically the X, Y and Z fields, respectively dimensioned with 10240, 10240 and 5120 points. The transfer can also involve the XYZ array which is an equivalent (using an EQUIVALENCE) form of the three previous fields. Since the use of the XYZ field is subordinated to the buffer ZZ located at the end of the Z array, the dimension of XYZ can be extended only up to 25088 (25600 - 512). The three arrays X512, Y512 and Z512 are used as buffers for the fast transfer mode and are equivalent to the 512 first values of the X, Y and Z arrays.

2.3.2 Transfer from the warehouse to the computing arrays

2.3.2.1 Preparation of the transfer

Before the transfer operations are performed, some preliminary checks are necessary to insure that the transfer will be possible. Since the operation presents some analogy with what a computing system does when it opens a file, the initialization of the transfer has been called OPEIN (= Open-in).

The first step is to verify if the experimental record has been found in the warehouse and this is materialized by a positive value of KN after a successful retrieval in the catalog. (See Description of the Main program). The validity of

COMMANDS:	OPEIN opens an experimental record stored in the warehouse, checks its existence and the validity of the request. OPEIN returns the arguments needed for further access.	NAME = OPEIN SYSTEM = TRANSFER ENTRY = see ADDEIN
None		

CALL OPEIN(KANW,KENW,RNAM,KN,KRAF,ZEYT,MAX,LKPT,FREQ,DAT)

LIST OF ARGUMENTS:

KANW is the first delimiter which carries the relative address of the first block to be read. KANW is returned as the absolute address of the block in the storage file.

KENW carries the address of the last block to be read and is also returned as an absolute address.

RNAM is the name of the experimental record to be read.

KN is the index of the experimental record in the record list. If KN < 1, the record does not exist.

KRAF is the sorting factor which will be applied to the input. If KRAF = 0, the default value KRAF = 1 is applied, if the value is negative, the sign is changed, but if KRAF > 100, the request is rejected.

ZEYT returns the time corresponding to the first transferred value.

MAX indicates the maximum of values which can be processed by the task.

LKPT returns the total number of points which results from the request expressed in blocks.

FREQ returns the resulting frequency after application of the sorting factor (default value: 1.0 Hz).

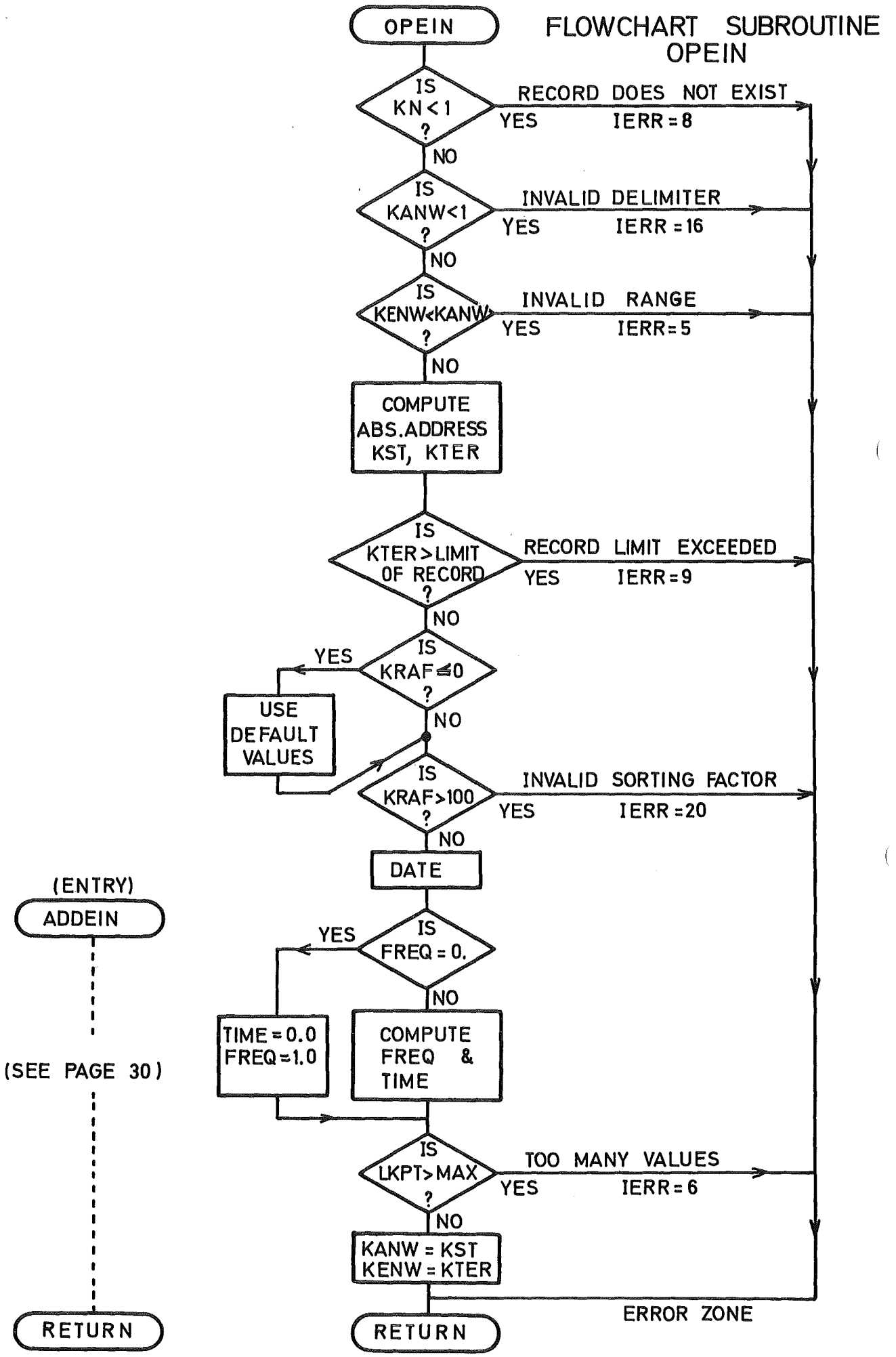
DAT returns the date of the record.

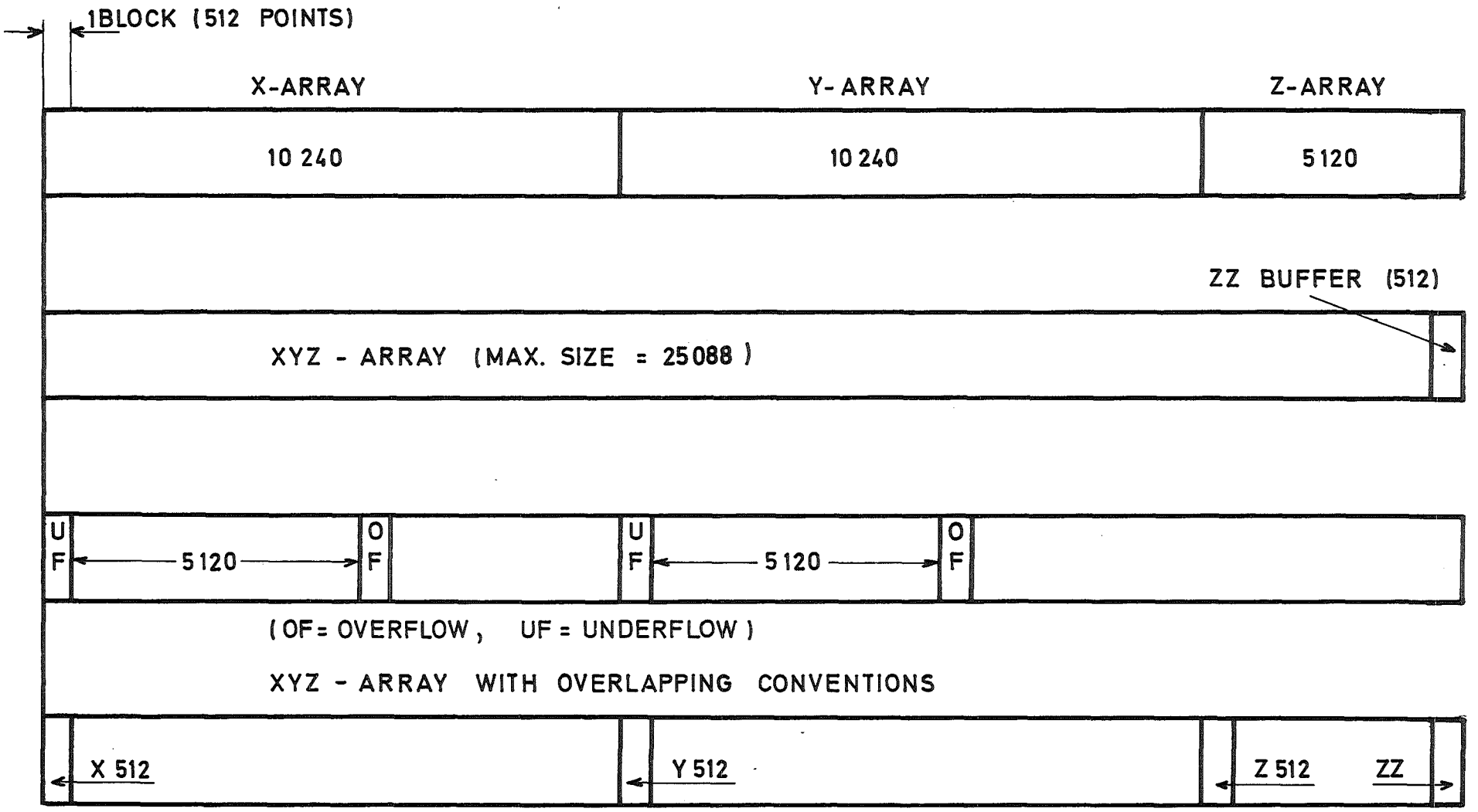
SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 4,5,6,8,16,20

INDIRECT:


FLOWCHART SUBROUTINE OPEIN
OPEIN
IS KN<1 ?
RECORD DOES NOT EXIST
YES
IERR=8
NO
IS KANW<1 ?
INVALID DELIMITER
YES
IERR=16
NO
IS KENW<KANW ?
INVALID RANGE
YES
IERR=5
NO
COMPUTE ABS.ADDRESS KST, KTER
IS KTER>LIMIT OF RECORD ?
RECORD LIMIT EXCEEDED
YES
IERR=9
NO
IS KRAF≤0 ?
YES
USE DEFAULT VALUES
NO
IS KRAF>100 ?
INVALID SORTING FACTOR
YES
IERR=20
NO
DATE
(ENTRY)
ADDEIN
IS FREQ=0.
YES
TIME=0.0 FREQ=1.0
NO
COMPUTE FREQ & TIME
(SEE PAGE 30)
IS LKPT>MAX ?
TOO MANY VALUES
YES
IERR=6
NO
KANW=KST KENW=KTER
ERROR ZONE
RETURN
RETURN

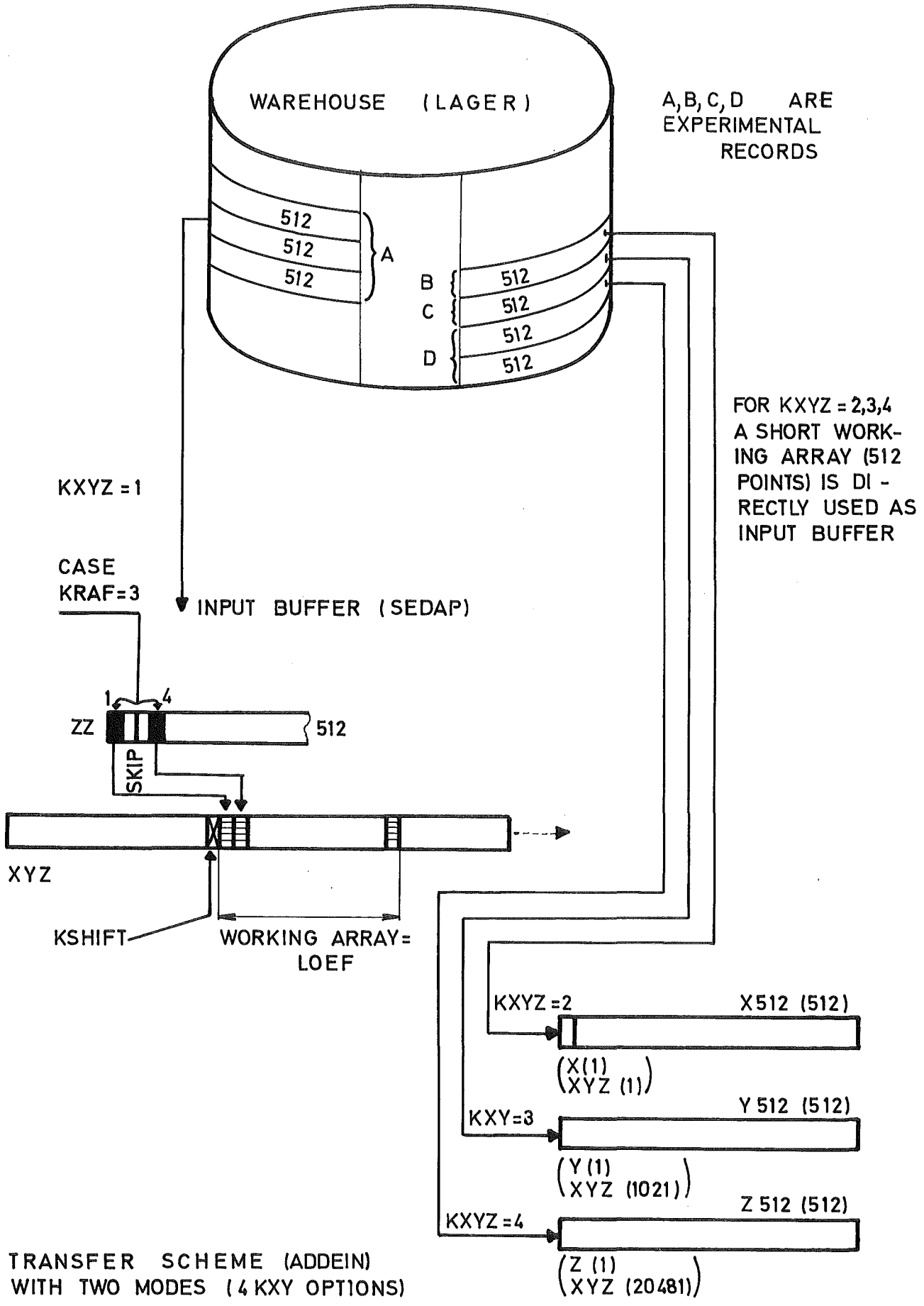


COMPUTING ARRAYS IN SEDAP
(SIZE AND EQUIVALENCE)

```
      SUBROUTINE JPEIN(KANW,KENW,RNAM,KN,KRAF,ZEYT,MAX,LKPT,FREQ,DAT)
C     ---------------------------------------------------------------
C
      COMMON X (10240),Y (10240),Z (5120),
     1 BENAM(512),NANF(512),NEND(512),WFREQ(512),ADAT(512),BZEIT(512),
     2 KDAT,KEND,NC,NP,IA,JRV,X1,X2,Y1,Y2,IERR,AERR,BERR,JERR,KERR
     3 ,KPF (512)
      DIMENSION XYZ(16384),ZZ(512),X512(512),Y512(512),Z512(512)
      EQUIVALENCE (XYZ(1),X(1),X512(1)),(ZZ(1),Z(4097)),
     1  (Y512(1),Y(1)),(Z512(1),Z(1))
C  VERIFY EXISTENCE OF RECORD (ERROR CODE = 8)
      IF(KN.LT.1) GO TO 97
C  CHECK IF THE FIRST DELIMITER IS POSITIVVE (ERROR CODE = 16)
      IF(KANW.LT.1) GO TO 95
C  CHECK THE POSITIVE PROGRESSION OF DELIMITERS (ERROR CODE = 5)
      IF(KENW.LT.KANW) GO TO 94
C  FIND THE ABSOLUTE ADDRESS OF THE DELIMITERS
C  CHECK IF THE UPPER LIMIT OF THE RECORD IS EXCEEDED(E.C. = 4)
      KST = NANF(KN) + KANW - 1
      KTER = NANF(KN) + KENW - 1
      IF(KTER.GT.NEND(KN)) GO TO 99
C  SET DEFAULT VALUES FOR AN INVALID SORTING FACTOR
      IF(KRAF.EQ.0) KRAF = 1
      IF(KRAF.LT.0) KRAF = -KRAF
      IF(KRAF.GT.100) GO TO 93
C  COMPUTE THE RESULTING NUMBER OF POINTS
      KPT = 0
      KBDIF = KENW - KANW + 1
      KPT = KBDIF * 512
      IF (KTER.EQ.NEND(KN)) KPT = KPT-512+KPF(KN)
      LKPT = KPT / KRAF
C  DETERMINE THE DATE,TIME AND THE FREQUENCY  FOR THE SEGMENT
      START = KANW - 1
C  FLOAT KRAF
      FARK = KRAF
      DAT = ADAT(KN)
      IF(WFREQ(KN).EQ.0.) GO TO 13
      FREQ = WFREQ(KN)/FARK
      ZEYT = BZEIT(KN) + (START * 512. / FREQ)
      GO TO 14
   13 ZEYT = 0.0
      FREQ = 1.0
C  CHECK IF THE MAX.   TRANFERABLE VALUE IS NOT EXCEEDED
   14 IF (LKPT.GT.MAX) GO TO 98
      KANW = KST
      KENW = KTER
      RETURN
C
      ENTRY ADDEIN(KANW,LKPT,LOEF,KRAF,KSHIFT,IMELD,IMENGE,KXYZ,ISTAT,
     1 IOFLOW,IUFLOW,KOFLOW,KUFLOW)
C
      ID = 8
      KSTORE = LKPT
      IUFLOW = 0
      IOFLOW = 0
      GO TO (10,40,40,40,40),KXYZ
C  IF NOT THE FIRST PART OF SEGMENT AND IF U-FLOW REQUESTED,SAVE IT
   10 IF(KUFLOW.EQ.0) GO TO 12
      IF (ISTAT.EQ.0) GO TO 12
```

```
      DO 8 I= 1,KUFLOW
      IU1 = KSHIFT -I + 1
      IU2 = KSHIFT + LOEF + 1 - I
    8 XYZ(IU1) = XYZ( IU2)
      IUFLOW = KUFLOW
C TRANSFER  THE VALUE OF LOEF INTO JOEF FOR DECREMENTING
   12 JOEF = LOEF
      IX = KSHIFT + 1
      IZ = 1
   15 IA = KANW
      READ (40'IA, ERR=96) ZZ
      KANW = KANW + 1
   20 CONTINUE
      XYZ (IX) = ZZ (IZ)
      LKPT = LKPT - 1
      JOEF = JOEF - 1
      IZ = IZ + KRAF
      IX = IX + 1
C  CHECK IF ALL THE POINTS HAVE BEEN TRANFERRED
   21 IF(LKPT.EQ.0) GO TO 22
C  CHECK IF THE BUFFERING ARRAY IS FULL
      IF(JOEF.EQ.0) GO TO 24
C  CHECK IF THE INPUT BUFFER IS  DEPLETED
      IF(IZ.LE.512) GO TO 20
      IZ = IZ - 512
      GO TO 15
   22 IMENGE = KSTORE
      JSTAT = 2
      GO TO 23
   24 IMENGE = KSTORE - LKPT
      JSTAT = 1
      IF(KOFLOW.EQ.0) GO TO 23
      IREST = LKPT / KRAF
      IF(IREST.GT. KOFLOW) IREST = KOFLOW
      IA = KANW
      READ(40'IA,ERR=96)ZZ
      DO 27 I = 1,IREST
      IO1 = KSHIFT + LOEF + I
      IO2 = 1 + (KRAF * (I-1))
   27 XYZ(IO1) = ZZ(IO2)
      IOFLOW = IREST
   23 IF(ISTAT.GT.0) GO TO 34
      IF(IMELD.EQ.0) GO TO 34
C  PRINT THE 8 FIRST VALUES FOR CONTROL PURPOSE
C  CHECK THE CASE OF A RECORD CONTAINING LESS THAN 8 VALUES
      IF(IMENGE.LT.ID) ID = IMENGE
      WRITE(NP,102)(XYZ(KSHIFT + I),I = 1, ID)
   34 ISTAT = JSTAT
      RETURN
C  FIND THE VALUE OF KSHIFT  FOR THE FIRST VALUE OF X512,Y512 AND Z512
   40 KSHIFT = 10240 * (KXYZ - 2)
      KUFLOW = 0
      KOFLOW = 0
      IF(KRAF.EQ.1) GO TO 48
C  IF THE SORTING FACTOR IS NOT 1 DOWNGRADE KXYZ TO 1(NORMAL CASE)
      LOEF = 512
      KXYZ = 1
   48 IA = KANW
      GO TO (10,50,60,70),KXYZ
```

```
   50 READ(40'IA,ERR=96)X512
      GO TO 80
   60 READ(40'IA,ERR=96)Y512
      GO TO 80
   70 READ(40'IA,ERR=96)Z512
   80 KANW = KANW + 1
      IF(LKPT.GT.512) GO TO 26
      LKPT = 0
      GO TO 22
   26 LKPT = LKPT - 512
      GO TO 24
   93  IERR = 20
      JERR = KRAFF
      JERR = KRAF
   90 RETURN
   94 IERR = 5
      AERR = BENAM(KN)
      JERR = KANW
      KERR = KENW
      GO TO 90
   95 IERR = 16
      JERR = KANW
      GO TO 90
   96 IERR = 3
      JERR = 40
      KERR = KANW
      GO TO 90
   97 IERR = 8
      AERR = RNAM
      GO TO 90
   98 IERR = 6
      JERR = LKPT
      KERR = MAX
      GO TO 90
   99 IERR = 4
      AERR = RNAM
      KERR = KENW
      JERR = NEND (KN) - NANF (KN) + 1
      GO TO 90
C
  102 FORMAT ( '  KONTROLLWERTE INPUT = ', 8(E12.6,1X))
C
      END
```



TRANSFER SCHEME (ADDEIN) WITH TWO MODES (4 KXY OPTIONS)

the two parameters KANW and KENW which delimit the selected segment must undergo the following tests:
KANW must be positive
KENW cannot be smaller than KANW
KENW must not exceed the limit of the experimental record.

The values of KANW and KENW which were provided by the command card and which were related to the experimental record are then replaced by their absolute value as pointers of the storage file.

Since the sorting factor KRAF must be comprised between 1 and 100 (both values included), the request is rejected for any value larger than 100 and the default value KRAF = 1 is automatically selected when the value is negative or equal to zero.

The frequency of the experimental record is divided by the sorting factor to become the new sampling frequency of the selected segment. If the origin of this segment is not the origin of the experimental record from which it has been extracted, the new time origin is shifted accordingly. If the frequency stored in the catalog is zero, this computation is not possible and the segment will be transferred with a time origin equal to zero associated with a sampling frequency of 1.0 Hz. The number of points involved in the transfer is computed and matched against a maximal limit set for MAX before returning the control to the calling program.

2.3.2.2 Execution of the transfer
The basic transfer method consists of moving the stored values into XYZ array by a successsion of elementary transfers until the input request has been satisfied. This operation requires a succession of ADDEIN (Add-in) calls. ADDEIN is an ENTRY in OPEIN and starts by reading the block pointed by KANW into the ZZ buffer. The LKPT values are transferred one by one into the XYZ array starting at the address immediately following the index KSHIFT and during this operation some values are dropped or skipped if a sorting factor has been specified. When the total number of points has been

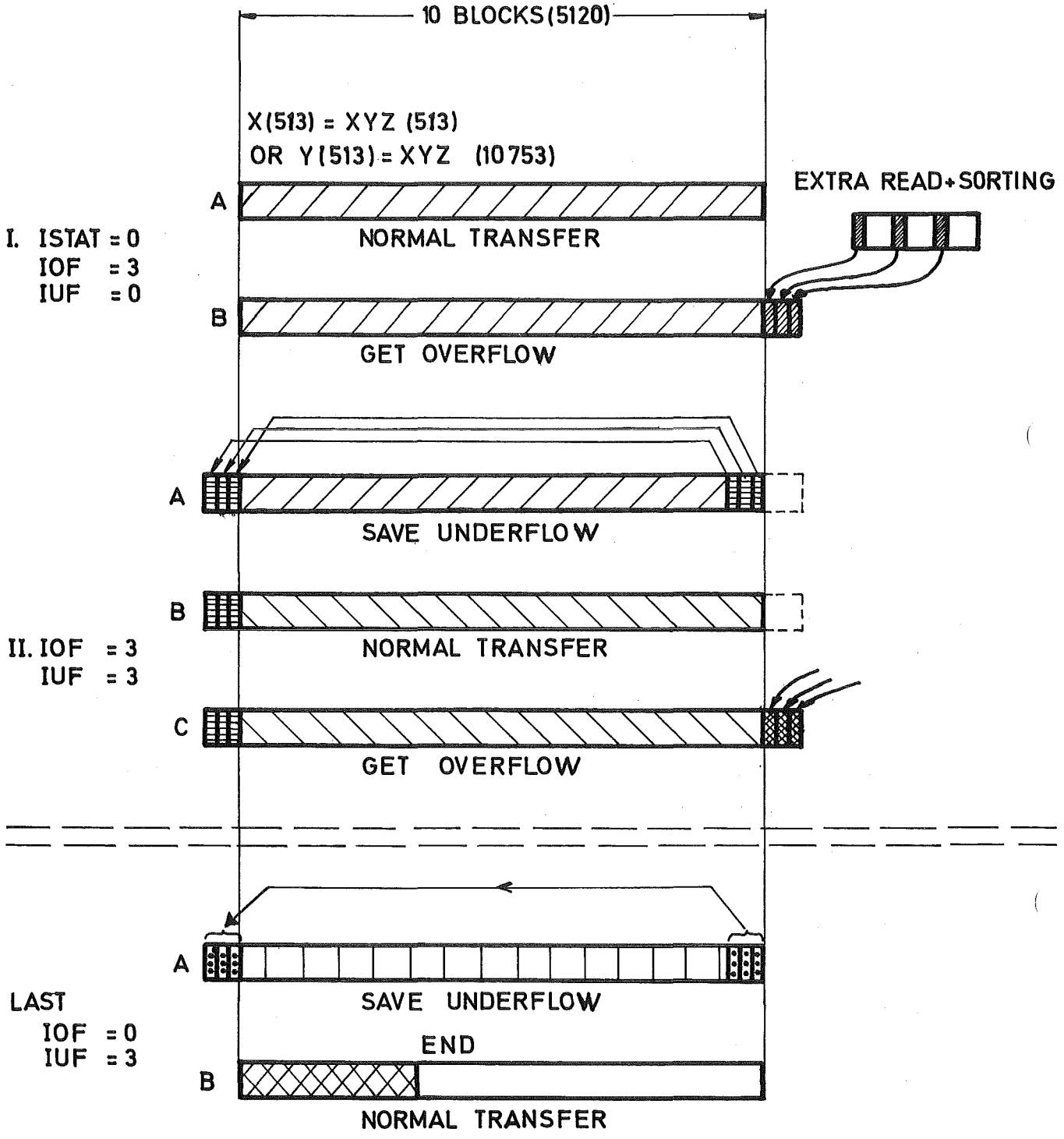
reached, LKPT is down to zero and the transfer is completed. If in the mean time the working quantity (= LOEF and always a multiple of 512) has been exhausted, the control is returned to the calling program to perform the specified computation and the process continues. Any of these operations can be shortly interrupted as soon as the input buffer is depleted and is resumed after the reloading with the next 512 values. The pointer KANW is incremented after every READ and does not need to be tracked by the calling subroutine, this remark is also applicable to LKPT which indicates the number of points transferred by an ADDEIN call is always equal to LOEF except for the last call (the first can be the last if it is the only one) and is given by the parameter IMENGE.

Furthermore, the parameter ISTAT carries an information about the status of the transfer. ADDEIN begins the transfer with a value of ISTAT which should be set to zero by the calling program before executing the first call. ADDEIN changes ISTAT to 1 if a continuation is expected or to 2 if the termination is acknowledged. If ADDEIN receives a message indicator IMEL = 1 from the calling program, the first available values (up to eight) of the transfer will be printed to provide the SEDAP user with a control of the operation. This feature requires the conjunction of IMEL= 1 with ISTAT = 0;

The necessity of segmenting the records into working quantities also called computing arrays can be a handicap when the computation involves not only the instantaneaous value but other adjacent values. The case is illustrated by the following widely used 5 point smoothing algorithm:

$$Y(I) = 0.2 * (X(I-2) + X(I-1) + X(I) + X(I+1) + X(I+2))$$

This mathematical expression will handle all the values from the third to the (n-2)th and a special treatment generally performed by a degenerated form of the previous algorithm will be required for the two first values as well as for the

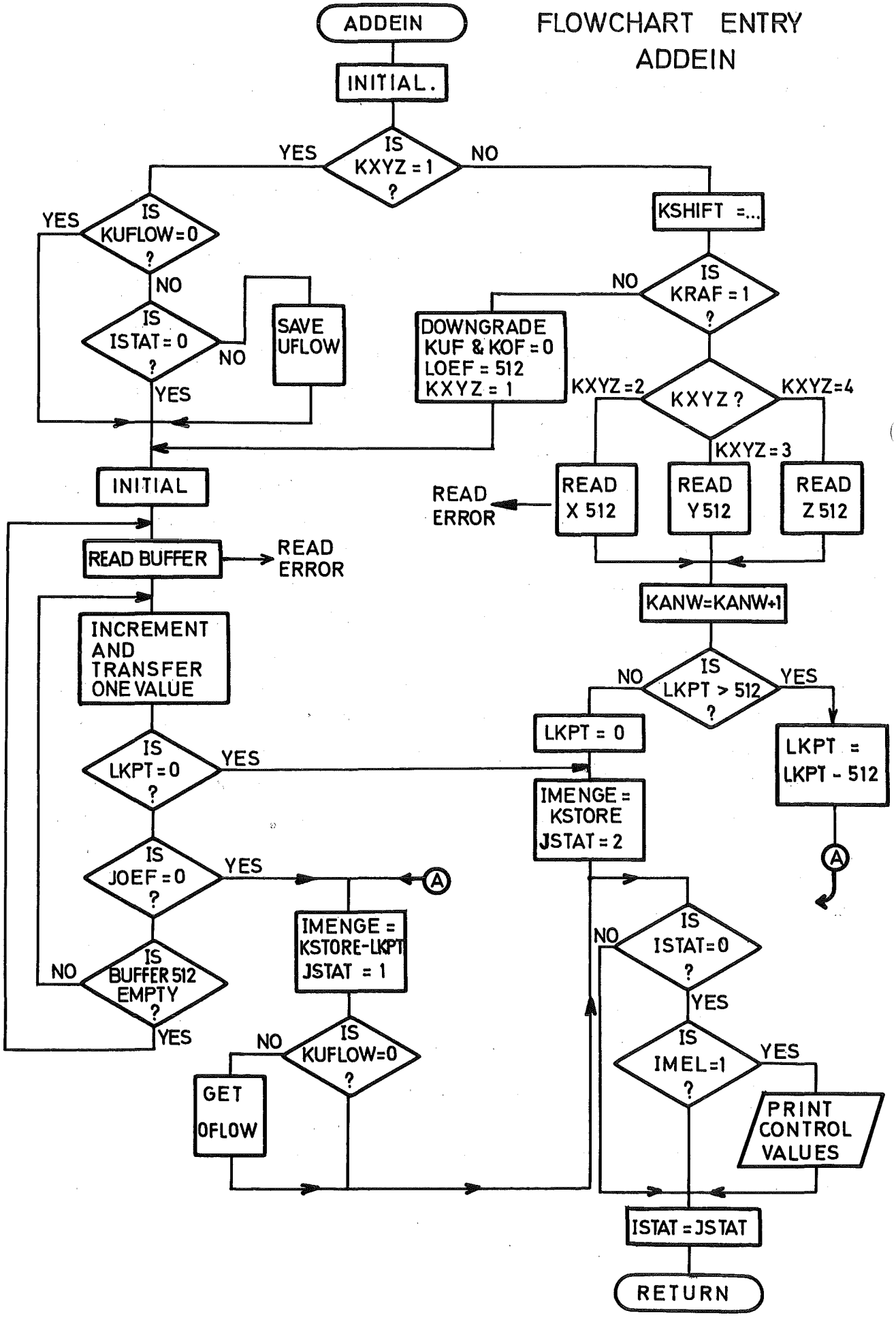


ADDEIN TRANSFER WITH OVERLAPPING
(CASE SHOWN WITH KOF=3, KUF=3)

two last values. To avoid this corner effect at the junction of two segments, provision has been made in the SEDAP concept to insure the uniform continuity of any computation which does not exceed five adjacent values in both directions. The limit is theoretically 512 values in both directions if a sorting factor of 100 were not to be guaranteed. In the previous example, an intermediary segment (for instance the second if there are at least three segments) should exhibit, besides his own values, the three last values of the previous segment and the three first values of the next one, to satisfy the condition of continuity. By analogy with the dynamic behaviour of a register, the five previous values are considered as an underflow area, while the five anticipated values will be stored in a so-called overflow area. From the previous considerations, it is obvious that the first segment can have only an overflow area, the last one only an underflow while any intermediary segment will have both of them. The over/underflow requirements of a segmented transfer are functions of the selected algorithm, and this is specified by the two parameters KUFLOW and KOFLOW (alias KOF and KUF) which can take any value from 0 to 5. ADDEIN fullfills the request whenever it can be carried out and stores the number of values effectively present in the two parameters IUFLOW and IOFLOW before returning the control to the calling program. To make the computations easier in the subroutines using the overlapping features of ADDEIN, the following non imperative rules have been adopted in such cases:

- The size of the computing array is specified as 10 blocks (LOEF = 5120)
- The value of the array starting index (KSHIFT) is 512 or 10752 according to the choice between the X and the Y zone.

The transfer of values with this general mode is flexible because it allows an overlapping of the segments, reduces the number of values by a user specified sorting factor and stores the values into the XYZ array with a variable starting address. These advantages have to be paid by a larger amount of executed instructions especially in the case of a straightforward


FLOWCHART ENTRY
ADDEIN
ADDEIN
INITIAL.
IS KXYZ = 1 ?
YES
NO
IS KUFLOW=0 ?
YES
NO
IS ISTAT = 0 ?
NO
YES
SAVE UFLOW
KSHIFT =...
IS KRAF = 1 ?
NO
DOWNGRADE KUF & KOF = 0 LOEF = 512 KXYZ = 1
KXYZ ?
KXYZ=2
KXYZ=3
KXYZ=4
INITIAL
READ X 512
READ Y 512
READ Z 512
READ ERROR
READ BUFFER
READ ERROR
KANW=KANW+1
INCREMENT AND TRANSFER ONE VALUE
IS LKPT > 512 ?
NO
YES
LKPT = 0
LKPT = LKPT - 512
IS LKPT = 0 ?
YES
IMENGE = KSTORE JSTAT = 2
IS JOEF = 0 ?
YES
A
A
IMENGE = KSTORE-LKPT JSTAT = 1
IS ISTAT=0 ?
NO
YES
IS BUFFER 512 EMPTY ?
NO
YES
IS KUFLOW=0 ?
NO
IS IMEL=1 ?
YES
GET OFLOW
PRINT CONTROL VALUES
ISTAT = JSTAT
RETURN

COMMANDS: None	ADDEIN performs the successive transfers of numerical values from the warehouse into the working arrays	NAME = ADDEIN SYSTEM = TRANSFER ENTRY = ADDEIN is an ENTRY into OPEIN

CALL ADDEIN(KANW,LKPT,LOEF,KRAF,KSHIFT,IMELD,IMENGE,KXYZ,ISTAT, IOFLOW,IUFLOW,KOFLOW,KUFLOW)

LIST OF ARGUMENTS:

KANW is the absolute address of the record to be read and is updated to be ready for the next call.

LKPT is the number of points still to be read, the value is updated and returned to be ready for the next call.

LOEF is the number of points to be returned by a call (N * 512)

KRAF is the sorting factor

KSHIFT is the index of the word preceding the first address of the XYZ array where the values will be stored.

IMELD =1 causes the first eight values to be printed for control. (No action if IMELD = 0).

IMENGE is the number of points transferred by the call (IMENGE $\leq$ LOEF).

KXYZ must be equal to 1 for the transfer on the whole XYZ array. The values 2,3,4 indicate a fast transfer of 512 values starting at X(1),Y(1), and Z(1).

ISTAT must be zero for the first call of a transfer, the value is updated to 1 if a continuation is expected and to 2 if the request is terminated (LKPT = 0).

IOFLOW,IUFLOW give the number of values ADDEIN has stored in the overlapping zones.

KOFLOW,KUFLOW give the number of values requested for the overlapping features (possible only if KXYZ = 1).

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 3

INDIRECT:

transfer. This is the reason why a second mode of transfer was introduced in ADDEIN. The method uses the 512 first values of the X, Y or Z arrays as input buffer for the READ statement and returns immediately to the calling program. The choice between the three arrays is specified by setting KXYZ equal to 2, 3 or 4 (KXYZ = 1 refers to the first transfer mode). The second type of transfer is obviously more effective and is often used because in the case of a sorting factor greater than one, it is automatically downgraded to the general case.

2.3.3 Transfer from the computing arrays to the warehouse

2.3.3.1 Preparation of the transfer

Like in the previous case, the execution of the transfer depends upon the successful completion of some validity tests. These tests are performed by OPAUS (= OPEN-OUT). The first step is to verify that the name proposed by the user for the new record is not already known to the catalog (KN must be equal to -1). OPAUS verifies also that the addition of the new name does not exceed the size of the catalog (512 names). The catalog is then updated but the warehouse endpointer KEND as well as the number of records KDAT are modified only at the end of the subroutine, once it has been verified that the end pointer of the new record will not exceed the limit of the storage file. The sorting factor KRAF is not used during the transfer from the computing arrays to the warehouse (OPAUS), because no sorting is done during this process.

2.3.3.2 Execution of the transfer

The transfer to the warehouse is simplified by the fact that there is neither sorting operation nor provision for an overlapping of the segments like in the ADDEIN case. The basic mode of ADDAUS (ADD-OUT) transfers the values of the XYZ array starting at the location KSHIFT + 1 into the ZZ buffer. The contents of the buffer are then moved into the warehouse block indicated by the pointer KPOINT. This value must be initially supplied by the calling program and is easily obtained by storing the warehouse end pointer KEND before the

OPAUS call. The value of KPOINT is then updated to be ready for the next access to the following warehouse block. The parameter ISTAT must be passed as Zero for the first call and if the message indicator IMEL is equal to 1, the first transferred values (up to eight if available) are printed for control purposes and the creation of the new experimental record is acknowledged. ISTAT is then updated to 1. The loading and unloading operations on the buffer are continued until the LKPT points have been transferred. If the transfer requires several ADDAUS calls, the value of LKPT must be a multiple of 512 with the exception of the last call.

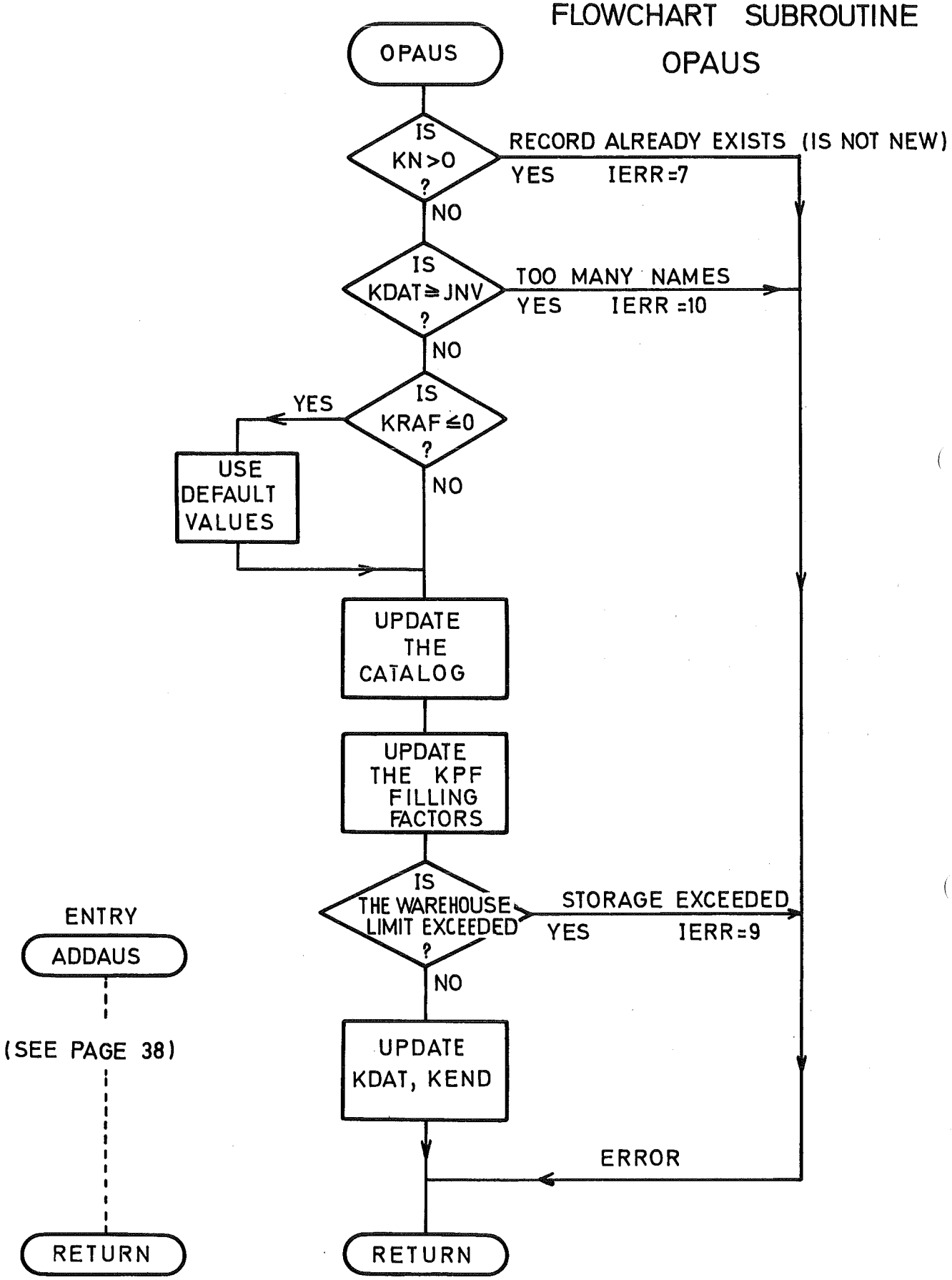
This general transfer mode is performed for the case KXYZ = 1. By setting KXYZ equal to 2,3 or 4 one obtains a faster mode which can be used with the 512 first values of the X, Y or Z arrays like in the ADDEIN case.

2.3.4 Remarks about the use of the transfer subroutines

The transfer subroutines are one of the central features of SEDAP and care must be taken to provide them with the proper arguments. One must be aware that many parameters which were initially passed by the calling program will be updated in such a way that it does not need to track them or to care for their incrementation. For instance, KANW is given as first integer in the command card and can be directly passed to OPEIN which transforms its relative pointer address into an absolute address ready for the ADDEIN call. The same ADDEIN will update the value of the pointer for the next call without any single action from the side of the calling program. This comfortable situation can become a disadvantage if one does not consider the evolution of the arguments when several transfers are parallel or nested. Such a situation arises when the following operation is performed:

A = B + C (A, B, and C are experimental records).

FLOWCHART SUBROUTINE OPAUS



COMMANDS:		
None	OPAUS opens a new experimental record in the warehouse, checks the validity of the request and updates the catalog	NAME = OPAUS SYSTEM = TRANSFER ENTRY = ADDAUS

CALL OPAUS(KPT,RNAM,KN,KRAF,FREQ,DAT,ZEYT)

LIST OF ARGUMENTS:

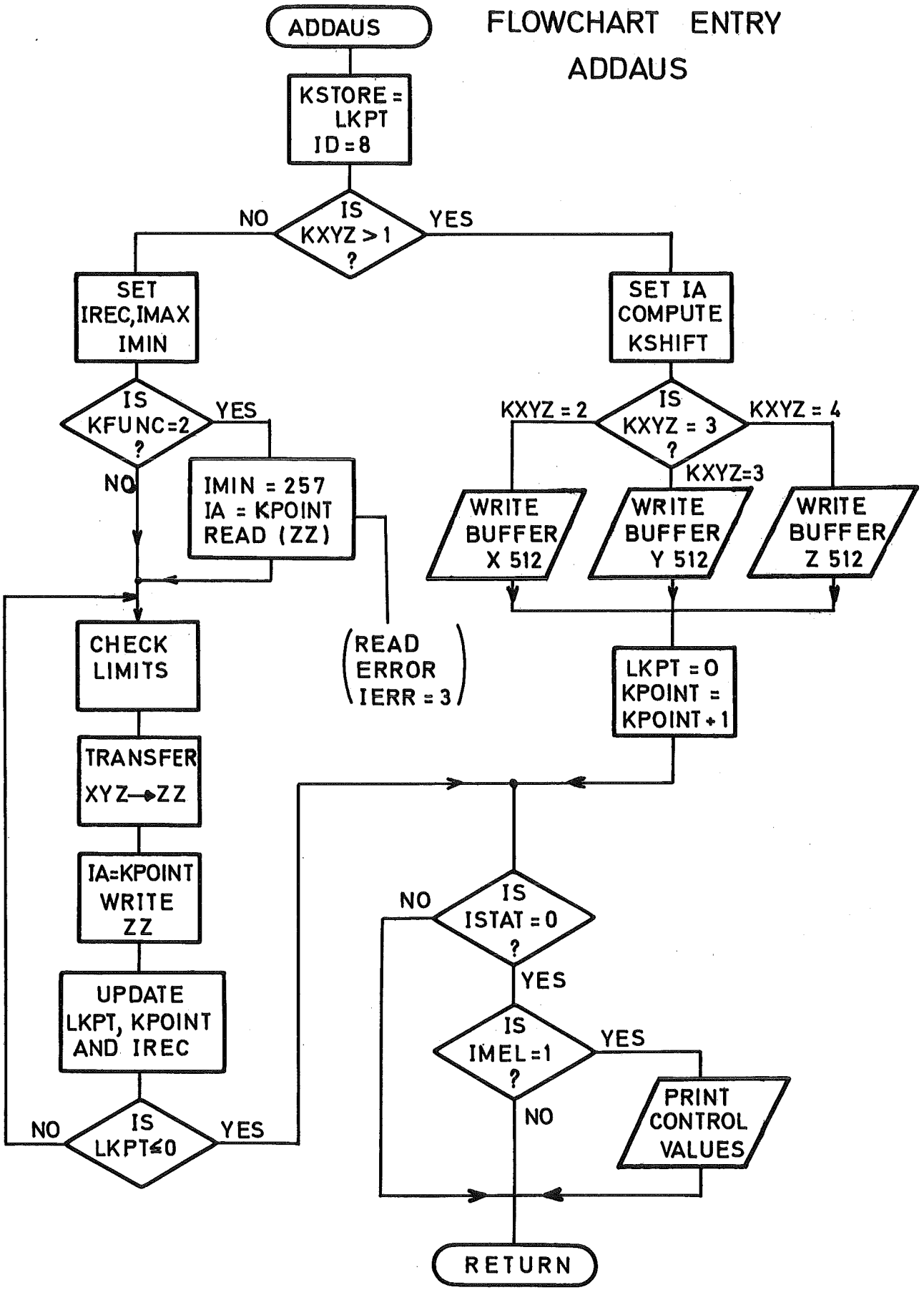
KPT	Number of points to be stored
RNAM	Name of the new experimental record to be stored
KN	Is the search index of RNAM (must be -1)
KRAF	Sorting factor used to obtain a new frequency(not used)
FREQ	Sampling frequency
DAT	Date of the record
ZEYT	Time corresponding to the first value to be stored

SUBROUTINES OR FUNCTIONS NEEDED:

ERRORS	DIRECT:	3,7,9,10
	INDIRECT:	

```
      SUBROUTINE JPAUS (KPT,RNAM,KN,KRAF,FREQ,DAT,ZEYT)
C     ------------------------------------------------
C
      COMMON X (10240),Y (10240),Z (5120),
     1 BENAM(512),NANF(512),NEND(512),WFREQ(512),ADAT(512),BZEIT(512),
     2 KDAT,KEND,NC,NP,IA,JRV,X1,X2,Y1,Y2,IERR,AERR,BERR,JERR,KERR
     3 ,KPF (512)
      DIMENSION XYZ(16384),ZZ(512),X512(512),Y512(512),Z512(512)
      EQUIVALENCE (XYZ(1),X(1),X512(1)),(ZZ(1),Z(4097)),
     1  (Y512(1),Y(1)),(Z512(1),Z(1))
      JNV = 512
C
C
C  CHECK IF THE RECORD IS REALLY NEW
      IF(KN.GT.0) GO TO 91
C     CHECK THE NUMBER OF NAMES LIMIT FOR THE CATALOG
      IF(KDAT.GE.JNV) GO TO 90
C    UPDATE THE PARAMETERS
      WFREQ(KDAT +1) = FREQ
      BENAM(KDAT + 1 ) = RNAM
      NANF(KDAT + 1) = KEND
      ADAT(KDAT + 1) = DAT
      BZEIT(KDAT + 1) = ZEYT
      MALK = KPT/512
C  COMPUTE THE FILLING FACTORS
      KGER = MALK * 512
      KREST = KPT - KGER
      KPROV = KEND + MALK - 1
      NEND(KDAT+1) = KPROV
      KPF (KDAT+1) = 512
      IF(KREST.EQ.0) GO TO 7
      NEND(KDAT+1) =KPROV + 1
      KPF (KDAT+1) = KREST
C  CHECK IF THE LAST RECORD DOES NOT EXCEED THE LIMIT OF THE WAREHOUSE
    7 IF(NEND(KDAT+1) .GT.JRV) GO TO 89
      KDAT = KDAT + 1
      KEND = NEND(KDAT) + 1
      RETURN
C
      ENTRY ADDAUS(KFUNC,ISTAT,KPOINT,RNAM,LKPT,KXYZ,KSHIFT,IMEL)
C
      KSTORE = LKPT
      ID = 8
      IF(KXYZ.NE.1) GO TO 45
   10 IREC = 0
      IMAX = 512
      IMIN = 1
      IF (KFUNC.NE.2) GO TO 14
C    SPECIAL CASE TO ADD VALUES IN THE SECOND HALF BLOCK
      IMIN = 257
      IA = KPOINT
      READ (40'IA,ERR=98) ZZ
   14 IF (LKPT.GT.512) GO TO 15
      IMAX = LKPT + IMIN - 1
   15 J = 0
      DO 20 I=IMIN,IMAX
      J = J + 1
C    FILL THE ZZ BUFFER
   20 ZZ(I) = XYZ (KSHIFT + J + IREC * 512 )
```

```
      IA = KPOINT
      WRITE (40'IA) ZZ
   25 LKPT = LKPT - IMAX + IMIN - 1
      KPOINT = KPOINT + 1
      IREC = IREC + 1
      IF (LKPT) 85,85,14
   45 KSHIFT = 10240 * (KXYZ - 2)
      IA = KPOINT
      GO TO(10,50,60,70),KXYZ
C    THIS IS THE DIRECT TRANSFER (KXYZ = 2,3 OR 4)
   50 WRITE(40'IA) X512
      GO TO 80
   60 WRITE(40'IA) Y512
      GO TO 80
   70 WRITE(40'IA) Z512
C   IF ISTAT = 0 AND IMEL = 1 PRINT THE 8 FIRST VALUES (CONTROL)
   80 LKPT = 0
      KPOINT = KPOINT + 1
   85 IF(ISTAT.NE.0) GO TO 39
      ISTAT = 1
      IF(IMEL.EQ.0) GO TO 39
C    CHANGE VALUE OF ID IF LESS THAN 8 POINTS
      IF (ID.GT.KSTORE ) ID = KSTORE
      IS = KSHIFT + 1
      IE = KSHIFT + ID
      WRITE(NP,101)(XYZ(I),I=IS,IE)
      WRITE(NP,102)RNAM
   39 RETURN
   89 IERR = 9
      AERR = RNAM
      JERR = KEND
      KERR = JRV
      RETURN
   90 IERR = 10
      JERR = KDAT
      KERR = JNV
      RETURN
   91 IERR = 7
      AERR = RNAM
      RETURN
   98 IERR = 3
      JERR = 40
      KERR = KPOINT
      RETURN
  101 FORMAT(' KONTROLLWERTE OUTPUT = ',8(E12.6,1X))
  102 FORMAT(/,' DIE WERTE SIND UNTER DEN NAMEN ',A4,' ADDRESSIERBAR')
      END
```


FLOWCHART ENTRY
ADDAUS
ADDAUS
KSTORE = LKPT
ID = 8
IS KXYZ > 1 ?
NO
YES
SET IREC, IMAX IMIN
IS KFUNC=2 ?
YES
NO
IMIN = 257
IA = KPOINT
READ (ZZ)
READ ERROR IERR = 3
CHECK LIMITS
TRANSFER XYZ→ZZ
IA=KPOINT
WRITE ZZ
UPDATE LKPT, KPOINT AND IREC
IS LKPT≤0
NO
YES
SET IA COMPUTE KSHIFT
IS KXYZ = 3 ?
KXYZ = 2
KXYZ=3
KXYZ = 4
WRITE BUFFER X 512
WRITE BUFFER Y 512
WRITE BUFFER Z 512
LKPT = 0
KPOINT = KPOINT + 1
IS ISTAT = 0 ?
NO
YES
IS IMEL = 1 ?
YES
NO
PRINT CONTROL VALUES
RETURN

COMMANDS: None	ADDAUS stores the numerical values into the warehouse after the record has been initialized and opened by OPAUS	NAME = ADDAUS SYSTEM = TRANSFER ENTRY = NAME in ADDAUS

CALL ADDAUS(KFUNC,ISTAT,KPOINT,RNAM,LKPT,KXYZ,KSHIFT,IMEL)

LIST OF ARGUMENTS:

KFUNC is equal to 2 if 256 values have to be stored in the second half-record, otherwise KFUNC = 1

ISTAT is given as zero for the first transfer and will be returned as 1

KPOINT is the value of the pointer which indicates the block where the values are stored. KPOINT is updated to be ready for the next call.

RNAM is the name of the record (used for documentation of errors)

LKPT is the number of points which have to be transferred

KXYZ = 1 transfer from the XYZ array (starting at KSHIFT + 1)
= 2,3,4 for a transfer of the first 512 values of the X,Y and Z arrays.

IMEL = 1 if the first output values (up to 8) are to be printed for control. In that case the storage is acknowledged. Otherwise IMEL = 0.

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 3

INDIRECT: None

The transfer scheme will be:

```
OPEIN (B)        (simplified writing form)
OPEIN (C)
OPAUS (A)

ADDEIN (B)
ADDEIN (C)       Iteration
.. Compute
ADDAUS (A)
```

and involve KANW, KENW, LKPT, ISTAT etc. ... in two separate OPEIN / ADDEIN structures. It is then advisable to initialize a double list of arguments like KAN1/KAN2, KEN1/KEN2, LKP1/LKP2, ISTA1/ISTA2 etc. ... which will be able to maintain their own independant evolution.

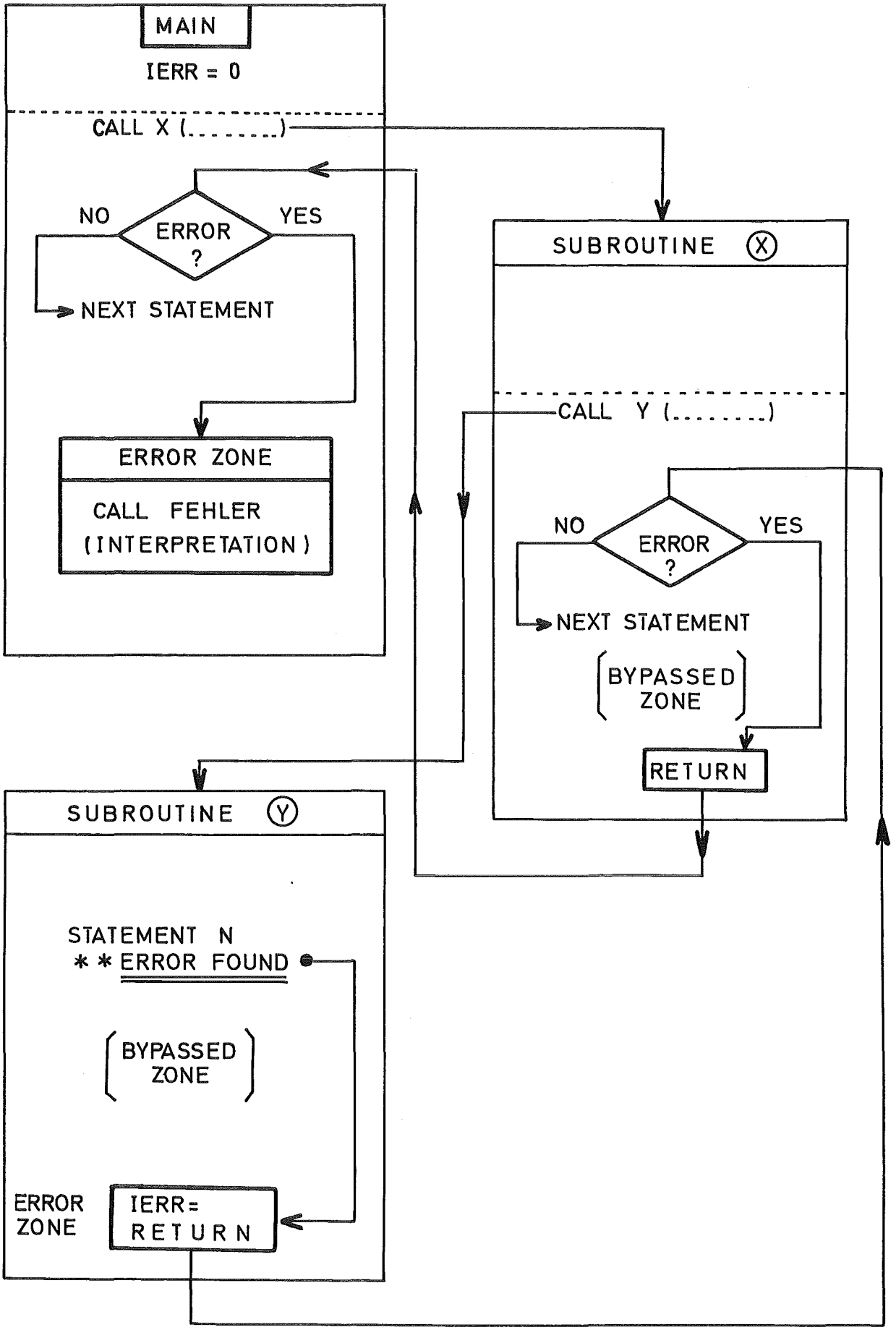
Although a strong similarity exists between OPEIN/ADDEIN and OPAUS/ADDAUS, their symmetrical structure could be misleading if the following points are disregarded.

There is an implicit master-slave relationship between ADDEIN and ADDAUS. ADDEIN is responsible for the input requests and provides the information concerning the end of the transfer. The ADDAUS call derives from ADDEIN or from the supervision of the calling program. This does not preclude the fact that in some situations there is an OPAUS without OPEIN and reciprocally. The difference should be noticed for two similar arguments like ISTAT and LKPT. ISTAT has three status values in ADDEIN but only two in ADDAUS. LKPT in ADDEIN refers to the number of points still to be read while LKPT in ADDAUS is the number of points which have to be transferred by the call, it corresponds to the parameter IMENGE of ADDEIN.

It can be noted that the end of an input request can be detected after the ADDEIN call by testing for LKPT = 0 or for ISTAT = 2, whichever is the most convenient.

2.4 Error Interpretation

Any error occuring during the execution of a program represents a very uncomfortable situation. The situation is even worse, if the error occurs in a large system of the size of



ERROR HANDLING WITH R E T U R N SCHEME

COMMANDS:	FEHLER is the error subroutine which is called at the end of a task if IERR ≠ 0. FEHLER processes twenty different error types and expects that the information necessary to the error interpretation has been orderly passed to the common	NAME = FEHLER SYSTEM = ERROR ENTRY = none
None		

CALL FEHLER

LIST OF ARGUMENTS: FEHLER has no argument but uses the five following parameters located in the common area according to the following conventions:

IERR is the error code
IERR = 0 if no error has been detected
IERR = 1 to 21 refers to one of the 21 error types

JERR and KERR are two integers which are used to pass the number of a unit, the address of a false block, the value of a wrong delimiter etc. ...

AERR and BERR are two decimal values used to pass an incorrect frequency value or which carry a record name corresponding to a FORMAT A4.

Note: Before issuing any IERR code it is necessary, in order to obtain a correct error interpretation, to update some of the four listed parameters according to the error table.

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: Does not apply

INDIRECT: Does not apply

SEDAP which was designed to be a problem oriented process, where the user should not be concerned with all the elementary steps of the computation. Most of the SEDAP errors can be classified according to the following types:

- the primitive errors. They are mainly due to punching errors. A user can punch SO32 instead SØ32 or TEMD when he means TEMP.
- the logical errors. They are mainly due to a lack of processing scheme or to an insufficient knowledge of the command specifications.
- a third class of errors is more difficult to detect and involves a type of errors inherent to the nature of any computing activity. Typical examples are the hardware errors (machine error, I/O parity error, the destruction of a card, the absence of a tape reel which was not delivered to the machine room etc. ..). These errors are generally known to the supervising system or to the operator but the related information is often extremely difficult to obtain at the FORTRAN level.

During the implementation of SEDAP it has been attempted to detect the largest possible number of errors and to stop the execution of the job before the consequences of an error become unpredictable. This is materialized by numerous tests located at critical points of the program. If the course of the program is not endangered, a warning will be issued but generally the error causes an immediate return to the error zone of the main program which terminates the job once the error has been interpreted. It has been our experience that a clearly described error will be corrected in one run while an exagerated indulgence can lead to a chain of errors which cannot be identified by the system's user. The error detection was implemented almost to the limit of the FORTRAN possibilities but the challenge cannot always be met and it is not claimed that all the errors will be detected.

A large number of errors of third class are out of reach and will cause an interruption which will be documented only by

the OS, i. e. without reference to our problem oriented application. This is the case for the machine errors, break-down etc....

When SEDAP addresses a new file (tape or direct access) the user will be informed of the operation by a special message like:

> "the DUMP command must now use the file 21 for the execution of the task, this requires the availability of a compatible tape and the correct specification of a corresponding control card
> //FT21F001 ....."

and may deduce that any interruption immediately following the message has been caused by one of the above mentioned points. Some errors like the register under/over-flow are not detected by SEDAP because they are caused by too many reasons and because their detection at the FORTRAN level would have to be paid by a too large increase in memory size and execution time. Most of critical divisions are protected against zero-divide.

The error status of SEDAP is represented by the integer IERR which is set to zero during the initialization or after an error interpretation in the case where a restart is allowed. Any detected error causes IERR to take a value greater than zero. When an error code is issued, the value of IERR must correspond to the type of error to be detected. Four other parameters JERR, KERR, AERR and BERR contain the information which must be supplied to the error interpreter according to the conventions listed in the errortable. JERR and KERR supply the information about integer values (file number, block numer etc. ...) while AERR and BERR are used to pass the record names or a decimal parameter. The five parameters of the error interpretation are located in the common area. Once an error code has been issued in a subroutine, the control must be immediately passed to the calling program (RETURN). This implies that after calling any of the subroutines which can issue an error code, the zero value of IERR

ERROR CODE LIST

IERR	Description	FORTRAN Reference	Remarks	AERR	BERR	JERR	KERR
1	Tape reading error	READ(KTAPE,IERR=...)	DD card, parity error	/	/	UNIT	BLOCK
2	End of file on tape	READ(KTAPE,END=...)	Too many blocks requested	/	/	UNIT	BLOCK
3	Direct access reading error	READ(40'IA,ERR=...)	DD card, damaged disk etc..	/	/	FILE	POINTER
4	Exp. record overflow	IA > NEND(KN)	Logical error(see Handbook)	NAME	/	KDIF	KENW
5	First delimiter > second del.	INT(1) > INT(2)	Reversed delimiters	/	/	KANW	KENW
6	Too many values	LKPT > MAX	A limit was set for the task	/	/	LKPT	MAX
7	New record name is not new	K3 > 1	Logical error(see Handbook)	NAME	/	/	/
8	Old record name is unknown	K1 < 1 or K2 < 1	Logical error(see Handbook)	NAME	/	/	/
9	Warehouse is full	NEND(N) > JRV	Use destroy or larger spec.	/	/	KEND	JRV
10	Catalog is full	KDAT > JNV	Only 512 names permitted	/	/	KDAT	JNV
11	1 of the 4 first cards is false	BEF(1).NE.'SEDA'	False initialization	HEAD	'SEDA'	/	/
12	Command is invalid	BEF.NE.BE(1...KBE)	Check commands list	BEF	/	/	/
13	Modifier is invalid	K4 < 1	See command description	NAM2	/	/	/
14	Less than N values	KPT < N	" " "	/	/	KPT	N
15	Frequency is ≤ 0.0	FREQ.LE.0.0	" " "	FREQ	/	/	/
16	First delimiter is ≤ 0	KANW ≤ 0	Logical error, punching err.	/	/	KANW	/
17	Warehouse specification exceeded	INT(1) > 5000	See Handbook	/	/	INT(1)	/
18	Binary conversion err.	(ERAKON)	Recording error (hardware)	/	/	UNIT	BLOCK
19	Warehouse is empty	KDAT=0	Logical error	/	/	/	/
20	Sorting factor 100	KRAF > 100	Logical error	/	/	/	/
21	Non standard error	(Listed errors)	See Handbook	/	/	CODE	/

must be checked. The operation provides a fast cascaded return to the main program where an error zone calls the subroutine FEHLER for the interpretation of the error. Twenty types of standard errors are interpreted by the system. The error code IERR = 21 is reserved for the non-standard errors and gives a reference number listed in the user's handbook.

2.5 Service subroutines

2.5.1 Command file transfer (Subroutine DAKA)

During the execution of a task, a copy of the original command card is provided for documentation purposes before the interpretation of the task is formulated. Since the main program already prints the complete list of the commands at the beginning of the job, a re-read operation must be provided. This function is performed by the subroutine DAKA which reads and prints all the input cards at the beginning of the job and transfers them to a new input file. During this operation, the comment cards which must begin with an arrow (symbol > i. e. greater than) are printed but are not transferred. The file 15 (blocksize 1680, logical record length 80) is used for this intermediary storage. Since the cards were read on the standard input file (file 5), the subroutine DAKA changes the value of the index NC from 5 to 15 to insure that all the subsequent READ will be made by addressing the new file.

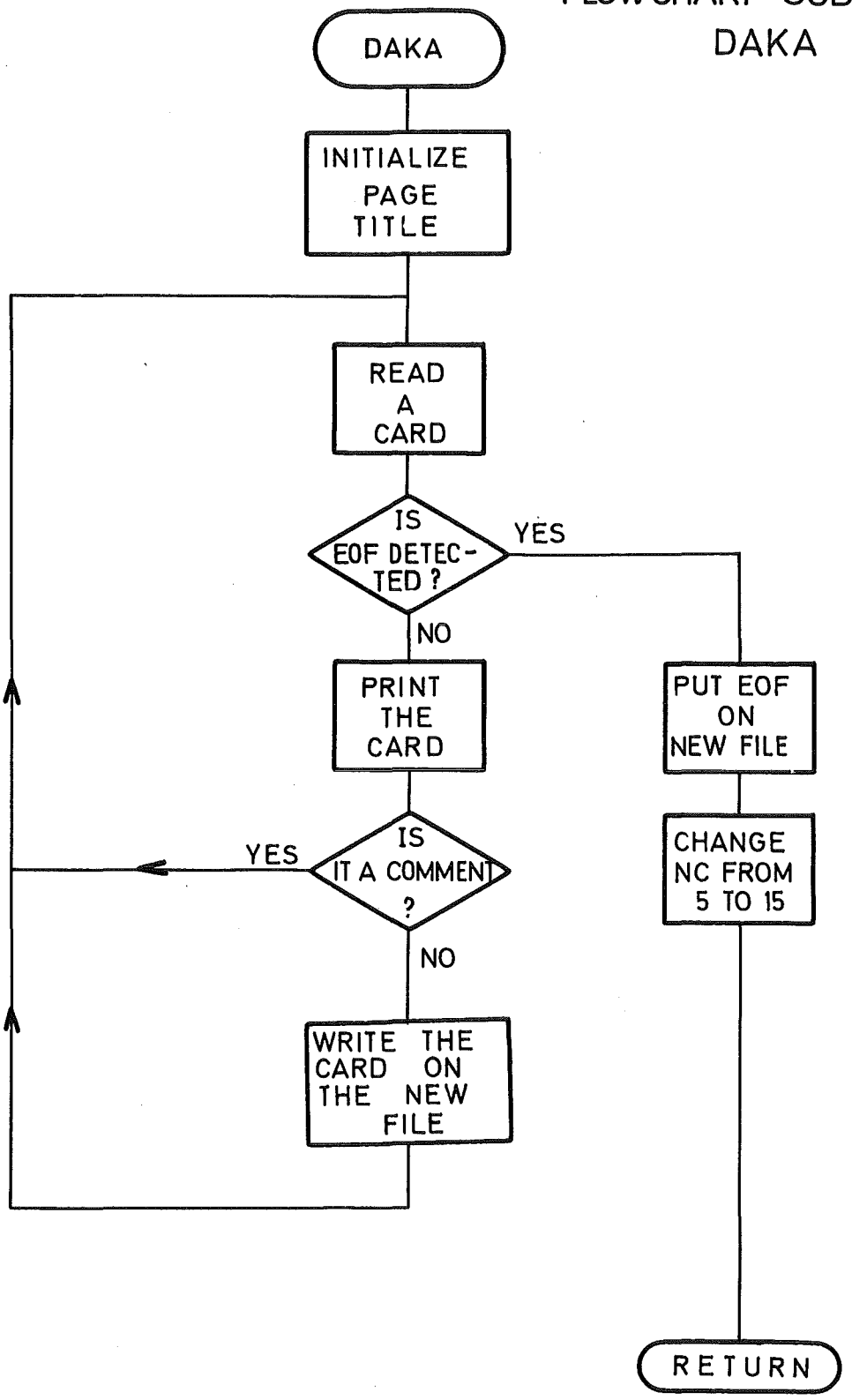
2.5.2 Status of the warehouse and the command list (Subroutine STATUT)

The subroutine STATUT maps the warehouse and gives the list of all the commands (keywords) which are acknowledged by the system. The subroutine can perform three types of tasks which can be classified according to the value of the variable KFUNC:

- for KFUNC = 1 the experimental records names are listed with all the related parameters.
- for KFUNC = 2 the previous case is extended to the block level and the eight first values of every block are listed.
- for KFUNC = 3 the keywords used as command names are listed together with the eight character titles which are used to report the initialization of a task.

The subroutine is straightforward and is mainly comprised of three DO loops, two of which are bypassed when the warehouse is empty.

FLOWCHART SUBROUTINE
DAKA



COMMANDS:	DAKA prints the list of all the command cards at the beginning of a job and provides a re-read possibility by transferring the commands to a new file. During this operation, DAKA skips the comment cards.	NAME = DAKA SYSTEM = Service ENTRY = None
None		

CALL DAKA(NC,NP,NN)

LIST OF ARGUMENTS:

NC is the index of the standard input file //G.SYSIN and is equal to five in the described configuration.

NN is the index of the intermediary file. In the described configuration, NN is equal to 15 and refers to the file allocated under //FT15F001 .....

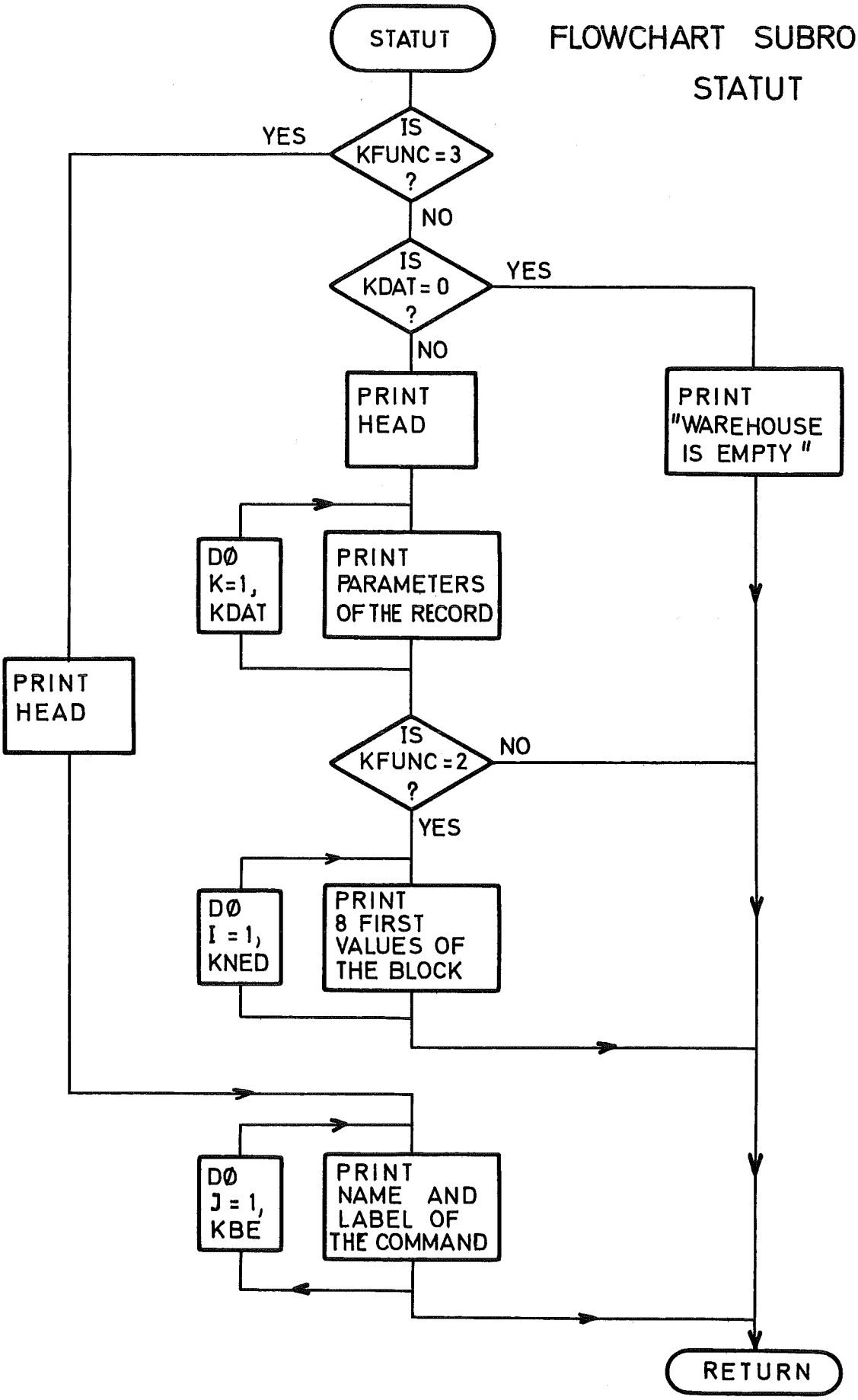
NP is the index of the standard output file or SYSPRINT file and is equal to 6 in the described configuration.

N.B. The value of NC to be returned by DAKA is the value passed for NN

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS	DIRECT: None
	INDIRECT: None

FLOWCHART SUBROUTINE STATUT



COMMANDS: BILD,ZUST	STATUT lists the records stored in the warehouse with their parameters and as option prints the eight first values of all the blocks contained in the warehouse. STATUT also gives a list of all the commands of the system	NAME = STATUT SYSTEM = Service ENTRY = None

CALL STATUT(KFUNC,KBE,BE,ME)

LIST OF ARGUMENTS:

KFUNC indicates the selected option with the following key:
KFUNC = 1 for the list of the records
KFUNC = 2 for the same list as KFUNC = 1 but with the addition of the eight first values of every block.
KFUNC = 3 for the system commands list with their label.

KBE is the number of implemented commands

BE is the array which contains the commands (4 characters)

ME is the array which contains the labels (8 characters)

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 3

INDIRECT: None

2.5.3 Destruction of records (Subroutine LAGER)

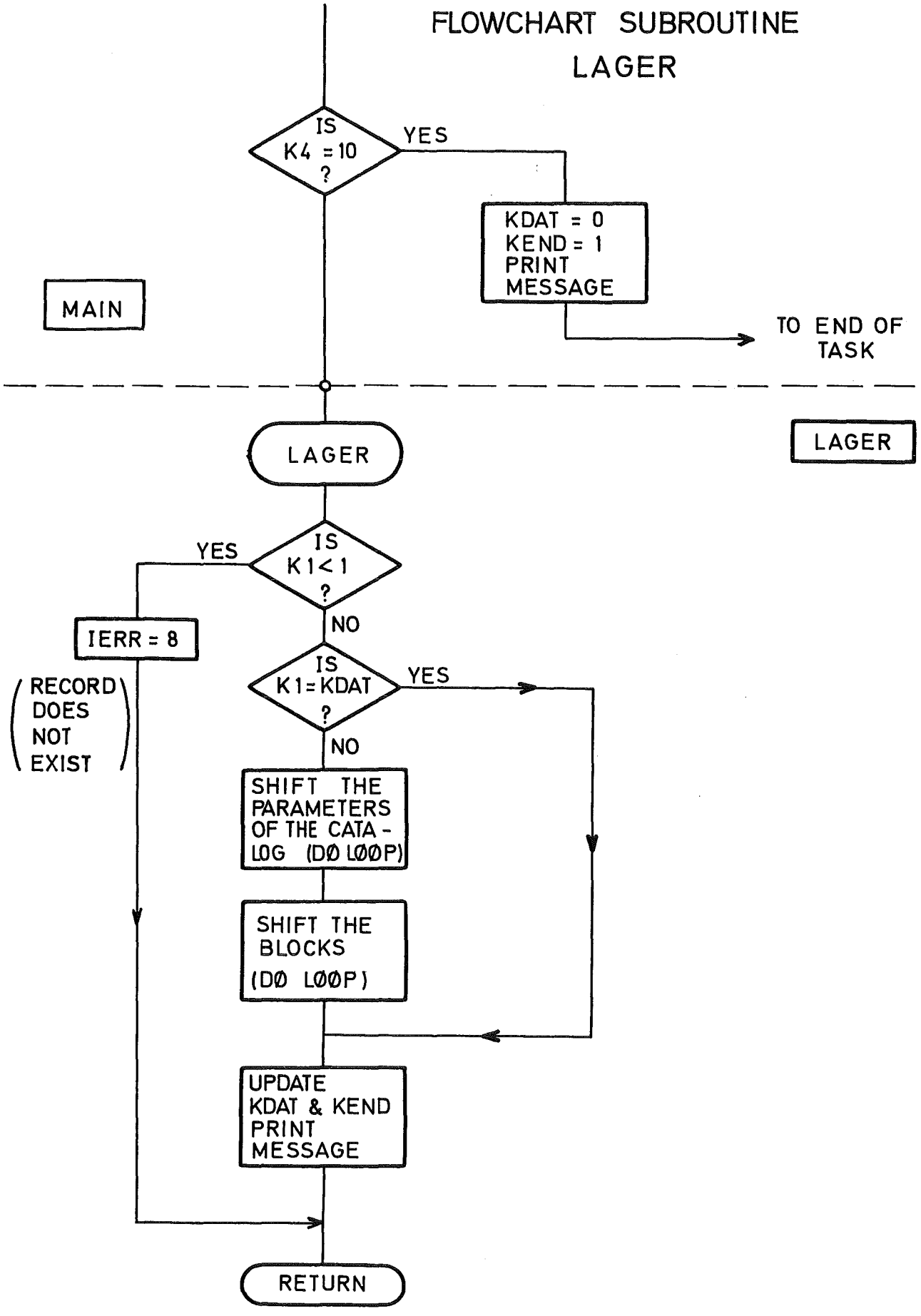
The subroutine LAGER destroys an experimental record from the warehouse. Such an operation can be necessary if a large experimental record is no longer needed (for example a multiplexed record is not used once it has been sorted) and if the warehouse free space has become insufficient. A special case is involved if the user intends to clear all the warehouse (ZERS ALLE for destroy all). Such a situation is looked upon in the MAIN and represents a simplified case directly handled by the MAIN by setting KDAT equal to zero and KEND equal to one. The destruction of a single record is performed by the subroutine LAGER which first checks the existence of the record and reorganizes the warehouse to erase the specified record. Such an operation is done by shifting all the parameters which follow the destroyed record to the preceding position and by shifting all the blocks which follow the last block of the destroyed record by an amount equal to the number of blocks occupied by this record.

It should be noted that another special situation arises if the record to be destroyed is the last record stored in the warehouse. A simplified treatment is applied to update KDAT and KEND without a shifting operation.

ENTRY CTLG

LAGER has a secondary function which is accessible by an ENTRY called CTLG. The purpose of CTLG is to systematically search the catalog to find whether a proposed name XNAM matches one of the existing KDAT record names contained in the warehouse. If the search has been successful, the search index KN will carry the index K of the record in the KDAT list. If the name is unknown to the catalog or if the warehouse is empty, KN is set to -1. SEDAP subsequently uses the index KN to address the record, to test his existence or to check the newness of new record names.

FLOWCHART SUBROUTINE LAGER



COMMANDS: ZERS=(destroy) MODIFIER=ALLE	LAGER destroys a record in the warehouse. If all the warehouse must be cleared, the MAIN does the job in a simpler way by setting KDAT = 0 and KEND = 1 CTLG is a LAGER-entry which searches a record-name in the catalog	NAME = LAGER SYSTEM = SERVICE ENTRY = CTLG

CALL LAGER(K1,RNAM)

LIST OF ARGUMENTS:

K1 is the catalog index of the record RNAM (if K1 = -1, the name has not been found)

RNAM is the name of the record to be destroyed

CALL CTLG(KN,XNAM)

KN is the resulting index after the search (KN = -1 if not found, KN = K if XNAM matches the Kth name)

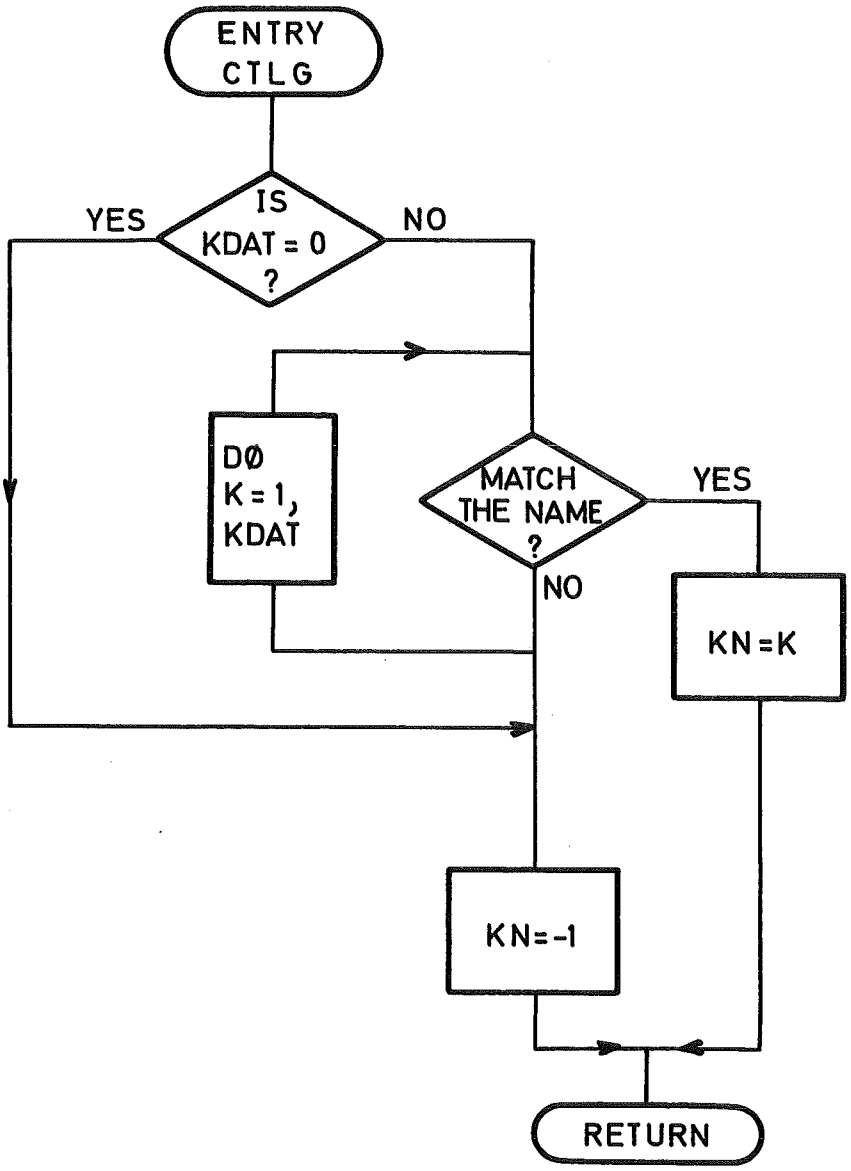
XNAM is the name which will be searched in the catalog

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 3 + 8

INDIRECT: None



ENTRY CTLG IN LAGER
(SYSTEMATIC CATALOG SEARCH)

2.5.4 Generation of simulated Data (Subroutine DAGEN)

SEDAP is a program system which was mainly designed for the treatment of experimental data but its range of application is extended by the system capability to generate his own data. This function is performed by the subroutine DAGEN.

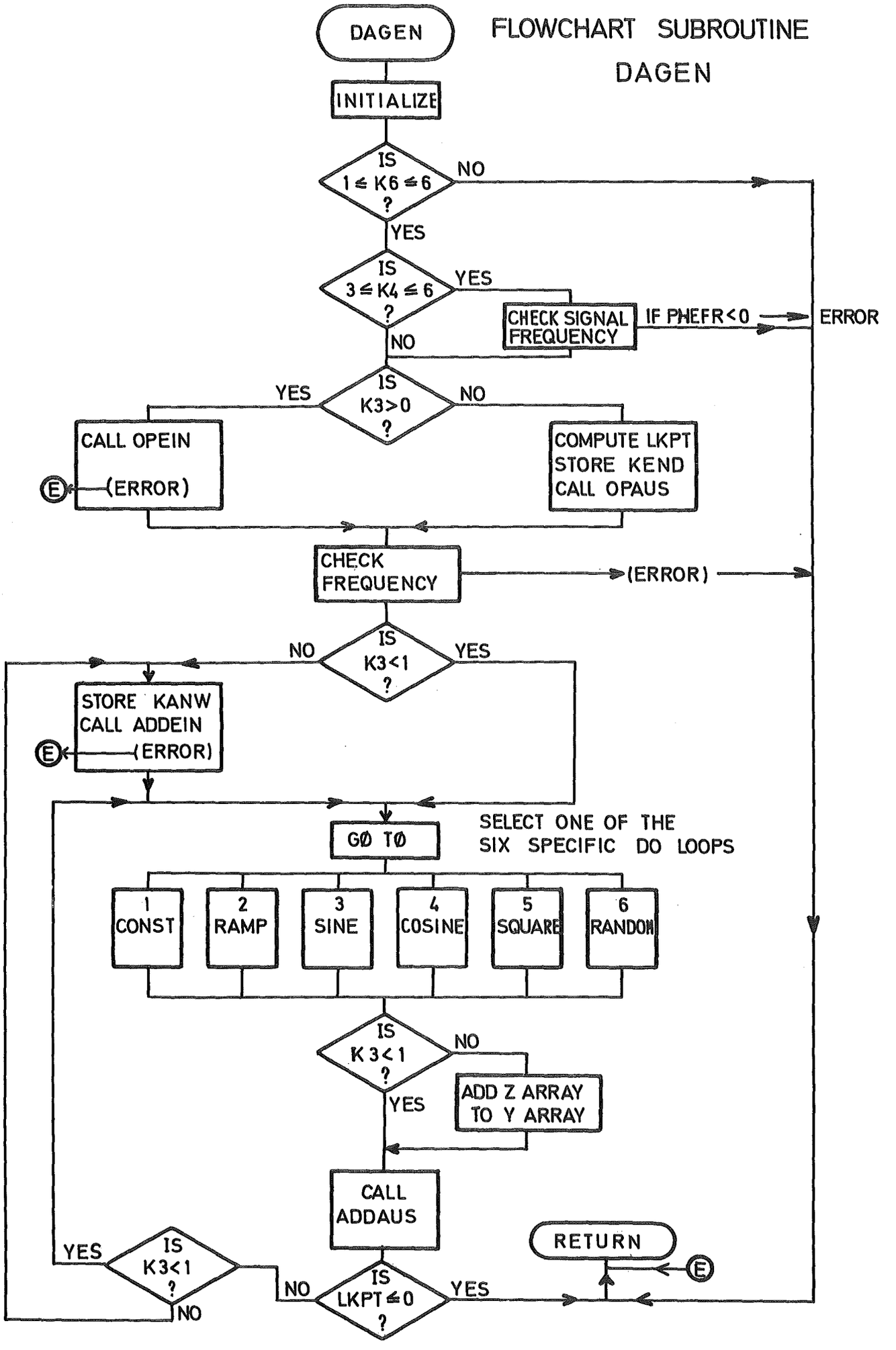
Two main reasons justify the existence of the data generation:

- The experimenter generally likes to test the SEDAP package in a "dry run" mode in order to gain some experience with the techniques of data reduction. The subroutine DAGEN allows the system's programmer and the system's user to produce data which are extremely convenient to test the system or to learn how it reacts.
- An advanced type of data reduction may call for some complex form of compensation which can be achieved by DAGEN.

The DAGEN subroutine initializes the parameters and checks the validity of the modifier (index K4) which is used to determine the type of generated data. The status of the proposed record name is then investigated. If the name is new (K3 = -1), a new record is opened by OPAUS but if the name already exists, DAGEN concludes that the user intends to add the generated data to an old record to obtain a compensation or to perform some type of complex waveform synthesis. In the last case OPEIN checks if the request is compatible with the record stored in the warehouse and uses the returned frequency as sampling frequency for the data to be generated. ADDEIN transfers immediately the first segment of 512 values to the Z array (KXYZ = 4).

In both cases the control is then passed to one of the six computing zones which generate the following type of data in the Y array:

1- constant of amplitude a
2- ramp (aX + b) with a and b as parameters
3- sine wave of amplitude a and frequency f_p
4- cosine wave of amplitude a and frequency f_p ,
5- square wave of amplitude (+a, -a) with a repetition rate f_p
6- random numbers comprised between 0 and +a.


DAGEN
FLOWCHART SUBROUTINE
DAGEN
INITIALIZE
IS 1 ≤ K6 ≤ 6 ?
NO
YES
IS 3 ≤ K4 ≤ 6 ?
YES
CHECK SIGNAL FREQUENCY
IF PHEFR<0
ERROR
NO
IS K3>0 ?
YES
NO
CALL OPEIN
(ERROR)
COMPUTE LKPT
STORE KEND
CALL OPAUS
CHECK FREQUENCY
(ERROR)
IS K3<1 ?
NO
YES
STORE KANW
CALL ADDEIN
(ERROR)
GØ TØ
SELECT ONE OF THE SIX SPECIFIC DO LOOPS
1 CONST
2 RAMP
3 SINE
4 COSINE
5 SQUARE
6 RANDOM
IS K3<1 ?
NO
YES
ADD Z ARRAY TO Y ARRAY
CALL ADDAUS
RETURN
IS K3<1 ?
YES
NO
IS LKPT ≤ 0 ?
NO
YES

COMMANDS:	DAGEN generates test data and stores the generated data in the warehouse. Six types of signals are possible (constant, ramp,sine,cosine,square and random), DAGEN can add the generated data on an existing record (Additive Process)	NAME = DAGEN SYSTEM = SERVICE ENTRY = None
DAGE		

CALL DAGEN(K4,FNAM,GNAM,K3,KANW,KENW,KBDIF,FREQ,PHEFR,AMP)

LIST OF ARGUMENTS:

K4 is the modifier index which indicates the desired type of signal

FNAM is the record name

GNAM is the modifier name

K3 is negative if the record is new, otherwise the output will be added to the record pointed by K3

KANW,KENW are the delimiters of the selected segment (in blocks)

KBDIF is the number of blocks to be generated

FREQ is the simulated sampling frequency (Hz)

PHEFR is the frequency of the generated signal (sine,cosine, square,wawe) or the amplitude of the increment (ramp).

AMP is the amplitude of the generated signal

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,ADDEIN,OPAUS,ADDAUS,SIN,COS,RANDU

ERRORS

DIRECT: 13 15

INDIRECT: OPEIN,OPAUS,ADDEIN

All the signals are generated with a sampling frequency f_s specified by the user or provided by the OPEIN call.

If the additive process has been selected, the segment of the existing record which has been stored in the Z array is added to the newly generated data of the Y array. In both cases the Y array is transferred to the warehouse with a pointer KSPUR which will direct the new data to the new record or to the old one (replacement of the block) according to the status of K3.

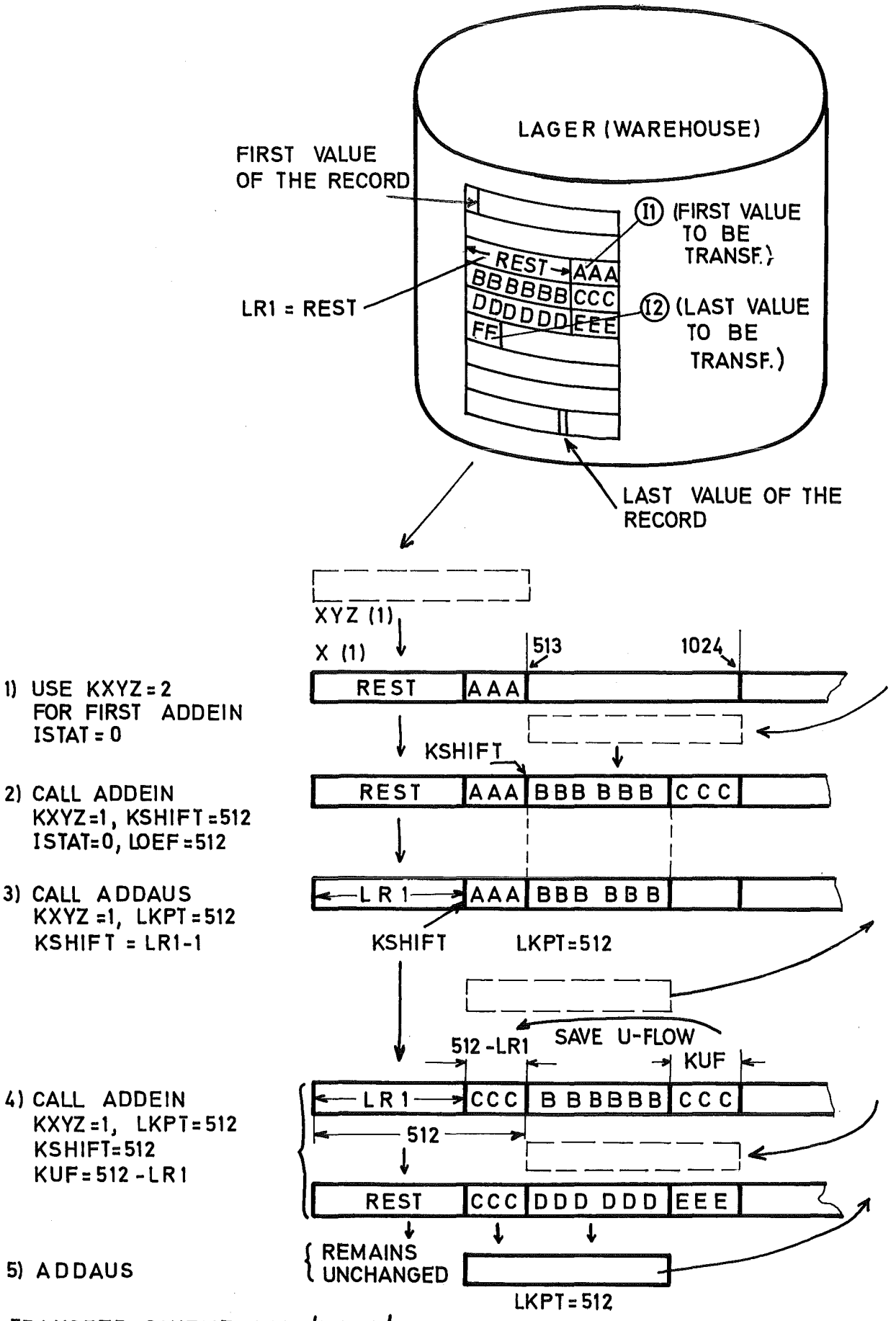
The data generation is terminated and the control is returned to the MAIN if the transfer request has been satisfied. Otherwise the process continues and the control will be passed to the next ADDEIN call or directly to the computing zone if the record is new.

Remark: The generation of uniformly distributed random real numbers (Type 6) requires the availability of the subroutine RANDU (IBM scientific subroutine package) which is specific to the system 360/370 /3/.

2.5.5 Record delimiting by values or time units (Subroutine WERT)

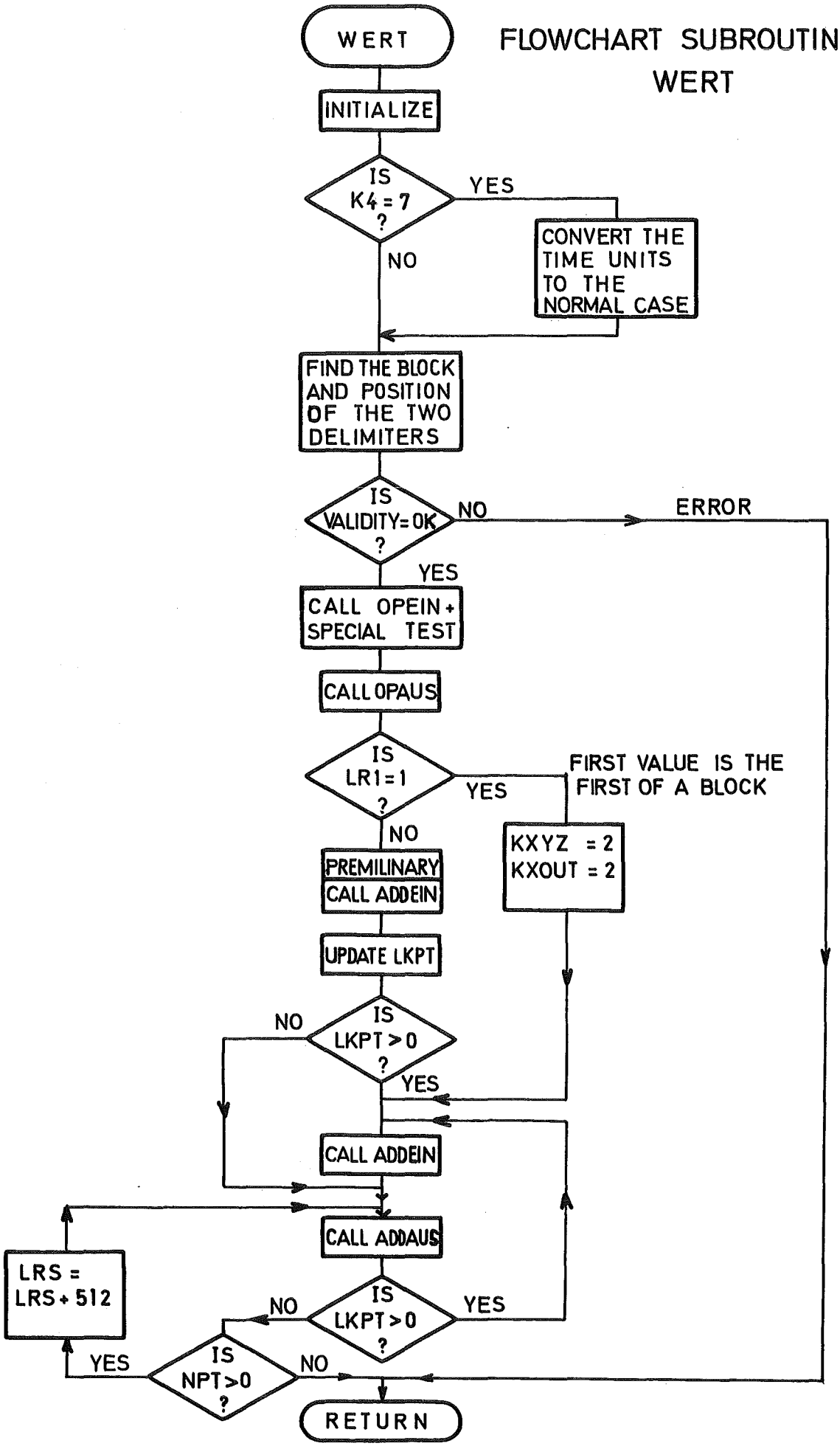
SEDAP handles the values by blocks, which means that the delimiters carried by the commands cannot retail the recorded values in quantities smaller than 512 (with the exception of the last block of a record which may not be completely filled). It has been initially planned to specify the delimiters in blocks, values or time units. Only the block option was implemented but the service subroutine WERT allows to transfer a part of record delimited in values or in time units into a new record.

The subroutine WERT first initializes the service parameters and verifies if the index K4 is not equal to 7, which would indicate that the delimiters were given in time units (seconds). This special case is first investigated and the time delimiters are converted to seconds in accordance with the time floating factor (FAK), which can be 0.001 for instance if the user has



TRANSFER SCHEME FOR 'WERT'
(CALL BY VALUES)

FLOWCHART SUBROUTINE WERT



COMMANDS:	WERT creates a new record by transferring a record segment whose delimiters have been specified, not in blocks as usual, but in points or in time units.	NAME = WERT SYSTEM = SERVICE ENTRY = None
WERT		

CALL WERT(ENAM,K1,GNAM,K3,I1,I2,K4,TA,TE,FAK)

LIST OF ARGUMENTS:

ENAM	is the name of the input record
K1	is the search index of ENAM (not found if K1 = -1)
GNAM	is the name of the new resulting record
K3	is the search index of GNAM (valid if K3 = -1)
I1,I2	are the two values (both inclusive) which delimit the selected segment
K4	is the modifier index. If K4 = 7, the delimiters are given in time units by TA and TE
TA,TE	are the two time delimiters normally given in seconds.
FAK	is a floating factor which will be applied to TA and TE before they are computed in seconds (FAK = 0.001 if the delimiters are given in msec).

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,OPAUS,ADDEIN,ADDAUS

ERRORS

DIRECT: 4,5,6

INDIRECT: see OPEIN,OPAUS,ADDEIN

used the option to specify the time units in milliseconds. An OPEIN call is used to obtain the time and the frequency of the record. A computation to transform the time delimiters into points is then possible, thus reducing the time option to the general case here after described.

The validity of the delimiters expressed in points is first verified and the position of these two points is investigated in order to obtain the address of the block where they are located and their position within that block. OPEIN is called to open the record and to verify the validity of the request. Since OPEIN was designed to handle the blocks, two complementary tests are necessary to insure a correct transfer under all conditions:

- The time origin must be shifted if the first value is not the first value of the block.
- The filling factor of the block which contains the last delimiter must not be exceeded by the position of this value.

OPAUS can then open the resulting new record in the warehouse. Since a special case is involved when the first delimiter value is the first value of a block (LRS = 0) the existence of this possible simplification is checked and causes the selection of a fast transfer mode for ADDEIN and ADDAUS with no underflow and both transfers are performed on the first 512 values of the X array.

The general case must provide a preliminary underflow zone (see chapter 2.3.2.2 and page 28) which cannot be obtained from the first ADDEIN call. This is done by a preparatory ADDEIN call with KXYZ = 2 which brings the first block into the first 512 storage locations of the XYZ array. If the transfer involves only one block, the input transfer is completed with the preliminary ADDEIN call and the control is shifted to the ADDAUS call. In the general case, the second ADDEIN call stores the next block into the 512 storage locations adjacent to the previous 512 stored values. ADDAUS can then transfer a complete block of 512 values by using the normal transfer mode KXYZ = 1 with a displacement KSHIFT

which is equal to LRS, i. e. to the position of the first value in the related block. All the subsequent ADDEIN calls will renew the initial zone by using the underflow feature which can cover the 511 possibilities.

It is important to note that in order to terminate the transfer operations two conditions must be met:
- the input request (number of points) must be satisfied.
- all the points must have been transferred by ADDAUS.

If the first condition is not met the process continues with the next ADDEIN but if only the second condition is not met the control must be passed to a last ADDAUS call which will be executed with a new displacement equal to the former KSHIFT incremented by 512.

Remark: Since the subroutine WERT takes advantage of almost all the possible features of the TRANSFER subsystem, some of them in tricky ways, the understanding of the individual operations of the WERT subroutines requires a detailed knowledge of the TRANSFER subsystem.

2.6 The input-output subsystem

SEDAP processes data which have been recorded on magnetic tape or paper tape during an experiment and communicates the results of the process to the user by directing the records or parts of the records to output files such as the printer file or the plotter file. All these files form an environment which will be shortly described.

1) The command file
   This is the standard card input file which contains the commands to direct the process. A second file is needed to transfer the list (file 15). These files are handled by the main program and DAKA and are described in relation with the specific parts of the system.
2) The magnetic tapes
   - standard magnetic tapes (usually 9 track - 800 bpi) are used to dump the records or to provide an interface to other programs. Sequential data sets on direct access

devices may be used for the same purpose.
- 7 track tapes in a special format are used to obtain the data from the data acquisition system (see ERAKON).

3) The printer file
   This is the standard output file of the computing system.
4) The plot file
   The plot file is installation dependent and is used by the subroutine GRAPH to produce plot output via offline Calcomp plotters.

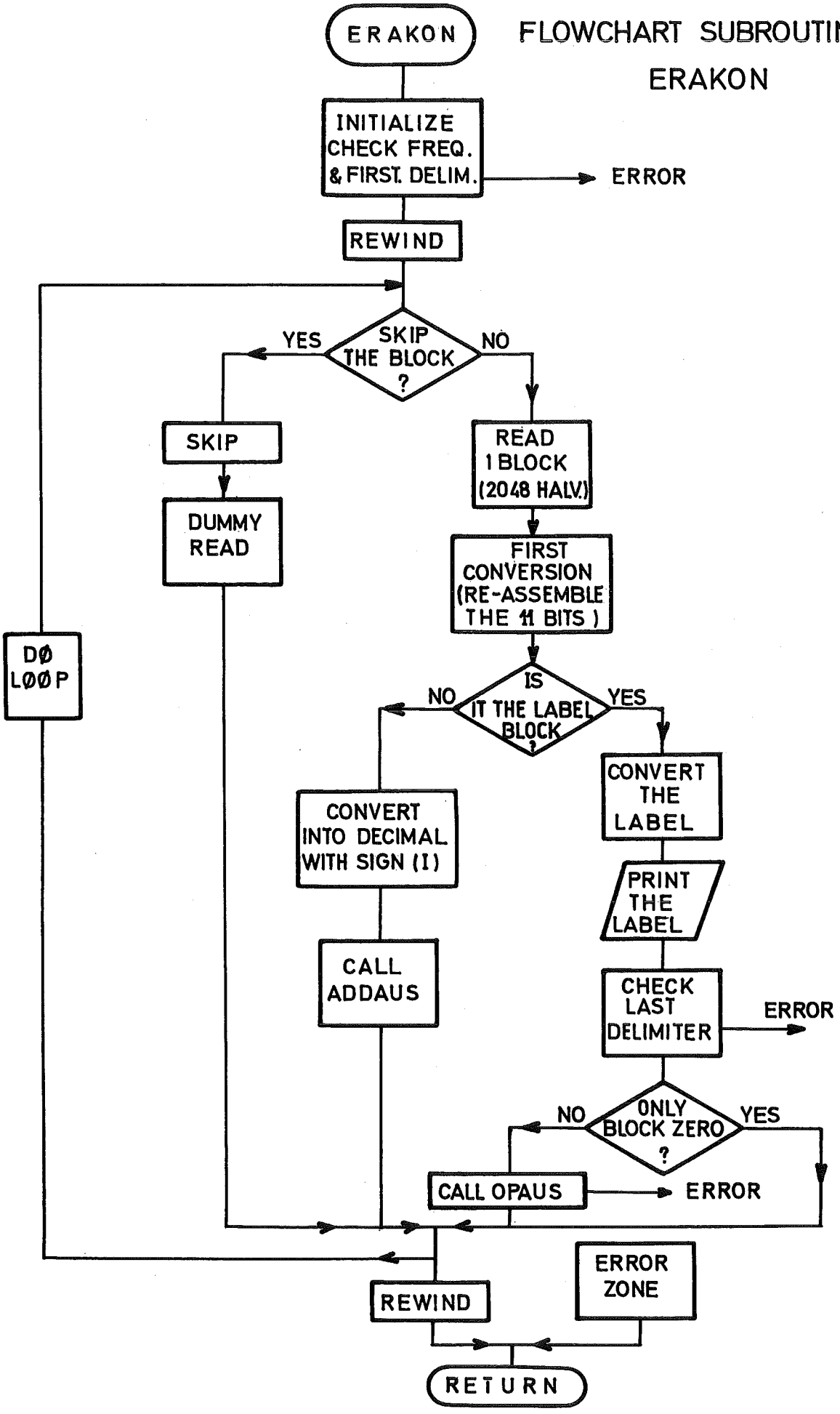
The input-output subsystem must provide the necessary interface between the warehouse and these files and this requires a custom-designed adaptation between the data structure implemented in the warehouse and the data structure of these files. This adaptation is quite straightforward for an output file like the printer file but may be rather complex for other files which depend from the installation or from the implementation of other subroutines (data acquisition system, plot subroutine). Since SEDAP is modular and since the input-output subsystem is a part of SEDAP which was built by assembling different submodules it is easy to substitute any other adaptation to a special input-output file.

2.6.1 Conversion of experimental data recorded by the ERA data acquisition system (SUBROUTINE ERAKON)

The subroutine ERAKON converts the data recorded by the data acquisition system of the Institut für Reaktorentwicklung and stores the resulting recordsin the warehouse. The structure of ERAKON is determined by the specifications of the recording system and they will be briefly described.

- All the input signals must be amplified in order to be compatible with the ± 10 Volt range of the analog to digital converter. It is expected that the user has correctly set the variable low pass filter built around the amplifier loop in order to avoid any aliasing. (Introduction of low frequency oscillations, which do not exist in the physical signal, due to the digital sampling method) (see § 3.3.2).
- The number of channels is always of the 2^N form which gives

FLOWCHART SUBROUTINE
ERAKON



COMMANDS:	ERAKON converts the data recorded by the ERA data acquisition system and stores the results into the warehouse.	NAME = ERAKON SYSTEM = Input ENTRY = None
ERAK		

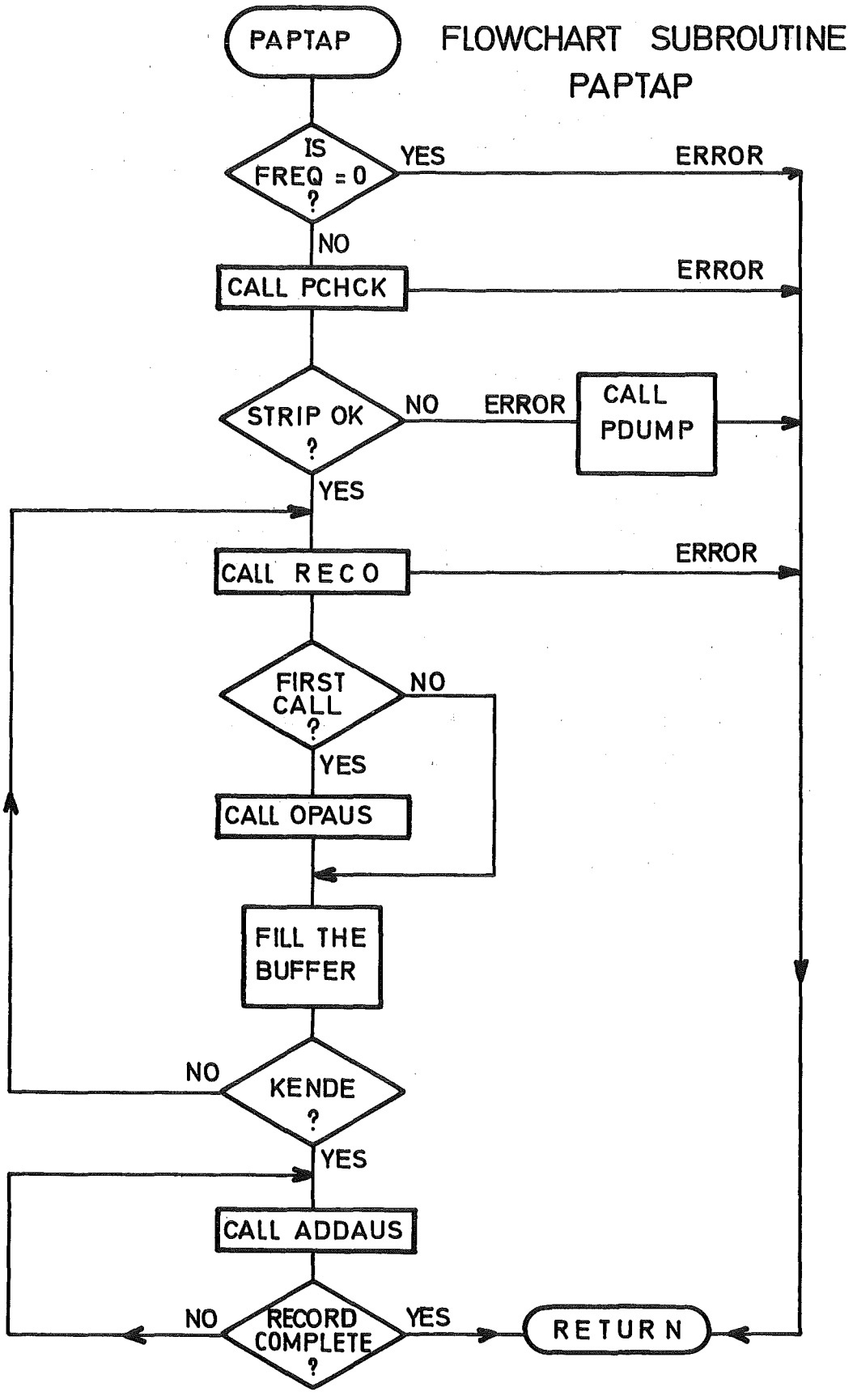
CALL ERAKON(KBAND,KBANF,KBEND,KBDIF,GNAM,K3,FREQ,DAT,ZEYT)

LIST OF ARGUMENTS:

KBAND	file number for the magnetic tape
KBANF	first block to be converted. (A block contains 1024 values and block 0 is the label block)
KBEND	last block to be converted.
KBDIF	number of blocks to be converted
GNAM	name of the resulting record
K3	search index of FNAM in the catalog (must be -1)
FREQ	is the sampling frequency
DAT	is the date (day, month, year) example: 0306.72 for June 3rd, 1972
ZEYT	is the time (seconds)

SUBROUTINES OR FUNCTIONS NEEDED: OPAUS, ADDAUS

ERRORS	DIRECT:	1,2,4,5,15,16,18
	INDIRECT:	OPAUS


PAPTAP
FLOWCHART SUBROUTINE
PAPTAP
IS
FREQ = 0
?
YES
ERROR
NO
CALL PCHCK
ERROR
STRIP OK
?
NO
ERROR
CALL
PDUMP
YES
CALL RECO
ERROR
FIRST
CALL
?
NO
YES
CALL OPAUS
FILL THE
BUFFER
NO
KENDE
?
YES
CALL ADDAUS
NO
RECORD
COMPLETE
?
YES
RETURN

COMMANDS:	PAPTAP converts experimental data originally recorded on paper tape and stores them into the warehouse. Faulty tapes are dumped into the print file.	NAME = PAPTAP SYSTEM = INPUT/OUTPUT ENTRY = None
PTAP		

CALL PAPTAP(KSTRIP,GNAM,FREQ,DAT,ZEYT,K3)

LIST OF ARGUMENTS:

KSTRIP is the number of the file containing the paper tape data.

FNAM is the name of record to be converted

FREQ is the sampling frequency

DAT is the date of the record

ZEYT is the time of the record

K3 is the search index of FNAM (K3 must be -1)

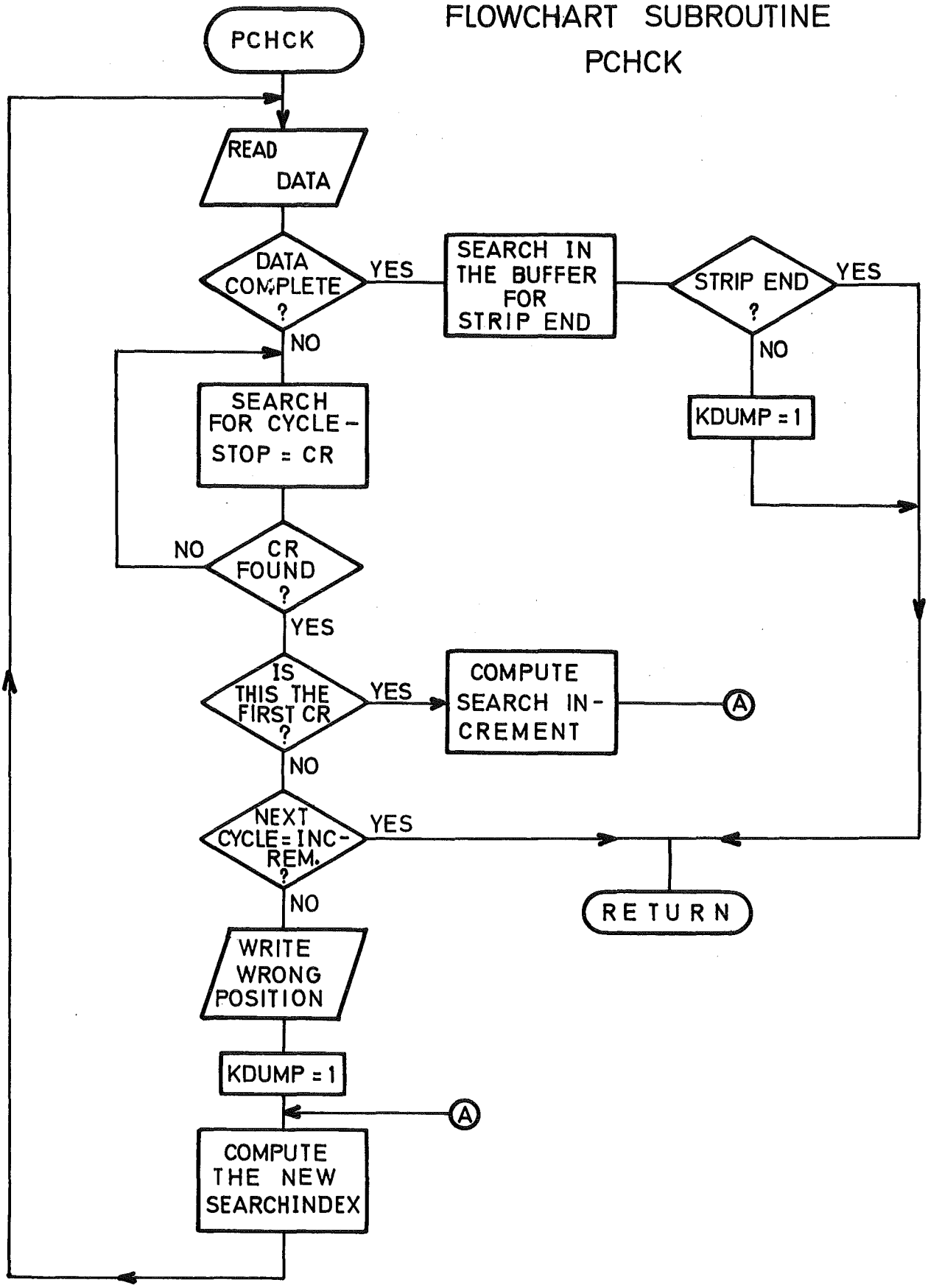
SUBROUTINES OR FUNCTIONS NEEDED: PCHCK,RECO,PDUMP,OPAUS,ADDAUS

ERRORS

DIRECT: 15,21

INDIRECT: see OPAUS

FLOWCHART SUBROUTINE PCHCK



COMMANDS:	PCHCK examines the data of a paper tape record. It computes the number of channels and cycles.	NAME = PCHCK SYSTEM = INPUT-OUTPUT ENTRY = None
None		

CALL PCHCK(KSTRIP,KDUMP,KZAHL,NP,IERR,JERR)

LIST OF ARGUMENTS:

KSTRIP is the number of the file containing the paper tape data.

KDUMP is an error indicator, KDUMP = 0 means no error, otherwise the paper tape is not correct.

KZAHL is the number of cycles recorded on the tape file.

NP is the file number of the printer

IERR is an argument of the error code

JERR is an argument of the error code for a further comment.

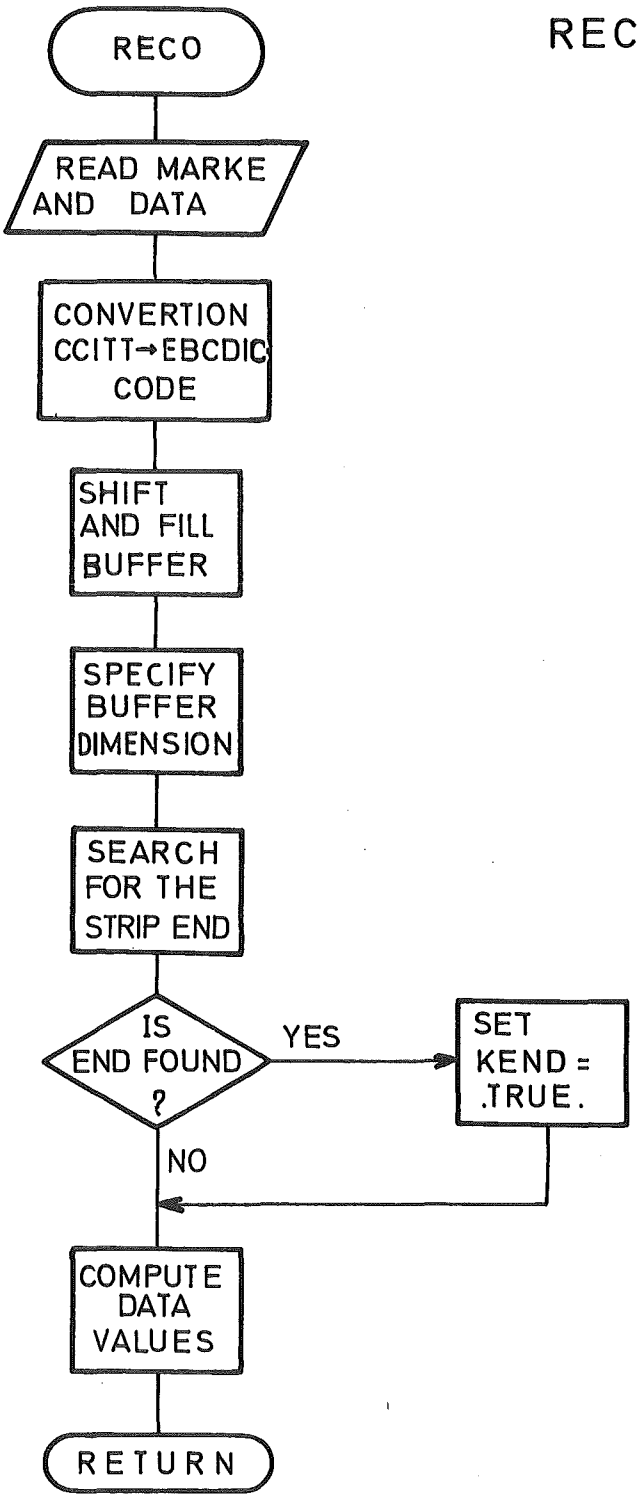
SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 1

INDIRECT: None

FLOWCHART SUBROUTINE RECO



COMMANDS: None	RECO converts the data of a paper tape from CCITT-2 code to EBCDIC code and floating point numbers	NAME = RECO SYSTEM = INPUT-OUTPUT ENTRY = None

CALL RECO(BEG,DATA,ZAZYK,KANAL,PUFFIN,KENDE,REZYK,KSTRIP)

LIST OF ARGUMENTS:

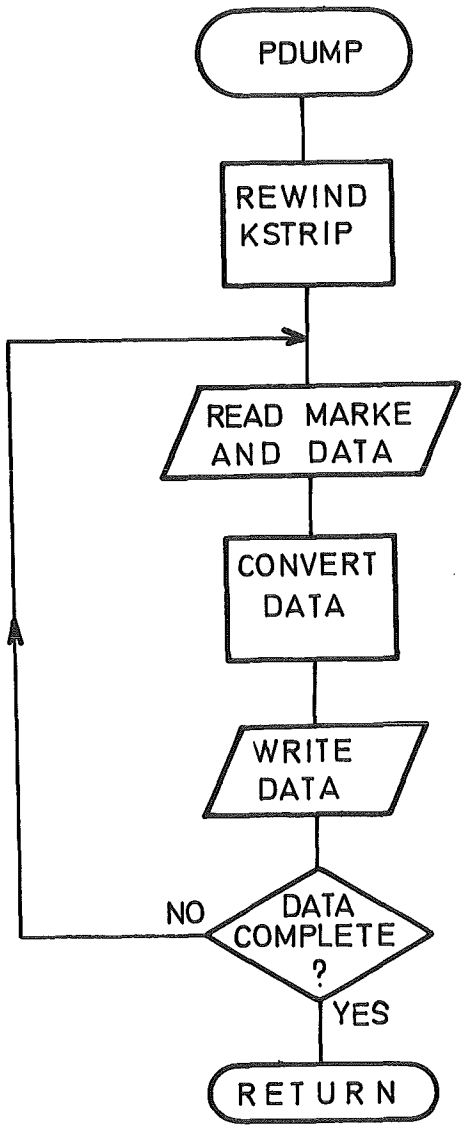
BEG	is a logical variable, initially set .true., after the first call of RECO it is altered to .false.
DATA	is an array containing the converted data.
ZAZYK	is the number of cycles of the data record
KANAL	is the number of the recorded channels
PUFFIN	is an index of buffer contents
KENDE	is a logical variable, initially set .true., and changed at the end of the data file
REZYK	is the rest of the buffer contents during any data cycles are returned to PAPTAP
KSTRIP	is the number of the file containing the paper tape data.

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS	DIRECT: 1 INDIRECT: None

FLOWCHART SUBROUTINE

PDUMP



COMMANDS:	PDUMP prints a dump of erroneous paper tape data.	NAME = PDUMP SYSTEM = INPUT-OUTPUT ENTRY = None
None		

CALL PDUMP(KSTRIP,NP)

LIST OF ARGUMENTS:

KSTRIP is the number of the file containing paper tape data
NP is the file number of the printer

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS
DIRECT: None
INDIRECT: None

the following combinations: 1, 2, 4, 8, 16 or 64 channels. The multiplexing of the channels and the analog to digital conversion are performed by a single unit (Raytheon Miniverter) with a maximum sampling rate of 40 KHz which has to be divided by the number of channels to obtain the maximum frequency of a channel. Different clock rates are available to control the fast sampling rates and a software loop which builds a variable time delay allows the use of very low sampling rates by using the so-called random mode.

The voltages are converted into a 11 bit complementary binary code and the first bit indicates the polarity. The converted values are stored in the memory of the Raytheon 703 computer and when 1024 values are stored, they are transferred as a complete block on the 7 tracks magnetic tape.

ERAKON verifies the delimiters and checks if the frequency is positive. The delimiters are incremented by one to take into account the fact that the label block is considered as the block No. 0. The label block is always interpreted since it provides useful information about the different parameters used to perform the recording. The 11 bits of data having been split into two six bit groups (5 + 6) on the tape are recombined. The necessary bit shifting operations beeing not available in FORTRAN are replaced by the appropriate integer division (or multiplication). The results are matched with a table which contains all the alphanumeric characters and the label is printed.

Once the label block has been interpreted a number of blocks may be skipped by a dummy READ if the first block to be converted is not the block No. 2. Since the label block contains information on how many blocks were recorded, the task will be rejected if the last delimiter exceeds this limit. The numerical conversion is easy since only two complementary operations are needed: a test for the polarity and a constant coefficient to convert the 4096 levels into the 10 Volt range. The first numerical conversion initializes the transfer by an OPAUS call and every block of 1024 values is transferred

to the warehouse to be stored as two 512 SEDAP blocks.

The first conversion program was written by P. Tack and made use of an assembler subroutine. The help of Mr. J. Krieger who made valuable suggestions to improve the speed and the structure of ERAKON is gratefully acknowledged.

2.6.2 Processing of data on paper tape (Subroutines PAPTAP, PCHCK, RECO, PDUMP)

At the Institut für Reaktorentwicklung (Institute for reactor development) of the Gesellschaft für Kernforschung a special data acquisition equipment exists for recording of experimental data on paper tape.

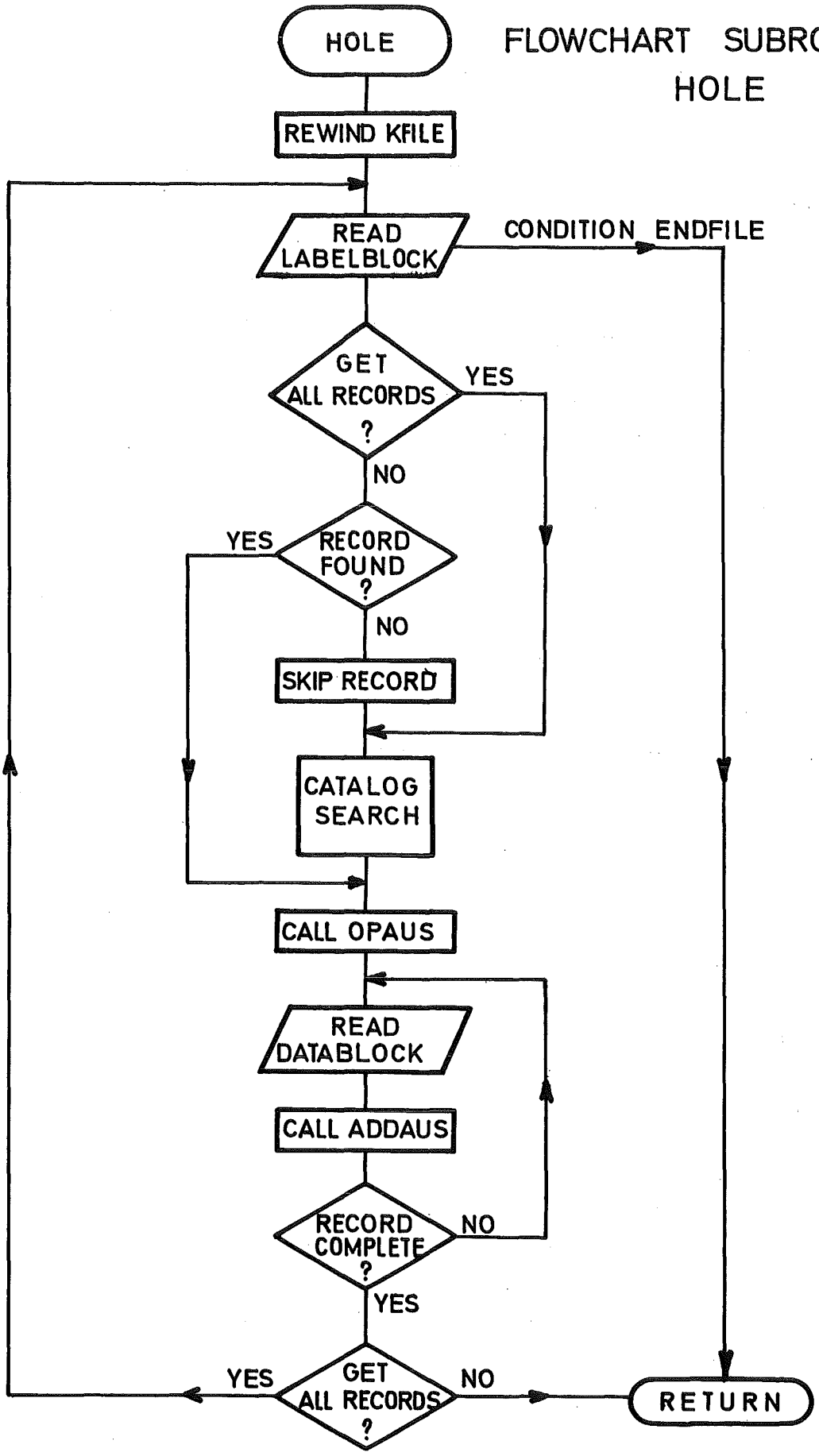
The specifications of the recording system will be briefly described.

- 40 channels may be treated. The first channel records the clock time, so up to 39 channels may be used for experimental data acquisition.
- A data cycle records the time plus any data channels. At the end of a cycle a controlsign CR (Carriage Return) is given.
- The data are recorded in CCITT-2 code.
- An experimental record is finished by a file limiter.

SEDAP does not read the paper tape directly. However, in a special step this data is read by a papertape reader and stored on a disk of the IBM 360/370 computer from which it is read by SEDAP in a subsequent step. The further processing of the experimental data by SEDAP is performed by the subroutine PAPTAP.

To examine the data file, PAPTAP calls the subroutine PCHCK because the results of the data acquisition are not always correct. Following verifications are executed: The length of the first cycle (i. e. the number of channels in this cycle) is detected. This increment is used to search the whole file for the correct position of the subsequent cycle-limiters (CR) and for the file limiter. In the case of stated errors appropriate messages are printed and a code is returned to the calling

FLOWCHART SUBROUTINE HOLE



COMMANDS:		
HOLE	HOLE transfers data from permanent storage (disk or tape) to the warehouse. This data must be stored according to the SEDAP-format (per default if generated from DUMP)	NAME = HOLE SYSTEM = INPUT ENTRY = None

CALL HOLE(KFUNC,FNAM,KFILE,KN)

LIST OF ARGUMENTS:

KFUNC is an option indicator
KFUNC = 1 if only one record is to be restored
KFUNC = 2 if all records are transferred

FNAM is the name of the record to be fetched

KFILE is the number of the dump-file referenced on the corresponding JOB-control-card

KN is the search index of FNAM

SUBROUTINES OR FUNCTIONS NEEDED: CTLG,OPAUS,ADDAUS

ERRORS

DIRECT: 1,2

INDIRECT: see OPAUS,ADDAUS

routine to decide if all data will be dumped (printed in a special format) by the subroutine PDUMP or whether it might be possible to correct the paper tape. If no error is found, the data are converted with the routine RECO into the EBCDIC-code and floating point numbers.

Finally PAPTAP fits the data to the SEDAP conventions and performs the transfer to the warehouse.

2.6.3 Restoring of data files (Subroutine HOLE)

Data files generated by the DUMP-command of SEDAP or other programs according to the SEDAP-format conventions can be transferred back to the warehouse with the subroutine HOLE (the command has the same name).

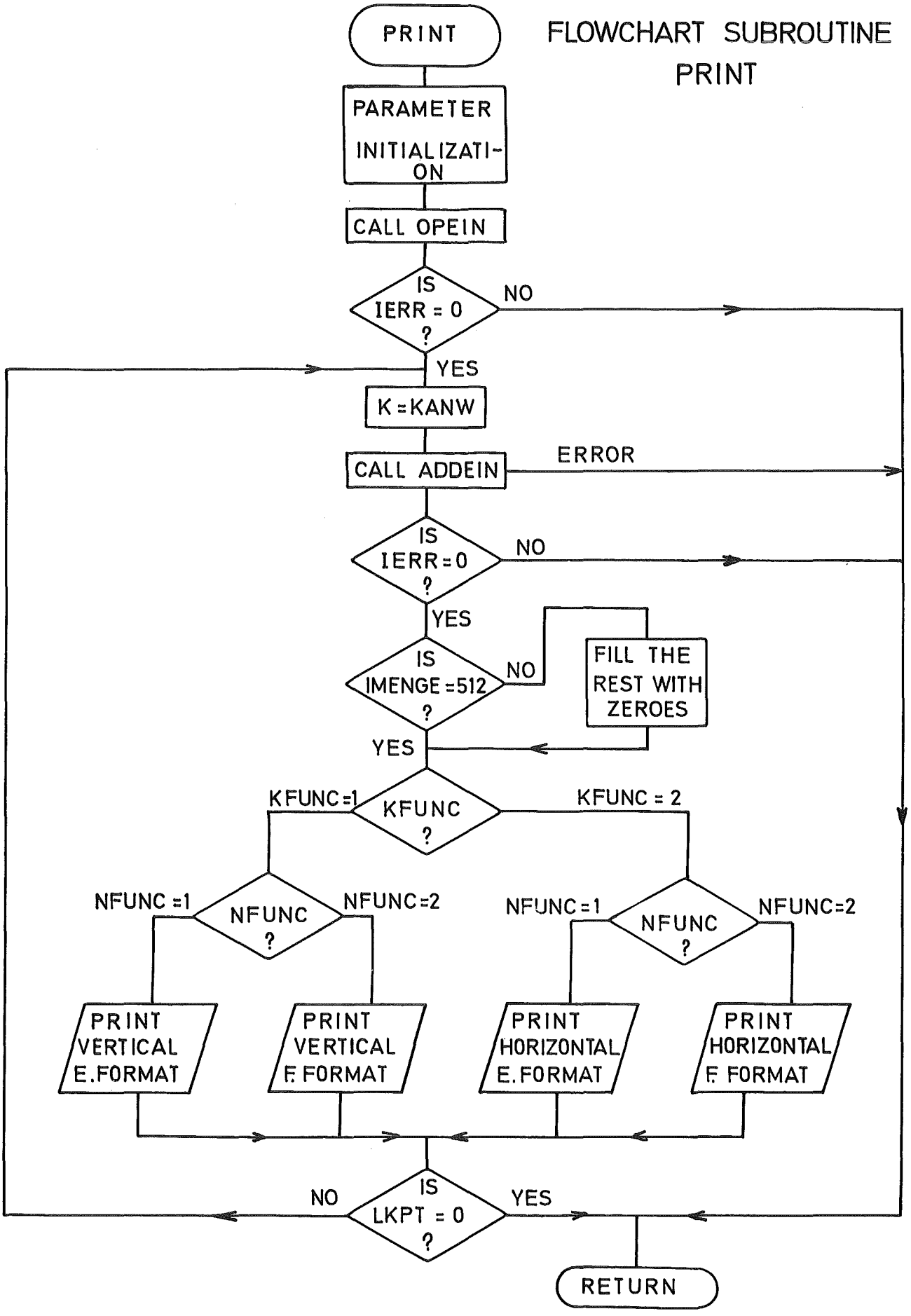
Every call of HOLE causes a REWIND of the dump file that contains the various data files. Now the label of the first file is read. In the case that only one data file should be restored, the label is searched for the name of the file. If it is the wanted name, the following data are carried to the warehouse using the TRANSFER routines of SEDAP. The describing parameters of the data are stored in the common storage. Otherwise, the data of the first file are skipped and the following files are looked up and possibly transferred.

If all records on the dump file should be restored the transfer is performed as described above, but without search for a name.

The user is warned not to use an uncontrolled series of mixed DUMP and HOLE commands because every dump may unintentionally destroy data files at the end of the dump file. In this case it is the users responsibility to program a logically correct succession of his commands.

2.6.4 Printed data output (Subroutine PRINT)

The subroutine PRINT prints the blocks of a record on the standard output file. The subroutine checks the input request by the standard OPEIN call and uses the fast transfer mode with KXYZ = 2 to bring a block of 512 values every time in the


PRINT
FLOWCHART SUBROUTINE PRINT
PARAMETER INITIALIZATI-ON
CALL OPEIN
IS IERR = 0 ?
NO
YES
K=KANW
CALL ADDEIN
ERROR
IS IERR=0 ?
NO
YES
IS IMENGE=512 ?
NO
FILL THE REST WITH ZEROES
YES
KFUNC=1
KFUNC ?
KFUNC = 2
NFUNC =1
NFUNC ?
NFUNC=2
NFUNC =1
NFUNC ?
NFUNC=2
PRINT VERTICAL E.FORMAT
PRINT VERTICAL F FORMAT
PRINT HORIZONTAL E.FORMAT
PRINT HORIZONTAL F FORMAT
IS LKPT = 0 ?
NO
YES
RETURN

COMMANDS: PBVE,PBHE, PBVF,PBHF	PRINT transfers the values from the warehouse to the output file and writes one data block per page. Four options are available to select a vertical or a horizontal order as well as an F or E-Format	NAME = PRINT SYSTEM = OUTPUT ENTRY = None

CALL PRINT(KFUNC,NFUNC,ENAM,KANW,KENW,K1)

LIST OF ARGUMENTS:

KFUNC indicates the selected printing scheme (vertical if KFUNC = 1, horizontal for KFUNC = 2)

NFUNC indicates the selected printing format (E-Format if NFUNC = 1, F-Format for NFUNC = 2)

ENAM is the name of the record to be printed

KANW,KENW delimit the selected segment of the record ENAM

K1 is the search index of ENAM (If K1 = -1, the record has not been found in the warehouse)

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN, ADDEIN

ERRORS

DIRECT: None

INDIRECT: see OPEIN and ADDEIN

first 512 locations of the X array. The subroutine checks if the block is the last block of a record. If this is verified, the filling factor is unvestigated and the rest of incompletely filled blocks is filled with zeroes. The two function indicators are then decoded to select a vertical or a horizontal printing order and to use the F- or E-Format. The transfer is terminated when the total number of points has been printed. This condition is detected by a nonpositive value of LKPT.

2.6.5 Graphical output (Subroutine GRAPH with entry GRAPH1)

Records generated by SEDAP may be plotted with the help of the subroutine GRAPH and its entry GRAPH1 as functions of time or frequency. It was tried to satisfy the different demands of the users with respect to comfort and flexibility.

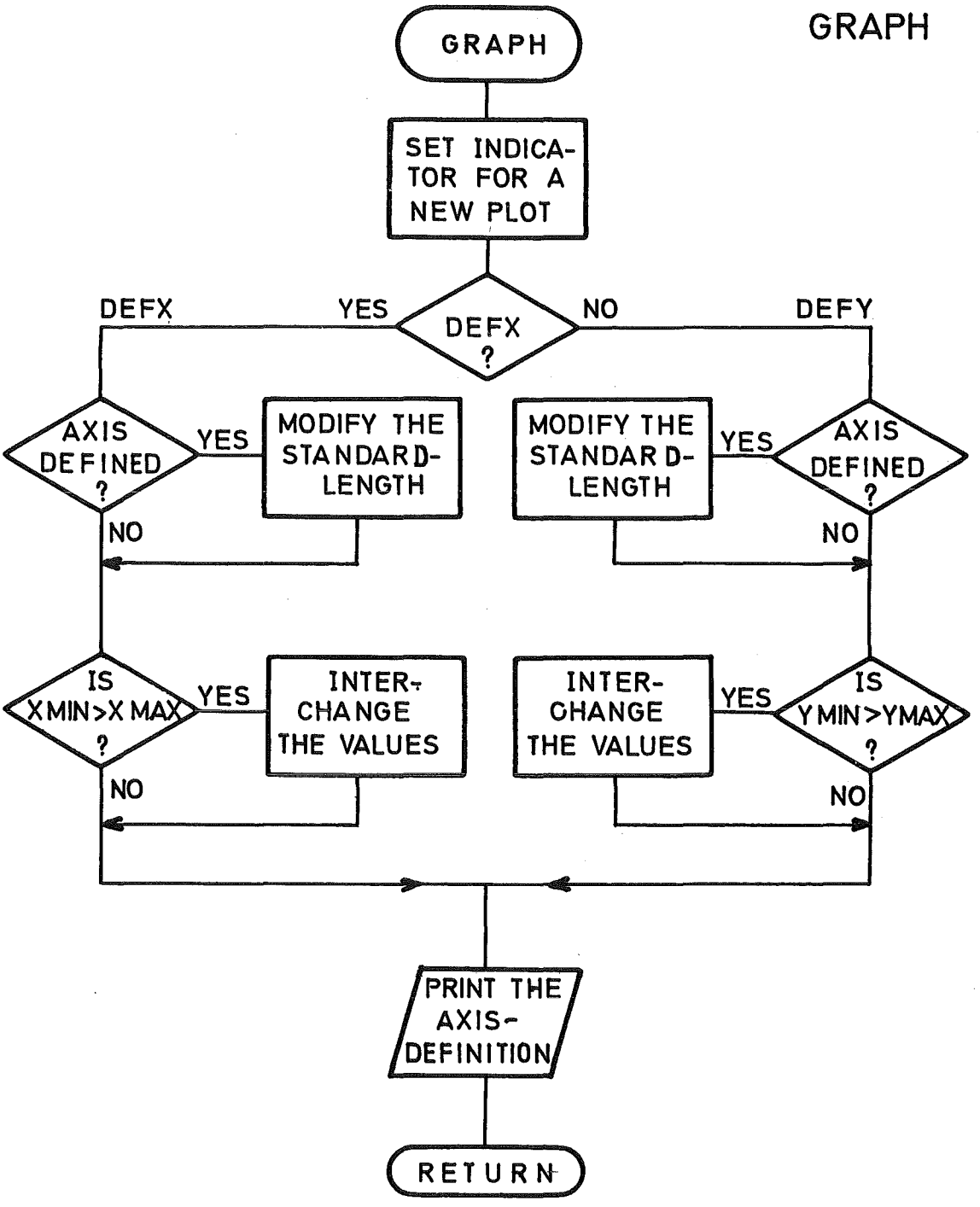
The following possibilities exist to produced plot output:

- The lengths and the scales of the coordinates may be predefined.
- Together with the first curve of a plot, a comment may be given to characterize it.
- Various curves may be plotted into one diagram.
- There also exist possibilities to choose the sort of ink and paper.

At the first part of the subroutine GRAPH the informations given with the commands DEFX, DEFY to specify the coordinates are verified. The standard values are overwritten by the input data for the coordinate definitions. If minimum values are defined greater than the maximum values, they are exchanged. The second part of GRAPH beginning with the entry GRAPH1 performs the plotting. First the parameters of the PLOT-command are verified.

The values of the X-axis are computed by using of the frequency and time parameters that are stored in the common storage for every experimental record. If the ordinates of a plot are not predefined, the record is searched for its minimum and maximum values to scale the plot axis.

FLOWCHART SUBROUTINE GRAPH



COMMANDS:	GRAPH specifies the limits and the scales of a following plot by overwriting default specifications	NAME = GRAPH SYSTEM = OUTPUT ENTRY = GRAPH1
DEFX DEFY		

CALL GRAPH (XLANG,YLANG,NSWI)

LIST OF ARGUMENTS:

XLANG length of abscissa expressed in centimeters

YLANG length of the ordinate experessed in centimeters

NSWI is an option indicator
NSWI = 1 is caused by the command DEFX and
NSWI = 2 by the command DEFY

SUBROUTINES OR FUNCTIONS NEEDED: None

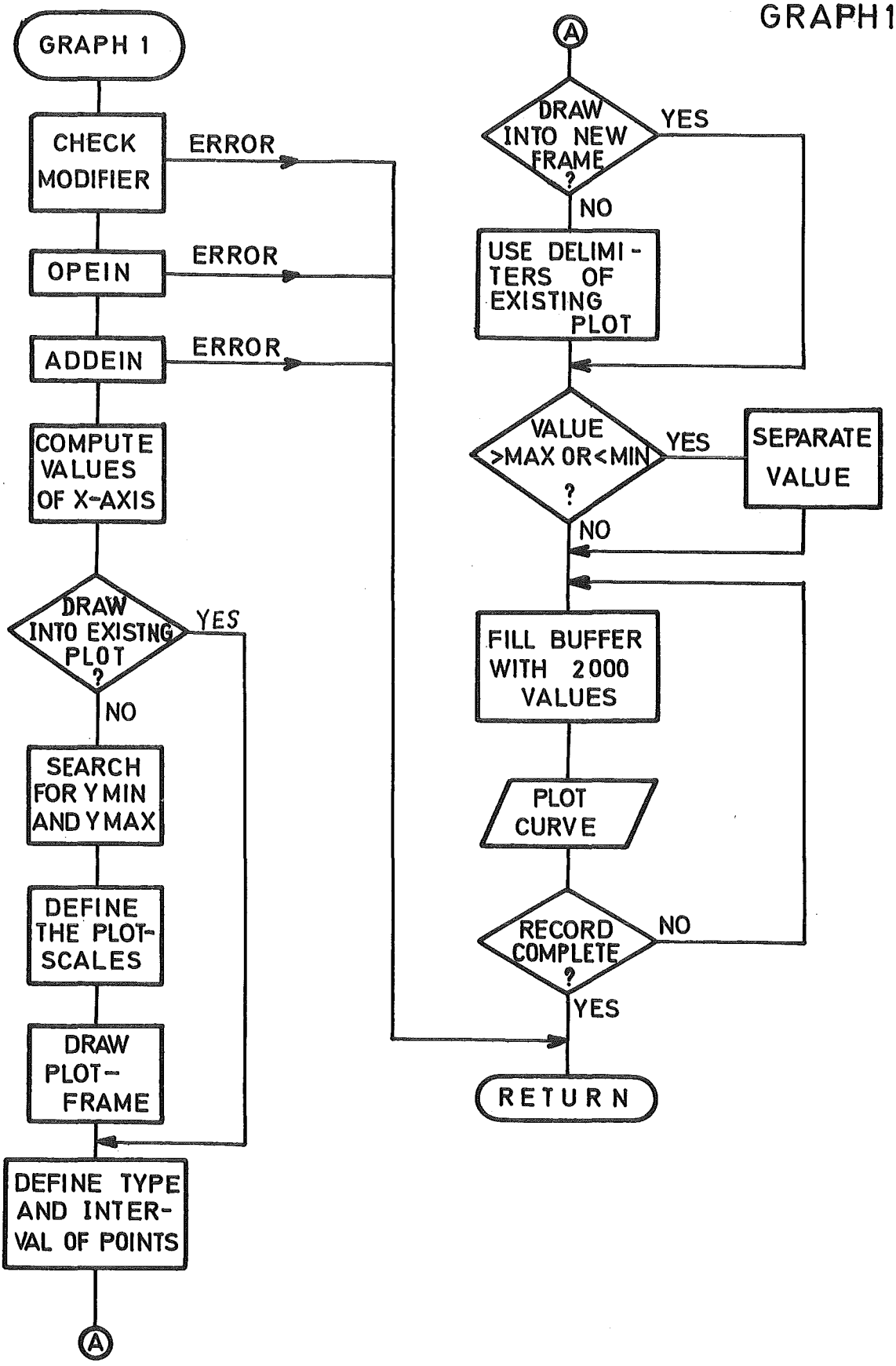
ERRORS

DIRECT: None

INDIRECT: None

ENTRY OF GRAPH

FLOWCHART ENTRY GRAPH1



COMMANDS:	GRPAH1 is an entry to the subroutine GRAPH. GRPAH1 performs the plotting of an experimental record as a function of time or frequency	NAME = GRAPH1 SYSTEM = OUTPUT ENTRY = GRAPH1 is an entry
PLOT		

CALL GRAPH1(NFUNC,KANW,KENW,KRAF,K1,NTX,FNAM,PLOTEN)

LIST OF ARGUMENTS:

NFUNC is an option indicator
NFUNC = 1 a curve is plotted into a newly opened plot
NFUNC = 2 a curve is drawn into an existing diagram

KANW,KENW are the delimiters of the selected record segment to be plotted

KRAF is a sorting factor

K1 is the search index of FNAM (if K1 = -1, the record has not been found in the warehouse)

NTX is an alphanumeric array which contains a comment to characterize the plot

FNAM name of the record to be plotted

PLOTEN is a logical variable. It is set to .TRUE. if a standard plot was opened.

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,PLOTA

ERRORS

DIRECT: None

INDIRECT: see OPEIN, PLOTA

The data to be plotted by the subroutine PLOTA are transferred from the warehouse to the computing array by OPEIN and ADDEIN. Different calls of the subroutine PLOTA /4/ may follow. The first call of PLOTA will only cause the drawing of the plot-frame and the characterizing comment.

The other calls of PLOTA causes the plotting of the datavalues into a newly drawn frame or into an existing plot.

PLOTA is a special Assembler routine for plotting at the computer center of the Gesellschaft für Kernforschung, Karlsruhe. However, an interface routine named PLOTA is also available which converts all calls to PLOTA to the appropriate calls of standard Calcomp software /5/.

2.6.6 Dump of the warehouse (Subroutine DUMP)

The subroutine DUMP enables the user of SEDAP to save his data stored in the warehouse beyond the end of the job on permanent storage files. These may be on disk or tape.

It is possible to dump either one record at a time or all experimental records contained in the warehouse.

To transfer the data from the warehouse to a permanent storage the computing storage (via OPEIN, OPAUS) of SEDAP is used as buffer.

The data can be restored later using the command HOLE. However, they may also be read by other programs.

2.7 Operators

2.7.1 Sorting the channels of a multiplexed record (Subroutine SORTIK)

The subroutine SORTIK separates the different channels of a multiplexed record obtained from the warehouse and stores the resulting new records into the warehouse. SORTIK is written to handle multiplexed records which contain always 2^N channels with the following possible options given by KSORT: 2, 4, 8, 16, 32, 64.

The first part of the subroutine is needed to generate the new names for the resulting records. When a system's user has up to 64 channels to sort, it would be a tedious work to provide a list of 64 names. The solution which was adopted for naming the new records was to derive the new names from the proposed record name by substituting the numbers '01' to '64' to the two last characters of this only name.

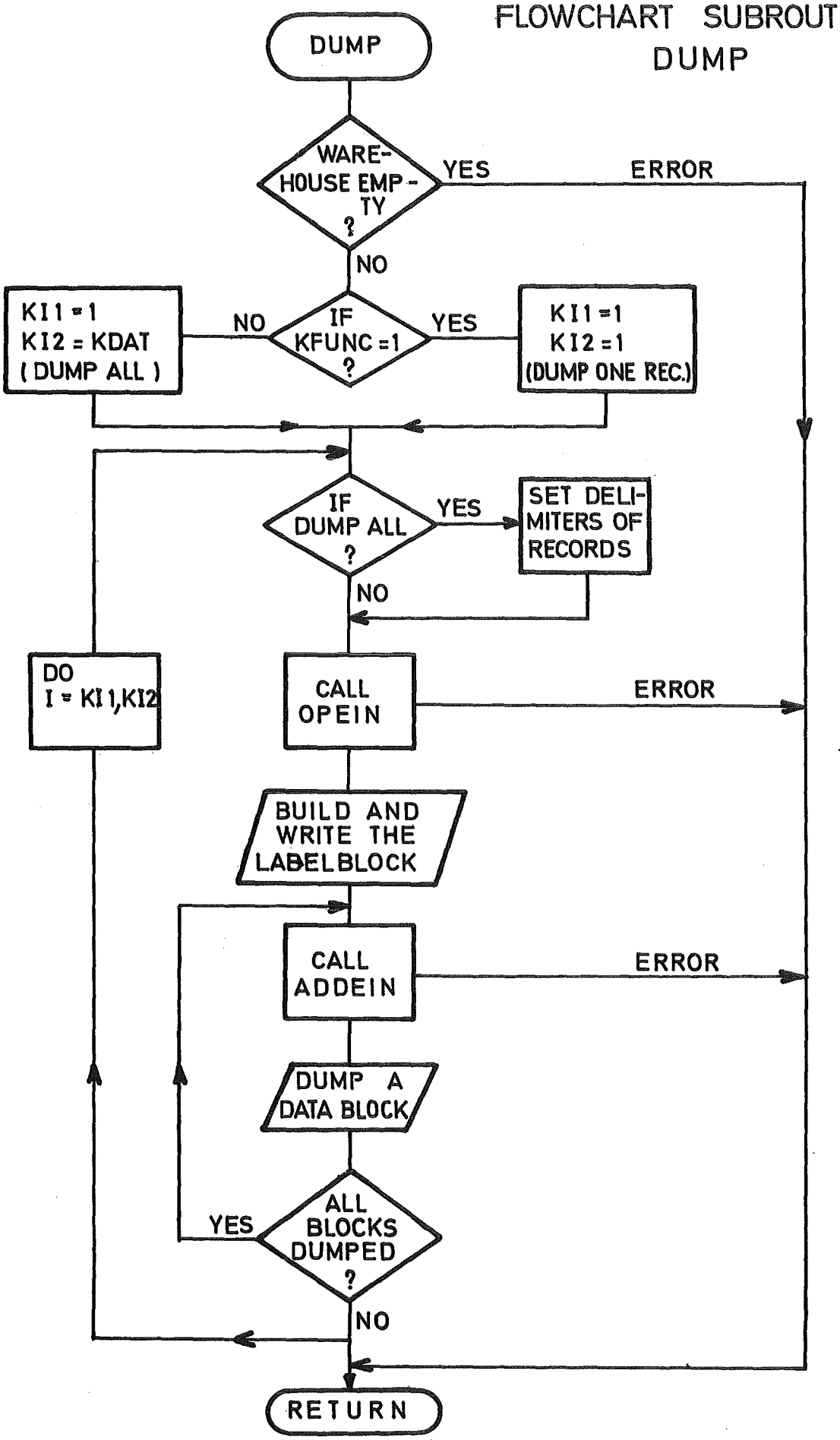
The proposed record name DA$$ will be transformed into DA01, DA02, ... DA64. The two characters $$ are not necessary but they are recommended to the user to keep him aware of the fact that they will be replaced. This substitution takes advantage of the fact that (the IBM-) FORTRAN compilers allow an equivalence of a REAL*4 with 2 INTEGER*2 or 4 LOGICAL*1. The numbers 0 to 9 are initially stored as characters in a data statement and are used by two nested DO loops in a counter-like generator to provide the second half of the names.

An OPEIN call verifies the validity of the input request and returns the necessary arguments. The frequency of the input record is divided by the floated sorting factor to become the frequency of the sorted records. An exception is made if the proposed record name is terminated by the two characters 'FT'. This option together with KSORT = 2 is reserved to separate the real and imaginary parts of a complex Fourier Transform (FT) and skips the frequency division.

Since the MAIN has not checked the newly generated names SORTIK calls the ENTRY CTLG to search the catalog with the resulting index to be used by OPAUS.

The output records are opened in the warehouse by a DO loop which calls OPAUS and computes the necessary parameters from the arguments given by OPEIN. It is important to note that the number of new records is defined by NFUNC and not by KSORT in order to limit the number of created records at the user's request. Such a situation arises for instance if 32 channels were recorded but only 25 connected to the experiment. The experimenter cannot be satisfied with the lower limit of 16 channels (2^N) and must select 32 channels. During the sorting

FLOWCHART SUBROUTINE DUMP



COMMANDS: DUMP	DUMP stores experimental records of the warehouse on permanent data files that may be on disk or on tape. The DUMP command must be followed by a comment card	NAME = DUMP SYSTEM = OUTPUT ENTRY = None

CALL DUMP(KFUNC,ENAM,KANW,KENW,KFILE,K1,LIST)

LIST OF ARGUMENTS:

KFUNC is an option indicator
KFUNC = 1 only one data file will be dumped
KFUNC = 2 the content of the warehouse will be dumped

ENAM is the name of the record to be dumped

KANW,KENW are the delimiters of the selected record segment

KFILE is a file reference number given on the according JOB-control card

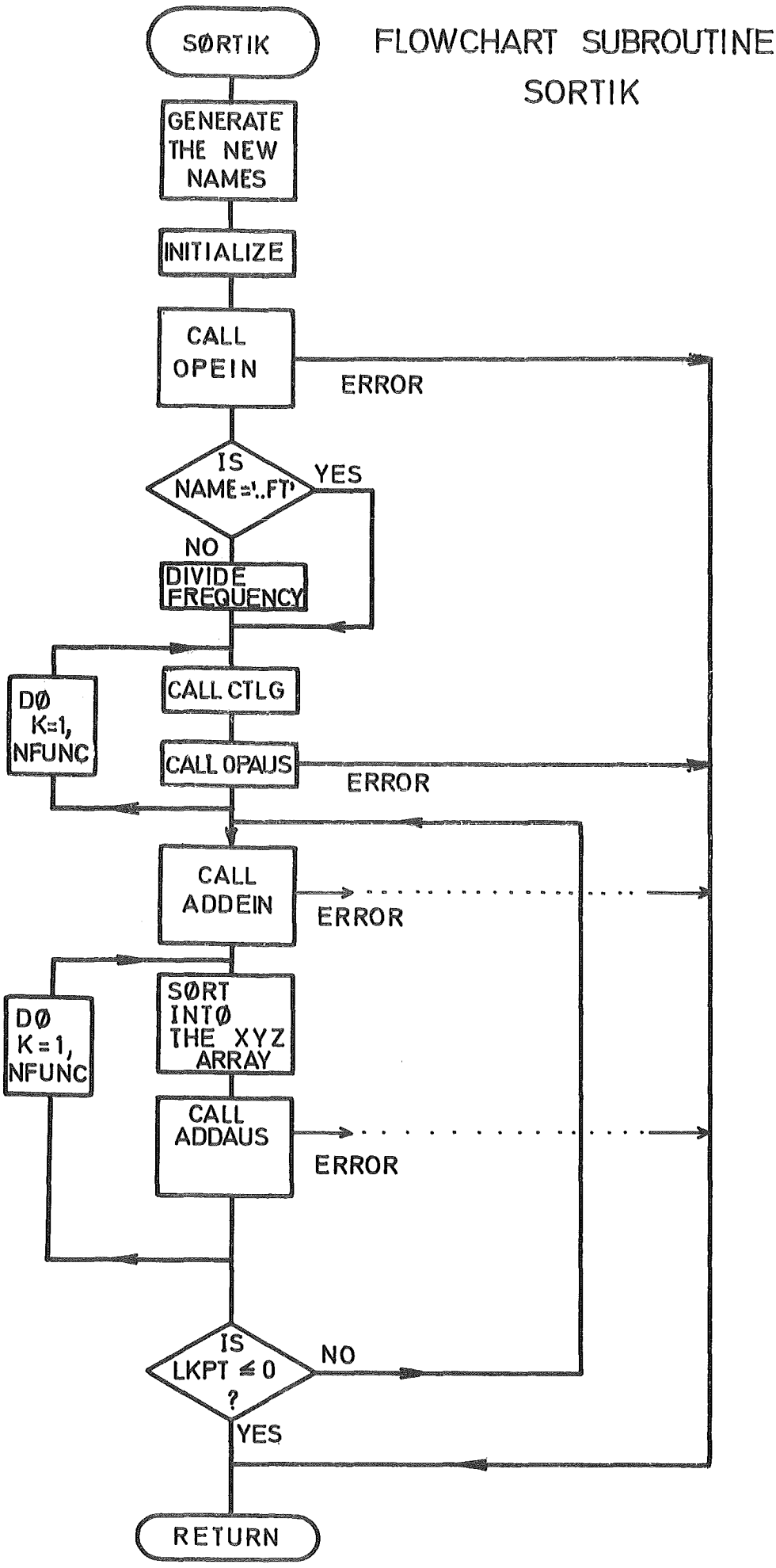
K1 is the search index of ENAM

LIST is an alphanumeric array of a length of 80 bytes. It contains the comment written on the card following the DUMP command.

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,ADDEIN

ERRORS	DIRECT: INDIRECT:

FLOWCHART SUBROUTINE SORTIK



COMMANDS:	SORTIK sorts the values of an multiplexed record. The number of channels must be 2^N ($1 \leq N \leq 6$). The sorted values are stored into 2^N new records of the warehouse.	NAME = SORTIK SYSTEM = OPERATORS ENTRY = None
SØ02,SØ04,SØ08 SØ16,SØ32,SØ64		

CALL SORTIK(KSORT,NFUNC,ENAM,FNAM,K1,KANW,KENW,K3)

LIST OF ARGUMENTS:

KSORT is the number of channels to be sorted (2^n)

NFUNC allows to limit the sorting operation to the NFUNC first channels ($1 \leq$ NFUNC $\leq$ KSORT)

ENAM is the name of the record to be sorted

FNAM is the name of the resulting records (the last two characters of FNAM are replaced by numbers from 01 to 64)

K1 is the search index of the record to be sorted (K1 = -1 if the record was not found)

KANW,KENW are the delimiters of the selected portion (in blocks)

K3 is the search index of FNAM (must be -1)

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,OPAUS,ADDEIN,ADDAUS,MOD,CTLG

ERRORS

DIRECT: None

INDIRECT: see OPEIN,OPAUS,ADDEIN,ADDAUS

operation, if he has specified a value of NFUNC = 25, he will limit the number of records stored in the warehouse to 25 since the last 7 records would not have any meaning for his data reduction.

The transfer of the values begins by an ADDEIN call which transfers the first 16384 values (if available) into the first part of the XYZ array. The sorting algorithm will be repeated NFUNC times with the location of the first transferred value being shifted every time. The sorted values are stored into the XYZ segment adjacent to the previous segment (i. e. starting at the location XYZ (16385) and are transferred immediately to the warehouse by ADDAUS for every channel. SORTIK cannot use the updating features of the pointers by itself. This is done by applying a formula closely related to the sorting algorithm. The number of blocks transferred by an ADDAUS call whenever ADDEIN has been called depends upon the sorting factor according to the following relation:

$$NB = \frac{16384}{512 \times KSORT} \quad \text{(16384 implies a filled segment)}$$

NB = 16 for KSORT = 2 or NB = 1 for KSORT = 32 but NB is only 0.5 for the maximum case when KSORT = 64. Since the computing arrays were limited because of the program size, the transfer of the resulting values in the case of KSORT = 64 can be made only with 256 values, i. e. an half block. Provision has been made in ADDAUS to store the second half block by specifying KFUNC = 2. This transfer mode requires a reading operation to get the first half block to which the 256 last values will be added. This is the only case where ADDAUS is followed by a test for error since it is the only case where ADDAUS involves a direct access READ.

The transfer operation continues until the total number computed by OPEIN has been exhausted.

2.7.2 Standard operations (Subroutine OPERA)

The subroutine OPERA performs 11 standard operations which can involve one to three records and which are characterized by the

fact that the overlapping of the segments is not necessary.

The subroutine first initializes the service parameters. A computed GO TO allows to print a specific message to complete the general sentence printed by the MAIN program.

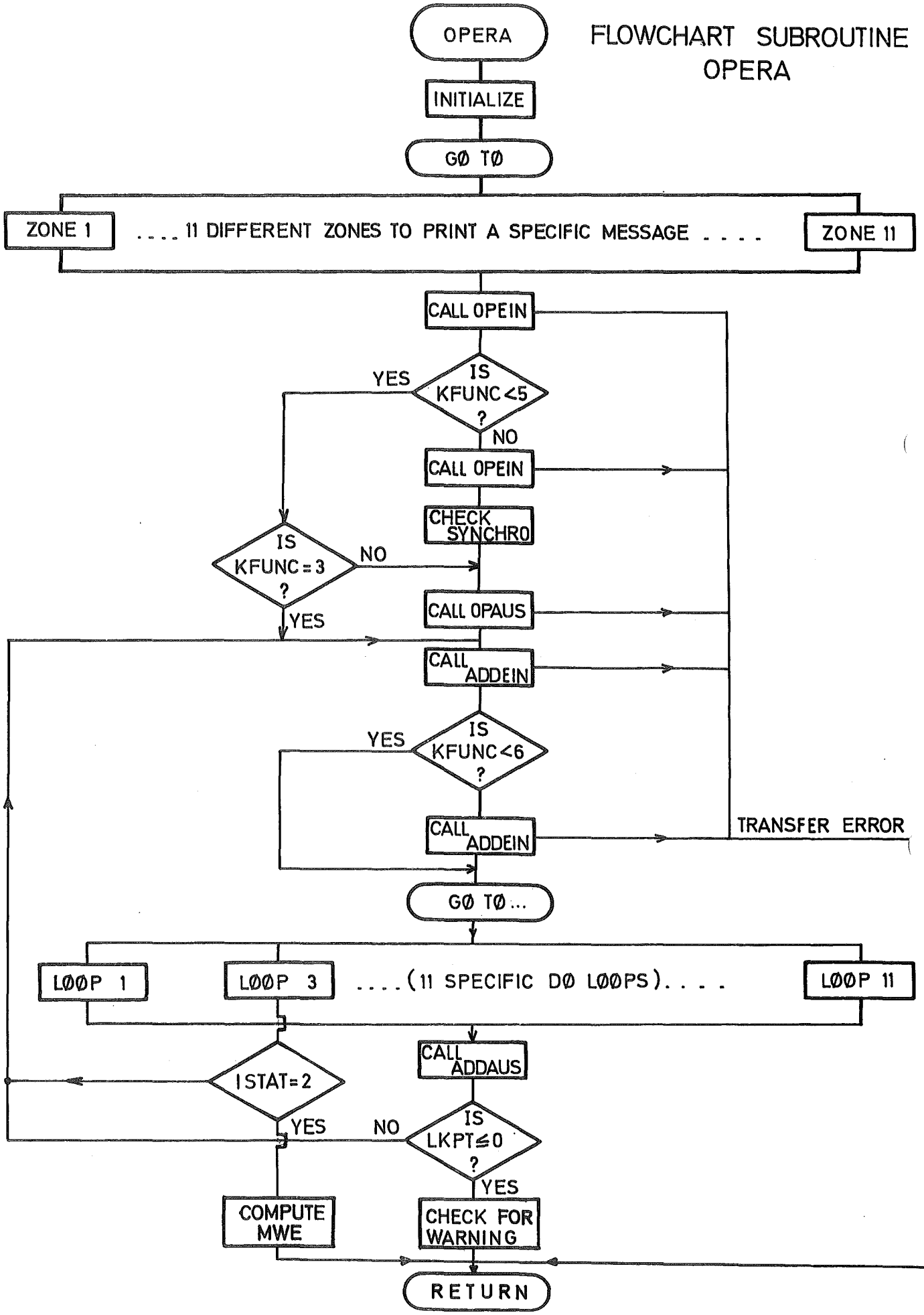
The following operations correspond to the 11 different options selected according to the value of KFUNC

1) Linear translation (aX + b)
2) Conversion of a NiCr thermocouple voltage into ^{o} C.
3) Computation of the mean value
4) Subtraction of the mean value found by (3)
5) Addition of two records
6) Subtraction of two records
7) Multiplication of two records
8) Division of two records
9) Multiplication of two complex records
10) Multiplication of a record by the complex conjugate of another record
11) Complex division of two complex records

The first input record is always needed and is opened by OPEIN according to the standard scheme. If the option index KFUNC is greater than 4, the specified operation requires a second input record and a second OPEIN is necessary. It will be explained in § 3.3.1 that the two input records must normally be synchronous and this property is investigated. If this condition is not met, a warning is printed to call the attention of the user but no action is taken since the selected parameters are derived from the first OPEIN and since the number of blocks has been checked by OPEIN in both cases.

The transfer begins by the first ADDEIN (KXYZ = KX = 2) and is followed by another ADDEIN (KXYZ = KY = 3) if KFUNC is greater than 4. One of the 11 different DO loops performs the specified computation and the results are stored in the warehouse by an ADDAUS call (KXYZ = KO = 3). The transfer process with ADDEIN - ADDAUS continues until the total number of points has been transferred, i. e. for LKPT $\leq$ 0.

FLOWCHART SUBROUTINE OPERA



COMMANDS:	OPERA performs one of 11 different standard operations on one or two records stored in the warehouse. The resulting record is stored in the warehouse with the exception of the MWEF case (no output record)	NAME = OPERA SYSTEM = Operators ENTRY = None
AX+B,TNIC,MWEF MWES,ADDI,SUBT MULT,DIVI,MUKO KOKO,DIKO		

CALL OPERA(KFUNC,ENAM,FNAM,GNAM,K1,K2,K3,KANW,KENW,KRAF,A,B,WWE)

LIST OF ARGUMENTS:

KFUNC is the option index which selects one of the 11 operations

ENAM is the name of the first input record

FNAM is the name of the second input record when needed

GNAM is the name of the output record

K1 is the search index of ENAM (must be positive)

K2 is the search index of FNAM (must be positive)

K3 is the search index of the record GNAM (must be -1)

KANW,KENW are the delimiters of the selected record segment and are expressed in blocks

KRAF is a sorting factor applied to the input records

A and B are the two arguments needed for the linear translation aX + b (for KFUNC = 1)

WWE mean value of a record, determined by a MWEF command and used in a subsequent MWES command

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,ADDEIN,OPAUS,ADDAUS,TNICR2

(The standard complex features of FORTRAN IV are required)

ERRORS

DIRECT: None (warnings are issued)

INDIRECT: see OPEIN, OPAUS, ADDEIN

This transfer scheme has only one exception when the mean value has to be found (KFUNC= 3). In that case the mean value is set to zero by the first part of the subroutine and will be stored by the value WWE after completion of the computation in order to be used by a subsequent call of type KFUNC = 4. (Subtract the mean value). This has to be considered for the Overlay version and implies that the mean value to be subtracted should be subtracted immediately after the task where it was computed. OPAUS and ADDAUS being unnecessary are bypassed. At the end of the DO loop the termination of the transfer is eventually detected and the WWE value which is the sum of all the processed values is divided by the total number of points to become the mean value.

It is important to note that the DO loop involving complex values use a special index which is only half of the normal index (three last options). The subroutine obviously requires a complex equivalenced array and the implementation of the standard complex arithmetic of FORTRAN IV.

OPERA avoids the critical situations like the ones caused by a zero divide or by the argument of a function which is out of range. The necessary IF conditions protect the critical operations, provide a standard fix-up and cause the warning index IWARN to be incremented by one. At the end of the task the number of warnings will be printed if it is greater than 0. OPERA requires the availability of the function TNICR2 which is a SEDAP adaptation of the function TNICRO. TNICR2 converts a voltage record of values comprised between 0. and 52.46 millivolts into ^{o} C ($0\ ^{o}$ C to $1300\ ^{o}$ C) according to the standard curve of a Nickel-Chromium thermocouple. If the voltage range of the function is exceeded or not reached the extreme values ($0\ ^{o}$ C or $1300\ ^{o}$ C) are provided as default values and the warning index is incremented.

2.7.3 Smoothing package (Subroutine FILTER)

The filter subroutine handles the six filter options provided for the different smoothing operations. The transfer scheme

of the program is straightforward and begins by the standard OPEIN and OPAUS calls. The succession of ADDEIN - ADDAUS calls continues until the input request has been satisfied (LKPT=0). FILTER uses the first transfer mode with overlapping capabilities which are specified by KOF and KUF. FILTER stores the values of KOF and KUF in two arrays indexed according to the value of the option parameter KFUNC. Between ADDEIN and ADDAUS, KFUNC is also used to select the appropriate smoothing subroutine.

2.7.3.1 Threepoint linear smoothing (Subroutine FIL13)

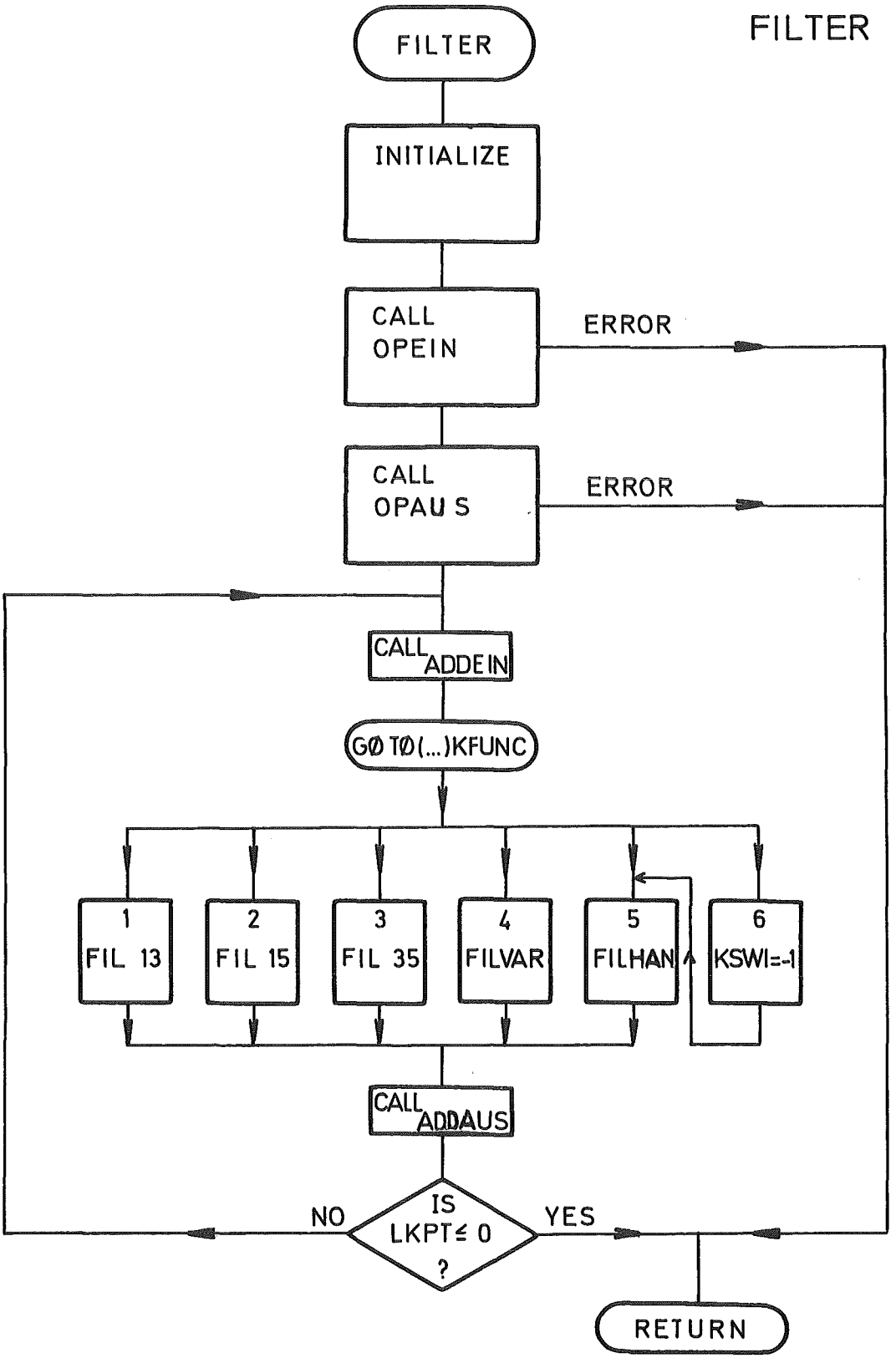
FIL13 (Filter option KFUNC = 1) computes a new record of smoothed values starting at the location XYZ(10753) from the input record starting at the location XYZ(513). FIL13 handles vectors divided into segments of 5120 points according to the following algorithm:

$$Y_1 = 0.5\ (X_1 + X_2)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Y_n = 1/3\ (X_{n-1} + X_n + X_{n+1})$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Y_1 = 0.5\ (X_{1-1} + X_1)$$

and requires an overlapping of one point in every direction (KOF = 1, KUF = 1).

The subroutine first checks if the underflow index IUF is equal to zero which means that the call is the first one (initial segment). This causes the first point to be computed according to the first formula and the computing DO loop to start with the index IANF = 2. Since this loop is only possible if at least three values are provided by the record, a smaller number of points causes the termination and an error code is issued. The upper limit of the DO loop IEND includes the overflow index IOF with the negative sign to allow the algorithm to be run up to the last point of the segment if an overflow of one value has been secured. If IOF = 0, the last point is treated according to the last line of the algorithm.

FLOWCHART SUBROUTINE
FILTER



COMMANDS:		
FIL1,FIL2,FIL3 FIL4,HAFU	FILTER computes a resulting smoothed record from an existing record according to one of the six options provided by five specific smoothing subroutines.	NAME = FILTER SYSTEM = Operators ENTRY = None

CALL FILTER(KFUNC,ENAM,GNAM,K1 ,K3,KANW,KENW,KRAF,PHEFR)

LIST OF ARGUMENTS:

KFUNC varies from 1 to 6 according to the selection of one of the six options

ENAM is the name of the input record to be smoothed

GNAM is the name of the resulting smoothed record

K1 is the search index of ENAM (invalid if K1 = -1)

K3 is the search index of GNAM (must be K1 = -1)

KANW,KENW are the delimiters of the selected segment specified in blocks

KRAF is the sorting factor to be applied to the input

PHEFR is the cut-off frequency (Hz) which must be provided for the option 4 (variable filter)

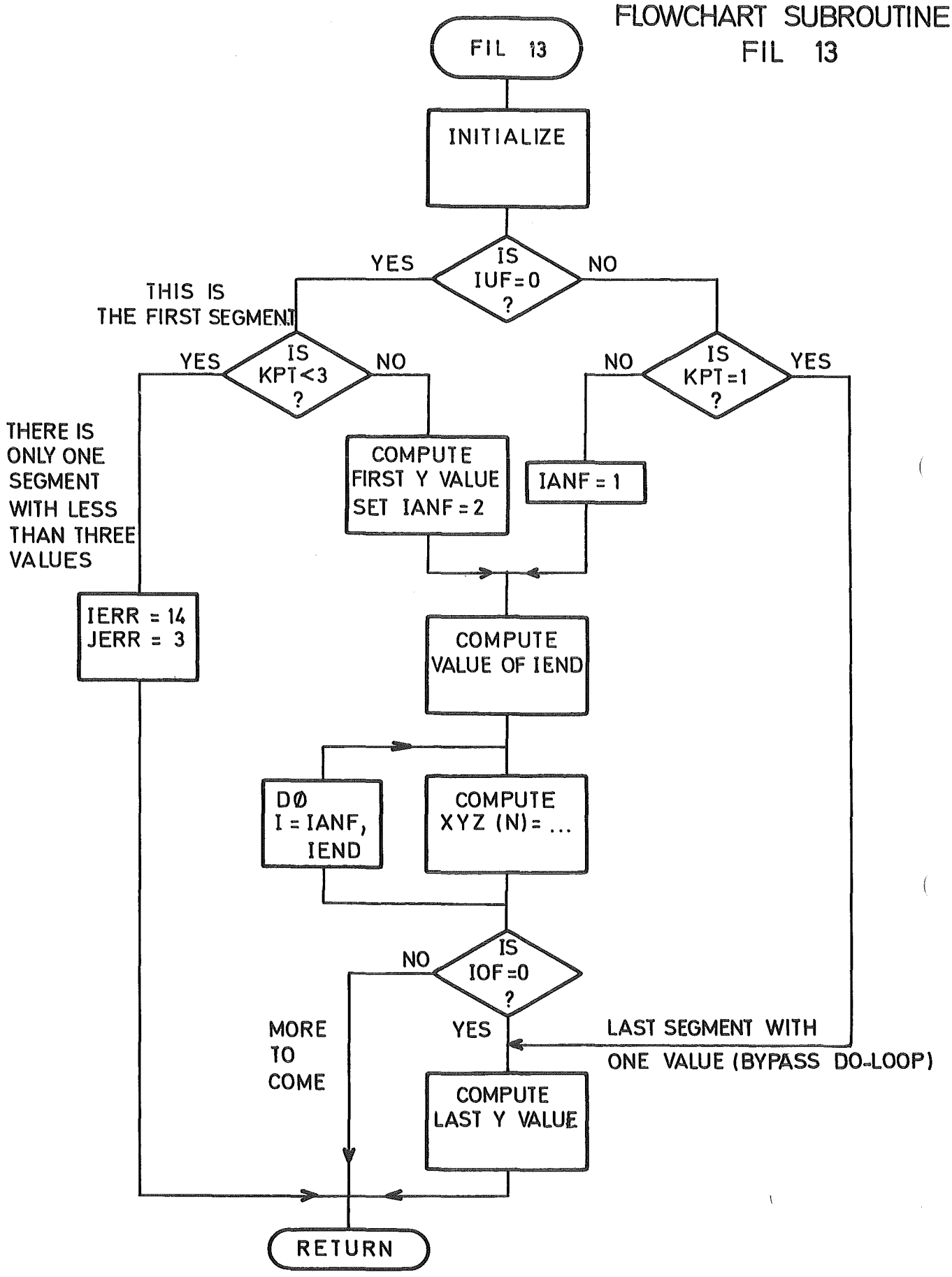
SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,OPAUS,ADDEIN,ADDAUS,FIL13, FIL15,FIL35,FILVAR,FILHAN and standard complex operations

ERRORS

DIRECT: None

INDIRECT: See OPEIN,OPAUS,ADDEIN,FIL13,FIL15,FIL35, FILVAR,FILHAN

FLOWCHART SUBROUTINE FIL 13



COMMANDS:	FIL13 generates a new record by smoothing a record already stored in the warehouse according to the hereunder described algorithm. FIL13 is written to perform the smoothing of a segmented array.	NAME = FIL13 SYSTEM = Operators ENTRY = None
See FILTER		

CALL FIL13(XYZ,KPT,IOF,IUF,IERR,JERR,KERR)

LIST OF ARGUMENTS:

XYZ is the computing array. The input values start at the location XYZ(513) and the output values at XYZ(10753)

KPT is the number of points contained in the segment

$$1 \leq KPT \leq 5120$$

IUF is the underflow index (0 for the first segment and 1 for all the further segments)

IOF is the overflow index (0 or 1)

IERR is equal to zero and will be returned as IERR = 14 (with JERR = KPT and KERR = 3) if the task involves less than three points.

Algorithm

$$Y_1 = 0.5\ (X_1 + X_2)$$
$$Y_n = 1/3\ (X_{n-1} + X_n + X_{n+1})$$
$$Y_l = 0.5\ (X_{l-1} + X_l)$$

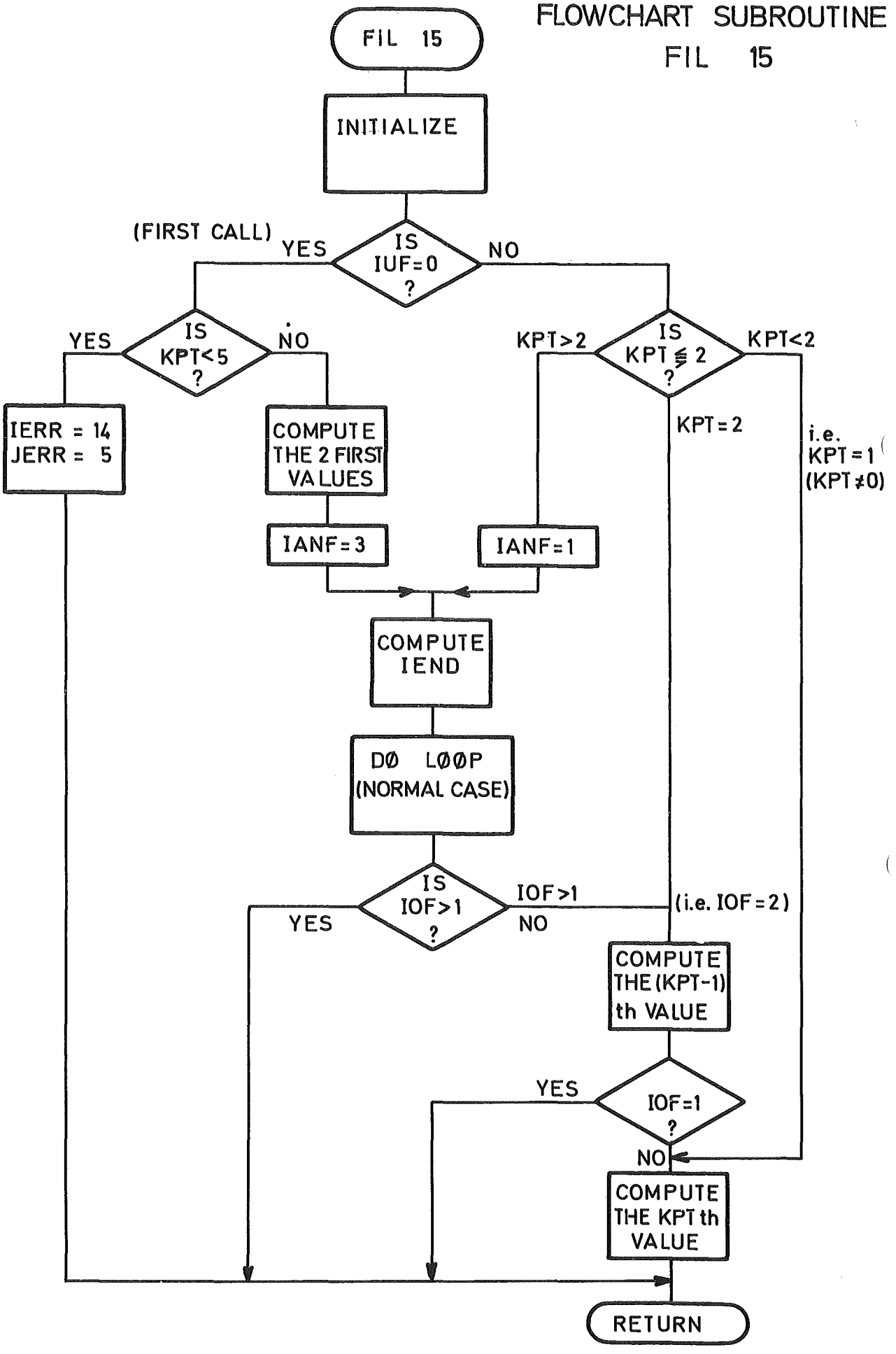
SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 14

INDIRECT: None

FLOWCHART SUBROUTINE FIL 15



COMMANDS:	FIL15 generates a new record by smoothing a record already stored in the warehouse according to the hereunder described algorithm. FIL15 is written to perform the smoothing of a segmented array.	NAME = FIL15 SYSTEM = Operators ENTRY = None
See FILTER		

CALL FIL15(XYZ,KPT,IOF,IUF,IERR,KERR)

LIST OF ARGUMENTS:

XYZ is the computing array. The input values start at the location XYZ(513) and the output values at XYZ(10753)

KPT is the number of points contained in the segment (it is not the number of points to be processed by the task)

IUF is the underflow index (0 for the first segment and 2 for all the further segments)

IOF is the overflow index (IOF = 0,1 or 2)

IERR is equal to zero as long as no error has been detected, and will be set equal to 14 (JERR = KPT and KERR = 5) if the task involves less than 5 points.

Algorithm

$$Y_1 = 0.25\ (2X_1 + X_2 + X_3)$$
$$Y_2 = 0.1\ \ (4X_1 + 3X_2 + 2X_3 + X_4)$$
$$\cdots\cdots\cdots\cdots$$
$$Y_n = 0.2\ \ (X_{n-2} + X_{n-1} + X_n + X_{n+1} + X_{n+2})$$
$$\cdots\cdots\cdots\cdots$$
$$Y_{l-1} = 0.1\ (4X_1 + 3X_{l-1} + 2X_{l-2} + X_{l-3})$$
$$Y_l = 0.25\ (2X_1 + X_{l-1} + X_{l-2})$$

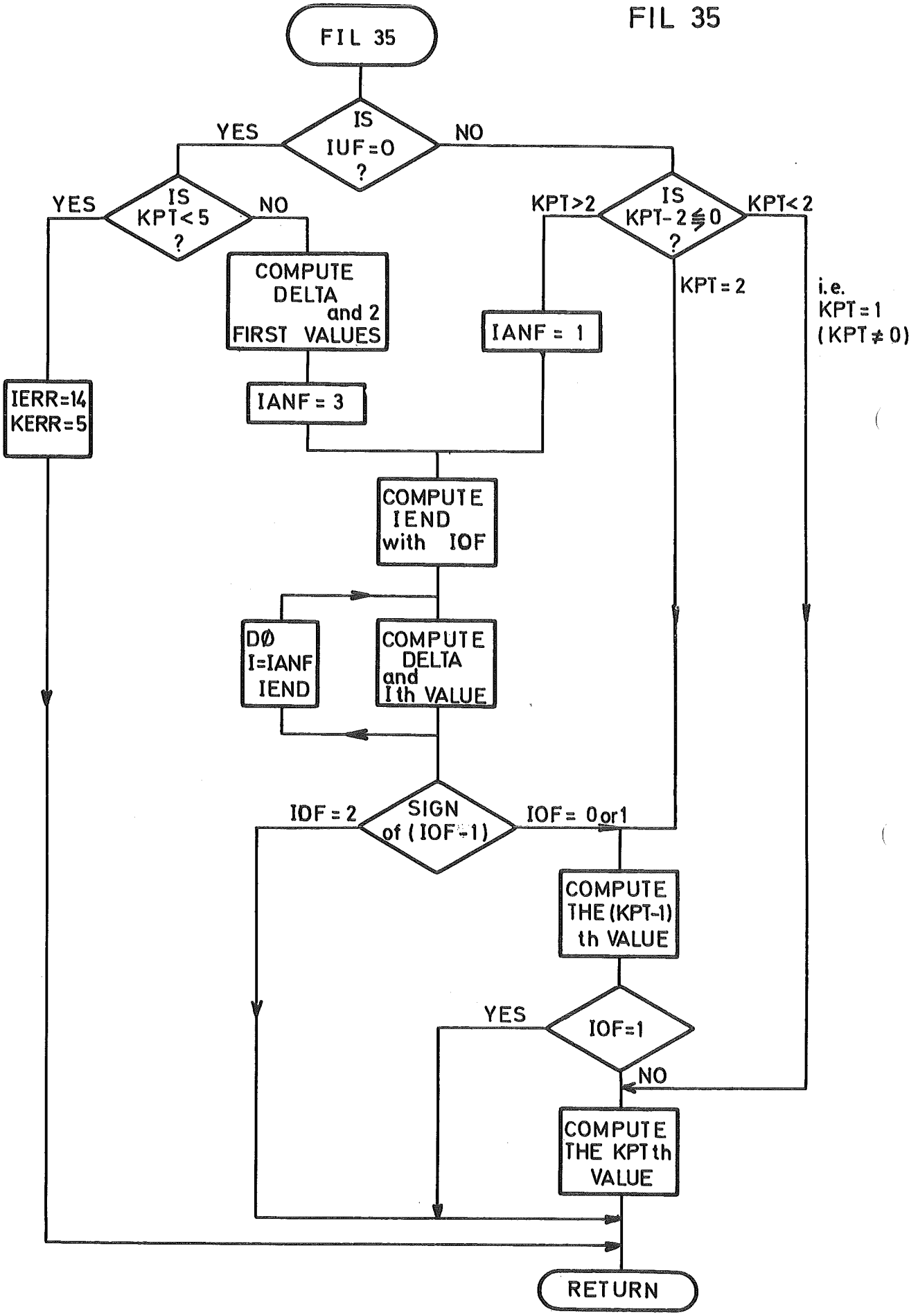
SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 14

INDIRECT: None

FLOWCHART SUBROUTINE FIL 35



COMMANDS:	FIL35 generates a new record by smoothing a record already stored in the warehouse. FIL35 evaluates the least-squares polynomial of degree 3 relevant to the five successive points for a segmented array.	NAME = FIL35 SYSTEM = Operators ENTRY = None
See FILTER		

CALL FIL35(XYZ,KPT,IOF,IUF,IERR,JERR,KERR)

LIST OF ARGUMENTS:

XYZ is the computing array. The input values start at the location XYZ(513) and the output values at XYZ(10753)

KPT is the number of points contained in the segment (it is not the number of points to be processed by a task)

IUF is the underflow index (= 0 for the first segment and 2 for all the further segments)

IOF is the overflow index (IOF = 0,1 or 2)

IERR is equal to zero as long as no error has been detected, and will be set equal to 14 (JERR = KPT and KERR = 5) if the task involves less than 5 points.

Algorithm

$$Y_1 = X_1 - 1/70\ D^4X_3$$
$$Y_2 = X_2 + 2/35\ D^4X_3$$
$$\ldots\ldots\ldots\ldots$$
$$Y_n = X_n - 3/35\ D^4X_n$$
$$\ldots\ldots\ldots\ldots$$
$$Y_{l-1} = X_{l-1} + 2/35\ D^4X_{l-2}$$
$$Y_l = X_l - 1/70\ D^4X_{l-2}$$

with $D^4X_n = X_{n-2} - 4X_{n-1} + 6X_n - 4X_{n+1} + X_{n+2}$

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 14

INDIRECT: None

Any subsequent call begins with IANF = 1 since an underflow necessarily exists and the segment has at least one point. However if there is only one point, the DO loop will be bypassed and the point will be treated as the final one.

2.7.3.2 Fivepoint linear smoothing (Subroutine FIL15)

FIL15 (filter option KFUNC = 2) computes a new record of smoothed values starting at the location XYZ(10753) from an input record starting at the location XYZ(513). FIL15 handles vectors which are divided into segments of 5210 points according to the following algorithm:

$$Y_1 = 0.25\ (2X_1 + X_2 + X_3)$$

$$Y_2 = 0.1\ (4X_1 + 3X_2 + 2X_3 + X_4)$$

.....

$$Y_n = 0.2\ (X_{n-2} + X_{n-1} + X_n + X_{n+1} + X_{n+2})$$

.....

$$Y_{l-1} = 0.1\ (4X_1 + 3X_{l-1} + 2X_{l-2} + X_{l+3})$$

$$Y_l = 0.25\ (2X_1 + X_{l-1} + X_{l-2})$$

and requires an overlapping of two points in both directions (KOF = 2, KUF = 2).

The subroutine first checks if the underflow index IUF is equal to zero which indicates that the call is the first one (initial segment). Since a minimum of five points is required, the number of points is checked and an error code will be issued if this condition is not satisfied. The two first values are computed and the DO loop first index is set equal to three. The second index of the DO loop IEND is calculated by adding the expression (IOF-2) to the number of points of the segment (KPT). This expression allows the central part of the algorithm to be run up to the last point of the segment (10752 + 5120) if an overflow of two values has been secured by the transfer subsystem. If the underflow is not present or if the array is not full, the DO loop will be run only until the (KPT-2)th value in order to allow the handling of the two last values by the final form of the algorithm. The value of IOF is also used after

the DO loop to select one of the three possible cases:

a) At least two values are left and the overflow contains two values. The two last lines are skipped and the control is passed back to FILTER for the next transfer.
b) There is only one value in the overflow area. The first of the two last formulas is used and the last point will require a subsequent call with only one point.
c) There is no overflow and the two last points are considered as the two final points.

Any subsequent call will involve a test of the KPT value to select one of the three possibilities:

a) KPT is larger than 2. The DO loop is run with IANF=1 since the underflow is always 2 after the first segment.
b) KPT = 2. The DO loop is bypassed and the two values are treated by the two last lines of the algorithm.
c) KPT = 1 (this is deducted from the fact that if KPT is less than 2 only the value 1 is possible). The last line of the algorithm is used.

2.7.3.3 Fivepoint cubical smoothing (Subroutine FIL35)

FIL35 (filter option KFUNC = 3) computes a new record of smoothed values starting at the location XYZ(10753) from an input record starting at the location XYZ(513). FIL35 handles vectors divided into segments of 5120 points by evaluating the least-squares polynomial of degree 3 relevant to the five successive points according to the following algorithm:

$$Y_1 = X_1 - 1/70\ D^4X_3$$
$$Y_2 = X_2 + 2/35\ D^4X_3$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Y_n = X_n - 3/35\ D^4X_n$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$Y_{l-1} = X_{l-1} + 2/35\ D^4X_{l-2}$$
$$Y_l = X_l - 1/70\ D^4X_{l-2}$$
$$\text{with } D^4X_n = X_{n-2} - 4X_{n-1} + 6X_n - 4X_{n+1} + X_{n+2}$$

The program structure of FIL35 is almost identical to FIL15 (KOF=2, KUF=2) with the exception of the delta value (D^4X_n) which is computed for the initialization and inside the DO loop.

2.7.3.4 Variable cut-off-frequency filter (Subroutine FILVAR)

The subroutine FILVAR (Filter option KFUNC = 4) computes a new record of smoothed values starting at the location XYZ(10753) from the input record starting at the location XYZ(513). FILVAR is written to handle segmented arrays according to the following algorithm which simulates a first order low-pass filter:

$$Y_1 = X_1$$

$$Y_n = Y_{n-1} \frac{(2T - t)}{(2T + t)} + \frac{t}{(2T + t)} (X_n + X_{n-1})$$

with t = dt = 1./freq
and T = Tau = time constant

FILVAR (variable filter) differs from the other smoothing subroutine by the fact that the effect of the filter does not depend only from the sampling frequency of the record but also from a cut-off frequency which can be adjusted and which must be supplied by the user. This is analog to the time constant setting of an RC network used as filter. Since the computation of this relation involves a feedback effect, FILVAR checks if the proposed cut-off frequency is compatible with the sampling frequency of the input record. Any cut-off frequency which exceeds this limit would cause numerical instability and will cause an interruption of the task with an error code IERR = 15.

FILVAR handles the segmented arrays without using the overlapping features of the TRANSFER subsystem. The initialization is detected when the switching index ISW has been found equal to -1, this causes the initial relation to be used and the DO loop is started with I = 2. Otherwise the loop starts with IANF = 1 since FILVAR always stores the last value of X and the last value of Y before the RETURN to FILTER is executed.

2.7.3.5 Smoothing of spectra (Subroutine FILHAN)

The subroutine FILHAN (filter options KFUNC = 5 or 6) computes a record of smoothed complex values starting at the location XYZ(5377) from a complex input record starting at the location XYZ(257). FILHAN handles vectors which are divided into segments of 5120 points (i. e. 2560 complex values) according to one of the following algorithms which are known as Hanning's method of smoothing:

Algorithm 1	Algorithm 2
$Y_1 = 0.5\ (X_1 + X_2)$	$Y_1 = 0.5\ (X_1 - X_2)$
.....	.....
$Y_n = 0.25\ (X_{n-1} + 2X_n + X_{n+1})$	$Y_n = 0.25\ (-X_{n-1} + 2X_n - X_{n+1})$
.....	.....
$Y_1 = 0.5\ (X_1 + X_{1-1})$	$Y_1 = 0.5\ (X_1 - X_{1-1})$

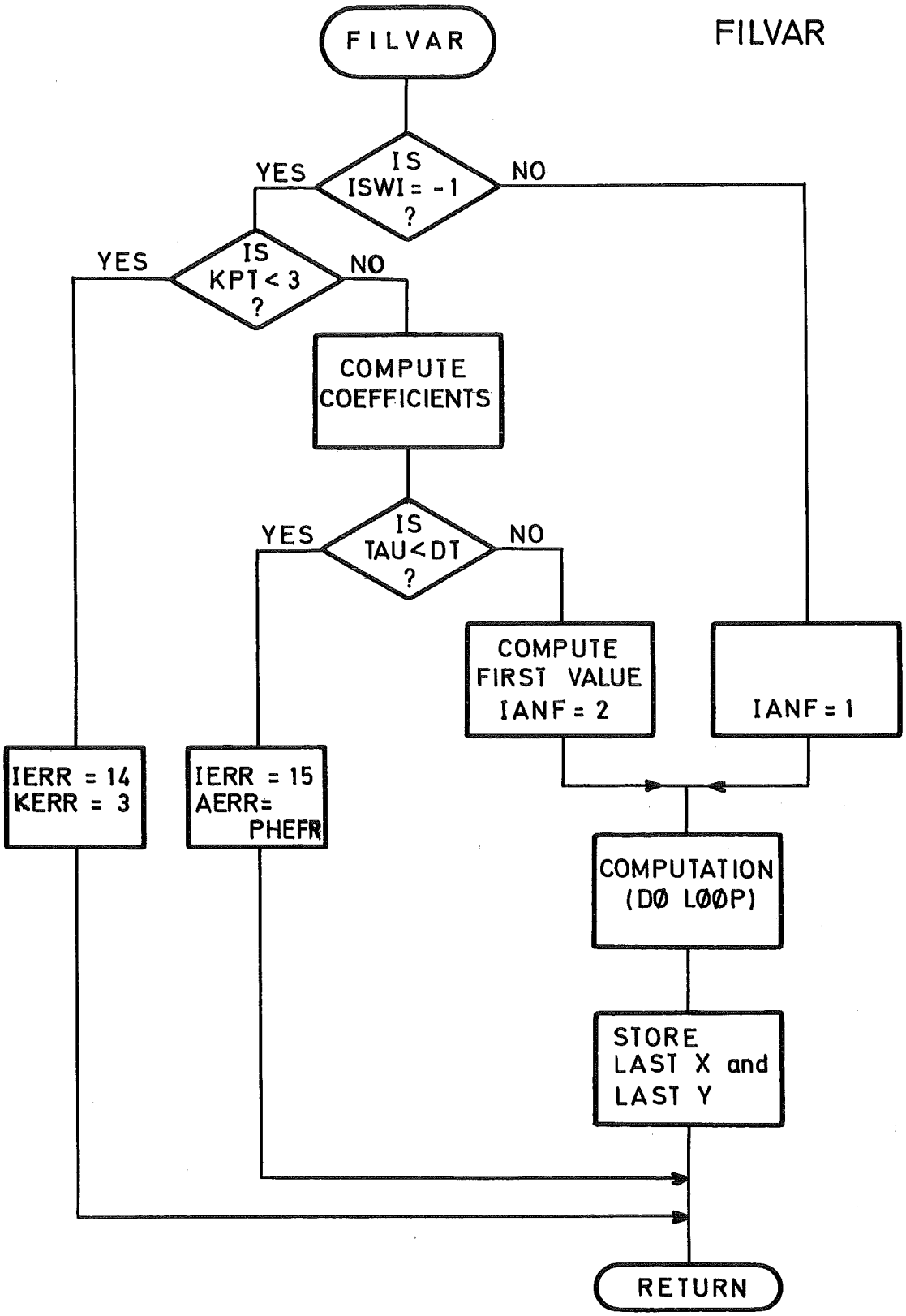
The choice between the two algorithms is made in the MAIN according to the sign of the first decimal value DEZ(1) and is passed to FILTER by the option index KFUNC (5 or 6). KFUNC is used to determine the sign of KSWI which selects one of the two algorithms in FILHAN.

The structure of the subroutine is almost similar to the structure of FIL13 which was explained before but some differences exist and are due to the following reasons:

- Since FILHAN handles complex values, the displacements are computed for the CXYZ array which is complex equivalent of the XYZ array. The DO loop indexes are also reduced accordingly.
- FILHAN requires an overlapping of one value in both directions but since the values are complex and since the TRANSFER subsystem handles a complex value as two adjacent REALx4, the two overlapping parameters KOF and KUF are equal to 2.

The subroutine checks the value of the option index KSWI to determine the sign of the parameter HALF needed to compute the first value. The absence of underflow (IUF = 0) indicates that the segment is the first segment and the first value will be-

FLOWCHART SUBROUTINE FILVAR



COMMANDS:	FILVAR generates a new record by smoothing a record already stored in the warehouse. FILVAR is written to handle segmented arrays and has the same properities as a 1st order filter (RC type) with variable cut-off frequency	NAME = FILVAR SYSTEM = Operators ENTRY = None
See FILTER		

CALL FILVAR(XYZ,KPT,ISW,FREQ,PHEFR,IERR,JERR,AERR,KERR)

LIST OF ARGUMENTS:

XYZ is the computing array. The input values start at the location XYZ(513) and the output values at XYZ(10753)

KPT is the number of points contained in a segment (it is not the number of points to be processed by the task)

ISW is equal to -1 for the first segment (initialization) and is equal to 1 for the following segments

FREQ is the frequency of the input record and corresponds to the sampling frequency

PHEFR is the cut-off frequency of the filter simulated by FILVAR

IERR,JERR, AERR and KERR are the error parameters.

Algorithm

$$Y_1 = X_1$$

$$Y_n = C_1 \times X_{n-1} + C_2 \times (X_n + X_{n-1})$$

with $C_1 = \frac{2T - t}{2T + t}$ and $C_2 = \frac{t}{2T + t}$

T = TAU = Time constant of the filter

t = 1.0 / FREQ

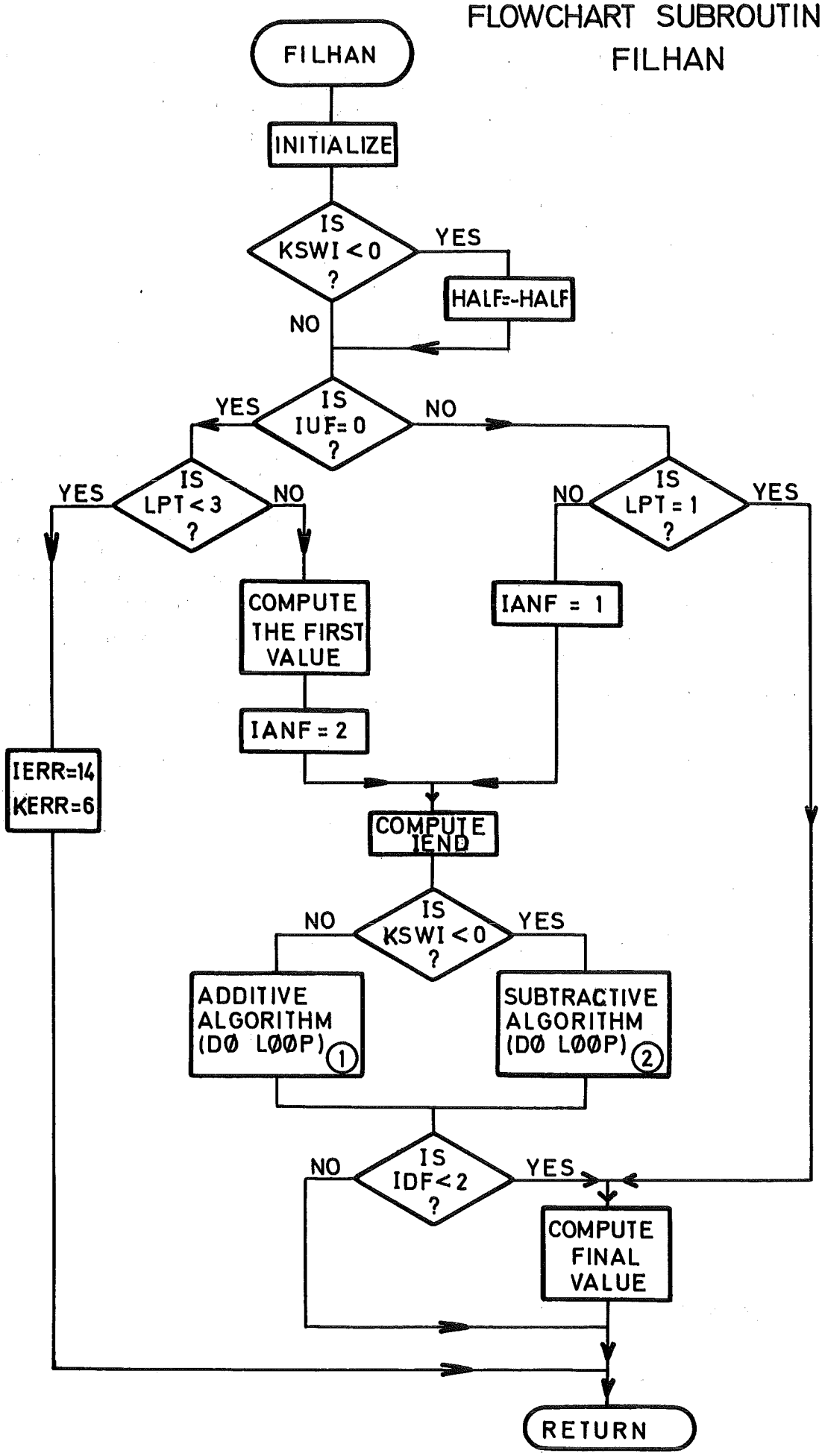
SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 14,15

INDIRECT: None

FLOWCHART SUBROUTINE FILHAN



COMMANDS:	FILHAN generates a complex record by smoothing a complex record already stored in the warehouse. FILHAN uses the Hanning's method and handles segmented arrays. The index KSWI is used to select one of the two possible algorithms.	NAME = FILHAN SYSTEM = Operators ENTRY = None
See FILTER		

CALL FILHAN(CXYZ,KPT,IOF,IUF,KSWI,IERR,JERR,KERR)

LIST OF ARGUMENTS:

CXYZ is the array equivalent to XYZ but used as complex array. The complex input values start at CXYZ(257) and the output values at CXYZ(5377)

KPT is the number of values contained in the segment and is equal to twice the number of complex values

IOF is the overflow index (IOF is equal to zero or to 2 if one complex value is stored in the overflow area, IOF = 1 is excluded)

IUF is the underflow index (IUF = 0 or 2)

KSWI selects one of the two possible algorithms
KSWI = 1 for the addition of the lateral values
KSWI = -1 for the subtraction of the lateral values

IERR is equal to zero as long as no error has been detected and will be set equal to 14 (JERR = KPT and KERR = 6) if the task involves less than 3 complex values (6 REAL x 4)

Algorithm 1	Algorithm 2
$Y_1 = 0.5\ (X_1 + X_2)$	$Y_1 = 0.5\ (X_1 - X_2)$
.....	.....
$Y_n = 0.25\ (X_{n-1} + 2X_n + X_{n+1})$	$Y_n = 0.25\ (-X_{n-1} + 2X_n - X_{n+1})$
.....	.....
$Y_1 = 0.5\ (X_1 + X_{1-1})$	$Y_1 = 0.5\ (X_1 - X_{1-1})$

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 14

INDIRECT: None

computed if at least three values are provided (this means six values for the transfer) and the first index of the DO loop is taken as IANF = 2. The second index of the DO loop is computed with the half IOF value to allow the algorithm to reach the end of the segment if a complex value has been secured in the overflow area (IOF = 2). One of the two available DO loops is selected according to the value of KSWI and after completion of the loop the last value will be computed as the final value if IOF is less than 2 (once again, IOF can be only 0 or 2 if the value is complex). Any subsequent call will run the DO loop with IANF = 1 but if there is only one complex value (LPT = 1) the loop will be bypassed and the last value will be computed as the final point.

2.7.4 Differentiation and integration (Subroutine DIFINT)

The subroutine DIFINT differentiates or integrates a record and stores the resulting record into the warehouse. DIFINT follows the standard transfer scheme with the overlapping features and that implies that the records are transferred by segments of 5120 points with the input values beginning at the location XYZ(513) and the output values at XYZ(10753). DIFINT controls four possible options which are executed by three special subroutines (TRAP, SIMP, DIF3).

1- Integration with reset of the integral to a preset level whenever the control record crosses over a specified threshold. This operation is also called "Integration with Switch".
2- Integration of a record by the trapezoidal rule.
3- Integration of a record by the Simpson's rule.
4- Differentiation of a record.

For the first option, the transfer of a second record is necessary and requires a second OPEIN call. Since the two OPEIN calls check only the validity of the delimiters given in blocks, the synchronism of the two records is verified. If the two sets of parameters brought back by the OPEIN calls are not identical (number of points, sampling frequency, time origin)

the two records are not synchronous and a warning is printed to report the fact. The values are transferred by ADDEIN and stored by ADDAUS after the total number of points LKPT has been processed. The intermediary computation depends upon the selected command and is directly done by one of the specific subroutines excepted for the first option where the supervision of DIFINT requires a detailed control of the second input record.

2.7.4.1 Integration with Switch

The integration with switch was implemented to allow an easier integration of pressure pulses recorded during some experiments performed with a sodium testing station. The integration can then be reset between two pulses. Such a situation is well known and arises when one integrates a sine wave signal. The results of the integration will be easier to follow if the integration is reset every time the signal crosses the zero-line, i. e. if the half waves are integrated separately. This features requires the storage of the switching record which can be the same record as the integrated record or any other record provided by the user.

2.7.4.2 Integration by the trapezoidal rule (Subroutine TRAP)

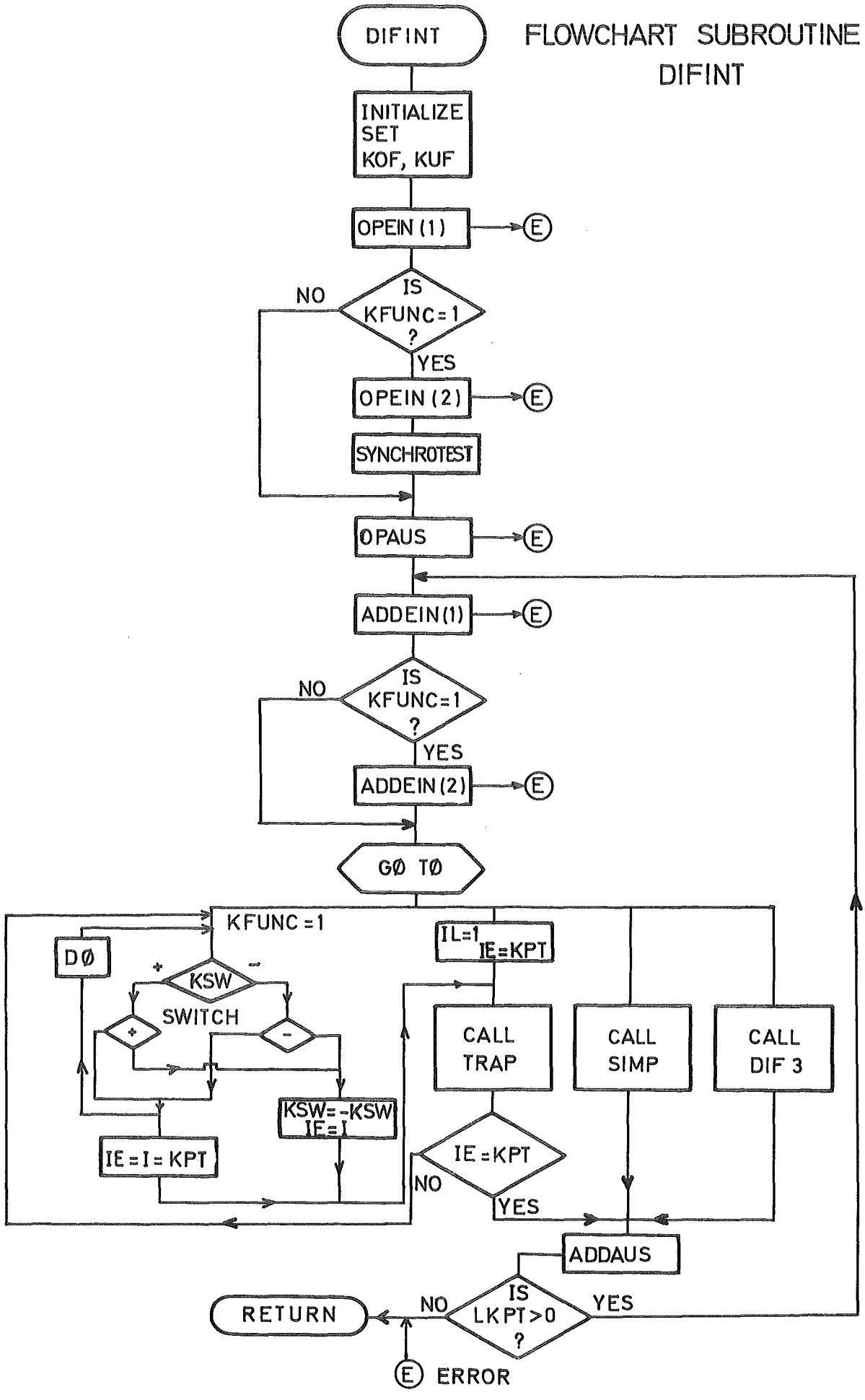
TRAP integrates a record segment which is delimited by the two parameters IL and IE. The subroutine TRAP was written to perform the integration of a record for the switching case as well as for the normal case. The integration is performed according to the following algorithm:

$$Y(1) = 0,\ Y(I) = Y(I-1) + 0.5\ (X(I-1) + X(I))/FREQ$$

and the first value of a segment is selected according to the value of the parameter ISWI.

If ISWI = -1, the call is the first one and the first value will be set equal to zero.

If ISWI = 0, the call is a subsequent call which has not been caused by the switch-interruption. In that case the overlapping feature is applied and the integration restarts by using XLAST and YLAST which are the two last values stored


DIFINT
FLOWCHART SUBROUTINE DIFINT
INITIALIZE SET KOF, KUF
OPEIN (1)
E
NO
IS KFUNC=1 ?
YES
OPEIN (2)
E
SYNCHROTEST
OPAUS
E
ADDEIN(1)
E
NO
IS KFUNC=1 ?
YES
ADDEIN(2)
E
GØ TØ
KFUNC=1
DØ
+
KSW
-
SWITCH
+
-
KSW=-KSW IE=1
IE=I=KPT
IL=1 IE=KPT
CALL TRAP
CALL SIMP
CALL DIF 3
IE=KPT
NO
YES
ADDAUS
IS LKPT>0 ?
NO
YES
RETURN
E
ERROR

COMMANDS:	DIFINT generates a new record in the warehouse by integrating or differentiating a record already stored in the warehouse. DIFINT uses three special subroutines to handle the segmented arrays.	NAME = DIFINT SYSTEM = Operators ENTRY = None
INSW,INTR, INSI,DIFF		

CALL DIFINT(KFUNC,ENAM,FNAM,GNAM,K1,K2,K3,KANW,KENW,KRAF,SWITCH,
RESET)

LIST OF ARGUMENTS:

KFUNC is the option index with the following code:
KFUNC = 1 integration with Switch (trap. rule)
KFUNC = 2 integration by the trapezoidal rule
KFUNC = 3 integration by the Simpson's rule
KFUNC = 4 differentiation

ENAM is the name of the input record which must be integrated or differentiated

FNAM is the name of the second input record which provides the switching record for KFUNC = 1

GNAM is the name of the resulting output record

K1,K2,K3 are the search indexes of ENAM,FNAM and GNAM. K1 and K2 must be greater than 0 and K3 must be -1

KANW and KENW are the delimiters (in blocks) which delimit the selected segment of the input records

KRAF is the sorting factor applied to the input

SWITCH is the value of the threshold which causes an interruption of the integration whenever it has been crossed over by the switching function (KFUNC = 1)

RESET stores the value to which the integral must be reset after a SWITCH interruption. (KFUNC = 1)

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN,ADDEIN,OPAUS,ADDAUS,TRAP,SIMP and DIF3

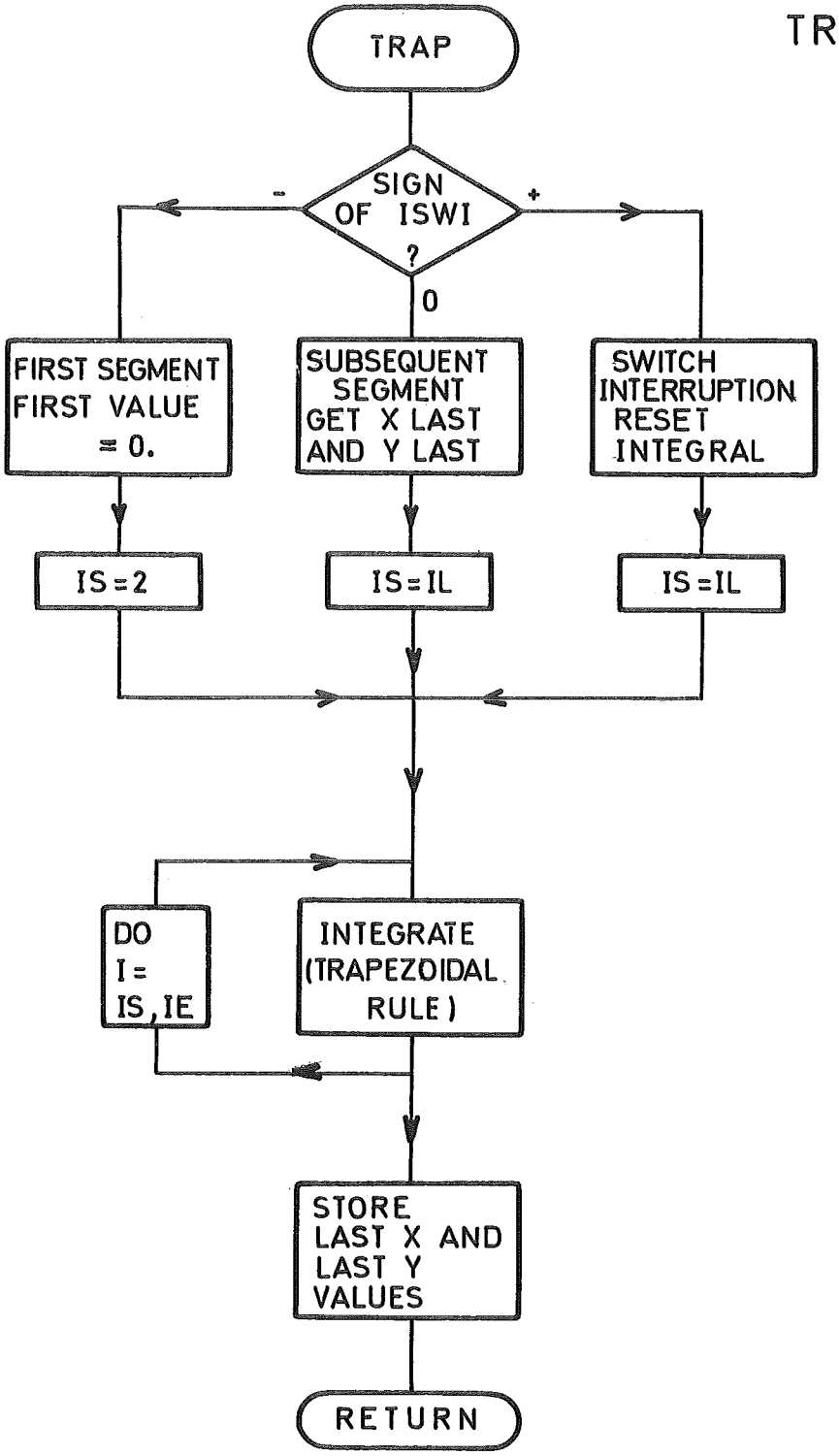
ERRORS

DIRECT: None, but warning is printed if the two input records are not synchronous

INDIRECT: See OPEIN,ADDEIN,OPAUS,SIMP,DIF3

FLOWCHART SUBROUTINE

TRAP



COMMANDS: See DIFINT	TRAP performs the integration of an equidistantly tabulated record by the trapezoidal rule. TRAP is written to perform the integration of a segmented array.	NAME = TRAP SYSTEM = Operators ENTRY = None

CALL TRAP(XYZ,IL,IE,ISWI,FREQ,RESET)

LIST OF ARGUMENTS:

XYZ is the computing array. The input values start at the location XYZ(513) and the output values are XYZ(10753)

IL is the position of the first value to be integrated. IL=1 when the first value is located at XYZ(513).

IL and IE delimit the part of the array which must be integrated by TRAP. In the normal case and for a filled array IL = 1 and IE = 5120 but with a switched integration IL and IE can take any value within these two limits.

ISWI is -1 for the first call
is 0 for a subsequent call
is +1 for a call due to a switch interrupt and causes the value to be reset

FREQ is the sampling frequency used to compute the interval h

RESET stores the value to which the integral has to be reset after a switch interruption.

Algorithm

$$Y_1 = 0.0 \quad \text{or} \quad Y_i = \text{RESET}$$

$$Y_n = Y_{n-1} + \frac{h}{2}(X_{n-1} + X_n)$$

$$h = 1/\text{FREQ}$$

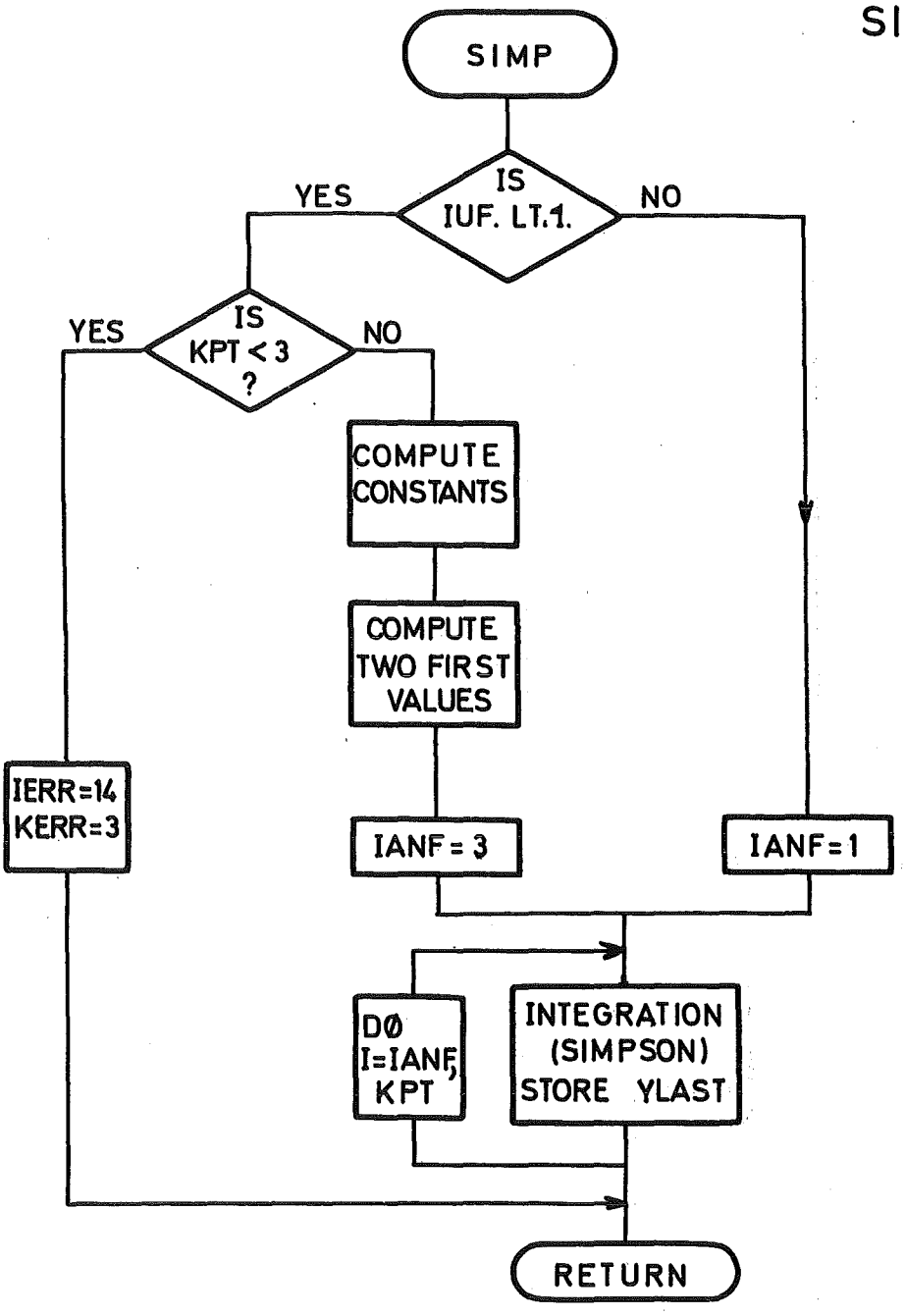
SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 14

INDIRECT: None

FLOWCHART SUBROUTINE SIMP



COMMANDS: See DIFINT	SIMP generates a new record by integrating a record already stored in the warehouse according to the Simpson's rule. SIMP is written to handle the segmented arrays of SEDAP.	NAME = SIMP SYSTEM = Operators ENTRY = None

CALL SIMP(XYZ,KPT,FREQ,IUF,IERR,JERR)

LIST OF ARGUMENTS:

XYZ is the computing array. The input values start at the location XYZ(513) and the output values at XYZ(10753)

KPT is the number of points contained in the segment (it is not the number of points to be processed by the task)

FREQ is the sampling frequency used to calculate the time interval h = 1/FREQ

IUF is the underflow index (IUF is 0 for the first segment and 1 for any of the following segments)

IERR is equal to zero as long as no error has been detected and will be set equal to 14 if the task involves less than 3 values.

Algorithm

$$Y_1 = 0.$$
$$Y_2 = h/3\ (1.25\ X_1 + 2X_2 - 0.25X_3)$$
$$Y_n = Y_{n-1} + h/6\ (X_{n-2} + 4X_{n-1} + X_n)$$

SUBROUTINES OR FUNCTIONS NEEDED: None

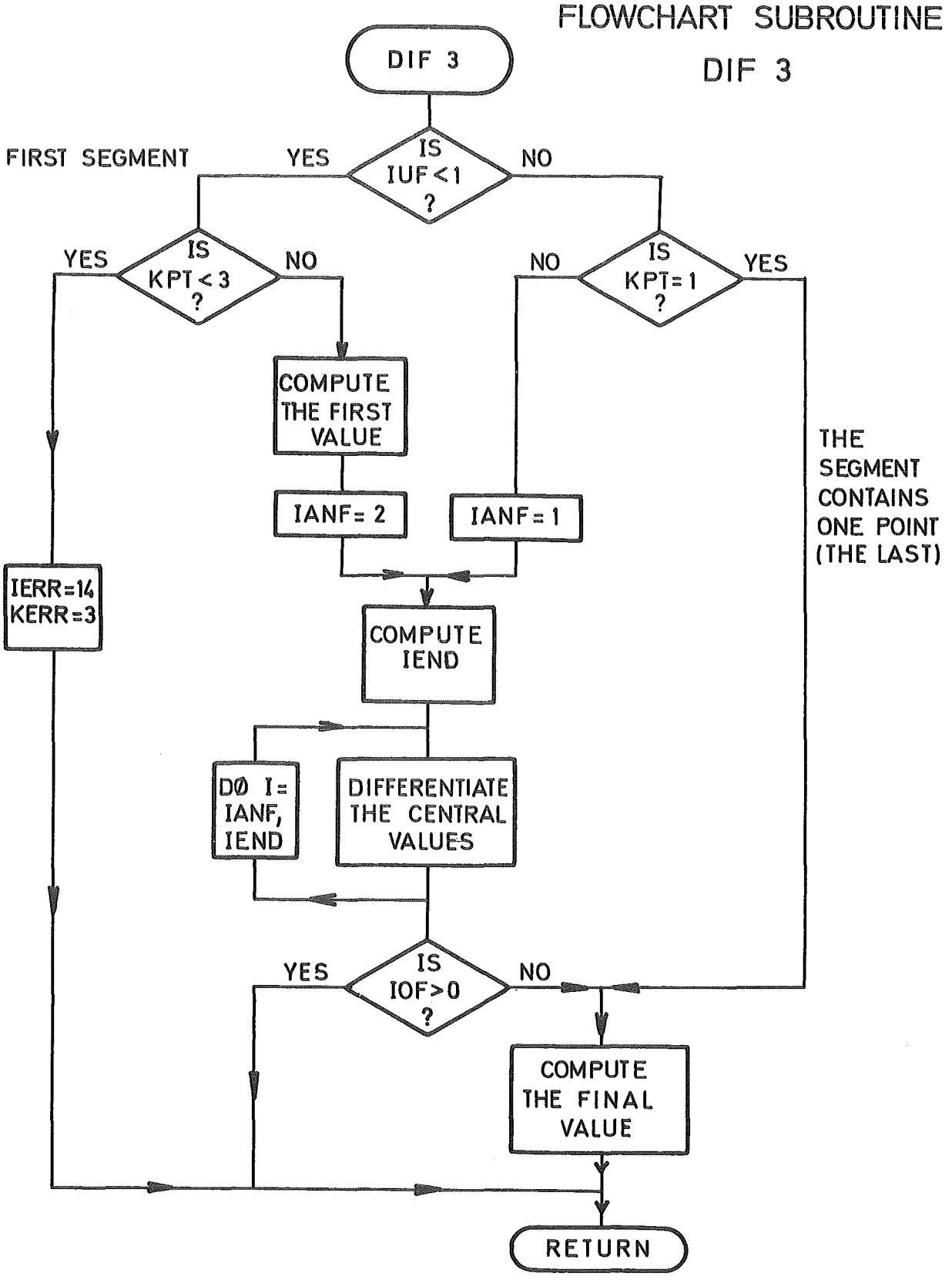
ERRORS

DIRECT: 14

INDIRECT: None

FLOWCHART SUBROUTINE

DIF 3



COMMANDS:	DIF3 generates a new record in the warehouse by differentiating a record already stored in the warehouse. DIF3 was written to handle segmented arrays.	NAME = DIF3 SYSTEM = Operators ENTRY = None
See DIFINT		

CALL DIF3(XYZ,KPT,FREQ,IOF,IUF,IERR,JERR)

LIST OF ARGUMENTS:

XYZ is the computing array. The input values start at the location XYZ(513) and the output values at XYZ(10753)

KPT is the number of points contained in the segment.(It is not the number of points to be processed by the task)

FREQ is the sampling frequency of the input and is used to compute the interval h = 1/FREQ

IUF is the underflow index (0 for the 1st segment and 1 after the following)

IOF is the overflow index (= 0 or 1)

IERR is equal to zero as long as no error has been detected and will be set to 14 if the task involves less than three points

JERR is an other argument for the error interpretation

Algorithm

$$Y_1 = 1/h\ (X_2 - X_1)$$
$$Y_n = 1/2h\ (X_{n+1} - X_{n-1})$$
$$Y_1 = 1/h\ (X_1 - X_{1-1})$$

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS

DIRECT: 14

INDIRECT: None

by the previous TRAP call. (TRAP does not use the overlapping feature of the TRANSFER subsystem).

If ISWI = 1 the interruption has been caused by the switch (or by both since SWITCH has the priority) and the first value of the segment is equal to RESET (default value = 0.)

2.7.4.3 Integration by the Simpson rule (Subroutine SIMP)

SIMP performs the integration of a record according to the Simpson's rule and handles segmented arrays passed by DIFINT. The following algorithm is used:

$Y_1 = 0.$

$Y_2 = h/3\ (1.25\ X_1 + 2X_2 - 0.25\ X_3)$

$Y_n = Y_{n-1} + h/6\ (X_{n-2} + 4X_{n-1} + X_n)$

h = 1/FREQ is the sampling time interval.

The subroutine verifies the presence of an underflow (index IUF) and if no underflow is present initializes the integration with the two first lines of the algorithm. If the task does not provide a minimum of three values the execution is not allowed. The first SIMP call must begin the computation of the DO loop with the first index IANF = 3 and when the DO loop is terminated the last Y value must be stored since the system does not have an underflow feature for the output value. Any subsequent call can start with IANF = 1, since the transfer has secured the two last X values.

2.7.4.4 Differentiation (Subroutine DIF3)

DIF3 performs the differentiation of a record passed by DIFINT and uses the same conventions for the values of the displacements along the XYZ array. DIF3 uses the standard overlapping features provided by the TRANSFER subsystem and specified in DIFINT with one extra value stored in both directions (KUF = 1 and KOF = 1). The differentiation is performed according to the following algorithm:

$Y_1 = 1/h\ (X_2 - X_1)$

.....

$$Y_n = 1/2h\ (X_{n+1} - X_{n-1})$$
.....
$$Y_1 = 1/h\ (X_1 - X_{1-1})$$

The first point is calculated according to the first line of the algorithm if IUF = 0 (first call). The task must involve at least three points otherwise the task is rejected with an error code IERR = 14. The first call initializes the DO loop with a first index IANF = 2 and includes the value of IOF (overflow index) in the computation of the second index IEND in such a way that the last line of the algorithm will be used for the last point of the segment if a value has not been secured in the overflow area. Any subsequent call will initialize the DO loop with IANF = 1 but if the call involves only one point the DO loop will be bypassed and the value will then be computed by the last line of the algorithm.

2.8 The FOURIER package

Many engineering and scientific problems require the treatment of experimental records not only in the time domain but also in the frequency domain. The standard method to pass from one of these domains to the other involves a Fourier transform or its reciprocal form which is called the Fourier antitransform. Since a few years the Fourier methods are more widely used and SEDAP presents the great advantage to offer a completely integrated Fourier package based upon the methods of the Fast Fourier Transform (FFT). The SEDAP Fourier package can be divided into two parts. The first part contains almost all the elements which are necessary to perform the different Fourier operations in many modular combinations. All the intermediary steps are separately programmed, but the Fourier transform is limited to 16 blocks or to 8 K values. The second part is centered around the subroutine MEPODE and is oriented toward a very effective evaluation of power density spectra. The efficiency is due to the fact that the operations are not slowed down by intermediary transfers to the warehouse and the overlapping techniques allow to process the very long records

which are not uncommon in the experiments involving noise analysis. It must be noted that the two parts can share some of the facilities of the package and that the user can combine the computations in order to perform all the standard tasks like auto- and cross-correlation, convolution integrals, determination of the various power spectra etc. ...

2.8.1 The algorithm of the Fast Fourier Transform (FFT)

Recently Cooley and Tuckey /6 - 10/ have devised an algorithm, which is called Fast Fourier Transform, whereby the sum of the form

$$x_k = \sum_{j=0}^{N-1} a_j \cdot \exp(2\pi i \cdot j \cdot k/N) \quad \text{and its inverse}$$

$$a_j = \sum_{k=0}^{N-1} x_k \cdot \exp(-2\pi i \cdot j \cdot k/N)$$

can be computed considerably more rapidly than by previous techniques provided $N = 2^M$ and M is an integer. Library subroutine programs to evaluate the sum have been written and one, available by IBM (FOUR1) /11/ was implemented in SEDAP. There are at least two somewhat different algorithmic approaches to implementing the Fast Fourier Transform, one due to Cooley and Tuckey and another programmed by Stockham and Forman /8, 9/. The Cooley-Tuckey algorithm was chosen because it needs only half of the storage places than that of Stockham-Forman, although it needs about 30 % more computing time. The subroutine FOUR1 may be used to perform a Fourier Transform or a Fourier Anti-transform. It uses one-dimensional complex arrays DATA(J), whose length N is a power of two. The discrete Fourier Transform defined by the summing equation above may be expressed with the following FORTRAN like written relation.

$$\text{TRANS(K)} = \sum_{J=1}^{N} \text{DATA(J)} \cdot \text{EXP(ISIGN} \cdot 2\pi i \cdot \text{(J-1)} \cdot \text{(K-1)/N)}$$

for all K from 1 to N. DATA(J) is a complex array, where real

and imaginary parts are stored adjacently. ISIGN is an option indicator and is equal to +1 for a Fourier Transform or to -1 for a Fourier Antitransform. If the input DATA(J) represents time-intervals equal to (J-1)•T, then the transform-values TRANS(K) correspond to the complex amplitudes at frequencies (K-1)•F with F = 2π/(N•T).

By periodicity, all frequencies above the "foldover frequency" π/T may be identified by a negative frequency reduced by an amount equal to 2π/T.About the algorithm see /6/, a special issue on the Fast Fourier Transform.

In comparing former non FFT-Methods /6 - 10/ with the FFT the time saving is expressed by the fraction $\log_2 N/N$. As an example, if there are 2^{10} values to be transformed the FFT is about 100 times faster than primitive transformation methods. Gentleman and Sande have shown /10/ that the FFT is the most accurate Fourier transform method. Their upper bound of the root mean square error is:

$$\varepsilon = 6 \cdot \sqrt{2} \cdot \log_2 N/2^b,$$

where b is the number of bits in the floatingpoint fraction.

Some preparations are needed to use FOUR1.

2.8.2 Implementation of the FFT in SEDAP (Subroutine FOUR)

The subroutine FOUR performs the Fourier transform or anti-transform of a record whose length must not exceed 16 blocks. The coefficients are normalized according to the standard conventions in such a way that the resulting value of a pure sine wave of amplitude A will be A/2. The transfer of the input record is performed by the TRANSFER subsystem and uses the standard subroutines OPEIN and ADDEIN. The subroutine verifies if the number of points is of the 2^N form and eventually fills the last part of the array with zeroes to meet this condition. The computations of FOUR vary slightly if the task specifies a Fourier transform or antitransform.

COMMANDS:	FOUR1 performs the Fast Fourier Transform (FFT) according to the algorithm of Cooley-Tuckey	NAME = FOUR1 SYSTEM = Fourier Package ENTRY = None
None		

CALL FOUR1(DATA,N,ISIGN)

LIST OF ARGUMENTS:

DATA is a complex array, equivalent to the XYZ-array (real and imaginary parts are adjacent in storage)

N is the number of points contained by the array time of computation. The length must be N = 2**M(M > 0, Integer)

ISIGN is an option indicator.
ISIGN = -1 for the Fourier transform (time → frequency)
ISIGN = +1 for the Fourier antitransform (frequency → time)

Algorithm: see 2.8.1

Remark:
This subroutine is written by N. Brenner of MIT Lincoln Laboratory and submitted by IBM (Program Order Number 360D.13.4.002)

SUBROUTINES OR FUNCTIONS NEEDED: SIN

ERRORS

DIRECT: None

INDIRECT: None

2.8.2.1 Fourier Transform

Since in most frequency analysis tasks the mean value is only of minor interest, SEDAP suppresses the amplitude at frequency 0. If, however, the user is interested in the mean value, he may obtain this information from the appropriate SEDAP command.

If the time signal contains less than 2^n sample values (n is integer) the signal is padded with zeroes up to the next higher number of this form (these extended records are called hyper-arrays) /12/.

Finally, before entering the transform algorithm the time signalrecord is converted from real to complex values (with zero imaginary part).

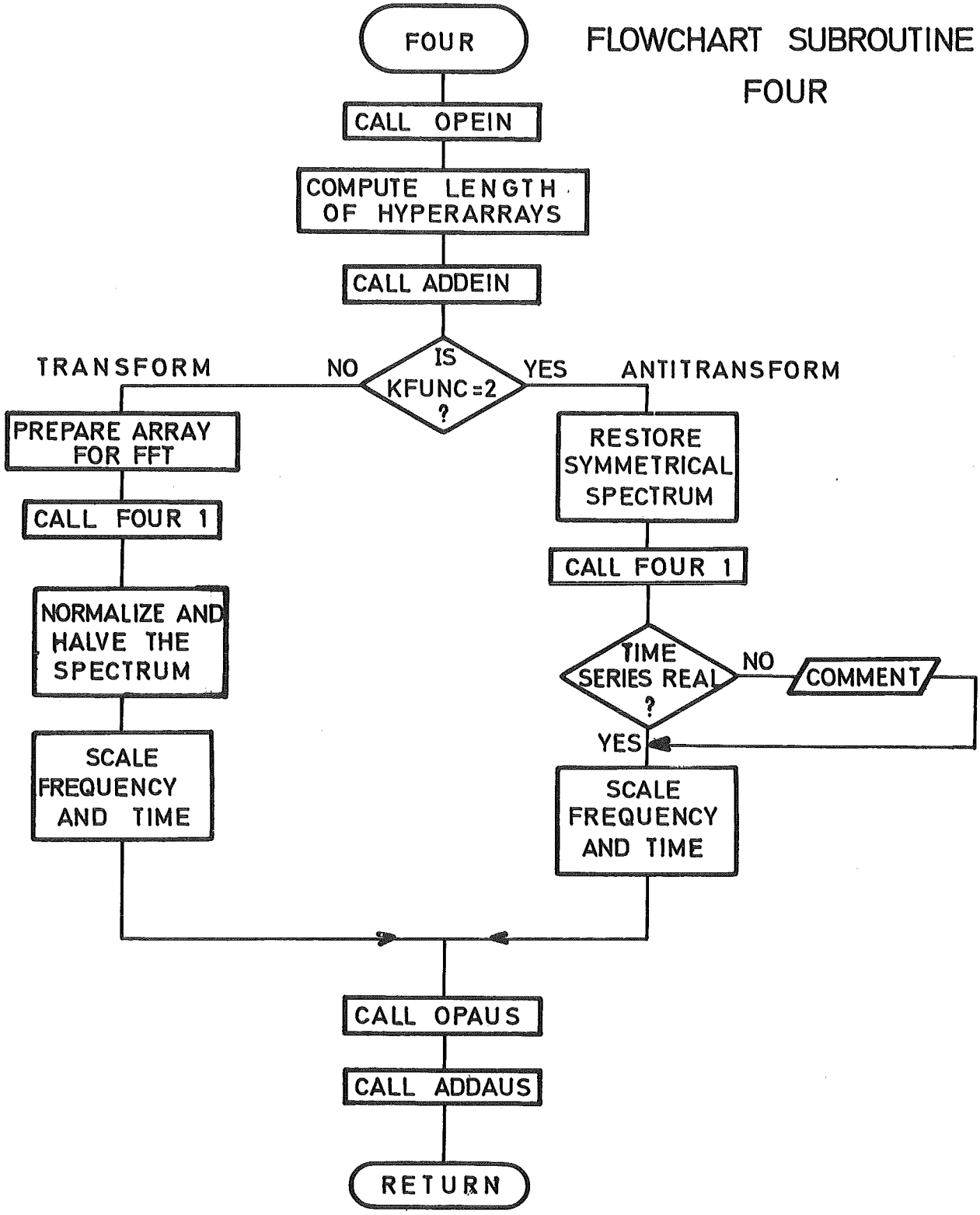
Now FFT is performed by calling the subroutine FOUR1. The raw spectrum is normalized by the factor 1/N (N = number of samples). It is not necessary to store the whole result array in the warehouse, because advantage can be taken from the symmetry properties of the FFT. Only the complex array represented by frequencies within the region of Freq = F_T/N and Freq = $F_T/2$ (F_T = sampling frequency, $F_T/2$ = Nyquist frequency) is stored into the warehouse. This action can be explained by the fact, that the complex spectrum calculated with FFT has a conjugate complex symmetry mirrored at the Nyquist frequency and that the value of the frequency zero contains no significant information, so the whole spectrum will be restorable at later times.

In the warehouse catalog the initial frequency value (which is F_T/N) is entered. (This corresponds to the time of the first sample value of time signal records). Also the reciprocal distance between the frequency samples (which is N/F_T) is entered (corresponding to the sampling frequency of time records).

2.8.2.2 Fourier Antitransform

The inverse of the discrete Fourier Transform, the Fourier Antitransform is in its form very similar to the Fourier Transform. So the FFT may be used to compute it. Before this transformation the whole complex frequency array must be restored, because SEDAP, as mentioned above, stores only part of the

FLOWCHART SUBROUTINE FOUR



COMMANDS:	FOUR performs a Fourier Transform or Antitransform with the help of FOUR1	NAME = FOUR SYSTEM = Fourier Package ENTRY = None
FOUT FANT		

CALL FOUR(KFUNC,ENAM,GNAM,K1,K3,KANW,KENW)

LIST OF ARGUMENTS:

KFUNC is an option indicator
KFUNC = 1 for Fourier Transform (time to frequency)
= 2 for Fourier Antitransform (frequency to time)

ENAM is the name of the input record

GNAM is the name of the output record

K1 is the search index of the record ENAM

K3 is the search index of the record GNAM

KANW, KENW are the delimiters of the selected record segment

Remark:
If the number of points is not a power of two, hyperarrays are generated and the mean value is subtracted from the time series before transforming.
The maximum resulting frequency is equal to the half of the Sampling frequency.

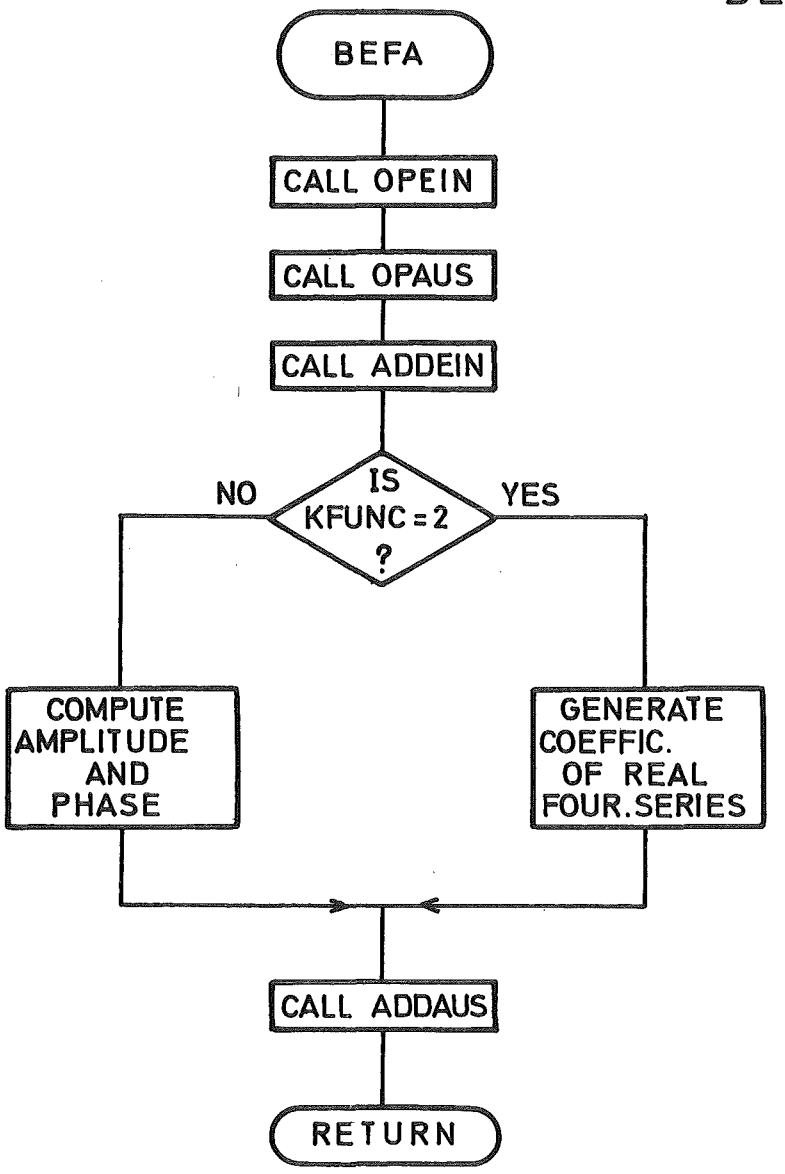
SUBROUTINES OR FUNCTIONS NEEDED: FOUR1, OPEIN, OPAUS, SQRT

ERRORS

DIRECT: None

INDIRECT: OPEIN, OPAUS

FLOWCHART SUBROUTINE BEFA



COMMANDS:	BEFA transforms a complex frequency record generated by the FFT into the amplitude and phase or into the normalized coefficients of the real Fourier series.	NAME = BEFA SYSTEM = Fourier Package ENTRY = None
BEFA FANA		

CALL BEFA(KFUNC,ENAM,GNAM,K1,K3,KANW,KENW)

LIST OF ARGUMENTS:

KFUNC is an option indicator
KFUNC = 0 for the computation of amplitudes and phases
KFUNC = 1 if the coefficients of the real Fourier analysis are to be computed from the complex Fourier Transform coefficients

ENAM is the name of the record to be transformed

GNAM is the name of the resulting record

K1 is the search index of the record ENAM

K3 is the search index of the record GNAM

KANW,KENW are the delimiters of the selected record segment

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN, OPAUS, SQRT, ATAN2

ERRORS

DIRECT: None

INDIRECT: OPEIN, OPAUS

spectrum in the warehouse.

If necessary there are zero values added to the frequency array to produce hyperarrays with a length equal to 2^N (N integer). It is necessary to perform the antitransform by beginning with the lowest frequency value stored in the warehouse (i. e. from block number 1), otherwise the result will not be correct. On the other hand, high frequency values may be disregarded, thus effectively using the FFT as a low-pass-filter. But it is not intended to be used as a standard possibility in SEDAP. (see description of the command FANT in chapter 3.2).

The Antitransform with FOUR1 should yield an array of real values in the time domain. To verify this, the imaginary part of the record is examined and if the maximum error(IMAG/REAL) is greater than 0.001 a comment is written.

Finally the real part of the computed record is transferred to the warehouse.

2.8.3 Real valued Fourier series and the computation of amplitude and phase (Subroutine BEFA)

Subroutine BEFA may be used for further reduction of spectra, calculated by the FFT. A BEFA execution may be caused by two commands:

BEFA generates amplitude and phase from real and imaginary data, while

FANA generates the normalized coefficients of the real Fourier series.

After the transfer of the data into the computing storage one or the other command is executed:

2.8.3.1 Amplitude and phase are calculated by

$$\text{ampl} = 2\sqrt{\text{real}^2 + \text{imag}^2}$$

$$\text{phase} = \arctg\left(\frac{-\text{imag}}{\text{real}}\right)$$

To compute the phase the FORTRAN-library-function ATAN2 (-IMAG, REAL) is used.

2.8.3.2 Fourier analysis

Since the FFT is a relatively new algorithm to perform spectral analysis, many users are still used to work with the coefficients of the real Fourier series. Therefore FANA transforms complex spectra, generated with the FFT into the normalized coefficients of the real Fourier series with the understanding that the time series had been real, according to the following relations:

If CR and CI are the coefficients of the complex FFT and COS and SIN are the coefficients of the real Fourier analysis then

$$COS(F) = 2 \cdot CR(F),$$
$$SIN(F) = -2 \cdot CI(F) \qquad \text{for } F = 1, \ldots, N.$$

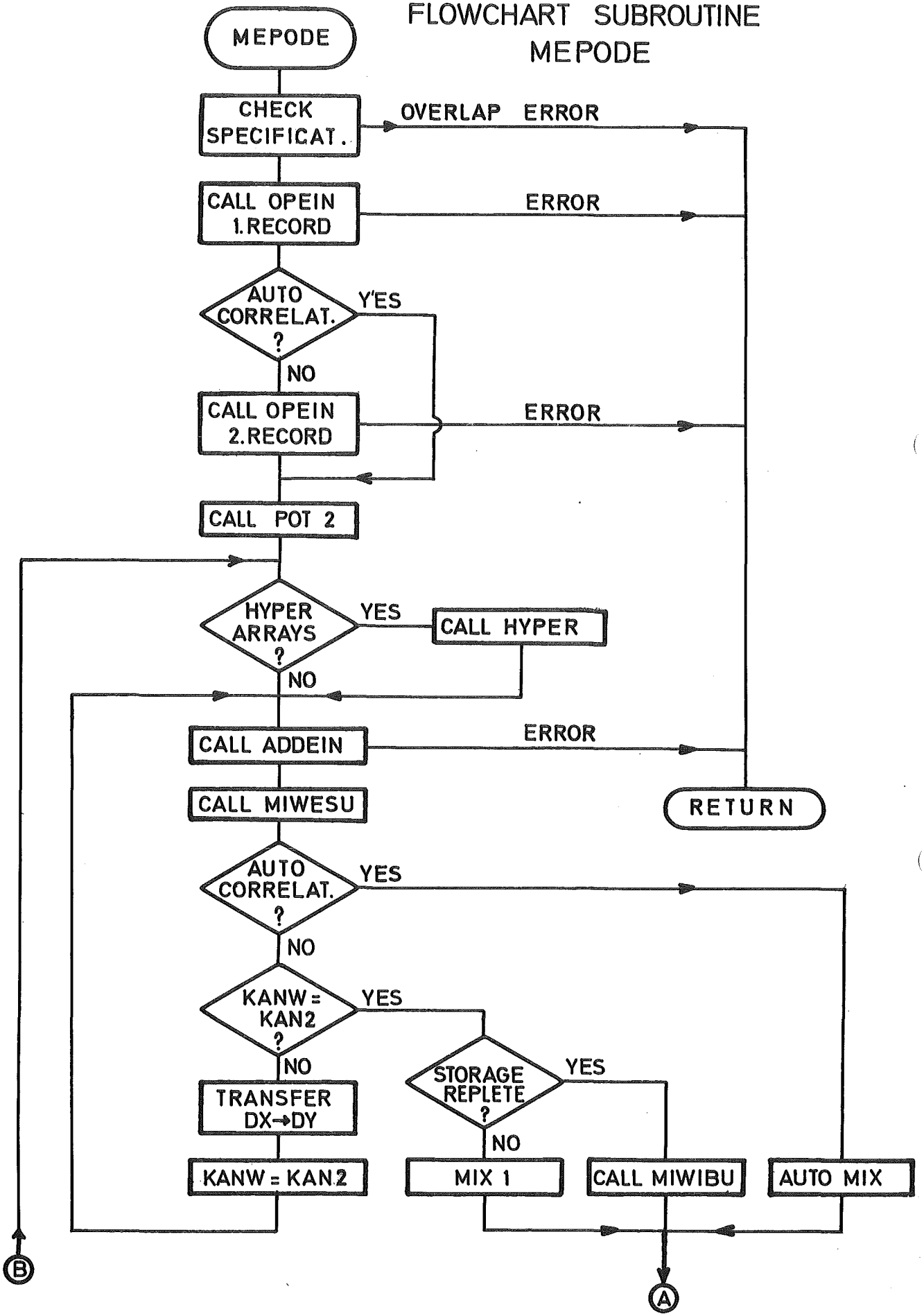
(Remark: Spectra generated by SEDAP do not contain coefficients for the frequency 0). The transformed spectra are transferred to the warehouse with Amplitude-Phase or else with SIN-COS adjacent in the storage. They may be sorted with the command SØ02. (See chapter 2.7.1 for special treatment of the frequency in the sorting algorithm).

2.8.4 Evaluation of power spectra (Subroutine MEPODE)

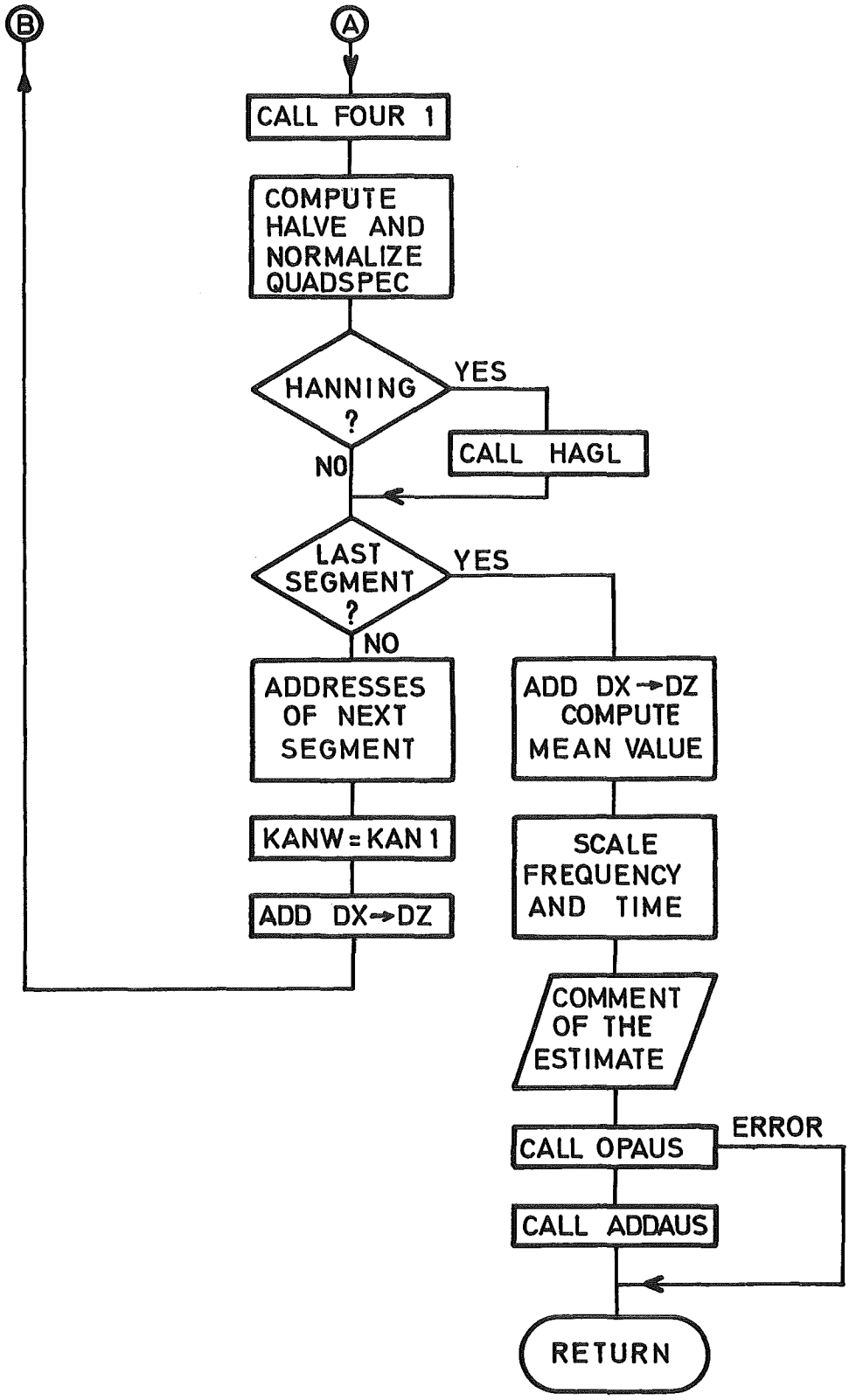
Subroutine MEPODE (Mean power density) uses a method /13/ for the application of the FFT algorithm to the evaluation of power spectra, which involves sectioning the record, taking modified spectra of these sections and averaging these modified spectra. In many instances this method involves fewer computations than other conventional methods /15/. Moreover, it involves the transformation of sequences which are shorter than the whole record and this is an advantage in SEDAP where the FFT is limited to sequences of 8192 points. Finally, it directly yields to a potential resolution in the time domain which is useful for testing and measuring nonstationarity.

MEPODE can be used to estimate cross- or auto correlated spectra. Furthermore correlation functions can be generated by using FANT for an antitransformation of correlated spectra.

FLOWCHART SUBROUTINE MEPODE



MEPODE
(CONTINUED)



COMMANDS:	MEPODE uses the FFT for the estimation of Auto- or Cross-Power Spectral Density by sectioning the experimental records and averaging modified periodograms of the sections	NAME = MEPODE SYSTEM = Fourier Package ENTRY = None
LEDI (Leistungs-dichte)		

CALL MEPODE(ENAM,FNAM,GNAM,K1,K2,K3,KANW,KENW,KSEG,UELAP,HANF, APCO,KDLT)

LIST OF ARGUMENTS:

ENAM is the name of the first input record

FNAM is the name of the second input record

GNAM is the name of the resulting record

K1,K2,K3 are search indexes of the three records

KANW,KENW are delimiters of the selected experimental record segment

KSEG is the length of a segment of the sectioned input record. It is also the length of the resulting record if no zeroes are added to the segments of the time series

UELAP is the length of the overlap of the segments

HANF is a repetition factor for Hanning smoothing

APCO is an option indicator
APCO = 1 a fully aperiodic correlated spectrum is generated. Otherwise it contains as many aperiodic spectral values as added zeroes.

KDLT is the whole length of the selected time series record to be transformed

SUBROUTINES OR FUNCTIONS NEEDED: OPEIN, OPAUS, FOUR1, POT2, MIWESU, HYPER, MIWIBU, HAGL

ERRORS

DIRECT: 21

INDIRECT: OPEIN, OPAUS

COMMANDS:	POT2 computes the number of zeroes which must be added to an array to obtain a length equal to a power of two, or to get aperiodically correlated spectra.	NAME = POT2 SYSTEM = Fourier Package ENTRY = None
None		

CALL POT2(KPT,N2,NZ,KAPCO)

LIST OF ARGUMENTS:

KPT	is the number of values of a data series
N2	is the length of the hyperarray
NZ	is the number of zeroes that must be added
KAPCO	is an option indicator if KAPCO = 1, N2 is doubled (aperiodic correlation) otherwise KAPCO = 0.

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS	DIRECT:
None	INDIRECT:

COMMANDS:	HYPER completes an array up to a specified length by filling the missing values with zeroes.	NAME = HYPER SYSTEM = Fourier Package ENTRY = None
None		

CALL HYPER(DATA,NKPT,NZ)

LIST OF ARGUMENTS:

DATA	is an array of data
NKPT	is the number of values of the data series
N2	is the length of the hyperarray which is completed by adding zero values to the end of the DATA-array.

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS	DIRECT:
None	INDIRECT:

2.8.4.1 Description of program flow

Mepode first checks the arguments and prepares the transfer of data from the warehouse to the computing arrays (OPEIN).

Two problems must be now considered, the first one involves the lags which are performed circularly (this means: the time record is assumed to be periodic in time) and the second must take into account the fact that the proposed number N of values is not always convenient for the use of the FFT-algorithm. The aperiodic correlation (which assumes that the time record is extended with zeroes both to the left and to the right) can be performed for as many points as there are zeroes added to the time series /7, 14/. One obtains a full aperiodic correlation if the third decimal argument DEZ(3) in the LEDI-command has been set to one.

If N is not a power of two, zeroes must be added to extend the arrays into the form suitable for FFT. These hyperarrays are generated by the subroutine HYPER, according to the following relations:

$$X(k) = X(k) \quad \text{for} \quad k = 1 \text{ to } N$$
$$X(k) = 0 \quad \text{for} \quad k = N+1 \text{ to } M$$

where M is the smallest power of two greater than or equal to N, or, in the case of a full aperiodic correlation, it is two times this value. The number of zeroes, that must be added is calculated by subroutine POT2. The calculation of mean power-spectra is then performed (see below). The initial frequency and the reciprocal of the frequency interval between the frequency samples are calculated and stored in the warehouse catalog.

Finally the transfer of data to the warehouse is initialized by OPAUS and executed by ADDAUS.

2.8.4.2 Method of sectioning time series

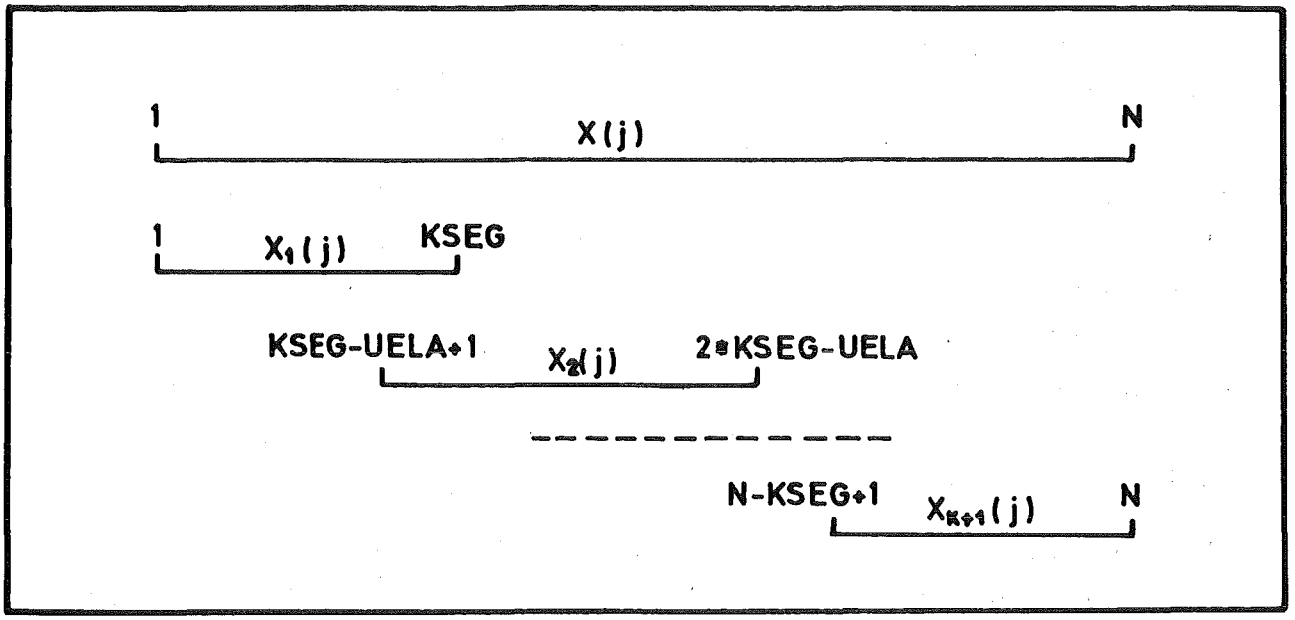


ILLUSTRATION OF RECORD SEGMENTATION

We consider an experimental record of length N from which segments of length KSEG are to be selected by taking into account a possible overlapping factor UELA. The starting points of the segments will be computed in the following manner:

Let X(j), j = 1, ..., KSEG be the first segment. Then

$$X_1(j) = X(j) \qquad \text{for } j = 1, \ldots, KSEG$$

Similarly

$$X_2(j) = X(j+KSEG-UELA)$$
$$X_K(j) = X(j+(K-1)\cdot(KSEG-UELA))$$

At the end of the experimental record an appropriate overlapping of the time segments is produced according to the technique proposed by Welch /13/.

We suppose that there are K+1 such segments, $X_1(j)$, ..., $X_{K+1}(j)$ and that they cover the entire record. The starting point of the last segment will be

$$X_{K+1}(j) = X(N-KSEG+j) \text{ for } j = 1, \ldots, KSEG$$

This segmentation is done for one experimental record in the case of an autocorrelated mean power density and for two experimental records when a cross power density is computed.

2.8.4.3 Handling of data in the computing arrays

The modest dimensions of the computing array require the use of economical methods. The computing array is segmented in three parts DX, DY, DZ, each of length 8192.

Data is transferred from warehouse into the first array DX. by ADDEIN. After the computation of hyperarrays by HYPER and of the residual time series by MIWESU (Mittelwertsubtraktion = compute and subtract the mean value) the segment of the first experimental record is transferred to DY in the case of cross-correlation and the equivalent segment of the second experimental record is loaded. Mixing algorithms are then applied to perform the storage allocation which is necessary for the FFT computation. Because both time series have only real parts, advantage is taken of the fact, that the two time series can be transformed at one time. One series is taken as the real part and the other series as the imaginary parts with

$$Z(k) = X(k) + iY(k)$$

and

$$Z(n) = \sum_{k=1}^{N} Z(k) \cdot \exp(-2\pi i \cdot n \cdot k/N)$$

It should be noted that Z has no physical meaning, but is introduced only for the sake of effectiveness of the algorithm. Using the Hermitian symmetry and its definition the spectra of X(k) and Y(k) are

$$X(n) = \frac{Z(n) + \bar{Z}(N-n)}{2}$$

$$Y(n) = \frac{Z(n) - \bar{Z}(N-n)}{2j}$$

($\bar{Z}$ is the complex conjugate of Z)

COMMANDS:	MIWESU (Mittelwertsubtr.) computes the mean value of an array and subtract it from the array.	NAME = MIWESU SYSTEM = Fourier Package ENTRY = None
None		

CALL MIWESU(DATA,NKPT,SUM)

LIST OF ARGUMENTS:

DATA	is an array and is equivalent to the XYZ-array
NKPT	is the number of values of the DATA array
SUM	is the meanvalue of the DATA array.

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS	DIRECT:
None	INDIRECT:

Three methods are available to mix X and Y into Z:

1. Mixing of arrays which are smaller or equal to 4096 points by using the free space in the computing storage.
2. Mixing of arrays of length 8192 points with the subroutine MIWIBU (Mix without buffer). This mixing algorithm uses only one storage allocation.
3. Autocorrelation-mix. Real and imaginary parts are equal: $Z(k) = X(k) + j\,X(k)$.

After mixing, the array DX + DY can be used for the FFT to allow the storage of 8192 complex values or 16384 real values.

From the result of the FFT the raw correlated spectrum

$$\frac{1}{2N} \cdot \overline{X} \cdot Y \quad (\overline{X} \text{ is the conjugate of } X)$$

is calculated normalized by the factor 1/2N so that an autocorrelated sinewave with amplitude A is transformed into the value $A^2/2$ /16/. Because of the symmetry of the spectrum, only frequencies up to the Nyquist frequency ($F_T/2$) are calculated (this economical method is also applied in FOUT and FANT). The spectrum is stabilized by the Hånning method which smoothes the spectrum by applying a frequency window (Subroutine HAGL). It is possible to specify between 0 and 10 successive smoothing passes by choosing the value of DEZ(2).

To calculate the mean power density of the whole experimental record all spectra (the auto- or cross-correlated mean power density of one segment of the time series) are added into the DZ array. Once the spectrum of the last segment has been added, the mean value of the spectra is calculated and the mean power density is expressed as

$$MPD = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{2N} \cdot \overline{X}_i \cdot Y_i \text{ with } i = 1, \ldots, M$$

for an ensemble of M samples.

2.8.4.4 Subroutine MIWIBU

If both input time series have a length of 8192 there is no free space in the computing array. To perform mixing of the

COMMANDS:	MIWIBU (Mix without buffer)	NAME = MIWIBU
None	The second half of an array is mixed with its first half to perform a quasi complex array out of two real series	SYSTEM = Fourier Package ENTRY = None

CALL MIWIBU(DATA)

LIST OF ARGUMENTS:

DATA is an array with a length of 16384 points. It contains two time series of 8192 points. DATA is equivalent to the XYZ array.

Method of mixing:
Only one free storage place is used to perform the mixing. This method is applied if no free storage is available in the computing arrays.

SUBROUTINES OR FUNCTIONS NEEDED: None

ERRORS	DIRECT:
None	INDIRECT:

COMMANDS: None	HAGL performs the smoothing of complex spectra with the Hanning function	NAME = HAGL SYSTEM = Fourier Package ENTRY = None

CALL HAGL(CX,N,KMAL)

LIST OF ARGUMENTS:

CX	is an array of complex values
N	is the number of the complex values
KMAL	is the repetition factor for the application of smoothing. It may be chosen between 0 and 10.

SUBROUTINES OR FUNCTIONS NEEDED:

ERRORS	DIRECT: INDIRECT:

time series at once, free space of at least half of the length of a time series should be available. The mixing is therefore done in 13 steps ($\log_2$ (8192)).

The method may be shown in the following example where the length of time series is 8. In the first step, the first half of the second time series is exchanged with the second half of the first time series, as illustrated in the following diagram:

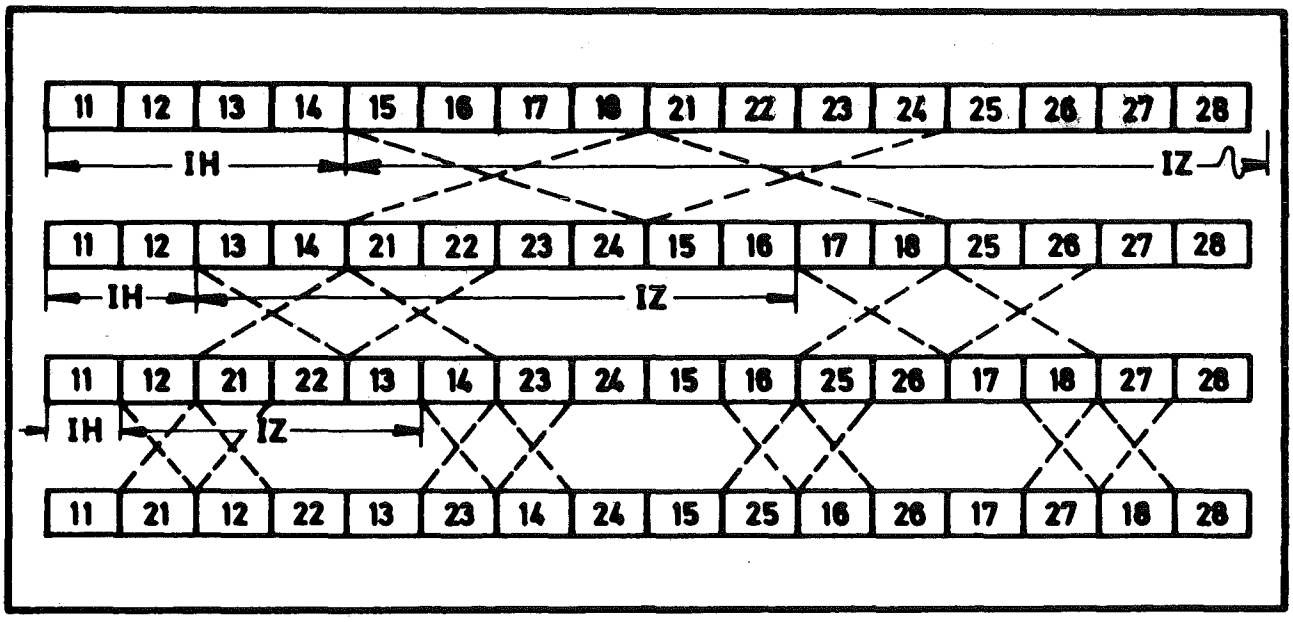


ILLUSTRATION OF THE MIXINGALGORITHM

Now all those values are in the first half of the whole array (16), which must be there at the end of the mixing. In the next step similar mixing is done as before, but for two arrays of half length. In every step the length of the arrays is divided by two, the number of arrays is doubled. The final result shows the right order after the execution of the third step.

2.9 User defined SEDAP commands (extending of SEDAP)

SEDAP helps the experimenter to conduct his own data reduction by the flexibility provided by its set of commands which can be used in many different combinations.

Furthermore the list of commands with fixed specifications can be extended by a special command: XTSD, the function of which may be specified by a user written subroutine. This subroutine must be written in FORTRAN with the name EXTSED and must be submitted to the linkage editor together with the object code of SEDAP in order to be available in a specific execution of SEDAP. To provide for this possibility SEDAP is not stored as a load module in the program library but rather as an object module. If no user supplied subroutine EXTSED is supplied, the system will provide a dummy subroutine with the same name and the command XTSD will have no effect.

The argument list of the subroutine EXTSED contains 13 arguments. Nine arguments correspond to the specifications of the nine parameter fields on the XTSD command card. The three first arguments correspond to the three first parameter fields of the command, also the six last arguments to the six last parameters of the command. Arguments four to seven are the results of checks in the main program and are used for further checks by the transfer routines (see chapter 2.2.2 and 2.3). Before using the parameters of the command, the description of the command XTSD should be studied with care. Some additional remarks may be useful:

- If a modifier is used (argument 2) it should be named with one of the existing modifier names.
- Only the three last parameters of the command may be deliberately used by the programmer.
- If the user needs more than these three parameters to control the execution of his subroutine, he may read additional information within EXTSED from any file which is not used by SEDAP itself (he should not use files 1, 5, 6, 7, 15, 40)
- Arguments of the transfer routines OPEIN, OPAUS, ADDEIN, ADDAUS may be defined as in example one, if the whole expe-

COMMANDS: XTSD	EXTSED is a user defined subroutine to extend the possibilities of SEDAP	NAME = EXTSED SYSTEM = ENTRY =

CALL EXTSED(ENAM,FNAM,GNAM,K1,K2,K3,K4,KANW,KENW,IX,DEZ1,
DEZ2,DEZ3)

LIST OF ARGUMENTS:

ENAM is the name of the first input record

FNAM is the name of the second input record or a modifier name

GNAM is the name of the resulting record

K1 is the search index of ENAM (must be positive)

K2 is the search index of FNAM (must be positive)

K3 is the search index of GNAM (must be -1)

K4 is the number of a modifier item in the modifier list

KANW,KENW are the delimiters of the selected record segment and are expressed in blocks

IX may be used as a sorting factor applied in ADDEIN or as any integer variable to control the subroutine algorithm, for example a input/output unit number

DEZ1,DEZ2,DEZ3 are user specified.

SUBROUTINES OR FUNCTIONS NEEDED: Defined by the user

ERRORS

DIRECT:

Defined by the user

INDIRECT:

rimental record can be stored in the computing storage (the 25088 first places in COMMON). If another transfer mode is used, a good knowledge of the possibilities offered by the transfer routines (chapter 2.3) is necessary.

- If a transfer from and to the warehouse is executed the program commonly will have the following structure:
  a) Specification of the common and other arrays
  b) Specification of transfer arguments
  c) Preparation of the transfer by OPEIN and OPAUS
  d) Transfer from the warehouse to the computing arrays
  e) Application of special data reduction algorithms to the experimental records
  f) Transfer from the computing arrays to the warehouse

As an illustration of programming with EXTSED two examples will follow.

Example 1

Let us assume that the data of an experimental record are disturbed by noise in the frequency range of 50 Hertz which should be eliminated. The converted and sorted signal shall be transformed by the Fast Fourier Transform (FOUT). Within the resulting frequency record the data-values in the frequency range 50 Hz ± 2 Hz shall be set to zero by XTSD. After the Antitransform of the resulting frequency record by FANT, the time signal will not contain any frequencies in this range.

Program flow of EXTSED

The whole common area of SEDAP is specified, a complex array CX is equivalent to the X-array.
The preparation of the data transfer follows. KRAF must be one if frequency records are transferred. The arguments of the transfer are chosen to execute the general transfer mode 1 (KXYZI = 1 , KXYZO = 1).
The preparation of the transfer is done by OPEIN, OPAUS.
The frequency interval of the spectral samples = 1/Freq is returned from OPEIN by the argument FREQ.
Now the delimiting numbers of the points corresponding to the frequency range to be deleted are computed and verified.

PROGRAM LIST OF EXTEND EXAMPLE NO 1

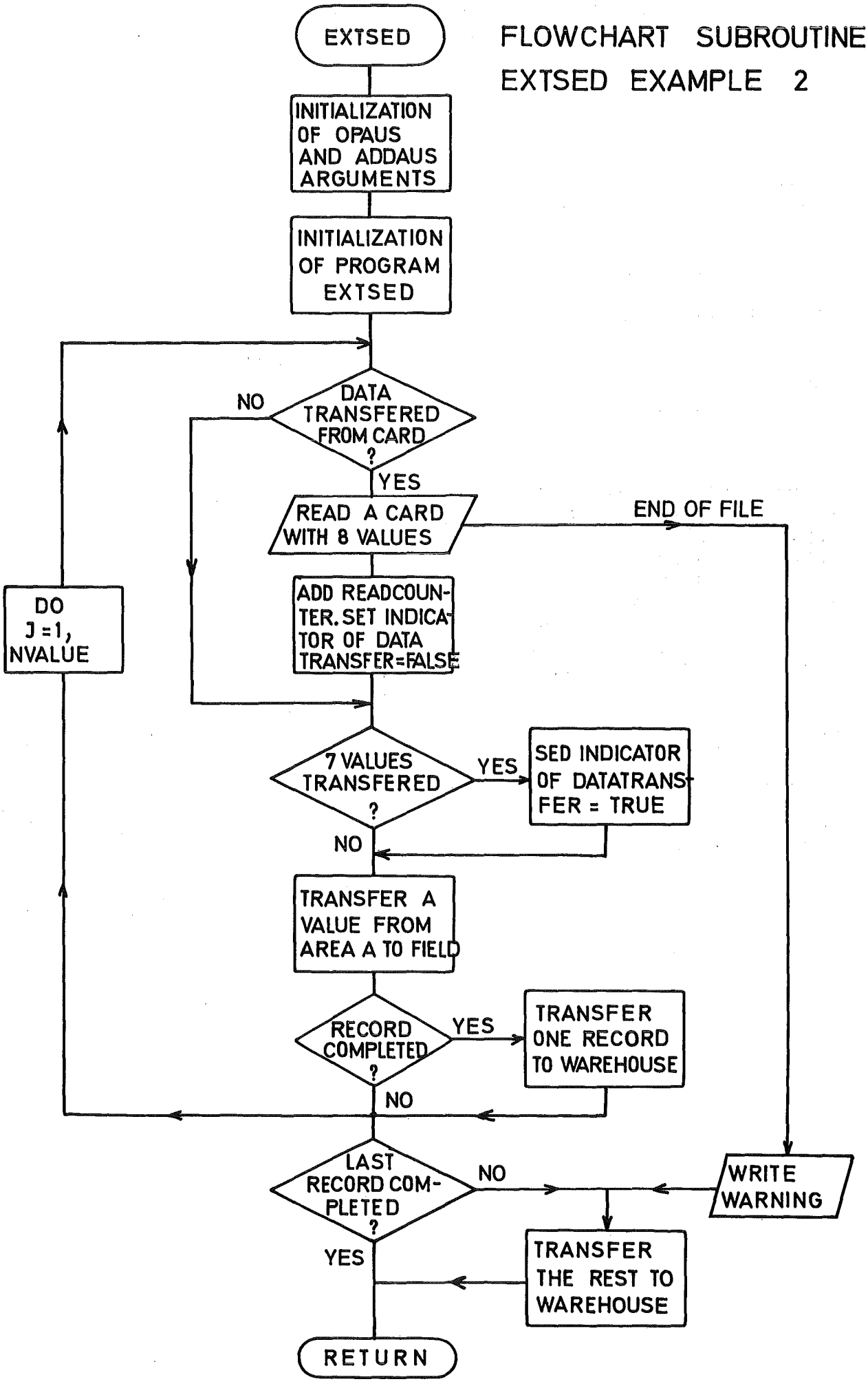
```
      SUBROUTINE EXTSED(ENAM,DUA,GNAM,K1,DUB,K3,DUC,KANW,KENW,DUD,FREDE,
     A FREDA)
C
C    USERDEFINED SUBROUTINE TO DELETE SOME VALUES OF A FREQUENCY RECORD
C    ------------------------------------------------------------------
C
      COMMON X (10240),Y (10240),Z (5120),
     1 BENAM(512),NANF(512),NEND(512),WFREQ(512),ADAT(512),BZEIT(512),
     2 KDAT,KEND,NC,NP,IA,JRV,X1,X2,Y1,Y2,IERR,AERR,BERR,JERR,KERR
     3 ,KPF (512)
      COMPLEX CX(4096)
      EQUIVALENCE (X(1),CX(1))
      KRAF = 1
      IMESS = 1
      ISTAT = 0
      ISTAK = 0
      MAX = 8192
      KUF=0
      KOF=0
      KXYZI = 1
      KXYZO = 1
      KPOINT = KEND
      KFUNC = 1
      KSHIFT = 0
      LOEF = 8192
      KNULL = 0
      CALL OPEIN (KANW,KENW,ENAM,K1,KRAF,TIME,MAX,LKPT,FREQ,DATE)
      IF (IERR.GT.0) GO TO 99
      CALL OPAUS (LKPT,GNAM,K3,KRAF,FREQ,DATE,TIME)
      IF (IERR.GT.0) GO TO 99
C  RANGE AND ADDRESSES OF FREQUENCIES
C DELETED FREQ = FREDE,MINFREQ = FREMI,MAXFREQ = FREMA
         FN = FREDE * FREQ
         FND=FREDA*FREQ
         NF = FN
         NDF=FND
         NMI = NF - NDF
         NMA = NF + NDF
         FREDE = NF / FREQ
         FREMI = NMI / FREQ
         FREMA = NMA / FREQ
         WRITE (NP,100) NMI,NF,NMA,FREMI,FREDE,FREMA
         IF (NMI.LT.1) GO TO 98
         IF (NMA.GT.LKPT) GO TO 98
      CALL ADDEIN (KANW,LKPT,LOEF,KRAF,KNULL,IMESS,IM,KXYZI,ISTAT,IOF,
     1 IUF,KOF,KUF)
      IF (IERR.GT.0) GO TO 99
C  DELETE
         DO 10 I=NMI,NMA
   10    CX(I) = (0.,0.)
      CALL ADDAUS (KFUNC,ISTAK,KPOINT,GNAM,IM,KXYZO,KSHIFT,IMESS)
      GO TO 99
   98 WRITE (NP,101)
   99 RETURN
  101 FORMAT(1H ,'FREQUENCIES DO NOT CORRESPONDENT TO THE',/
```

```
     A1H ,'FREQUENCY RANGE OF THE RECORD'/)
  100 FORMAT (1H ,3(I10,2X),3(F10.4,2X))
      END
```

COMMAND LIST OF EXTEND EXAMPLE NO 1

```
SEDAP      XTSD50HZ
           50 HZ FILTER
TEST OF EXTEND SEDAP,TAPE  ISPIK3
SEDA                       2400
ERAK                  ISPI    3   10   21 5000.
SOO4       ISPI       IS$$    1   16    2
FOUT       ISO2       SPEC    1    4
DEFX                                     0.        1.        12.7
DEFY                                    -600.     900.       5.08
PLOT       ISO2 TEXT               1    1
SIGNAL WITH    50HZ NOISE
XTSD       SPEC       FILT    1    4        50.       2.
FANT       FILT       ISFI    1    4
PBHE       FILT               1    4
PBHE       SPEC               1    4
BILD            ALLE
BEFA       SPEC       AMPN    1    1
BEFA       FILT       AMPF    1    1
SOO2       AMPN       ANFT    1    1    1
SOO2       AMPF       AFFT    1    1    1
DEFX                                     0.        1.        12.7
DEFY                                    -700.     800.       5.08
PLOT       ISFI TEXT               1    1
SIGNAL WITHOUT 50HZ NOISE
DEFX                                     0.        125.      12.7
DEFY                                     0.        500.      5.08
PLOT       ANO1 TEXT               1    1
SPECT. WITH    50HZ NOISE
DEFX                                     0.        125.      12.7
DEFY                                     0.        500.      5.08
PLOT       AFO1 TEXT               1    1
SPECT. WITHOUT 50HZ NOISE
STOP
```

FLOWCHART SUBROUTINE EXTSED EXAMPLE 2



COMMANDS: XTSD	Example 2 shows the use of EXTSED to transfer data punched on card deck into the warehouse	NAME = EXTSED SYSTEM = INPUT ENTRY = None

CALL EXTSED(S1,S2,GNAM,K1,K2,K3,K4,I1,NVALUE,IUNIT,FREQ,DATE,
TIME)

LIST OF ARGUMENTS:

S1	dummy argument in this example
S2	dummy argument in this example
GNAM	is the name of the resulting record
K1	dummy argument for this example
K2	dummy argument for this example
K3	is the search index of GNAM (must be -1)
K4	dummy argument for this example
I1	dummy argument for this example
NVALUE	is the number of values to be read
IUNIT	is the unit number of the input file
FREQ	is the frequency of the record
DATE	is the recording date of the data
TIME	is the recording time of the data

Remark:
S1, S2, GNAM correspond to NAM1, NAM2, NAM3 (page 166), K1, K2, K3, K4 are the search indexes defind on page 13 through 16, I1, NVALUE, IUNIT as well as FREQ, DATE and TIME correspond to I1, I2, I3 and X1, X2, X3 in the list given on page 166.

SUBROUTINES OR FUNCTIONS NEEDED: OPAUS,ADDAUS

ERRORS

DIRECT: 1

INDIRECT: see OPAUS,ADDAUS

PROGRAM LIST OF EXTEND EXAMPLE NO 2

```
      SUBROUTINE EXTSED (S1,S2,GNAM,K1,K2,K3,K4,I1,NVALUE,IUNIT,FREQ,
     1 DATE,TIME)
C
C  ACQUISITION OF DATA PUNCHED ON CARD DECKS
C  -----------------------------------------
C
      COMMON X (10240),Y (10240),Z (5120),
     1 BENAM(512),NANF(512),NEND(512),WFREQ(512),ADAT(512),BZEIT(512),
     2 KDAT,KEND,NC,NP,IA,JRV,X1,X2,Y1,Y2,IERR,AERR,BERR,JERR,KERR
     3 ,KPF (512)
      DIMENSION FIELD (512),A (8)
      EQUIVALENCE (X(1),FIELD(1))
      LOGICAL AEMPTY
C
C INITALISATION OF OPAUS AND ADDAUS ARGUMENTS
C
      WRITE (NP,102) NVALUE,IUNIT
      ISTAT = 0
      IM1 = 512
      KXYZ = 2
      IMESS = 1
      KSHIFT = 0
      KFUNC = 1
      KSORT = 1
      KPOINT = KEND
      CALL OPAUS (NVALUE,GNAM,K3,KSORT,FREQ,DATE,TIME)
C
C INITIALISATION OF PROGRAM EXTSED
C
      MUCH = 0
      INDA = 0
      INDF = 0
      IREAD = 0
      AEMPTY = .TRUE.
      IQUANT = NVALUE
C
      DO 10 J=1,NVALUE
      IF(.NOT.AEMPTY) GO TO 5
      READ (IUNIT,100,END=20,ERR=98) A
      IREAD = IREAD + 1
      AEMPTY = .FALSE.
      INDA = 0
    5 INDA = INDA + 1
      IF (INDA.EQ.8) AEMPTY = .TRUE.
      INDF = INDF + 1
      FIELD (INDF) = A (INDA)
      IF (INDF.NE.512) GO TO 10
      CALL ADDAUS (KFUNC,ISTAT,KPOINT,GNAM,IM1,KXYZ,KSHIFT,IMESS)
      IQUANT = IQUANT - 512
      MUCH = MUCH + 1
      INDF = 0
   10 CONTINUE
      IF (INDF.EQ.0) GO TO 99
      IM1 = IQUANT
   15 CALL ADDAUS (KFUNC,ISTAT,KPOINT,GNAM,IM1,KXYZ,KSHIFT,IMESS)
```

```
      GO TO 99
   20 IM1 = IREAD * 8 - MUCH * 512
      NR  = IREAD * 8
      WRITE (NP,101) NVALUE,NR
      GO TO 15
   98 IERR = 1
      JERR = IUNIT
      KERR = NR
   99 RETURN
  100 FORMAT (8F10.4)
  101 FORMAT (1H ,'THE END OF THE CARD DECK FILE WAS FOUND BEFORE ALL ',
     1 I4,' COULD BE READ.THERE ARE ONLY ',I4,' VALUES STORED IN THE ',
     2 'WAREHOUSE'/)
  102 FORMAT (1H ,I4,' VALUES ARE TO BE READ WITH FORMAT (8F10.4)',
     1' WITH THE FILE NUMBER FT0',I1,'F001'/)
      END
```

COMMAND LIST OF EXTEND EXAMPLE NO 2

```
SEDAP      EXTSED
  EXAMPLE NO. 2
CONVERSION OF DATA STORED ON CARDDECK INTO SEDAP BLOCKS
SEDA                          100
XTSD                  RECA        1002     1     1.      1006.72       0.0
PBVF       RECA                  1     2

        .
        .
        .
        .
```

The data are transferred to the CX-array, the values corresponding to the frequency range 50 Hz ± 2 Hz are zeroed and the resulting record is transferred back to the warehouse.

Example 2

SEDAP provides possibilities to reduce data recorded on magnetic or paper tape. This example shows how experimental data recorded on a card deck can be transformed into experimental records of the SEDAP format.

We assume the data to be punched on cards. Every card contains eight values with FORTRAN-format F 10.4.

The data is read by EXTSED (see flowchart) into an array FIELD of length 512*(4 bytes). Every time the array FIELD contains 512 values the data is transferred into the warehouse by ADDAUS.

If the last block to be transferred does not contain 512 values a special call of ADDAUS is executed.

If there are less data cards than specified by IX on the command card, a warning message will be written.

The following short command list gives an example how 1002 data values are read by XTSD and stored into the warehouse of SEDAP as an experimental record with the name RECA specified by the parameters: sampling frequency, recording date and time.

3. USING SEDAP

The basic idea behind the SEDAP concept was to relieve the experimenter from all standard programming work and to provide a detailed report of the processing activities. The experimenter must however direct the process and this is achieved by an experiment oriented language known as the SEDAP language.

As a long range target it was planned to implement a rather sophisticated command interpreter. In the present version, however, the structure of the command was restricted to the normal Fortran IV conventions with the advantages and the limitations such a choice necessarily implies.

3.1 Running a SEDAP job

The system's user is mainly concerned with the three following steps which are necessary to run a SEDAP job.

3.1.1 Description of the files

SEDAP requires the availability of different files which must be defined by the corresponding job control cards. The definition of the files is a task of a very specific nature and depends not only upon a given machine configuration but also upon the release or the type of software available. The job control cards which were used in November 1972 to run SEDAP on an IBM 360/65 - 370/165 computer are listed in appendix A to show the typical file environment of SEDAP. SEDAP users have access to a German handbook which informs them from any change in the procedure.

3.1.2 System initialization

A SEDAP job must always begin with the four following cards:

Card 1 - Title card
Card 2 - First comment card
Card 3 - Second comment card
Card 4 - Warehouse card.

The first card begins with the word SEDAP (column 1 to 5) and carries the title word which will be printed in big characters on the first page. The title can be made of any valid 8 characters combination starting in the column 11.

The second and third card will be printed at the bottom of the first page and allow the user to give a short description of the job he intends to process. The 160 characters (2 x 80) are completely free and will be printed in the same format. These two cards belong to the formal initialization and must always be present, they can be replaced by two blank cards but they should never contain the arrow (>) in the first column. The fourth card is a command card of the SEDA type which specifies the size of the warehouse and the use of the standard options. The description of the SEDA card is given in the list of the commands.

3.1.3 The SEDAP commands

The SEDAP commands were specified according to a general scheme. A SEDAP command card is generally formulated by one card and occasionally the card must be followed by a description card. This card is expected for instance after a new plot or a dump command and has nothing to do with the normal comment card which begins with an arrow (>) and which is skipped after the listing operation. The general structure of a SEDAP command is organized according to the list given on page 166.

3.1.4 Programming of the tasks

The modular principle of SEDAP allows to select different schemes as long as the basic requirements of the commands are respected. The user must generally begin by converting the data or by generating the test data. Various operations are then possible but the user must take care when specifying names, that new names are really new and old names are already known to the catalog of the warehouse. One common source of errors involves the segmenting of records which must be always compatible with the number of points really stored in the record. The use of an input sorting factor reduces the output by the

same factor and a new task must take the reduction into account when the new limits of a resulting segment are used. It is recommended to insert a few "BILD" commands to document the storage organization.

3.2 Description of the commands

The list of the commands uses a simplified syntax for the representation of the command language. This scheme shows the three groups of three command parameters included in parentheses with two commas as delimiters. Parameters, which are omitted, will be ignored by the command interpreter. As an example

```
ADDI (RECA, RECB, RECC) (IB1, IB2, IB3)
```

indicates that this command requires three record names and three integers as parameters, while no real data are necessary.

TABLE OF VALID COMMANDS

ADDI - ADD TWO RECORDS
AX+B - LINEAR TRANSFORMATION
BEFA - CONVERSION FROM CARTESIAN TO POLARKOORDINATES
BILD - CONTENT OF WAREHOUSE
DAGE - DATAGENERATION
DEFX - DEFINE THE X - AXIS
DEFY - DEFINE THE Y - AXIS
DIFF - DIFFERENTIATION OF A RECORDS
DIKO - COMPLEX DIVISION
DIVI - DIVIDE TWO RECORDS
DUMP - DUMP RECORDS ON A DATASET
ERAK - CONVERSION OF DATA RECORDED ON MAGNETIC TAPE
FANA - FOURIER ANALYSIS
FANT - FOURIER ANTITRANSFORM
FOUT - FOURIER TRANSFORM
FIL1 - LINEAR THREE POINTS SMOOTHING
FIL2 - LINEAR FIVE POINTS SMOOTHING
FIL3 - CUBICAL FIVE POINTS SMOOTHING
FIL4 - SMOOTHING WITH VARIABLE CUT-OFF FREQUENCY
HAFU - SMOOTHING OF SPECTRA WITH HANNING FUNCTION
HOLE - RESTORING OF DUMPED RECORDS
INSI - INTEGRATION WITH SIMPSON METHOD
INSW - INTEGRATION WITH SWITCH
INTR - INTEGRATION BY THE TRAPEZOIDAL METHOD
KOKO - COMPLEX CONJUGATE MULTIPLICATION
LEDI - ESTIMATION OF POWER SPECTRA
MUKO - COMPLEX MULTIPLICATION
MULT - MULTIPLICATION OF TWO RECORDS
MWEF - COMPUTATION OF THE MEAN VALUE
MWES - SUBTRACTION OF THE MEAN VALUE FOUND BY MWEF
PBHE - PRINT RECORDS IN HORIZONTAL ORDER WITH E-FORMAT
PBHF - PRINT RECORDS IN HORIZONTAL ORDER WITH F-FORMAT
PBVE - PRINT RECORDS IN VERTICAL ORDER WITH E-FORMAT
PBVF - PRINT RECORDS IN VERTICAL ORDER WITH F-FORMAT
PLOT - PLOT WITH AUTOMATIC SCALING
PTAP - CONVERSION OF DATA RECORDED ON PAPERTAP
RENA - RENAME A RECORD
SEDA - SPECIFY THE SIZE OF THE WAREHOUSE
SO02 - SORT 2
SO04 - SORT 4
SO08 - SORT 8
SO16 - SORT 16
SO32 - SORT 32
SO64 - SORT 64
SUBT - SUBTRACT TWO RECORDS
STOP - STOP THE COMMANDS INPUT STREAM
INI1 - CONVERT THERMOCOUPLE VOLTAGE TO TEMPERATUR
WERT - CREATION OF RECORDS DELIMITED BY VALUE- OR TIMEUNITS
XTSD - POSIBILITY TO DEFINE COMMANDS BY THE USER
ZERS - DESTROY ONE OR ALL RECORDS
ZUST - LISTS THE EXISTING COMMANDS

LIST OF VALID MODIFIERS

KONS - USED WITH COMMAND DAGE
AX+B - USED WITH COMMAND DAGE
SINF - USED WITH COMMAND DAGE
COSF - USED WITH COMMAND DAGE
VIER - USED WITH COMMAND DAGE
RAND - USED WITH COMMAND DAGE

ZEIT - USED WITH COMMAND WERT

TEXT - USED WITH COMMAND PLOT
ALT* - USED WITH COMMAND PLOT

ALLE - USED WITH COMMANDS DUMP HOLE BILD

MOD1 - USED WITH COMMAND XTSD
MOD2 - USED WITH COMMAND XTSD
MOD3 - USED WITH COMMAND XTSD
MOD4 - USED WITH COMMAND XTSD

FORMAT OF THE SEDAP COMMAND LANGUAGE

```
COMM      NAM1 NAM2 NAM3  I1   I2   I3      X1        X2        X3
----      ---- ---- ----  ----|----|----|-----.---|-----.---|-----.---|
```

EXPLANATION : COMM : COMMAND NAME
THE COMMAND MUST BE SPECIFIED BY ONE OF THE 51 KEYWORDS.ONLY THE FOUR FIRST CHARACTERS ARE CHECKED AND THE USER CAN EXTEND THE FOUR CHARACTERS TO ANY COMBINATION WICH DOES NOT EXCEED THE 10 CHARACTERS SPACE (MULTIPLY IS A VALID EXTENSION OF MULT).

NAM1 : FIRST RECORD NAME
IS GENERALLY THE NAME OF A RECORD WHICH IS TO BE FOUND IN THE WAREHOUSE.

NAM2 : SECOND RECORD NAME OR THE MODIFIER
IS THE NAME OF A SECOND RECORD OR THE NAME OF A MODIFIER WHICH IS SPECIFIED BY THE COMMAND.

NAM3 : THIRD RECORD NAME
IS THE NAME OF A NEW RECORD

I1 : FIRST INTEGER
IS THE FIRST BLOCK OF A SELECTED SEGMENT

I2 : SECOND INTEGER
IS THE LAST BLOCK OF A SELECTED RECORD SEGMENT

I3 : THIRD INTEGER
IS IN MOST CASES A SORTING FACTOR APPLIED TO THE INPUT (1 OF N VALUES).FOR SOME COMMANDS,WHICH DO NOT PERMIT SORTING,THIS NUMBER HAS A DIFFERENT MEANING.

X1 X2 X3 : THE THREE LAST PARAMETERS ARE DECIMAL NUMBERS AND THEIR MEANING IS EXPLAINED IN THE DESCRIPTION OF THE COMMANDS.

DATASTUCTURE :

STRUCTURE	COLUMNS	FORMAT	COMMENT
COMMAND	1 - 4	A4	LEFT JUSTIFIED
NAM1	11 - 14	A4	LEFT JUSTIFIED
NAM2	16 - 19	A4	LEFT JUSTIFIED
NAM3	21 - 24	A4	LEFT JUSTIFIED
I1	26 - 30	I5	RIGHT JUSTIFIED
I2	31 - 35	I5	RIGHT JUSTIFIED
I3	36 - 40	I5	RIGHT JUSTIFIED
X1	41 - 50	F10.4	
X2	51 - 60	F10.4	
X3	61 - 70	F10.4	

ADDI

adds two records and stores the resulting record in the warehouse.

ADDI (RECA, RECB, RECC) (IB1, IB2, IS)

RECA is the name of the first record to be added
RECB is the name of the second record to be added
RECC is the name of the record resulting from the addition (C = A + B)
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor to be applied at the input

Example

ADDI (CH22, CH23, TEMP) (1, 6, 2)

Add the six blocks (1 to 6) of the channel 22 to the six blocks of the channel 23 and store the resulting record (three blocks since IS = 2) under the name TEMP

Remarks

Maximum number of blocks = 2500 blocks
Sorting factor from 2 to 100
See note on synchronous records
ADDI can be used to add complex values

AX+B

performs the linear translation of a record according to the relation y = ax + b and stores the results in the warehouse

AX+B (RECX,,RECY) (IB1,IB2,IS) (A,B,)

RECX is the name of the input record (x in the formula)
RECY is the name of the resulting record (y in the formula)
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor to be applied at the input
A,B are the two coefficients a and b of the formula

Example

AX+B (CH11,,NC11) (1,10,5) (10.0,3.0,)

Multiply every 5th value of the 10 first blocks of the record CH11 by 10.0, add +3.0 to the product and store the 2 resulting blocks under the name NC11

Remarks

Maximum number of blocks = 2500 blocks
Sorting factor from 2 to 100
If A = 0.0, AX+B transforms the record in a constant B
If B = 0.0, AX+B multiplies a record by a constant A
If A = 1.0, AX+B adds a constant to the record
If A = 1.0 and B = 0.0, AX B becomes a "DO NOTHING" operator and transfers the input to the output. Since SEDAP recognizes this case and speeds the transfer accordingly without executing the operation, AX+B should be used to sort one of IS values of a record segment.

BEFA

transforms a complex record by computing amplitude and phase analog to the conversion of cartesian coordinates to polar coordinates

BEFA (COSP,,AMPH) (IB1,IB2,)

COSP is the name of the input record (complex spectrum)
AMPH is the name of the resulting record (amplitude, phase)
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment

Example

BEFA (KART,,POLA) (1,8,)

Transform the 8 first blocks of the record KART and store the result under the name POLA

Remarks

Maximum number of blocks = 2500
A sorting factor cannot be applied
Amplitude and phase values of any frequency are stored in two adjacent memory locations
They are normalized as if they had been generated by real Fourier analysis
For separating amplitude and phase see command SOnn

BILD

(means Snapshot or picture) maps the contents of the warehouse
Two forms are possible:

BILD (no argument)
or
BILD (,ALLE,)

BILD gives the names of all the records stored in warehouse and lists the parameters of the catalog for each record, if the modifier ALLE (=all) is specified, the task is extended to a list of the first eight values of each block contained in the warehouse.

Remark

BILD is especially useful to understand the way the system stores the records and should be called a few times by the new users of the system to check the properties of the records stored in the warehouse.

DAGE

Data generation

DAGE (,TYPE,REC?) (IB1,IB2,) (X1,X2,X3)

TYPE specifies the type of generated signals and is a generic name which must be replaced by one of the following modifier names:

KONS generates a constant signal with a sampling frequency X1 and an amplitude X3 (X2 is not used)

AX+B. generates a ramp with a sampling frequency X1. The first point of the signal has the value X3, the second X3 + X2 and the n-th point X3 + X2 (n-1). X2 and X3 can be positive or negative.

SINF generates a sine wave with a sampling frequency X1, a sine frequency X2 (Hz) and an amplitude X3. X1 and X2 must be always correctly defined (no default) and it is recommended to satisfy the condition $X1 \geq 2.0 \cdot X2.X3$ is generally positive and the user can use a negative value for X3 if he intends to cause a 180 ^{o} shift.

COSF generates a cosine wave and uses the same convention as SINF

VIER (viereck = square) generates an alternated (+/-) squarewave with a sampling frequency X1, a repetition rate X2 (Hz) and an amplitude X3. (The first halfwave is equal to X3 and the second to -X3). X1 and X2 must be correctly defined and a negative X3 value causes a shift of 180 ^{o} (inversion)

RAND generates a random signal with a sampling frequency X1 and an amplitude comprised between 0.0 and X3. X3 can be negative and X2 is disregarded.

REC? is the name of the generated record and the question mark indicates the status duality of the record:
The record name can be new and a new record will be generated in the warehouse. In that case IB1 is expected to be 1 and IB2 is the last block to be generated.
The record name can also be already known and the genera-

ted data will be added to the existing record. In that case the sampling frequency X1 has no meaning since the sampling frequency will be given by the catalog. IB1 and IB2 delimit the segment of the record and must satisfy to the requirements of a normal input request.

Examples

```
DAGE  (,AX+B,RAMP)   (1,10,)   (512.0,1.0,2.0)
```

A new record of 10 blocks will be generated under the name RAMP. The signal is a ramp with a sampling frequency of 512. Hz. The first value will be equal to 2.0, the second to 3.0 and the last value to 5121.

```
DAGE  (,SINF,SINE)   (1,1,)   (100.0,10.0,5.0)
```

A new record of 1 block will be generated under the name SINE. The signal is a sinewave with a sampling frequency of 100.0 Hz and a frequency of 10 Hz (there are ten full sine cycles in an interval of 100 points) and an amplitude of 5.0.

```
DAGE  (,KONS,MIXD)   (1,5,)    (400.,,10.0)
DAGE  (,VIER,MIXD)   (2,2,)    (400.,8.5, 1.)
DAGE  (,RAND,MIXD)   (4,4,)    (400.,,1.0)
```

The three previous commands will generate a constant signal of 5 blocks with an amplitude of 10.0 volt. The second block of the record will be "disturbed" by the superposition of a squarewave of 1.0 volt and the fourth block by a random signal of also 1.0 volt.

Remarks

Maximum of blocks = 1000
Sorting factor has no meaning and will be disregarded.

DEFX

defines the X axis of a plot frame

DEFX (,,) (,,) (XMIN,XMAX,XLENGTH)

XMIN is the minimum value specified for the X axis (the X axis is related to the time and is expressed in seconds)

XMAX is the maximum value specified for the X axis. If the relation XMIN > XMAX is not respected the system will interchange the two values.

XLENGTH specifies the physical length of the plot length (in centimeters). If XLENGTH has been omitted, the default value X = 35.0 cm will be substituted.

Example

DEFX (,,) (,,) (11.5,11.7,20.0)

A length of 20.0 cm is reserved to plot the X values which will be comprised between 11.5 and 11.7 seconds.

Remark

Since PLOT handles only 20 blocks in a task it is possible to extend the limit to several times 20 blocks if all the plot tasks are directed to the same frame which has been specified for all the values of different tasks. (See PLOT)

DEFY

defines the Y axis of a plot frame

DEFY (,,) (,,) (YMIN,YMAX,YHEIGHT)

YMIN is the minimum value specified for the Y axis

YMAX is the maximum value specified for the Y axis
If the relation YMIN < YMAX is not respected, the system will interchange the two values

YHEIGHT specifies the height of Y on the physical plot frame (units = cm). If YHEIGHT has been omitted, the default value DY = 26. cm will be substituted. In the present configuration the limit is 101.0 cm and any height exceeding 25.4 cm will cause the plot to be drawn on the large size plotter.

Example

DEFY (,,) (,,) (5.0,105.,50.0)

A height of 50 cm is reserved to plot the Y values which are expected to be comprised between 5. and 105. arbitrary units.

Remarks

See DEFX and PLOT

DIFF

differentiates a record and stores the resulting record in the warehouse

DIFF (RECA,,RECB) (IB1,IB2,IS)

RECA is the name of the record to be differentiated
RECB is the name of the resulting record
IB1 is the first block of the selected segment
IB2 is the last block of the selected segment
IS is the sorting factor to be applied to the input

Example

DIFF (SPID,,ACCE) (1,9,3)

the 9 first blocks of the record SPID are differentiated and the three resulting blocks are stored under the name ACCE. (Obtain the value of an acceleration by differentiating a velocity)

Remarks

- Maximum number of blocks = 2500 blocks
- Sorting factor from 2 to 100
- Minimum number of points = 3

DIKO

performs the complex division of two complex records and stores the resulting complex record in the warehouse

DIKO (RCXA,RCXB,RCXC) (IB1,IB2,)

RCXA is the name of the complex record to be divided by RCXB
RCXB is the name of the second complex record
RCXC is the name of the resulting complex record C = A/B
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment

Example

DIKO (SPK1,WEIG,QUOT) (2,3,)

the blocks 2 and 3 of the record SPK1 are divided by the blocks 2 and 3 of the record WEIG and the two resulting blocks are stored under the name QUOT

Remarks

- Maximum number of blocks = 2500
- The input sorting factor is not allowed for a complex operation
- see note on synchronous records
- a SEDAP block contains 512 values which must be considered as 256 complex values when the record is complex.

DIVI

divides two records and stores the resulting record in the warehouse

DIVI (RECA,RECB,RECC) (IB1,IB2,IS)

RECA is the name of the record to be divided by RECB
RECB is the name of the second record
RECC is the name of the resulting record (C = A/B)
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor applied during the input transfer (the three last parameters concern both RECA and RECB)

Example

DIVI (CH15,CH16,RATE) (2,21,10)

Divide CH15 by CH16 (20 blocks) and store the resulting 2 blocks (IS = 10) under the name RATE

Remarks

Maximum number of blocks = 2500 blocks
Sorting factor from 2 to 100 (included)
See note on synchronous records
DIVI should not be used to divide complex values (see DIKO).
If the record RECB contains values equal to zero, the division is impossible and the zero will be replaced by 1.0 as fix-up. This will be reported by one or several warnings at the end of the task.

DUMP

dumps a record or the complete warehouse on a user supplied sequential data set (usually tape). Two options are possible.

DUMP (RECA,,) (IB1,IB2,IFILE)
or
DUMP (,ALLE,) (,,IFILE)

RECA is the name of the record to be dumped (option1)
IB1 is the first record block to be dumped (option1)
IB2 is the last record block to be dumped (option 1)
IFILE is the file number for the user supplied 9 track tape
ALLE is the modifier name which selects the second option and causes the complete warehouse to be dumped

Example

DUMP (TEMP,,) (2,4,22)

Dump three blocks (2 to 4) of the record TEMP on the file 22.

DUMP (,ALLE) (,,23)

Dump the complete warehouse on the file 23

Remarks

DUMP must always be followed by a comment card. (80 characters are free, the first one should not be the > sign).
Since DUMP may be used to interface SEDA with other programs the user should be informed of the method to access the dumped values. Every dumped experimental record is preceded by a label. To read the label (SEDAP label, i. e. not the tape label), the following unformatted statement may be used.

READ (KFILE) (IWØRD(I),I=1,24), (FWORD(I),I=1,8)
IWØRD(1) length of the label (124 bytes)
IWØRD(2) record number
IWØRD(3) number of points
IWØRD(4) Filling factor (last block)

IWØRD(5 to 24) Text
FWØRD(1) Name of the record
FWØRD(2) Frequency
FWØRD(3) Date
FWØRD(4) Time
FWØRD(5 to 8) Unused.

The corresponding JOB-control card which must be supplied by the user for the SEDAP JOB STEP, might be the following:

```
//G.FTnnF001 DD UNIT=TAPE9,DSN=D1,DISP=(NEW,KEEP),
//           DCB=(BLKSIZE=3303,RECFM=VBS)
```

nn is the file number specified in the third integer-parameter of the DUMP-command.

ERAK

converts the experimental data recorded on a magnetic tape by the ERA data acquisition system and stores the resulting record in the warehouse

ERAK (,,NREC) (ITB1,ITB2,ITAPE) (FREQ,DATE,TIME)

NREC is the name of the record which results from the conversion

ITB1 is the first converted block

ITB2 is the last converted block

The following conventions are due to the special features of the ERA data acquisition system and must always be applied to ITB1 and ITB2 (Tape blocks):

a) The tape blocks contain 1024 values (and not 512). That means that 10 tape blocks will be converted into 20 SEDAP blocks.

b) The "label block" which contains a short information about the nature of the recording is always printed and is referred as the block No. 0. The blocks 1 and 2 are test blocks which are used to record the offset values of the amplifiers before the experiment is run for good. This standard practice implies that the "real" experimental values begin with the block No. 3.

c) If ITB1 = ITB2 = 0, the system will conclude that the user intends to convert only the label block. Since this operation is performed without storing a record, the name NREC can be omitted.

ITAPE is the file number used to specify the data set and the related type reel. This number should correspond to the definition given to the system for the tape and should be comprised between 20 and 29 to avoid any confusion with the standard units.

FREQ is the sampling frequency used to perform the recording and must be given in Hz. FREQ is the total frequency of the multiplexer or the sum of the frequencies used by all the recorded channels. If the experimenter records four

channels at the 5 kHz sampling rate, FREQ must be specified as 20000 Hz. Since many further operations (integration, differentiation, plot etc...) depend upon the value of the frequency, it is especially important that the user specifies correctly the value of FREQ. A frequency of 0.0 Hz will cause the task to be rejected.

DATE is the date of the experiments in the following order: day, month, year which must be coded as
2604.72 for the 26th day of April 1972.

TIME is always the time origin of the first value of the third block and must be given in seconds. If no time is given, the first value of the third block will have assigned the default origin 0.0 sec. This solution is recommended as long as the data reduction of the experimental phase does not involve a cross-reference of several files or tapes.

Example

ERAK (,,TA33) (3,22,29) (16000.,2604.72,0.0)

Twenty blocks of the file 29 (//FT29F001 ......) recorded with a sampling frequency of 16 kHz will be converted into the record TA33 (40 SEDAP blocks).

Remarks

The maximum number of blocks to be converted is limited only by the size of the warehouse.
It is possible during a job to convert more than one file or more than a tape. The user should be aware that the computing installation cannot simultaneously handle too many tapes and that even with a few tapes the job can seriously impede the smooth flow of a job stream by blocking several units. The user should clearly indicate that he intends to call the different tapes in a sequential order and not in the parallel mode. This can be achieved by requesting a deferred mounting or by specifying the affinity of different volumes for the same unit. This detailed information can be obtained from the specifications of the job control language.

FANA

transforms a complex record C(k) generated by the FFT into the usual coefficients A(k), B(k) of the Fourier analysis according to the relation:

$$C(k) \rightarrow 0.5(A(k) - jB(k)) \quad k = 1,\ldots,N/2$$

FANA (COMP,,COSI) (IB1,IB2)

COMP is the name of the complex input record
COSI is the name of the resulting sin-cos series
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment

Example

FANA (COMP,,COSI) (1,4,)

The 4 first blocks of the complex record COMP generated by the FFT are transformed. The resulting coefficients of the cos-sin series are stored in pairs into the warehouse under the name COSI.

Remarks

Maximum number of blocks = 2500
A sorting factor cannot be applied
Before separating the cos-sin series see description of command SOnn.

FANT

is used to antitransform a complex spectrum into the time domain with the Fast Fourier Transform (FFT) algorithm.

FANT (SPEC,,TIME) (IB1,IB2,)

SPEC is the name of the record to be antitransformed
TIME is the resulting record
IB1 is the first block of the record
IB2 is the last block of the selected record segment

Example

FANT (SPEC,,TISE) (1,3,)

The first blocks of the record SPEC are antitransformed. The resulting time series has a length of 4 blocks and is stored under the name TISE.

Remarks

Maximum number of blocks = 16
A sorting factor cannot be applied
FANT expects that the complex spectrum originally was generated by the FFT (commands FOUT,LEDI).
FANT may be used as a low pass filter by cutting of the higher frequencies (see example). The cutting of the lower frequencies will lead to erroneous results.
For instance the command
FANT (SPEC,,TISE) (2,4,)
will produce incorrect time series.

FIL1
FIL2
FIL3

smoothes a record and stores the resulting record in the warehouse

FIL3 (RECA,,RECB) (IB1,IB2,IS)

RECA is the name of the record to be filtered
RECB is the name of the resulting record
IB1 is the first block of the record to be filtered
IB2 is the last block of the record to be filtered
IS is the sorting factor applied at the input

FIL3 uses the third of three different algorithms which are given in the part II of the report (see FILTER).

Example

FIL3 (RAW1,,SMØ1) (1,2,2)

Smooth the two first blocks of the record RAW1 and store the resulting block (IS = 2) under the name SMØ1.

Remarks

Maximum number of blocks = 2500
Sorting factor from 2 to 100 (included)
See note on the use of sampling frequency.

The use of the filter subroutines requires a minimum number of values:

- FIL1 3 values
- FIL2 5 values
- FIL3 5 values

FIL4

filters a record with an user specified cut-off frequency and stores the resulting record in the warehouse.

FIL4 (RECA,,RECB) (IB1,IB2,IS) (FREQ,,)

RECA is the name of the input record
RECB is the name of the resulting record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor to be applied at the input
FREQ is the cut-off frequency of the filter (in Hz)

Example

FIL4 (CH21,,DA21) (1,10,2) (10.0,,)

Filter the 10 first blocks of the record CH21 by removing the frequencies above 10.0 Hz and store the five resulting blocks under the name DA21.

Remarks

FIL4 simulates a first order low-pass filter analog to the wellknown RC filter. FIL1, FIL2 and FIL3 provide a smoothing effect which is always adapted to the sampling frequency of the record, whereas FIL4 is a very effective variable filter which must be used with some care.

- The effect of this filter depends upon the setting of FREQ. FREQ has been normalized as the reciprocal value of the time constant (RC=TAU) of the filter, which means that a value of FREQ = 0.1 corresponds to a time constant of 10 sec.
- The cut-off frequency must be smaller than the sampling frequency (otherwise an error code will terminate the job) and the user must keep in mind that the effective sampling frequency is the sampling frequency divided by the sorting factor.
- The user is warned against the use of two large time-constants i.e. too small cut-off frequencies. The effect of a cut-off frequency f = 0.01 (100. sec time constant) on a record samp-

led at 50 Hz will be disastrous and will "dilute" or "smear" 10000 points since the filter is still effective after a time lag of two time constants.

Sorting factor from 2 to 100

Maximum number of blocks = 2500

FOUT

performs the Fast Fourier Transform (FFT) of a time series into a complex spectrum.

FOUT (TIME,,SPEC) (IB1,IB2,)

TIME is the name of the record to be transformed
SPEC is the name of the resulting record (complex spectrum)
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment

Example

FOUT (TIME,,SPEC) (1,,5)

The 5 first blocks are transformed. The time series array is completed to a hyperarray by adding 3 blocks filled with zero values. The resulting 8 blocks of the complex spectrum are stored under the name SPEC.

Remarks

Maximum number of blocks = 16
A sorting factor cannot be applied
The FFT expects the number of points to be a power of two. In the other case the array of the time series is extendet adding extra points with zero values.
The mean value of the time series is calculated and subtracted before transformation.
Only part of the complex spectrum is stored in the warehouse according to the frequency domain from $F = F_T/N$ up to $F = F_T/2$ (F_T = sampling frequency).

If hyperarrays be transformed the results may be corrected by a factor NPZ/NP (NP = is the number of data values, NPZ = NP plus the number of added zero values). Use command AX+B !
Before separating the real and imaginary parts, see description of command SOnn.
Detailed descriptions of the FFT can be found in /6 - 12/.

HAFU

smoothes complex spectra according to the Hanning's method.

HAFU (DATA,,DATB) (IB1,IB2,) (DSIGN,,)

DATA is the name of the record to be smoothed
DATB is the name of the modified spectral record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
DSIGN is 1. or -1. (see the remarks)

Example

HAFU (RADA,,MOSP) (1,4,) (1.,,)

The 4 first blocks of the record RADA containing the raw data are smoothed with the positive Hanning function. The resulting modified spectrum is stored under the name MOSP.

Remarks

Maximum number of blocks = 2500
A sorting factor cannot be applied
Algorithms of the Hanning smoothing method:

DSIGN = 1. Smoothing of quadratic spectra

$$MOSP(1) = 0.5 \cdot (RADA(1) + RADA(2))$$
$$MOSP(K) = 0.25 \cdot (2. \cdot RADA(K) + RADA(K-1) + RADA(K+1))$$
$$MOSP(N) = 0.5 \cdot (RADA(N-1) + RADA(N))$$

DSIGN = -1. Smoothing of linear spectra

$$MOSP(1) = 0.5 \cdot (RADA(1) - RADA(2))$$
$$MOSP(K) = 0.25 \cdot (2. \cdot RADA(K) - RADA(K-1) - RADA(K+1))$$
$$MOSP(N) = 0.5 \cdot (RADA(N) - RADA(N-1))$$
$$K = 2,3,\ldots,N-1$$

For detailed description see /6 - 8/.

HOLE (means GET)
restores a dumped record or several dumped records into the warehouse.
Two options are possible

HOLE (RECA,,) (,,IFILE)
or
HOLE (,ALLE,) (,,IFILE)

RECA is the name of the record to be transferred (option1)
IFILE is the file number of a user supplied 9 track tape which was produced by a DUMP or by a special interface.
ALLE is the modifier name which causes all the records of the records of the file to be transferred into the warehouse.

Example

HOLE (TEMP,,) (,,22)
The record TEMP is to be found on the file 22 and will be transferred to the warehouse.

HOLE (,ALLE) (,,23)
Restore the records of the file 23 in the warehouse.

Remarks

It is possible to transfer only one record if the transfer has been performed by DUMP ALLE, and HOLE ALLE will be accepted if only one record has been dumped.

INTR

integrates a record according to the trapezoidal rule (see definition of the algorithm in DIFINT) and stores the resulting record in the warehouse.

INTR (RECA,,RECB) (IB1,IB2,IS)

RECA is the name of the record to be integrated
RECB is the name of the resulting record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor applied at the input

Example

INTR (CH15,,IN15) (1,4,4)

The four first blocks of the record CH15 will be integrated and the resulting block will be stored under the name IN15.

Remarks

- Maximum number of blocks = 2500
- Sorting factor from 2 to 100
- The task should involve at least two points

INSI

integrates a record according to the Simpson's rule (see definition of the algorithm in DIFINT) and stores the resulting record in the warehouse.

INSI (RECA,,RECB) (IB1,IB2,IS)

RECA is the name of the record to be integrated
RECB is the name of the resulting record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor applied at the input

Example

INSI (CH21,,PR21) (1,1,4)

The first block of the record CH21 will be integrated and the resulting block (128 values) is stored under the name PR21.

Remarks

- Maximum number of blocks = 2500 blocks
- Sorting factor from 2 to 100
- The task must involve at least three points (required minimum)

INSW

(integration with switch) integrates a record by the trapezoidal rule and resets the integration to a preset level every time the "switching record" crosses a user specified threshold. The resulting record is stored in the warehouse. Typical application is the integration of periodic signals (sine) or pseudo-periodic waveforms (pulse shaped shockwaves) which is easier to interprete if the integration is reset periodically.

INSW (RECA,SWIT,RESL) (IB1,IB2,IS) (TRIG,RESET,)

RECA is the name of the record to be integrated
SWIT is the name of the record which causes the integration of RECA to be reset to a value RESET every time it crosses over the value of TRIG. SWIT can be the same record as RECA
RESL is the name of the resulting record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor to be applied at the input
TRIG is the threshold value of the "switching record"
RESET is the value to which the integration must be reset (usually 0.)

Example

INSW (SINE,SINE,HALF) (1,2,4) (,,)

the two first blocks of the record SINE are integrated and the resulting block (256 values) is stored under the name HALF. Since the two values TRIG and RESET are taken as 0.0 per default and since SINE itself provides the switch function, the integration will be restarted at the end of every half cycle.

Remarks

- Maximum number of blocks = 2500 blocks
- Input sorting factor between 2 and 100
- At least two values should be provided
- See note on synchronous records
- Records which cross the threshold between every two points, should be avoided as control records (SWIT)

KOKO

performs the multiplication of a complex record by the conjugate of another complex record.

KOKO (RCXA,RCXB,RCXC) (IB1,IB2)

RCXA is the name of the complex record whose conjugate is to be multiplied by RCXB
RCXB is the name of the second complex record
RCXC is the name of the resulting record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment

Example

KOKO (SPC1,SPC2,XREC) (1,4)

The 4 first blocks of the complex record SPC2 are multiplied by the complex conjugate of the 4 first blocks of the record SPC1 and the resulting complex record (i.e. a power spectrum) is stored under the name XREC.

Remarks

Maximum number of blocks = 2500
A sorting factor cannot be applied
See note on synchronous records
A SEDAP block contains 512 values which must be considered as 256 complex values if the record is complex. The conjugate complex multiplication is used to generate auto- or cross-power spectra.

LEDI

uses the FFT for the evaluation of auto- or cross-power spectral density by sectioning the experimental records and averaging modified periodograms of the sections.

LEDI (TIMA,TIMB,SPEC) (IB1,IB2,ISEG) (XLAP,XSMO,XAPER)

TIMA is the name of the first input record (time series)
TIMB is the name of the second input record (time series)
SPEC is the name of the resulting record (complex spectrum)
IB1 is the first block of the selected experimental record segment to be transformed
IB2 is the last block of this selected record segment
ISEG is the length of the partial record segments into which the total selected record segment is sectioned, and also the length of the resulting record if no zeroes be added to the segments of the time series.
XLAP is the length of the overlap of the segments
XSMO is a repetition factor for the application of the Hanning smoothing algorithm
XAPER is an option indicator. XAPER = 1. leads to a fully aperiodic correlation, otherwise the spectrum contains as many aperiodic spectral values as zero values were generated to extend the time series record (hyperarrays).

Example 1

LEDI (SIGA,SIGA,APSD) (1,100,4)

The 100 first blocks of record SIGA (signal A) are used to estimate an auto-correlated power spectral density (APSD). The FFT is performed in sections of 4 blocks, that is, first the blocks 1 to 4 are transformed, next the following blocks 5 to 8, and so on, until the whole signal record has been processed. The choosed segments do not overlap, the spectra are not smoothed, the correlation is cyclically performed. The resulting spectral estimation has a length of 4 blocks and is stored under the name APSD.

LEDI (continued)

Example 2

LEDI (TIMA,TIMB,CPSD) (1,16,3) (1.,3.,1.)

The cross-power spectral density is evaluated for the 16 first blocks of the times series records TIMA and TIMB. The transformation is performed with sections of length 3 blocks one block overlapping (section 1 = block 1 to 3, section 2 = block 3 to 5, and so on). To power spectrum the Hanning smoothing is applied three times.
Hyperarrays are generated of the time series sections. One block of zero values is added according to the requirement of the FFT for array length of a power of two, other 4 blocks of zero values are added to perform a fully aperiodic correlation.
The linear meanvalue of the 8 computed spectra is stored under the name CPSD with a record length of 8 blocks.

Remarks

Maximum number of blocks = 2500
A sorting factor cannot be applied
See note on synchronous records and on complex values
Linear mean values are computed and subtracted for every section of the time series.
The length of the segments must be a power of two, otherwise hyperarrays are performed. Their maximal length is 16 blocks.
In case of aperiodic correlation the length of the segments is doubled by adding zero values, the maximal length of the segments is then 8 blocks.
The length of the resulting record is in blocks:
(512 • ISEG + Number of added zero values) /512

LEDI (continued)

The adding of zero values causes a too small amplitude. It may be corrected with the help of the command AX+B by multiplication by a factor:
FA = ((512 • ISEG + number of added zero values)/(512•ISEG))^2
(see 2.8.2.2). The resulting spectrum is the linear meanvalue of the computed quadratic spectra.
It is stored according to the spectral range from F = F_{sample}/N to F = $F_{sample}/2$ (Nyquist frequency).
For separating real and imaginary parts see command SOnn.

Special remarks

The evaluation of power spectra without the use of LEDI:
The command LEDI has been defined to offer the user further programming comfort. All the operations executed by the example 2 could have been performed with the existing more special commands. However, this requires a longer command list, more records in the warehouse and because of the many transfer-operations much more computing time. To demonstrate the difference between the application of LEDI and the programming of commands without LEDI a schematic list to compute a cross-power density as in example 2 follows on the next page.

LIST OF COMMANDS TO ESTIMATE THE CROSSPOWER DENSITY
AS IN EXAMPLE 2 WITHOUT LEDI

```
AX+B    TIMA        ZERO    1    8    0.          0.
AX+B    TIMA        TSA1    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        TSA8   14   16
  .                                                 .
  .                                                 .
  .                                                 .
                                                    8
ADDI    MSP1  MSP2  ADD1    1    8
ADDI    MSP3  ADD1  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.
```

LEDI (continued)

The first command AX+B produces a record filled with zero values, but with the same parameter as those of the time series records. To get the hyperarrays the sections of the time series are added to the zero record (see note on synchronous records). Following are the FFT of the hyperarrays, the correlation by complex conjugate multiplication and threetimes the Hanning-smoothing. This must be done 8 times. Finally the mean value is computed by adding and normalizing the spectra.
During normalizing two corrections are applied:

1) The result must be multiplied by a factor 7.1 to correct the deflection of the amplitude caused by the adding of zero values (see the remarks above).
2) The computation of the power density via the complex conjugate multiplication KOKO yields only to the half value of the expected amplitudes.

So the normalizing factor is 7.1•2/8 (8 is the number of samples).

This detailed program of commands produces indeed many intermediate informations, but additional 78 commands are needed and 624 more blocks stored in the warehouse.

MUKO

performs the complex multiplication of two complex records and stores the resulting complex record in the warehouse

MUKO (RCXA,RCXB,RCXC) (IB1,IB2,)

RCXA is the name of the complex record to be multiplied by RCXB
RCXB is the name of the second complex record
RCXC is the name of the resulting record
IB1 is the first block of the selected segment
IB2 is the last block of the selected record segment

Example

MUKO (SPC1,SPC2,XREC) (5,6,)

The blocks 5 and 6 of the complex record SPC1 are multiplied by the blocks 5 and 6 of the complex record SPC2 and the two resulting complex blocks are stored under the name XREC.

Remarks

- Maximum number of blocks = 2500
- Input sorting factor is not allowed and will be disregarded
- See note about synchronous records
- A SEDAP block contains 512 values which must be considered as 256 complex values if the record is complex.

MULT

multiplies two records by each other and stores the resulting record in the warehouse.

MULT (RECA,RECB,RECC) (IB1,IB2,IS)

RECA is the name of the first input record to be multiplied by RECB
RECB is the name of the second input record
RECC is the name of the resulting record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor applied during the input transfer

Example

MULT (DATA,DATB,PROD) (2,3,4)

Multiply DATA by DATB (2 blocks) and store the resulting half filled block (IS = 4) under the name PROD.

Remarks

Maximum number of blocks = 2500
Sorting factor from 2 to 100
See note on synchronous records
MULT should not be used to multiply complex values (see MUKO, KOKO).

MWEF

computes the meanvalue of a data series and saves it for a subsequent command to substract the meanvalue (MWES). Also it is listed.

MWEF (RECA,,) (IB1,IB2,IS)

RECA is the name of the input record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor

Example

MWEF (TISE,,) (1,3,)

Of the 3 first blocks of TISE the meanvalue is computed and saved.

Remarks

Maximum number of blocks = 2500
Sorting factor from 2 to 100
If the command MWES succeeds, it should be immediately, in case of the overlay version.
By the command MWEF a formerly computed meanvalue is destroyed.

MWES

may be used to subtract from an experimental record its meanvalue. This must be computed by a preceding MWEF command.

MWES (RECA,,RECB) (IB1,IB2,IS)

RECA is the name of the input record
RECB is the name of the resulting record
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor applied to the input

Example

```
MWEF  (TISE,,)   (1,3,)
MWES  (TISE,,RESI)   (1,3,)
```

Both commands are put together in order to suggest to use them always in this sequence immediately following each other. By the command MWES the values of the 3 first blocks of TISE are transformed. After the meanvalue computed by MWEF has been subtracted, the result is stored under the name RESI.

Remarks

Maximum number of blocks = 2500
Sorting factor from 2 to 100
The parameters of the corresponding MWEF and MWES commands (RECA,IB1,IB2,IS) must be identical; otherwise the resulting record RECB will be incorrect.

PBxy

prints the blocks of a record on the standard printing file. PBxy is a generic name where x and y must be replaced by the following characters to select one of the four options:

x = H for a horizontal list
  = V for a vertical list

y = F for a FORTRAN F-format
  = E for a FORTRAN E-format.

The four valid combinations define the four following commands: PBHF, PBHE, PBVF and PBVE

PBxy (RECA,,) (IB1,IB2)

RECA is the name of the record whose blocks will be printed
IB1 is the first block to be printed
IB2 is the last block to be printed

Example

PBVF (TEMP,,) (1,5,)

The five first blocks of the record TEMP will be printed in vertical order and the values will be written with a F-format.

Remarks

- Maximum number of blocks = 2500 blocks
- An input sorting factor is not possible
- There is always one block by page

PLØT

plots a record on the plotter.

PLØT (RECA,MØDI,) (IB1,IB2,IS)

RECA is the name of the record to be plotted
MØDI stands for the three possible modifiers and must be replaced by either "ALT*" or "TEXT" or blank
IB1 is the first block to be plotted
IB2 is the last block to be plotted
IS is the sorting factor applied at the input

Two cases are possible:

1) The modifier is blank or "TEXT".
   PLOT uses the coordinates which were given by DEFX and DEFY and cuts the values which are not compatible with the frame. Otherwise the subroutine determines a frame. If the modifier has been specified as "TEXT" a comment card must follow immediately the PLØT card, otherwise a default text will be generated.
2) The modifier is "ALT*".
   The PLOT is drawn on the former frame and the values which are not compatible with the frame are cut off. A new frame will be used if no former frame was available.

Example

```
PLOT  (TEMP,,)   (1, 8,)
```

The first 8 blocks of the record TEMP are to be plotted with the default text.

Remarks

- Maximum number of blocks = 20 blocks

PTAP

converts a paper tape and stores the converted values into the warehouse.

PTAP (,,RECA) (,,IFILE) (FREQ,DATE,TIME)

RECA is the name of the new record which results from the conversion

IFILE is the file number of the corresponding file declaration (paper tape reader)

FREQ is the sampling frequency in Hz which corresponds to the cycle frequency multiplied by the number of channels effectively recorded

DATE (see ERAK)

TIME (see ERAK). The default value is 0.0 sec

Example

PTAP (,,LØØP) (,,17) (0.03,2604.72,0.0)

Convert the paper tape referenced under the file 17 and store the resulting record which will be known as "LØØP".

Remarks

If an error is detected during the conversion, an auxiliary subroutine will be automatically called and the error will be identified. Furthermore the values will be printed.

Maximum number of blocks = 40 blocks

RENA

renames a record stored in the warehouse

RENA (OLDN,,NEWN)

OLDN is the old name of the record. The name must be known
NEWN is the new name of the record. The name must be new

Example

RENA (CH34,,TEMP)

The record CH34 will be renamed TEMP and must be thereafter called by the name TEMP.

Remarks

RENA can be used to change the names provided by a sorting operation.

SEDAP

is a special command which specifies the size of the warehouse and the options of the system.

SEDA (DUMP,,PASS) (ISIZE,,IFILE)

"DUMP" is the keyword which causes the contents of the warehouse to be automatically dumped on a magnetic volume if an error has been detected before the end of the job.

PASS is a password for system testing and should be left blank.

ISIZE specifies the size of the warehouse in blocks. The minimum size is 100 blocks. The maximum size of 5000 blocks should not be exceeded.

IFILE is the file number of a user supplied 9 track tape where the records will be dumped.

Example

SEDA (,,) (500,,)

defines a 500 blocks warehouse without the DUMP option.

SEDA (DUMP,,) (5000,,29)

specifies a 5000 blocks warehouse and requires that the contents of the warehouse be dumped on the file 29 if an interruption occurs.

Remarks

The SEDAP command must be the fourth card of the deck, i.e. the first command. If a second SEDAP card is read thereafter, the option status may be changed but the size of the warehouse remains unchanged.

SØnn

sorts a multiplexed channel into nn channels. SØnn is a generic command name which must be replaced by one of the six possible options to obtain the six following commands:

SØ02, SØ04, SØ08, SØ16, SØ32, and SØ64

SØnn (RECA,,REØØ) (IB1,IB2,ILIMIT)

RECA is the name of the record which will be sorted.

REØØ (REFT) is the name which has been selected for the nn resulting records. The new names are given automatically by SEDAP during the execution of the task by replacing the two last characters of REØØ by the serial number of the channels. The new names will be RE01, RE02, ... RE64 for nn = 64 and the ØØ ending is not a requirement but is recommended to the user to keep him aware of the fact that they will be replaced. If the user uses the command SØ02 to separate the real and imaginary parts of a complex record like in the case of a complex Fourier spectrum (Fourier Transform), he must indicate his intention by specifying a new name ending by the two characters 'FT'. In that case the record REFT will produce two record names RE01 and RE02 but their sampling frequency will be equal to the sampling frequency of RECA (i.e., not divided by 2).

IB1 is the first block to be sorted

IB2 is the last block to be sorted

ILIMIT is a user specified limit which must be comprised between 1 and nn and which causes only the first ILIMIT records to be stored in the warehouse. This option is especially useful when 64 record channels were recorded with only ILIMIT connected to the experiment.

Example

SØ16 (DAX1,,CHØØ) (1,32,11)

the 32 first blocks of DAX1 will be sorted into 16 channels. The resulting 11 new records are CH01, CH02 ... and CH11 and are comprised of two blocks.

Remarks

Maximum number of blocks = 2500 blocks
No input sorting factor
See AX+B if only one of n values has to be sorted for a non-multiplexed record.

STOP

is the last command of a job

STOP (no arguments)

Remark

STOP provides the system with a command which orderly terminates a job and should not be forgotten.
Any commands following the STOP command will be listed at the beginning of the job but they will not be processed.

SUBT

subtracts a record from another record and stores the resulting record in the warehouse

SUBT (RECA,RECB,RECC) (IB1,IB2,IS)

RECA is the name of the first input record
RECB is the name of the record to be subtracted from RECA
RECC is the name of the record resulting from the subtraction (C = A - B)
IB1 is the first block of the selected record segment
IB2 is the last block of the selected record segment
IS is the sorting factor to be applied at the input

Example

SUBT (DA15,DA16,RDIF) (1,6,2)

Subtract the six first blocks of DA16 from the six first blocks of DA15 and store the three resulting blocks (IS=2) under the name RDIF.

Remarks

Maximum number of blocks = 2500 blocks
Sorting factor from 2 to 100
See note on synchronous records
SUBT can be used to subtract complex records

TNI1

converts a millivolt record originated from a Ni-Cr-Ni thermocouple into ^{o} C and stores the resulting record in the warehouse

TNI1 (MILV,,DEGR) (IB1,IB2,IS)

MILV is the name of the input record to be converted
DEGR is the name of the resulting record (see remarks)
IB1 is the first block to be converted
IB2 is the last block to be converted
IS is the sorting factor applied at the input

Example

TNI1 (CH15,,TE15) (1,3,3)

Convert the two first blocks of the record CH15 and store the resulting block (IS=2) under the name TE15.

Remarks

Maximum number of blocks: 2500
Sorting factor from 2 to 100 (included)
Since the voltage produced by a thermocouple is physically limited and since the range of the function is comprised between 0 ^{o} C and 1300 ^{o} C, the input record MILV must contain positive values comprised between 0.0 and 52.46 mV. Any value not comprised within this range will be converted to the minimum or to the maximum (0 or 1300 ^{o} C) and a warning will be printed at the end of the task to indicate the number of times the function has been found exceeded.
The user should be aware that the thermocouples signals are often amplified and that the function is defined for a reference temperature equal to 0 ^{o} C. The operator AX+B allows this double correction in one step.
The user can also use AX+B in a following step if he wants to obtain a temperature in ^{o} F or in ^{o} K.

WERT

creates a new record by transferring a segment of an old record with the peculiarity that the segment is delimited by time units or by the position of the limiting points. Two forms are possible:

WERT (RECA,,RECB) (IP1,IP2,)

WERT (RECA,ZEIT,RECB) (,,) (T1,T2,TFLOAT)

RECA is the name of the old record which must be stored in the warehouse

RECB is the name of the resulting record

For option 1

IP1 is the IP1th point of the record RECA and will become the first point of the record RECB

IP2 is the IP2th point of the record RECA and will be the last point of the new record RECB

conditions IP1 < IP2 ≤ 99999 (I5 Format)
IP1 > 0

For option 2

ZEIT (=Time) is the modifier name which causes the selection of the second option and the interpretation of the parameters T1 and T2 instead of IP1 and IP2

T1 is the time coordinate of the value of RECA which will become the first value of RECB

T2 is the time coordinate of the value of RECA which will become the last value of RECB (T1 < T2)

TFLOAT is a floating factor which is applied to T1 and T2 (multiplication) if the user wishes to use another unit. If T1 = 10. and T2 = 20., the system will transfer all the values comprised between 10.0 and 20.0 sec. for a value of TFLOAT equal to 1.0 or 0.0 (blank and 0.0 are replaced by the default option 1.). If the user has used a TFLOAT factor equal to 0.001 the two values will be interpreted as milliseconds.

Examples

WERT (TEMP,,T200) (1,200,)
Transfer the 200 first values of the record TEMP to build the record T200

WERT (TEMP,ZEIT,TCUT) (,,) (15.0,25.0,0.001)
Transfer the values of TEMP which are comprised between 15. and 25. ms to build the record TCUT

Remarks

- WERT requires a detailed knowledge of the parameters of the record RECA (number of points, sampling frequency) and it is recommended to the new users to use BILD as preceding command to facilitate the interpretation of any possible error.
- The output control values printed by WERT are the new values but the input values are the first values of one or two blocks of the record RECA.
- A WERT task can be terminated with an error code of type 4 if the number of blocks is not exceeded but if the filling factor of the last block is too small to allow the execution of the task.

XTSD

is a user specified command and the conventions must be given by the user which has programmed the EXTSED subroutine.
Following conventions are only indicative:

XTSD (RECA,WORD,RECD) (IB1,IB2,IX) (X1,X2,X3)

RECA is the name of the first input record
WORD is the name of the second input record or the name of a modifier
RECD is the name of the resulting record
IB1 is the first block to be processed
IB2 is the last block to be processed
IX is a sorting factor or a input/output unit number

X1,X2,X3 are user specified.

Two examples are given in the description of the subroutine EXTSED (chapter 2.9).

ZERS

(zerstören = destroy) destroys a record stored in the warehouse or clears the warehouse
Two forms are possible:

ZERS (RECA,,)

or

ZERS (,ALLE,)

RECA is the name of the record to be destroyed (the record must be stored in the warehouse to be destroyed)

ALLE (=all) is the modifier name which causes the second option to be selected. In that case all the records contained in the warehouse are destroyed, i.e. the warehouse is cleared.

Example

ZERS (CH22,,)
destroy the record CH22

ZERS (,A11E,)
destroy all the records of the warehouse

Remarks

The user needs to destroy only if there is a risk to exceed the capacity of the warehouse. Prior to such situations he should investigate the possibility to select a larger warehouse size. He should be aware that the destruction of a record implies a reorganization of the warehouse and that it is more efficient to destroy a record as soon as it has served his purpose in order to avoid the shifting of many following records. When several records have to be destroyed it is always more efficient to begin by the last record.

ZUST

lists all the command names (keywords) which are acknowledged by the system and lists the corresponding 8 character labels which are printed as heading of a task.

Example

ZUST

Remarks

ZUST is comprised of only a keyword and has no parameters.
ZUST gives also the date corresponding to the last version of the system and the user is advised to verify if the date of his handbook matches the information provided by ZUST.

3.3 Some special features in the reduction of data series

3.3.1 Synchronism of two records

The basic scheme for a SEDAP task is to obtain a record from the warehouse and to store the results of a specific mathematical operation into the warehouse by creating a new record. This operation involves an input record or a segment of the input record and the new record will derive his new parameters from the values of the warehouse parameters: date, time, frequency and number of points. If the user has specified a sorting factor and the transfer of only a record segment, the relation still exists after application of the frequency reduction and of the shift of the time origin. Some other tasks involve an operation performed on two input records,

for instance $C = A + B$

where A, B and C represent experimental records.
It is expected that when the user specifies such an operation, the two input records A and B will be synchronous. Two SEDAP records (or segments of records) will be synchronous if:

a) the two records have the same time origin. This must also be valid if the two selected segments are specified by a delimiter other than 1 and should be extended to the date.
b) the two records have the same sampling frequency, i. e. the time interval between two points of both records will be the same.
c) the number of points involved by the task is the same for the two records.

The synchronism of two input records is important since the delimiters and the sorting factor are specified for the two input records by a single set of values and since the transfer subsystem is mainly concerned with blocks. A special situation arises if two input records contain respectively e.g. 612 and 614 points. Both records will be accepted by the TRANSFER subsystem which has received 1 and 2 as delimiters. The transfer zone will however correctly compute the two numbers, of points (612 and 614) since the filling factors of the last blocks will be

respectively 100 and 102. The computing subroutine could theoretically choose between two possibilities:

- stop the process after 612 values and disregard the two last points which are present on only one of the records
- execute the operation up to the 614th value and since nobody knows what are the values stored into the last positions of the shorter record, the results can become at least unpredictable.

Such a situation would speak in favor of a radical solution (reject the task) but there are other situations where the lack of synchronism could be tolerated.

- An experimenter can justify the comparison (for instance subtraction) of two records which have the same frequency, the same number of points but two different time origins because he compares the runs of two different days.
- One would like to compensate the drift of an integrator by subtracting the ramp signal generated by DAGEN.

The most important thing is that the user should be aware of what he does and the following rules which reflect the duality of the previous considerations are applied to the detection of non synchronous records:

1) Any operation involving two input records supposes that the user has selected two synchronous records.
2) The synchronism of two records is always verified and the lack of synchronism does not cause an interruption of the task but a warning message will be issued if:
   - the two numbers of points are not identical
   - the two time origins are not the same
   - or if the two sampling frequencies are not identical.
3) The resulting record derives always its parameters from the first record listed in the command card.

The last point leads to suggest that the first record should always be the shorter record if the two records don't have the same length (it is obvious that the difference of length concerns only the case of the last block of a record when both

don't have the same filling factor). This is very easy to perform operations where the commutativity is accepted like for the addition or the multiplication of two records:

$$C = A + B = B + A$$
$$G = E * F = F * E$$

but requires some more care for other operations.

The user can find numerous ways to solve similar problems and although it should be considered as a very minor point the following examples are given to provide a few complementary explanations about the way how the system handles the records.

Assuming two records R612 and R614 which contain respectively 612 and 614 values:
The following operation

```
SUBT  (R612,R614,REST)  (1,2,1)
```

is easy because the shorter record is the first listed.

```
SUBT  (R614,R612,REST)  (1,2,1)
```

will be followed by a serious warning and can be replaced by the following list

```
AX+B  (R614,Z614)   (1,2,1)   (-1.0,0.0)
ADDI  (R612,Z614,REST)   (1,2,1)
```

where the operator AX+B with a = -1 and b = 0 has inverted the record to replace the subtraction by an addition which is commutative.

The following command

```
DIVI  (R614,R612,QUOT)   (1,2,1)
```

could be replaced by the following list

```
AX+B  (R612,,FRAM)   (1,2,1)   (0.0,1.0)
DIVI  (FRAM,R612,X612)   (1,2,1)
MULT  (X612,R614,QUOT)   (1,2,1)
```

which follows almost the same pattern. It is interesting to note that the operator AX+B has created a record filled with values equal to 1. but with the same parameters as the previous one. Such records filled with 1 or 0 are called "frame-records" because they carry only the former frame of the record and can be used in many different "tricky combinations".

There is obviously a shorter way to replace the two first lines of the previous example:

```
WERT  (R614,,D612)   (1,612,1)
```

with the last line changed to

```
DIVI  (D612,R612,QUOT)   (1,2,1)
```

The example of the compensation of the drift of an integrator can also illustrate the flexibility of the system if the user knows how to take advantage of the modularity. Any D.C. offset at the input of an integrator will cause a drift of the integration and must be compensated by a ramp. If the record to be compensated contains only 614 values,the record produced by DAGE with the modifier AX+B will simulate a ramp of 1024 values. Since the larger record is subtracted from the first one, the system will not issue a warning. The user could replace the subtraction by an addition by generating a negative ramp if the commutativity would be involved but a better solution would be to use the additive option of DAGE and to spare the intermediary record.

3.3.2 The sampling frequency

The digital data acquisition systems record the different state variables not as continuous signals but as sequences of points which are considered as equispaced. The experimenter must always be conscious of the sampled nature of the recording and this requires to treat the data reduction with some extra care. Most of the scientists are familiar with the applications of the Stroboscope which substitutes an apparent frequency to the real rotation,everybody knows the imperfections of an optical sampling like a cinematographic sequence which often gives an unsatisfactory representation of a motion (the wheels of the stage coaches seem always to challenge the motion's laws). But many experimenters disregard the importance of the sampling frequency in their own data reduction. This is mainly due to the fact that many experimenters have had considerable experience with the techniques of continuous analog recording where the inertia of the galvanometers have a strong limiting influence on the frequency bandwidth of the signals. If the

basic relationship between the sampling frequency and the signal bandwidth is not respected, the resulting record may be aliased./17,18,19/. The aliasing of a record is not only dangerous because of the inaccuracy of the results but especially by the fact that a serious aliasing can be interpreted as a new phenomenon which has nothing to do with the real experiment.

It is therefore recommended to pay the greatest attention to the sampling frequency at the different steps involved in the recording of the reduction of numerical values.

- The first step is to select a sampling frequency which is at least twice as high than the highest frequency one wishes to investigate. (A signal which must be evaluated up to 200 Hz could be sampled at 500 Hz).
- The experimenter must verify that the frequencies which represent a higher spectrum are correctly eliminated prior to the sampling process. That implies that the variable low pass filter be correctly adjusted in order to cut off the frequencies which exceed the folding frequency (250 Hz in the previous example).
- These preliminary steps are extremely important and must be followed by others which deal with the data reduction. The user must supply the correct sampling frequency when the records are passed by the input system (ERAK,PTAP). The value of the frequency is extremely important for the operation which involve the time interval (for instance integration or differentiation).
- Many users have a tendency to select the highest sampling frequency and justify this excess by saying that "one never knows" if a fast transient will not require such a high resolution. The discussion of this viewpoint does not belong to the frame of this report but it should be pointed out that such a "safe" viewpoint generally involves the use of a high sorting factor in the data reduction to compress the records into shorter ones. The use of the sorting factor necessarily implies that the setting of the low pass filter is no more

valid for the new "sorted frequency". It is therefore advisable to perform a smoothing of the record before attempting to reduce its length by a sorting factor.

3.3.3 Complex values

The values recorded by the data acquisition system are always real. When the recorded values are processed by a Fourier Transform the resulting values will be complex and build a complex record which must be treated with some special care. Like in most of the computing systems, complex data are stored as couples of scalar values, the first scalar value being the argument of the real part, the second one the argument of the imaginary part. The experimenter who directs the process of complex values generally knows the meaning of the operations he has planned and it is his responsibility to select the appropriate operators which are designed to handle such complex records. A complex record may be printed in a horizontal format (PBHE) but if the same record is directly plotted, the graph will be of little use because of the alternation of real and imaginary parts. The user who wants to plot a complex record will generally use a preliminary sorting task (SØ02) and plot separately the real and imaginary parts. The user will find the Fourier Package especially easy to handle and the modular structure allows to perform all the standard operations with a very good flexibility. He must however pay some attention to the following points:

- The commands which handle the complex records will generally disregard the sorting factor which is always set to one by the system itself. If the user, in a separate task specifies a sorting factor, he will obtain a new record which may be of no further use.
- The use of the command WERT (call by values) may also destroy the structure of a complex array. If the first value is even (2 to 127 for instance), the first value will be skipped and the imaginary parts will be stored where the real values were expected.

4. EVALUATION OF SEDAP

The first version of SEDAP was implemented in 1970 and was used for different tasks of data reduction involving up to several millions of sampled values. The system was enlarged to include the Fourier Package and a few other components /20, 21/. During the first months of 1972 the system was slightly modified to insure a better uniformity of the subroutines and a better efficiency and to allow the processing of larger records. This version was designed to form a complete and consistent package which includes not only the master deck with the listing but also the test runs, the user's handbook and the documentation with a detailed description of the commands and the associated procedures.[1)]

SEDAP has been widely used already in different experiments mainly related to thermodynamics or to the sodium technology (sodium boiling, simulation of fuel rod failures, performance of sodium loops etc.) The SEDAP approach has been so far considered as very successful /22/. An average experimenter can learn the SEDAP language in one or two hours and after a few runs he is able to conduct very delicate data reductions which otherwise would have required many days of programming work, should a conventional computing technique have been used. It is obvious that a user will need more time if he intends to acquire a perfect grasp of the system and if he tries to master all the tricky applications which are possible within a complex modular structure like SEDAP.

The use of experimental records, the very simple command language and the possibility to name the records have been found very valuable and are especially appreciated by the scientists who are not familiar with computing sciences. The modularity of the system has provided the expected versatility and most of the problems of data reduction were solved with the standard features of SEDAP without resenting the limitations of the system. It must be added that the SEDAP package includes a user's subroutine (EXTSED) which gives the

---

1) As time proceeds, SEDAP will of course be modified to accommodate more user wishes.

possibility to join a user written Fortran subroutine to the system in order to solve any specific problem which has not been treated under the organization of the official version.

SEDAP was also used for some applications which were not originally foreseen. The interfacing capabilities of the system were used to perform analysis of data which were produced by digital simulation programs and which were then analysed with the Fourier Package. The flexibility of the data generation provides many possibilities for the theoretical investigations of different types of signals and any experimenter can try the system in the dry run mode.

SEDAP was created according to some preliminary guidelines which were exposed in the first part of the present report and the system was progressively extended within the limits of the original frame. It would have been tempting during the development to change some details of the frame but this tendency was resisted because it was always possible to extend SEDAP without changing the shape of the basic scheme. The fundamental structure of the system is likely to remain actually stable. and that is the reason why the documentation of the program was undertaken at the present time. It is interesting however to summarize the few points where the frame has been found somewhat narrow and to discuss the improvements which could be contributed without great changes to the whole system. In other words a basic question can be formulated as the following: Should it be done again, would it be done the same way?

4.1 The command interpreter

It was already stated that the SEDAP language was designed in the rigid context of a Fortran input. The general scheme has been very satisfactory, but in some cases a few limitations have become apparent. One can refer to the example of two input records which block the use of a modifier or to the legitimate wish to have two modifiers. In some cases an option has to be passed as a decimal number because the three integers are already assigned.

It seems therefore that a more sophisticated command interpreter would greatly improve the system without changing the basic structure. The interpretation of the language might be preceded by a syntax check which would allow a detection of all syntactical errors before the execution. According to our experience, most of the errors are of trivial nature and we evaluate to more than 90 % the percentage of the errors which could be detected by a syntax check. This would involve the investigation of a catalog belonging to a "dummy warehouse" and would considerably alleviate the burden of the error checking procedures at the time of the execution. The efficiency could be increased further by introducing a compilation of the input language rather than an interpretation. It is interesting to mention that in such a case the record number would be substituted to the record name, the absolute address to the relative address etc. ... A great improvement could be achieved if the command interpretation would be executed in a time sharing environment with an interactive mode to allow an immediate correction. Another advantage of such a modification could be obtained by combining the results of the interpreter to a dynamic linkage. In that case only the necessary modules would be considered and many of the unnecessary elements could be dropped according to the list of the commands.

In many applications, where the same sequences of operations ought to be executed on a number of signals, the capability of defining and executing subroutines was found very desirable. Another method to solve the same problem would be the introduction of a macro facility.

4.2 Type dependent operations

One feature which was included in the very early planning for SEDAP and which was dropped later, was the introduction of various record types and the sensitivity of the operations with respect to these types. At present, the user is requested to use different commands for the multiplication of two records, whether they contain real data (signals in the timedomain) or

complex data (frequency spectra). If the warehouse catalog would be extended to include the appropriate type information, the same command syntax could be used in either case and many user errors could be avoided.

4.3 Size of the system

The size of the complete executable SEDAP load module amounts to around 240 K bytes. The last reorganization of the system brought it down from 300 K to 238 K and by trimming the overlay version the limit can be expected at about 200 K. Such a size is a compromise between the priority of a job and the input/output load for the present computer installation but can be a disadvantage for the smaller computers. The reduction of the size is possible with the use of the overlay version but the reduction factor is rather modest and this is mainly due to the large size of the common area which includes the catalog and the computing arrays. It has been explained that the use of a preliminary compilation would almost eliminate the catalog during the execution and we can add that most of the checking features which are scattered all over the system would be reduced in such a way that the combined savings can be estimated in the range of 25 - 35 K bytes.

The reduction of the computing arrays would obviously contribute further to the size reduction of the module. The size of the computing arrays was determined before the introduction of a systematic segmenting of the transfer operations. A reduction of the computing arrays would be perfectly feasible for most of the commands with a penalty on the input/output efficiency which is not the sensitive issue of small configurations. Two SEDAP complexes would however be seriously offended by such a drastic change, they are the sorting subroutine and the Fourier Package. The sorting subroutine could be easily modified to sort 2^n channels by a succession of elementary steps applied with lower factors like 2 and 4. SØ08 will be iteratively treated as a 4 and 2 cascade, SØ64 would require the triple cascade 4, 4 and 4. It would be also possible to reduce the size of a block to 256 or 128 and this

reduction can be combined with the previous step. In the two cases the results would influence very negatively the performance of the input/output operations.

The reduction of the computing arrays would be more disastrous on the Fourier Package which cannot be implemented with smaller arrays without loosing either the capability to treat the presently rather high number of frequencies or the advantages of the Fast Fourier Transform. The involved segmentation would require a bulky bookkeeping and tremendously increase the computing time . Hence, the reduction of the computing arrays would most likely go along with a reduction in the frequency range of the FFT.

From the previous considerations it is clear that a separate handling of the command interpretation and of the execution would bring a considerable improvement for the user and an interesting reduction of the size of the module. Further reductions would be more difficult to justify and narrowly depend upon the types of configurations on which SEDAP is run as well upon the types of processed records (length in 1000 or millions - with or without Fourier Package etc.). The version which has been documented represents a good compromise for the machines which are generally available in the scientific computing centers.

4.4 Data management

There are two areas of data management to be considered:
a) the computing arrays and b) the warehouse.

As mentioned above, the computing arrays might be reduced in size if one is willing to accept more input/output operations and a reduction of the frequency range which can be handled by the FFT. Since in the present version the computing arrays are located in COMMON, almost all source programs must be modified and recompiled for such a modification. This would be a nontrivial job and would be acceptable only for the implementation on another computer installation. However, even on the same installation a more flexible version would be desirable

in order to save core space and to gain priority for SEDAP jobs which do not require the full capability. This would require a modification in such a way that all subroutines would obtain their computing arrays through the argument list and would use adjustable dimensions.

The present data management in the warehouse is extremely simple. The warehouse is always filled consecutively with the experimental records, which are kept contiguous to each other even if intermediate records are to be scratched. A great deal of input/output is required for copying of data in this case. A new version of SEDAP would certainly contain an address table in the catalog which would permit scattered storage of the records in the warehouse.

4.5 Conclusion

Like for many software projects the time which was necessary to develop and to document SEDAP has exceeded the original estimation. This is partly due to the fact that the system has been welcome in its early stage and that because of the favourable resonance of the user's group, it was decided to adapt the package several times to user wishes and to include a complete detailed documentation of SEDAP. One of the main achievments in dealing with this problem oriented computing application was the excellent cooperation between the scientists involved in the experimental work and the designers of the system. The aim of the system was to obtain a better quality and a better efficiency of the data reduction which is one of the most important problems of the research work in the field of the fast breeder project. The target has been reached and the authors are indebted to the different users for their outstanding cooperation and for the numerous discussions which have contributed to the progress of the system. The assistance of G. Rittirsch who is responsible for the data acquisition system is especially acknowledged.

References

/1/ Hoare C. A. R.: Record Handling in Programming Languages, F. Genuys Academic Press London and New York 1968

/2/ Eggenberger O.: A8FORM Programm zum Ausdrucken von Texten im Großformat, Programmbeschreibung 230-GFK (not published)

/3/ RANDU Subroutine in System /360 Scientific Subroutine Package (360A-CM-03X) Version III, Programmer's Manual (H20-0205-3), IBM

/4/ Heine S. et al.: PLOTA Ein allgemeines Plotprogramm, Programmbeschreibung Nr. 117, GFK (not published)

/5/ Leinemann K.: PLOTA FORTRAN-IV-Routinen zur Umsetzung von 1130-PLOTA-Aufrufen in Aufrufe der CALCOMP-Software, IRE-Programmbeschreibung Nr. 96/1972, GFK (not published)

/6/ Cooley J. W. et al.: Application of the Fast Fourier Transform to Computation of Fourier Integrals, Fourier Series, and Convolution Integrals. IEEE Trans. Vol. AU-15, No. 2, June 1967, Special issue on the Fast Fourier Transform

/7/ Webb C.: Practical Use of the Fast Fourier Transform (FFT) Algorithm in Time-Series Analysis, Texas Univerity, Austin Texas, June 1970, AD713166

/8/ Cochran W. T., Cooley J. W. et al.: What is the Fast Fourier Transform? IEEE Trans. Audio and Electroacoustics AU-15, 45-55, June 1967

/9/ Kremer H.: Praktische Berechnung des Spektrums mit der Schnellen Fourier-Transformation, electronische datenverarbeitung 6/69

/10/ Gentleman W. M. and Sande G.: Fast Fourier Transform - For Fun and Profit, 1966 Fall Joint Computer Conf. AFIPS Proc., Vol. 29, Washington, Spartan Books, pp. 563 - 578, 1966

/11/ Brenner N.: Cooley-Tuckey Fast Fourier Transform FOUR1 IBM Corp. PID, 40 Saw Mill River Road, Hawthorne, New work 10532, Progr. Order Number: 360D-13.4.002 , 1968

/12/ Bingham C. et al.: Modern Techniques of Power Spectrum Estimation, IEEE Trans. Audio and Electroacoustics, Vol. AU-15, pp. 56 - 66, June 1967

/13/ Welch P. D.: The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms, IEEE Trans. Audio and Electroacoustics, Vol. AU-15, pp. 70 - 73, June 1967

/14/ Stockham T. G.: High speed convolution and correlation, 1966 Spring Joint Computer Conf., AFIPS Proc. vol. 28. Washington, Spartan Books, pp. 229 - 233, 1966

/15/ Tack P.: Program to Compute Correlation Coefficients, Spectral Density Functions and Cross Spectral Density Functions, Kernforschungszentrum Karlsruhe, Germany, KFK 1237, July 1970

/16/ Bendat J. S.: Principles and applications of random noise theory, Wiley, New York,1968

/17/ Audoux M.: Grundlagen der digitalen Erfassung primär analoger Meßwerte, Kernforschungszentrum Karlsruhe, Germany, Ext. Bericht 8/69-6, Nov. 1969

/18/ Truxal J. G.: Automatic Feedback Control System Synthesis, New York, 1955

/19/ Rittirsch G.: Kriterien zur Wahl der Meßfilter und Abtastfrequenz sowie Methoden zur Meßfehlerkorrektur, angewandt bei Temperaturmessungen in Natrium, Kernforschungszentrum Karlsruhe, Germany, Ext. Bericht 8/71-3, 1971

/20/ Audoux M.: SEDAP A Systematic Approach to the Processing of Experimental Data, Proc. ifip congress 71, Booklet TA-6, North-Holland, Publishing Co., Netherlands P.O. Box 211, Amsterdam

/21/ Audoux M., Katz F.W., Schlechtendahl E.G.: SEDAP Rechnergestützte Auswertung technischer Versuche, KFK-Nachrichten 3/71, Kernforschungszentrum Karlsruhe

/22/ Audoux M., Katz F., Rittirsch G.: Praktische Erfahrungen mit einem modularen leicht programmierbaren System zur Auswertung von Meßsignalen (SEDAP), Angewandte Informatik 8/72

Appendix A

Job Control Cards for SEDAP

```
//SEDAP     PROC BAND=NULLFILE
//L         EXEC PGM=IEWL,COND=(4,LT),PARM=OVLY
//SYSLIN    DD   DSN=DATA.IRE(SEGVLY),DISP=SHR
//          DD   DDNAME=SYSIN
//SYSLIB    DD   DSN=SYS1.FORTLIB,DISP=SHR
//          DD   DSN=GFK.FORTLIB,DISP=SHR
//          DD   DSN=LOAD.IRE,DISP=SHR
//LOAD      DD   DSN=LOAD.IRE,DISP=SHR
//PLOT      DD   DSN=LOAD.CALCOMP,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   UNIT=(CTC,,DEFER),LABEL=(,NL),
//          DCB=(BLKSIZE=968,LRECL=121,RECFM=FBM)
//G         EXEC PGM=*.L.SYSLMOD,COND=(5,LT)
//FT05F001  DD   DDNAME=SYSIN
//FT06F001  DD   UNIT=(CTC,,DEFER),LABEL=(,NL),
//          DCB=(BLKSIZE=931,LRECL=133,RECFM=FBA)
//FT15F001  DD   UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(1680,(20,1)),
//          DCB=(BLKSIZE=1680,LRECL=80,RECFM=FB)
//FT40F001  DD   UNIT=SYSDA,DISP=(NEW,DELETE),SPACE=(2048,(2500))
//PLOTTAPE  DD   UNIT=(TAPE9,,DEFER),LABEL=(,NL),DSN=&BAND,
//          VOL=(,RETAIN,SER=(&BAND))
//          PEND
```

Appendix B

EXAMPLE

FOLGENDE BEFEHLE SIND IN DIESEM JOB ENTHALTEN

```
SEDAP      NSK
AUSWERTUNG DES NSK VERSUCHES NR.1 VOM 19.APRIL 1971.GENAUE BESCHREIBUNG SIEHE :
EXTERNER BERICHT 8/71-3 G.RITTIRSCH - INSTITUT FUER REAKTORENTWICKLUNG AUGUST 71
SEDA                        1000
ERAK                  DATE     3  130   21 10000.0   2805.71
SO16       DATE       DA$$     1  256    9
AX+B       DA06       AX06     1   16          0.005      0.0
TNI1       AX06       CX06     1   16
DIFF       CX06       EX06     1   16
AX+B       EX06       FX06     1   16          0.0142     0.0
ADDI       FX06 CX06  IX06     1   16
FIL2       CX06       JX06     1   16
FIL2       IX06       KX06     1   16
BILD
DEFX                                             5.8      6.57         25.4
DEFY                                           650.      800.          17.8
PLOT       JX06 TEXT           1   16
 NSK-VERSUCH NR.1 /VOM 29.4.71/ MESS.-ST. T16 MIT TEMP.-KORREKTUR
PLOT       KX06 ALT*           1   16
> ES FOLGT EINE FEHLERHAFTE SEDAP-ANWEISUNG
ADDI       KX06 NYDA ERR       1    1
STOP
```

ENDE DER EINGABEBEFEHLE

```
**************                                                                  DATUM = 20.12.72
*            *                                                                  ZEIT  = 11.04.11
*   AX + B   *
*            *
**************

 BEFEHL WAR WIE FOLGT CODIERT :

AX+B       DA06      AX06      1   16    0    0.0050    0.0       0.0
____       ____ ____ ____ *   |*   |*   |*    .   |*    .   |*    .   |

 AUFTRAG IST WIE FOLGT WEITERGELEITET WORDEN :

 VON BLOCK   1  BIS BLOCK   16  SOLLEN   16  BLOECKE  DES EXPER. RECORDS DA06    TRANSFORMIERT WERDEN
  UND DEN EXPER.RECORD AX06   BILDEN



DIE TRANSFORMATION ERFOLGT MIT EINEM OPERATOR VOM  TYP :
AX+B (LINEAR VERSCHIEBUNG)

 KONTROLLWERTE INPUT = 0.425293E 04 0.424804E 04 0.424804E 04 0.425293E 04 0.425293E 04 0.425293E 04 0.425293E 04 0.425293E 04
KONTROLLWERTE OUTPUT = 0.212646E 02 0.212402E 02 0.212402E 02 0.212646E 02 0.212646E 02 0.212646E 02 0.212646E 02 0.212646E 02

DIE WERTE SIND UNTER DEN NAMEN AX06 ADDRESSIERBAR


**                                                                  STEP UM 11.04.12 BEENDET
**  AUFTRAG ERFUELLT **
**                                                                  BENOETIGTE CPU-ZEIT:    0.1132 SEK
```

```
**************                                                                  DATUM = 20.12.72
*            *                                                                  ZEIT  = 11.04.28
*  ADDIEREN  *
*            *
**************

 BEFEHL WAR WIE FOLGT CODIERT :

ADDI       FXO6 CXO6 IXO6      1   16     0     0.0         0.0         0.0
____       ____ ____ ____ *    |*   |*    |*     .    |*     .    |*     .    |

 AUFTRAG IST WIE FOLGT WEITERGELEITET WORDEN :

 VON BLOCK   1  BIS BLOCK   16  SOLLEN   16  BLOECKE  DER RECORDS FXO6 UND CXO6  TRANSFORMIERT WERDEN
  UND DEN EXPER.RECORD IXO6   BILDEN



DIE TRANSFORMATION ERFOLGT MIT EINEM OPERATOR VOM  TYP :
 ADDIEREN ZWEIER DATEIEN

 KONTROLLWERTE INPUT = -.507669E 01 -.253834E 01 0.253834E 01 0.253834E 01 0.0          0.0          0.0          0.0
 KONTROLLWERTE INPUT = 0.514530E 03 0.513958E 03 0.513958E 03 0.514530E 03 0.514530E 03 0.514530E 03 0.514530E 03 0.514530E 03
KONTROLLWERTE OUTPUT = 0.509453E 03 0.511419E 03 0.516496E 03 0.517068E 03 0.514530E 03 0.514530E 03 0.514530E 03 0.514530E 03

DIE WERTE SIND UNTER DEN NAMEN IXO6 ADDRESSIERBAR


**                                                                  STEP UM 11.04.32 BEENDET
**  AUFTRAG ERFUELLT **
**                                                                  BENOETIGTE CPU-ZEIT:    0.1531 SEK
```

****
*   *
* LAG.BILD *
*   *
****

DATUM = 20.12.72
ZEIT = 11.04.38

BEFEHL WAR WIE FOLGT CODIERT :

```
BILD                      0    0    0    0.0       0.0       0.0
____      ____ ____ ____ *  |*   |*   |*   .   |*   .   |*   .   |
```

AUFTRAG IST WIE FOLGT WEITERGELEITET WORDEN :

LAGER ZUSTAND DRUCKEN

FOLGENDE 17 EXPERIMENTAL RECORDS SIND IM LAGER GESPEICHERT

INDIZ	NAME	ANFANG	ENDE	BLOCK	FREQ	DATUM	UHRZEIT	PUNKTE	FUELLFAKTOR
1	DATE	1	256	256	10000.000	2805.710	0.0	131072	512
2	DA01	257	272	16	625.000	2805.710	0.0	8192	512
3	DAC2	273	288	16	625.000	2805.710	0.0	8192	512
4	DA03	289	304	16	625.000	2805.710	0.0	8192	512
5	DAC4	305	320	16	625.000	2805.710	0.0	8192	512
6	DA05	321	336	16	625.000	2805.710	0.0	8192	512
7	DA06	337	352	16	625.000	2805.710	0.0	8192	512
8	DA07	353	368	16	625.000	2805.710	0.0	8192	512
9	DAC8	369	384	16	625.000	2805.710	0.0	8192	512
10	DAC9	385	400	16	625.000	2805.710	0.0	8192	512
11	AXC6	401	416	16	625.000	2805.710	0.0	8192	512
12	CXC6	417	432	16	625.000	2805.710	0.0	8192	512
13	EXC6	433	448	16	625.000	2805.710	0.0	8192	512
14	FXC6	449	464	16	625.000	2805.710	0.0	8192	512
15	IXC6	465	480	16	625.000	2805.710	0.0	8192	512
16	JXC6	481	496	16	625.000	2805.710	0.0	8192	512
17	KXC6	497	512	16	625.000	2805.710	0.0	8192	512

**
** AUFTRAG ERFUELLT **
**

STEP UM 11.04.38 BEENDET

BENOETIGTE CPU-ZEIT: 0.0399 SEK

```
**************                                                    DATUM = 20.12.72
*            *                                                    ZEIT  = 11.04.39
*  PLOTTEN   *
*            *
**************

 BEFEHL WAR WIE FOLGT CODIERT :

PLOT      JX06 TEXT          1   16    0    0.0       0.0       0.0
----      ---- ----  ---- *   |*   |*   |*    .   |*    .   |*    .   |

 AUFTRAG IST WIE FOLGT WEITERGELEITET WORDEN :

 VON BLOCK   1  BIS BLOCK   16  SOLLEN   16  BLOECKE  DES EXPER. RECORDS JX06  GEPLOTTET WERDEN
 ES WIRD EIN NEUER PLOT ERGEFFNET

 KONTROLLWERTE INPUT = 0.514244E 03 0.514243E 03 0.514301E 03 0.514301E 03 0.514415E 03 0.514530E 03 0.514530E 03 0.514530E 03

 SKALENBESCHRIFTUNG :  NSK-VERSUCH NR.1 /VOM 29.4.71/ MESS.-ST. T16 MIT TEMP.-KORR

 DIE KURVE HAT FOLGENDE DIMENSIONEN :
                                                  XMIN = 0.580000E 01
                                                  XMAX = 0.657000E 01
                                                  YMIN = 0.650000E 03
                                                  YMAX = 0.800000E 03


**
**  AUFTRAG ERFUELLT **                                           STEP UM 11.04.43 BEENDET
**
                                                                  BENOETIGTE CPU-ZEIT:    0.2729 SEK
```

```
**************                                                    DATUM = 20.12.72
*            *                                                    ZEIT  = 11.04.44
*  ADDIEREN  *
*            *
**************

 BEFEHL WAR WIE FOLGT CODIERT :

ADDI      KXC6 NYDA ERR      1    1    0    0.0       0.0       0.0
____      ____ ____ ____ *   |*   |*   |*    .    |*    .    |*    .    |

 AUFTRAG IST WIE FOLGT WEITERGELEITET WORDEN :

 VON BLOCK    1  BIS BLOCK     1  SOLLEN     1  BLOECKE  DER RECORDS KXC6 UND NYDA  TRANSFORMIERT WERDEN
  UND DEN EXPER.RECORD ERR     BILDEN


DIE TRANSFORMATION ERFOLGT MIT EINEM OPERATOR VOM  TYP :
 ADDIEREN ZWEIER DATEIEN

FEHLER AUFGETRETEN (FEHLER CODE - IERR =    8 )

>>> FEHLERERKLAERUNG :

ES WURDE DER EXPERIMENTAL RECORD NYDA VERLANGT, OBWOHL ER NICHT IM LAGER IST


AUFTRAG NICHT ERFUELLT


DIESER SCHWERWIEGENDE FEHLER VERHINDERT EINE  SINNVOLLE WEITERFUERUNG DES PROGRAMMES UND VERURSACHT  DIE SEDAP UNTERBRECHUNG

                    ****************
                   *                *
                  *                  *
                 *                    *
                *      ERROR           *
               *                        *
              *                          *
             *                            *
            *                              *
           *                                *
          *                                  *
```