NASA CR-
141597

AEROPHYSICS RESEARCH CORPORATION
TECHNICAL NOTE

JTN-10
VOLUME II

THE DLG PROCESSOR - A DATA MANAGEMENT EXECUTIVE FOR THE ENGINEERING DESIGN INTEGRATION (EDIN) SYSTEM

VOLUME II - PROGRAMMERS' MANUAL

(NASA-CR-141597) THE DLG PROCESSOR: A DATA MANAGEMENT EXECUTIVE FOR THE ENGINEERING DESIGN INTEGRATION (EDIN) SYSTEM. VOLUME 2: PROGRAMMERS' MANUAL Final Report, Jun. 1973 - Dec. 1974 (Aerophysics Research Corp., G3/61
N75-17123
Unclas
09644

By: C. R. Glatt and W. N. Colquitt

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Johnson Spacecraft Center
Houston, Texas 77058

December 1974

AEROPHYSICS RESEARCH CORPORATION
TECHNICAL NOTE

JTN-10
VOLUME II

THE DLG PROCESSOR -
A DATA MANAGEMENT EXECUTIVE FOR THE
ENGINEERING DESIGN INTEGRATION (EDIN) SYSTEM

VOLUME II - PROGRAMMERS' MANUAL

By: C. R. Glatt and W. N. Colquitt

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Johnson Spacecraft Center
Houston, Texas 77058

December 1974

1. Report No.
NASA CR-

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle
THE DLG PROCESSOR - A DATA MANAGEMENT EXECUTIVE FOR THE ENGINEERING DESIGN INTEGRATION (EDIN) SYSTEM. VOLUME II - PROGRAMMERS' MANUAL

5. Report Date
December 1974

6. Performing Organization Code

7. Author(s)
C. R. Glatt and W. N. Colquitt

8. Performing Organization Report No.
JTN-10 VOLUME II

9. Performing Organization Name and Address
Aerophysics Research Corporation
Houston, Texas 77058

10. Work Unit No.

11. Contract or Grant No.
NAS9-13584

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Johnson Spacecraft Center
Houston, Texas 77058

13. Type of Report and Period Covered
Cost Plus Fixed Fee

14. Sponsoring Agency Code
EX42

15. Supplementary Notes
Final Report

16. Abstract

The DLG Processor is a Univac 1100 series Exec 8 computer program designed to read, modify, manipulate and replace symbolic images. DLG is controlled by a set of user supplied directives which augment the data being processed. A number of data management functions can be performed that include the construction of input data files, data base maintenance and control of program sequencing.

17. Key Words (Suggested by Author(s))
Data management, design, data base, EDIN, language

18. Distribution Statement
Unclassified - Unlimited

19. Security Classif. (of this report)
Unclassified

20. Security Classif. (of this page)
Unclassified

21. No. of Pages
178

22. Price*

*For sale by the National Technical Information Service, Springfield, Virginia 22151

PREFACE

This report describes a computer program called The DLG Processor - A Data Management Executive for The Engineering Design Integration (EDIN) System. The program was written in support of NASA Contract NAS9-13584, "Extended Optimal Design Integration (Extended ODIN) Computer Program." The study was conducted during the period from June 1973 through December 1974, with funds provided by the National Aeronautics and Space Administration, Johnson Spacecraft Center, Engineering Analysis Division. Mr. Robert W. Abel was the technical monitor. The contract was monitored by the Launch Analysis Section. The report is presented in two volumes:

VOLUME I - Engineering Description and Utilization Manual

VOLUME II - Programmers' Manual

The report specifically describes a user-developed data processor which is integrated with the Univac 1100 executive system and is interfaced to the EDIN data base.

TABLE OF CONTENTS

PRECEDING PAGE BLANK NOT FILMED

TABLE OF CONTENTS

TABLE OF CONTENTS

THE DLG PROCESSOR -

A DATA MANAGEMENT EXECUTIVE FOR

THE ENGINEERING DESIGN INTEGRATION (EDIN) SYSTEM

VOLUME II - PROGRAMMERS' MANUAL

By: C. R. Glatt and W. N. Colquitt
Aerophysics Research Corporation

SUMMARY

The DLG Processor is a Univac 1100 series Exec 8 computer program designed to read, modify, manipulate and replace symbolic images. DLG is controlled by a set of user supplied directives which augment the data being processed. A number of data management functions can be performed that include the construction of input data files, data base maintenance and control of program sequencing. Functions are illustrated in figure 1.

The primary purpose of the DLG Processor is to link one application program to another through a common information source. The procedure is to read output data from one applications program, insert a selected subset of the data into a structured data base and then selectively extract this and other stored data and place it into the input stream for other applications programs.

A considerable capability for manipulating data files is available with DLG which is not available from any other processor. DLG currently has about 20 directives implemented but the basic commands are 'CREATE' for the construction of a new data base element, 'PROCESS' directive which is used to process special output files from an application program, the 'ADD' command for generating or modifying data in the data base and the replacement function which uses a retrieval technique that substitutes delimited data base names for the current values in the data base. Many other useful data manipulation directives are available to the user. Arithmetic expressions are available as part of the language.

The overall design of the computer program has allowed its integration into the Exec 8 environment in an extremely sophisticated manner. The program loads in less than 20,000 words and

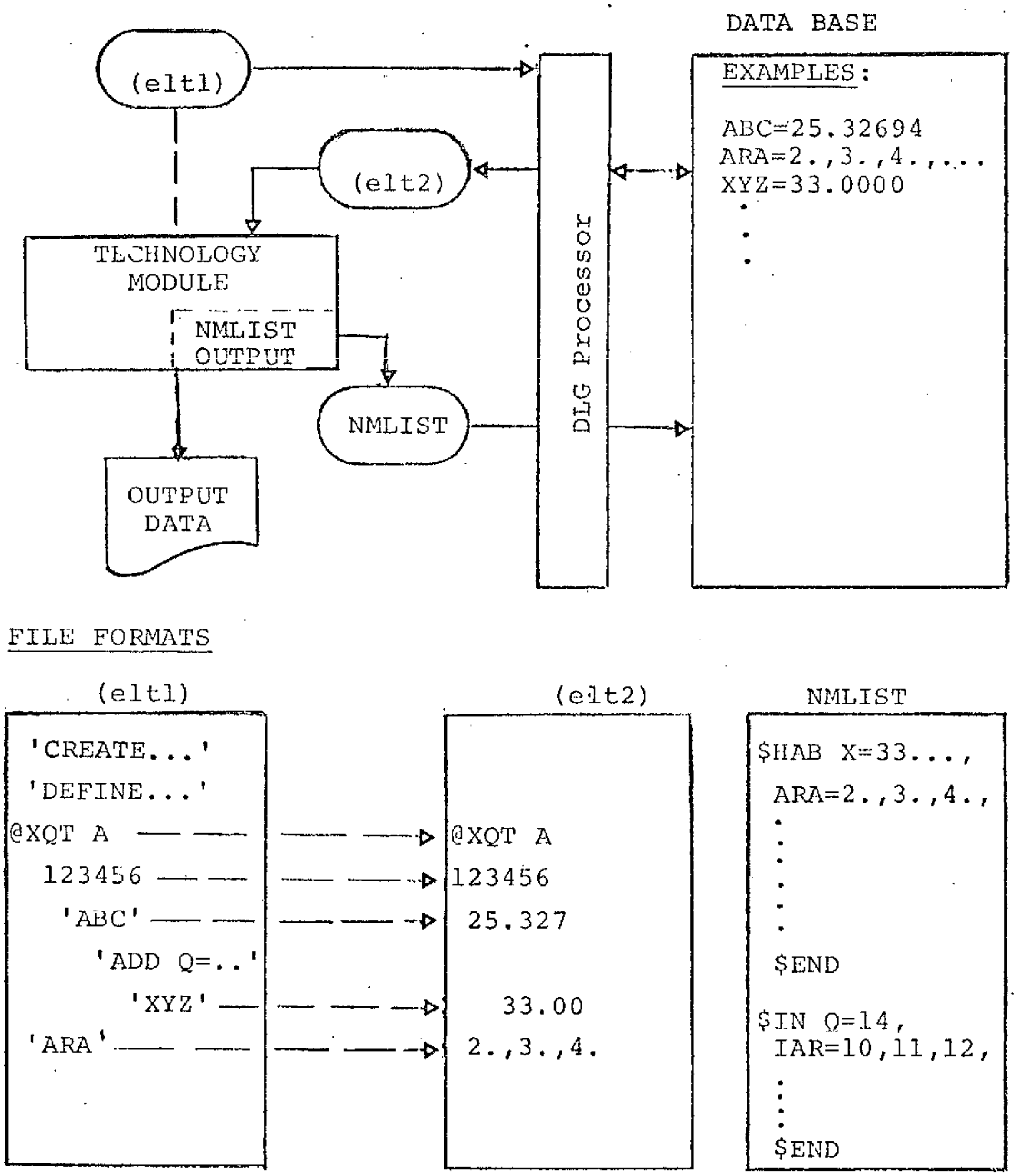


FIGURE 1 DLG PROCESSOR FUNCTIONS.

uses dynamic core allocation to minimize the impact of using large data bases. It uses double blocked buffering to read and write data and manipulates character strings in an extremely efficient manner, thereby reducing the preprocessing overhead to a minimum. It effectively allows looping in control streams, thereby offering a capability not previously available on this standard Exec 8 system. Finally, it can be used just as effectively in the demand as in the batch mode of operation.

INTRODUCTION

The EDIN system provides a balance of data management techniques which consider the inherent capabilities of the computer operating system, past efforts in the storage and retrieval of stratified data and the recent development of some flexible paging techniques for the transfer of information between the computer core and the mass storage of the computer. The Univac Exec 8 system provides the resources for the storage of large complex data files, for the storage and retrieval of the files and for the cataloguing protection and backup of the files. The executive system has several processors with instruction sets for manipulating the data retained in mass storage. A limitation on the operating system capabilities arises in accessing the subfile level of information in the system files once the file is addressed.

The EDIN data management system is designed to subdivide the files in a manner that will allow the data which is retained in mass storage to be accessed at any level from the single parameter level to a large matrix of data. Rather than constructing an extensive single computer program that attempts to be everything to everyone, the EDIN data management system provides a three-level data management capability. This approach permits the individual designer using the system to make his own decisions with regard to the storage method and techniques. It also permits the flexibility of using existing data sources not specifically created for EDIN.

The three levels of the EDIN data management system are built upon one another as illustrated in figure 2. The lowest level deals with the interface between the data in mass storage and the computer operating system. The file level of the data management system is provided by the Exec 8 software and consists of the file utility processor FURPUR, the file administration processor SECURE and other system level processors. The system processors are accessed using Exec 8 control statements. Therefore, file level software may be used directly by the designer for transmitting large structured blocks of data or the files

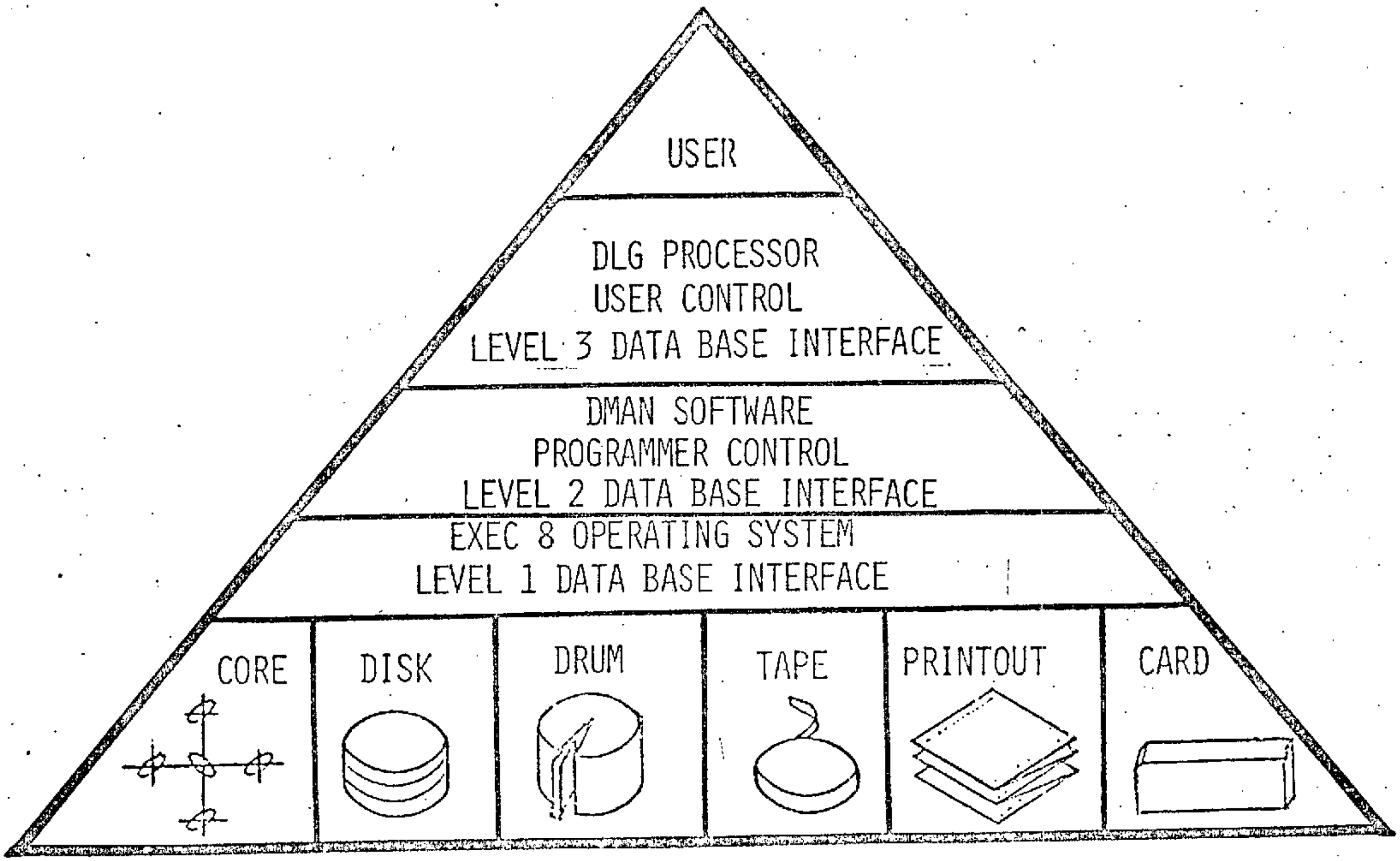


FIGURE 2 EDIN DATA MANAGEMENT SYSTEM.

themselves to be accessed by the programmer who seeks economy above all else. The file level constitutes the foundation for all higher level data management components.

The second level of the EDIN data management system provides the mechanism whereby the files can be organized into blocks of data called pages. Pages of information can be organized in a number of ways and names can be given to each page. A pointer system or directory is maintained by a Fortran callable software package, called DMAN, a subroutine utility package maintained in the EDIN library.

The third and highest level of the data management system is provided to make the system more usable to the designer who may not be a programmer. The capability is provided in the DLG processor which is designed to maintain a data base of stratified information, the stratified data can be selectively accessed and merged with the input stream of the EDIN technology programs. This level also provides the interactive language structure which allows the designer to sit at a remote terminal and interact with the data base directly as he develops a design. The DLG processor also contains routines for processing the output from the technology programs for the storage of design information in the data base.

Although the user may access the data base through any of the three levels, it is the lowest level maintained by the Exec 8 system which actually stores and retrieves the data. Exec 8 handles all of the underlying data management functions including file assignments, file directories and maintenance and security procedures as well as the data block transfer to and from mass storage. The Exec 8 system is discussed in reference 1, and a thorough treatment of the first level data management is provided by Univac in the appropriate User Documentation. This document deals primarily with the third level of the EDIN data management system (i.e. the DLG Processor).

However, the second level is a general software package which can be used in any program and is specifically applied to the DLG program for accessing the data base pages in which stratified design data is stored. Therefore, some discussion of DMAN is presented here.

PROGRAM STRUCTURE

The DLG Processor consists of a main driver routine (DIALEK) which controls the initialization and the selection of the processor functions illustrated in figure 2. The three major functions are data base interrogation (INMOD), data management (RSPOND) and data storage (NLADD). The functions are defined by a directive language which is read and interpreted by the program. After each directive is processed, control is returned to DIALEK and another directive is read. Processing is continued until another control statement is encountered.

Concepts and Definitions

The following concepts and definitions which may be new to the reader will be helpful in understanding this document:

Processor

An absolute program element which is executed with a special Exec 8 processor control statement:

```
@name elt1,elt2
```

and which is interfaced with the elements named on the processor control statements.

Data Base

File of information which is subdivided into named pages of data accessible by the DLG processor. Each page is further subdivided into named parameters and arrays.

Technology Module (Application Program)

An independent computer program which will receive or generate data base information.

Interrogation

The process of retrieving information from the data base. The disposition of the retrieved data is dependent upon the directive employed.

Directive (Also Command)

A language element used to specify a DLG Processor's action or function.

Data Management

A class of DLG functions which control and manipulate data base information. These functions include the creation of data base pages, the adding and defining information in the data base, printing and many others.

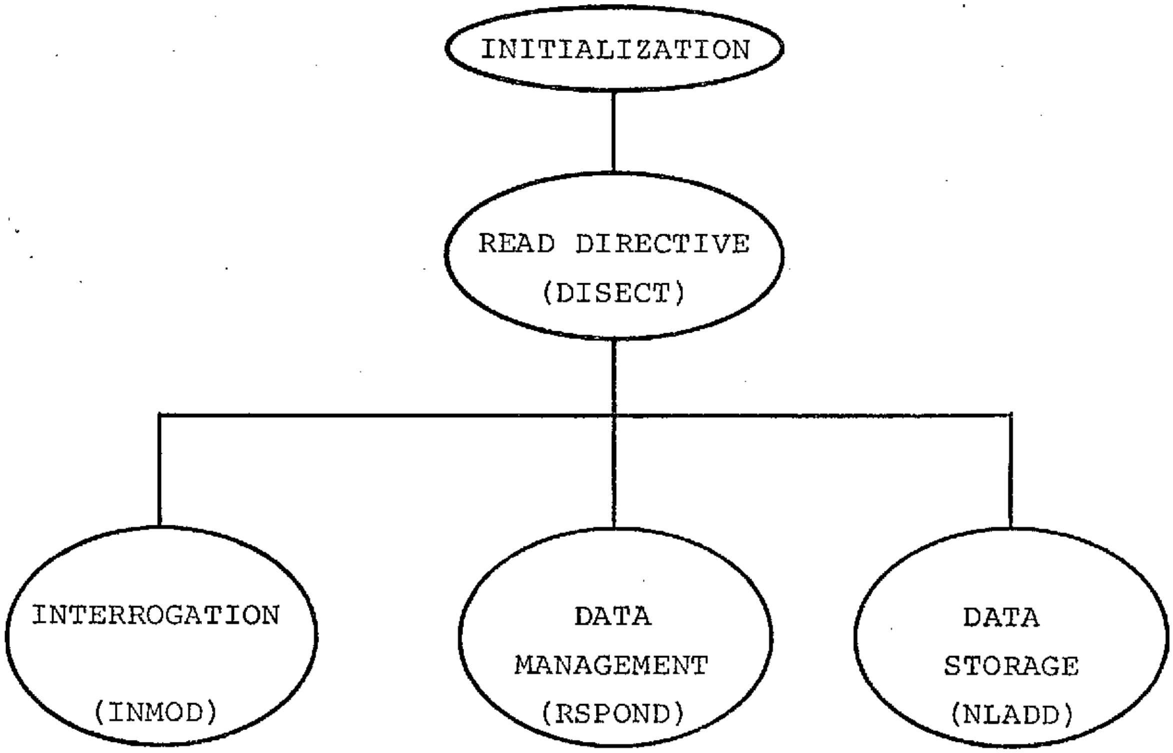


FIGURE 3 PROGRAM STRUCTURE.

Data Storage — A special class of data management functions which are designed specifically to store data generated by a technology module.

Run Stream — A sequence of data images which constitute a computer run.

Partial Run Stream — A portion of a run stream which can be merged at any point in the run stream through an @ADD control statement.

File Designations

Unit 5 — The System Card Reader.

Unit 6 — The System Printer.

Unit 14 — Temporary Data File for Incoming Data Base Data.

Unit 25 — Internal Logical Unit usually Attached to the EDIN Design Data Base.

Processor Specifications

Control Statement. -

```
@DLG.DLG,options lfn.eltl,lfn.elt2
     lfn.eltl       Source Input (See I Option)
     lfn.elt2       Source Output.
```

Option Specifications. -

I — Source input will follow the processor card. Source output will be placed in eltl.

L — Source input data will be listed.

O — Source output data will be listed.

D — Card cracking information will be listed.

E — Solicitation and result of directives will be printed.

S — List interrupt mode will be invoked.

M New data base files will be generated with this execution.

B Build option will be invoked. This option specifies that all data directives of the form:

```
'name name=value---
```

or

```
$name name=value---
```

This will permit the addition of data to the data base regardless of the directive name. Otherwise, only those data base variable names, which were previously defined in the data base, will be updated unless the data directive name is ADD or DEFINE.

The B option may not be invoked via the "ON" command. If desired, it must be present on the processor call card.

Syntax Definition. -

name	Must be six (6) or less alphanumeric characters and begin with an alphabetical character.
' (quote or prime)	The DLG delimiter. Strings that occur between pairs of delimiters will be processed by DLG. Strings external to primes will be passed "as is" into the output element.
-	The underline on a command indicates an optional character string which may be used as a directive.
value	Indicates a data base value in real, integer or hollerith format.
i,j,k	Indicates integer constants used in the directives.
elt	Exec 8 file element name in program file format.
lfn	Exec 8 logical file name in system data format.
text	Textual information.
[ ]	Indicates optional items on the line.

Summary of DLG Directives. - The DLG directives are summarized below. Underlines are optional character strings. All commands are excluded data base names.

'name'	Replace name with information from the data base.
'ADD	Replace specified information in the data base.
'CHANGE	Change values in common IDLOG.
'COMMENT or '.	User description with null effect.
'CREATE	Create a new data base.
'CSF or 'ER	Submit executive control statement.
'DBLIST	Print the names of all random access data bases on the data base file.
'DEFINE	Place description in data base directory.
'FORMAT	Format free data base information in place.
'INSERT	Insert binary SDF data in place.
'ON	Mode activation.
'OFF	Mode suppression.
'PRINT	Print data base information.
'USE	Specify a circular data base search.
'UPDATE	Update a specified data base.

Descriptions of Control Directives. -

'ADD name' - Specifies that information will be added to data base.

```
'ADD name=value'

'ADD name=name'

'ADD name=value,value,---'

'ADD name=name,name,---'

'ADD name=name op name, name op value,---'
```

where op =

+	Add
-	Subtract
/	Divide
*	Multiply
**	Exponentiation

'CHANGE number=value' - Using the integer number 'number' as an index into the master common block, IDILOG, the current value is replaced by 'value.'

'COMMENT ______' - This is a null card and is discarded by DLG.

'CREATE name,DIRLEN=number,LENDES=number,LTOTAL=number' - The data of name 'name' is brought into existence on the data base file. Optional parameters are DIRLEN - the directory length (This should be a prime number.).

LENDES - Length in computer words of the description.

LTOTAL - Total size, in computer words, reserved for the data base.

'CHANGE' -

Example 'CHANGE 27=3'

Location 27 of the common block IDILOG will have its value replaced by an integer 3.

'COMMENT' - A null card. The delimited field is removed from the card. If the resulting card is BLANK, the card will be removed from the run stream.

'CSF @ Control Statement' - Specifies that an execution control statement will be processed using the standard CSF$ package. The following control statements may be used:

@ADD
@ASG
@BRKPT
@CAT
@CKPAR
@CKPT
@FREE
@LOG
@MODE
@QUAL
@RSPAR
@RSTRT
@START
@SYM
@USE

Example:

'CSF @USE 25, DBASE'

'CSF @ADD DUSEFIL.DLOG'

'CSF @QUAL B'

'DEFINE name=value,text' - Stores a textual description with the name in the data base directory. If the name is a new directory entry, the value is the number of data base entries allotted. Existing data is unaffected and new data is not added.

'DEFINE A, LETTER 1' - Stores the description, LETTER 1, with the name A.

'DEFINE B=10, BARRAY' - Stores the description, BARRAY, with the variable name B and allots 10 data base entries for B.

'FORMAT name=value/value, (Fortran compatible format statement)' Extracts freely stored data from the data base and places into the output elements in accordance with the given format.

'FORMAT A=6/3,(1X,3F15.3)'

The six items of A are output into the named element, 3 on a line through the (1X,3F15.3) format.

'INSERT name=value/value' - Specifies that binary coded information the SDF file name will be placed in the source output element in 14A6 format.

'INSERT A' - Entire file of data in A will be transferred to source output element.

'INSERT B=5-13' - Insert data from B from records 5 through 23.

'INSERT C=5*EOF' - Insert records from file C records 5 to the end-of-file.

Other Examples - 'INSERT A,B=5-23, C=5*EOF'

'name' - Specifies a simple replacement of named information with data base parameters or arrays.

'REAL'	Real parameter or array.
'INTEG'	Integer parameter or array.

'HOLITH'	Hollerith parameter or array.
'LOGICL'	Logical parameter or array.
'ARRAY(j)'	Real or integer element of an array, j must be a constant greater than 1. A value of j=1 will cause the transfer of all of j.

'ON name,name---' - Mode activation directive.

'OFF name,name---' - Mode suppression directive.

P or PAGDMP	Print card cracking information.
O or OUTDMP	List logical file 1 data.
N or INDUMP	List source output element.
C or CONTINUE	Activate continuation card option.
L or LIST	List source input information.
S or SPLIT	Interrupt mode.
E or EDIT	Edit mode (demand response to printer).

'PRINT name' - Specifies that data information will be printed.

'PRINT name=A,Z'	Print all information in name.
'PRINT name=n,m'	Print entries n through m alphabetically.
'PRINT name'	Directory and first data base entry of named data base.
'PRINT'	Directory and first data base entry of current 'USE' assigned data bases.

'USE' -

'USE A,B,C' - The data bases named will be circularly searched in the order given for variables used in replacements. All will be searched once before a NO FIND is declared. It should be noted that this command may cause very excessive SUP changes if not carefully used.

'UPDATE name' - Specifies that the named data base will be updated with the information which follows:

'UPDATE A' - Specifies that the data base A will be updated with the data which follows.

Processor Interface

The processor interface is a Univac 1100 series EX8 utility subroutine, IF, written in assembly language. IF is designed for use by the FORTRAN programmer in the construction of a processor. It allows the information on the processor call card to be made available to the user program. Two fields on the processor call card are available to the user. The first is the input field, and the second is the output field. The I option implies only the first field will be used. This field will be the output field.

Usage. - The programmer is assumed to have a minimum working knowledge of Univac's (R) EX8 operating system and the use of such system processors as ELT, FOR and FURPUR. There are three entry points into the subroutine: SIREAD for reading from the SI field, PGMOUT for writing to the SO field, and DONE for closing the file. The calling sequences and the associated arguments are as follows:

CALL SIREAD ($err,$eof,IMAGE,'word')

$err — Statement number to be transferred to in case of error.

$eof — Statement number to be transferred to when an end-of-file is reached.

IMAGE — An array containing the image you want written out. Normally, this is 14 words long.

'word' — This word is used to delete words from the right, back to the left to make the image as short as possible in order to conserve disk space. For card images, this would be a word of blanks: for binary information in internal machine format, zero would be best.

SIREAD — Stands for source input read.

CALL PGMOUT ($err,$eof,IMAGE,'word')

$err — Statement number of location to be transferred to in the event of an I/O error.

$eof A dummy argument.

IMAGE The array containing the image of words to be written out (normally dimensioned 14).

'word' As the image is compressed on disk with the trailing null words dropped, this word is used to fill out the image so that when it is returned to the user, it is the full 14 words long.

PGMOUT Stands for program output.

CALL DONE ($err)

This call must be executed prior to conclusion of the program. It will drain any uncompleted buffers, close and release to their original status any attached files. If this entry is not called prior to program termination, the created element of SO field will not be properly created.

Restrictions. - There are three important limitations on the use of IF:

1. There can be only 2 fields on the processor card.

2. SI READ must be called prior to any reads from the standard system input device, the card reader (unit 5 in FORTRAN); otherwise, read errors will occur.

3. Once the entry DONE is called, none of the entries into IF may again be referenced (the program will error off if this rule is violated).

DMAN Software Package

The storage and retrieval of the multitude of data pages which constitute a design data base are managed by DMAN. When a data page is stored, it is given a page name. DMAN keeps a directory of all the names of data pages on a file and the disk addresses where those pages may be found on the file. This makes it possible for a symbolic name rather than a numerical index to be used to access a data page during its residence on the file.

DMAN provides all of the basic data management functions to handle variable length data pages while allowing them to be referenced by name. A data page may be stored on any file which has been established for data base use. All or portions of a data page contents may be retrieved. Modification of the contents of a data page is permitted, including that which requires increasing or decreasing the size of a page. Finally, removal of a data page from a file may be accomplished.

DMAN Usage. - The DMAN data management system is a Fortran callable software package which has been written for access and retrieval of data from the EDIN data base. The package consists of the following subroutines which must be included in the calling program:

DMAN	Basic Read/Write Controller.
NXTAD	Extend File Routine.
UPACK7	Character Unpack Routine.
RITBF	Write Routine.
PACK7	Character Packing Routine.
REDBF	Read Routine.
NWBLK	Create a New Block for Data.

The use requires the following declarations in the user program:

```
COMMON/UNITS/IAREA(273)
DATA IAREA/0,n,271*0/
INTEGER IT(5),IBUF(256)
```

where n is the file number where the data base is stored. The usage is as follows:

```
CALL DMAN(IOP,IT,N,IDATA,IBUF,IAREA(1),IAREA(2))
```

IOP The read/write option. A further discussion of these options is given later.

IT A five word array containing the data title. A further discussion of the titles is given below.

N This variable contains the number of words in IDATA to be read or written. When reading, and the requested list cannot be satisfied, this value is reset to the number of words actually read, so this item must always be a variable when reading data.

IDATA An integer or real array containing the data to be stored in the data base. There is no restriction on the length of this array.

IBUF A 256 word buffer area for use by DMAN.

IAREA This is a unit dependent area needed by DMAN. It must be dimensioned 273. One IAREA is required for each unit using DMAN. The double appearance of this array in the calling sequence is required for interal addressing purposes. This area must be protected, such as in COMMON, and must be reserved for use by DMAN while this file is being used.

A Discussion of IT. - There are two significant portions to the five word array IT. The first three words of the title are user supplied hollerith words which represent the name of the data item which is to be accessed or stored in the data base. If this is the first access of this data in the data base, the fourth word must be set to zero. This zeroing of the fourth title word will also return access to the beginning of the data set stored under the title given in the first three words.

The fourth and fifth words of the title are reserved for use by DMAN. If the fourth word is zero, a search is made of index arrays to find the address of the desired data set. This address is then inserted into these two words. Each time some activity occurs using this title, the address stored in these two words is updated so that this address always refers to the next word after the last word accessed. This eliminates the need to search the index arrays for each access of the data.

A Discussion of IOP. - IOP controls the type of reading or writing done by DMAN. The I/O options are:

IOP = 10 - write a matrix. The complete data set to be stored under the title IT is present in IDATA.

= -10 - read a matrix.

= 20 - write a single fixed length record.

= -20 - read a single fixed length record.

= 21 - write a single variable length record. Using this type of write option, an end-of-record mark is inserted after the end of the record. Any variable length record read will not pass this mark when reading. If the read is a fixed length record read, however, this mark will be ignored.

= -21 - read a variable length record. In this case, N is the number of words requested. The read will continue until N words have been read, and end-of-record mark is found, or the data set is exhausted, whichever comes first. The value of N will be set to the number of words actually returned.

= 30 - extend a data set with a fixed length record. The data in IDATA is to be appended to the existing data set stored under the title in IT.

= 31 - extend a data set with a variable length record.

NOTE: If a read attempt is made, which will extend the read past the end of the stored data set, or the data set requested has not been stored, the following values will be returned by DMAN:

N=0 and IDATA(1)=3LEOD.

IOP = 6HPURGE - this option will cause the title given in IT to be purged from the index array.

IOP = 6HCLEAR - this action will cause the buffer IBUF to be cleared. That is output to disc if necessary. This action is necessary before releasing the buffer to other uses, or existing a subroutine or overlay under conditions which will not protect the buffer.

IOP = 6HCLOSE - this action conditions the data base so that the entire contents of the data base do in fact reside on disc. It is necessary to execute this statement on any catalogued data base to insure that its entire contents are on disc. Normal activity may proceed after the function is called, and this function may be called as many times as desired.

Technology Module Interface Package

The communication of information from a technology program to the EDIN data base generally requires modification of the applications program. This modification is usually trivial and requires little programming knowledge to accomplish. The objective of the modification is to create a special file of information which contains a format suitable for reading by the DLG processor. The information is placed on the special file by the technology program. The file is later integrated by the DLG for possible placement of the information into the EDIN data base.

A series of four routines for printing the common types of data in a format readable by DLG are available. They may be called at any point in the calculation sequence for generating EDIN output. The format simulates the control directives format used in the DLG processor.

ADDREL - For printing real variables and arrays.

ADDINT - For printing integer variables and arrays.

ADDHOL - For printing Hollerith variables and arrays.

ADDLOG - For printing Logical variables and arrays.

The output is similar to the format of NAMELIST for one variable name only with any number of associated values. Each subroutine has the same calling sequence characterized as follows:

```
CALL ADDREL (LU, NAME, NUM, VALUE)
```

LU - Logical unit or special output file.

NAME - Desired name chosen by the analyst/programmer. It may be a stored name set by a Fortran data statement or can be set in the calling sequence as nHname.

NUM - Number of values in the array. For a single variable NUM=1.

VALUE - Internal variable or array name (starting location).

The subroutines for the other variable types have the same calling sequence. The primary difference among them is the format used for writing the variables and the special output file. Each output is a DLG control directive format. The name associated with the directive is set by a data statement in the individual subroutines. The data statement may be set at the time the technology program is modified. Usually it is desirable to use a name which is reminiscent of the application program name. The selected name may be precisely the same as the acronym used to execute the application program in EDIN. The reason for such a choice is that the directive name is stored in the EDIN data base. A print of the data base prints the last directive which updated each variable in the data base.

For most technology programs, the use of the software described above is adequate. However, certain programs generate data base information in a Fortran "DO LOOP." In these instances, the package (by itself) can not satisfy the EDIN requirement of separate names for different data elements and arrays.

The most convenient way to make this program and others of this type compatible with EDIN is to provide some name-generating capability with the applications program. Function subroutines which provide this capability can be called as illustrated below:

```
NAMGEN (NAME, K, J)
```

NAME = The desired root name.

I = Concatenated number occupying the first one or two BCD character positions beyond the root name.

J = Concatenated number occupying the second one or two BCD character positions beyond the root name.

An example would be:

```
NAM=NAMGEN (4HNAME,1,2)
```

In the above illustration, the name NAME would be extended by the BCD characters 1 and 2 concatenated to it and stored in NAM.

```
NAM=6HNAME12
```

A maximum of 6 characters may be generated. This limit is imposed by the word size limit for EDIN data base names.

Usually the NAMGEN function is used in conjunction with the NAMELIST simulator described above in the following manner:

```
CALL ADDREL(LU,NAMGEN(NAME,I,J),NUM,VALUE)
```

In the illustration, the name is generated within the calling sequence of the subroutine which prints the simulated namelist for the generated name.

Subroutine Descriptions

Subroutine ADDER. - ADDER is a Fortran subroutine for controlling the placement of names and values into the data base. Its purpose is to process the ADD commands and place the specified information into the data base. If the information going into the data base is new, a new entry will be created for the data. If the entry already exists, the information will replace what is already there. The subroutine is designed to handle not only real, integer, logical, hollerith variables, but arrays as well. It will also perform simple arithmetic operations upon a given element before entering it into the data base.

Subroutine ADDONE. - ADDONE is a Fortran subroutine for adding names and/or values to the data base. This is a standard storage routine for the design data base information. The subroutine has five calling arguments, I,B,S,F, and L. I is the name of the data base entry. B is the value of the data base variable being installed. S is the element number. F defines whether this is the first element. L is the logical variable defining whether or not it is the last variable.

Subroutine ANLSIS. - A small subroutine used just as DLG terminates normally to process the "A" option. It produces a 4-line report that includes a count of IO operations and a collision record for RANDAC.

Subroutine BCDDB. - BCDDB is a Fortran subroutine for transferring one element of information from the BCD array to the data base. The BCD array is a temporary array which is loaded with information to be transferred to the data base from some other subroutine.

Subroutine BCDDEC. - BCDDEC is a Fortran subroutine which converts BCD character strings to equivalent decimals word (integer or real). The subroutine has three calling arguments BCD, NCHR and DEC. BCD is a string of characters to be converted. NCHR is the number of characters, one per word, left justified and blank filled. DEC is the resultant real or integer variable.

Subroutine BCDINT. - BCDINT is a Fortran subroutine which converts BCD characters to integer equivalents. The subroutine has three calling arguments, BCD, NCHR and INT. BCD string contains NCHR characters, one character per word, left justified and blank filled. INT is the resultant integer variable.

Subroutine BCDVAL. - BCDVAL loads the decimal equivalent of one or more BCD words into the variable VAL. It also loads the BCD array with one BCD character for each input BCD character and determines the type of resulting variable in VAL (real, integer, hollerith or logical\.

Subroutine BILDOP. - Subroutine BILDOP is a Fortran routine that determines whether previously defined information is to be added to the data base or ignored. The criteria is the existance and/or the data base value of BUILD. If the word BUILD does not exist in the data base, all incoming information will be added. The same is true if BUILD exists in the data base and has a value of 1. However, if the variable exists and has a value of 0, no new variable will be added.

Subroutine CCDUMP. - CCDUMP is a Fortran subroutine for printing of data base information. It processes the control directive 'PRINT name'. The routine sorts the data base name alphabetically in groups of 100 and calls the routine DBWRT to actually print the information. It also prints the data base parameters for the data base being printed.

Subroutine CDINIT. - This Fortran routine initializes the index values of commands and their corresponding character string names for use by RSPOND.

Subroutine CHANGE. - This Fortran subroutine is used to modify the contents of any location in the IDILOG common block. It presents the value of the indexed location both before and after the change.

Subroutine CHARS. - This highly efficient assembly language routine strips out characters from 6 to a word to 1 per word - L.J.S.F. Also returned is the last valid character position.

Subroutine CHRNUM. - CHRNUM is a Fortran subroutine which determines the integer equivalent of a single BCD digit.

Subroutine CREATF. - CREATF is a subroutine for equivalencing external (system) file names to internal logical unit numbers. Two arguments which have significance are LU and LFN. LU is the internal logical unit number to be equivalence and LFN is the external logical name to be equivalent. CREATF uses the system routine ERTRAN to dynamically perform the USE assignment.

Subroutine CSF. - This subroutine passes the control image from the 'CSF' directive to ERTRAN. In this manner any legal control card may be submitted to Exec 8 while DLG is in execution.

Subroutine DBADD. - DBADD is a control routine for processing information to be added to the data base. It is called for initially loading the data base and updating the data base with information from previously executed programs. It is called from INITIZ, NLADD and EXECUT. The single calling arguments specify the origin of the namelist like files to be read.

Subroutine DBINIT. - DBINIT processes that portion of the CREATE control directive which specifies the five data base parameters DIRLEN, LTOTAL, KEYLEN, LENDES and NWORD, if they exist on the CREATE control directive. The values are set into the corresponding location of the DILOG common block.

Subroutine DBLOAD. - DBLOAD is a Fortran subroutine for writing out the data base which is currently in core and reading in the data base which has been requested in the calling sequence.

Subroutine DBWRT. - DBWRT is a subroutine which collects all of the names of data base variables in groups of 100 and sorts them alphabetically and prints the names and values in groups.

Subroutine DECBCD. - DECBCD is a Fortran subroutine that converts a real decimal value to a specified field width of BCD characters, left justified and blank filled. The routine insures maximum significance within the specified field width and uses either E or F format to accomplish this end. A maximum of two BCD words is used to characterize the decimal number.

Subroutine DECIDE. - DECIDE is a Fortran subroutine that builds an array of BCD words, one character per word from a packed BCD array. It determines the type of the input BCD array and number of characters in that word.

Subroutine DELETE. - DELETE is a Fortran subroutine that deletes an entry from the data base directory. It does not however delete the space which has been used in the data base proper.

Subroutine DIALEK. - DIALEK is the main routine for controlling the DIALEK Executive System. It initializes all data and directory through calls to the appropriate routine. It processes the namelist output from other programs by a call to NLADD. It then begins reading control directives and processing them through appropriate calls to the actual processing subroutines. All control directives are read and processed from the program DIALEK.

Subroutine DISECT. - It is a Fortran subroutine which reads and cracks BCD card input and places the information into the IPAG array for future processing. Once an apostrophe is encountered, processing begins until a second apostrophe is encountered. Within the apostrophe delimiters, the card is broken down by DILOGS which are delimited by commas, an operation, which is delimited by the normal operators (plus, minus, multiply, divide or exponentiation) the pattern of storage of the information read is picked up by the processing routine which interprets the commands and directives specified on the card input.

Main Program DLGDVR. - This is the main program and it is essentially a dummy so that DIALEK can be a subroutine and have more than one entry point.

Subroutine DMAN. - This Fortran subroutine is the random access package to mass storage. The basic technique is through the use of DEFINE FILE statements in conjunction with the associated random read and write operations. DMAN uses the utilities NWBLK, NXTAD, PACK7, REDBF, RITBF and UNPACK7.

Subroutine DYNCOR. - This very powerful assembly language is used to contract/expand Fortran array sizes through LCORE$/MCORE$ executive requests. The addressing of these arrays must be done via statement functions but otherwise use is quite general. Total program size is limited to 262K and any one array to 64K. The use of DYNCOR allows a Fortran program to execute in the absolute minimum size needed to handle the current amount of data - thereby significantly lowering system impact.

Subroutine ENDFL. - ENDFL is a Fortran subroutine which places a Fortran end-of-file on normal sequential files. However, it places the character string *EOF into the current record of random access files.

Subroutine EOFTST. - EOFTST is a Fortran subroutine which tests the current card image to determine if the first four characters contain the character string *EOF. If so, the logical variable MYEOF is set to true.

Subroutine FLDATA. - The entry point INTFLD of the assembly language routine FLDATA is used to convert an internal binary integer into a 12 character FLDATA representation.

Subroutine FORMAT. - This code is used to process data from the data base through a Fortran format and then passes it into the output element according to the format being used.

Subroutine GET. - GET is a utility routine written in assembly language which can be called as a subroutine or function. It has three arguments, S, I and T. The function of the subroutine is to get the I symbol from the string S and place it left justified.

Subroutine GETSUB. - Extracts from the data base the values of a variable and converts it to internal integer for use as a subscript in an expression.

Subroutine IDENT. - IDENT is a Fortran subroutine for processing the DEFINE command directive. The subroutine has the function of reserving space in the data base and inserting descriptive information with regard to the specified variable in the data base directory. If the name was not previously defined, by a DEFINE command or an ADD command, the name and description are entered into the data base and the number of entries specified are reserved in the data base. If the name previously existed in the directory, the action of this subroutine is simply to insert the description in the directory.

Subroutine IF. - This interface routine, written in assembly language, provides the capability for DLG to be invoked as a processor rather than an ordinary program. This technique allows considerable use of Exec 8 in file handling and access. Images may be both passed and received to/from mass storage via the IF interface.

Subroutine IGNORE. - The subroutine simply blanks out all of the characters associated with the comment directive on the input image.

Subroutine INITDM. - This one time called Fortran routine initializes some of the values of common block /MS/.

Subroutine INITIZ. - INITIZ is a subroutine for initializing the design data base for the control card data base. It processes the 'CREATE directive' by determining which data base is to be initialized. It then calls the subroutine DBINIT to process the remainder of the 'CREATE directive' to determine deviations in the data base parameters such as the length and width of the directory, etc. INITIZ then initializes the directory and data base and calls the data base load routine.

Subroutine INITL. - Subroutine INITL initializes the DILOG common area and some positions of the files that are used in DIALEK.

Subroutine INMOD. - INMOD processes the 'name' command. It performs the simple replacement function for data base variables and arrays and, if required, performs the arithmetic operations which are provided for in the language.

Subroutine INSRT. - INSRT processes 'INSERT' command by attaching the named system file and copying the specified BCD records from the file to the modified input stream for the next program to be executed.

Subroutine INTBCD. - INTBCD converts an integer into two words of BCD characters for storage into the data base.

Subroutine IOPT. - This three (3) line assembly subroutine returns the option word, in master-bit notation, both in the calling argument and as its value, if referenced as a function.

Subroutine IVCALC. - This subroutine loads the description arrays, which are stored in the data base directory. The description array consists of the data base location, the origin of the most recent update and the user's specified number of words of arbitrary descriptive information.

Subroutine IVDESC. - This subroutine extracts the descriptive information from the data base directory and places it in the IDESC array.

Subroutine LOCP. - LOCP is a Fortran function which determines the equivalent singly descripted array location corresponding to a three dimensional array location.

Subroutine MOVER. - MOVER is a highly efficient assembly routine for transferring information from one place in core to another. The increment used in both arrays need not be equal, therefore, one word, with zero increment, can be used to fill another array. Transfer method is via the BT instruction therefore DO-LOOPS are better if 5 or less words are to be moved.

Subroutine NLADD. - NLADD processes the NMLIST file which was generated by the last program in the execution sequence. The Fortran namelist like format is assumed in the processing. Therefore, the delimiter, which is normally an apostrophe, is changed to a dollar sign and the record width is changed from card width (80 columns) to the normal namelist record width (132). In addition, the start column for processing the data is changed from 1 to 2. This is because all namelist data starts in column 2 and column 1 sometimes contains carriage control information.

Subroutine NUMNIT. - NUMNIT initializes the numbered directory which correlates the BCD representation of the numbers 0 through 9, +, - and . to their integer representation 0 through 13.

Subroutine ONROFF. - The Fortran code will turn on or turn off, through entry points ON and OFF, the effect of any of the allowed option bits (letters) on the processor call card except the "B" option.

Subroutine OPINIT. - OPINIT initializes the operator directory -, *, /, **, $ and ' with the names equal, plus, minus, mltply, divide, expon, dollar and noteql. The operators can be changed by changing their character representations in the data base.

Subroutine OPTION. - This Fortran routine is used one time only to make .TRUE. those variables in common blank IDILOG that appeared upon the processor invoking card. The following table gives options and corresponding common locations:

OPTION LETTER	LOCAL NAME	IDILOG LOCATION	DESCRIPTION
A	ANALY	340	Analysis print at end of execution.
B	STORE	292	All new data stored in data base.
C	CONTIN	36	End-of-card signifies end-of-directive.
D	PGDUMP	304	Dump page array.
E	EDIT	250	Requests and responses printed.
I	*		Source input will follow.
L	LISTI	300	Source input will be listed.
M	INIT	311	Make data base file.
O	LISTO	301	Source output will be listed.
S	SPLITR	305	First interrupt mode.
T	TRACE	307	Trace information printed.

*Standard Processor Option.

Subroutine PAGDMP. - This routine prints the card cracking information from the IPAG-array which was loaded by the DISECT routine. Each entry in the IPAG-array consists of a start column for the operator, the operator character, a name and a subscript. If the entry is a number, both the name and the subscript locations are used to represent that number. Function page determines the equivalent single subscripted location in the page array corresponding to a three dimensional array call and transfers the information from the IPAG-array into the function name page.

Function PAGE. - Integer function PAGE uses LOCP to return a value from the IPAG array.

Subroutine PRINTF. - PRINTF copies a specified file to output.

Subroutine PRTT. - PRTT is used to process the DBLIST command in a manner similar to the FURPUR command @PRT,T. It returns the names of all the data bases residing on the file that are attached to logical unit number 25.

Subroutine PUT. - PUT is an assembly language routine which can be called as a subroutine or a function. The routine has three calling arguments, S, I and T. The function of the subroutine is to put the left most symbol of T into the Ith position of string S.

Subroutine RANDAC. - RANDAC is a Fortran utility routine for locating information in the data base directory by name. There are four main entries to initialize the directory, to find information in the directory, to install information in the directory and to delete information from the directory. The directory information contains pointers to the actual data.

Subroutine READBR. - This subroutine is used to read binary information. The subroutine has three calling arguments, LU, INREC, and NW. The routine reads one record of width NW from file LU into the array INREC.

Subroutine READCR. - READCR reads coded records. It has three calling arguments, LU, INREC and NW. READCR reads one record of NW words from file LU into the array INREC.

Subroutine RPLACE. - RPLACE performs the simple and array replacement function for delimited data base names by retrieving the information from the data base and placing the current data base values in the image array. In the case of simple replacement, the routine uses the column position between delimiters to format the data base information. In the case of array replacement, column positions are not preserved. The replacement begins at the first delimiter and the array is placed in the image array three values per card separated by commas. The above format is suitable for namelist and other read routines.

Subroutine RSPOND. - RSPOND performs the "switching function" of logic control by identifying the command, getting its numeric equivalent and using that value in a computed GOTO. It is basically a routine to decrease size of a demand program without adding any overhead noticeable by the terminal operator.

Subroutine SCALE. - SCALE is a Fortran utility routine for processing simple arithmetic operations such as add, subtract, multiply and divide, which are specified by the data base language.

Subroutine SHELL. - SHELL is a Fortran subroutine for sorting an independent array of names. SHELL has three calling arguments, IARRAY, KEY and N. SHELL sorts an independent array of size N into ascending order (algebraically leased first) and provides a key array which will allow the companion subroutine SHELLX to return dependent arrays in the original correspondence with the independent array. IARRAY is the name of the independent array (dimensioned at least N in the calling program) key is the name of the key array (dimensioned at least N in the calling program) and N as the number of elements in both IARRAY and KEY.

Subroutine STRMOV. - This subroutine has five calling arguments, OBCR, ICOLD, NUMCHR, NEWBCD and ICNEW. STRMOV is a Fortran subroutine which uses the routines GET and PUT to move characters from one location to another. STRMOV moves NUMCHR characters from OLDBCD starting a column ICOLD to the array NEWBCD starting at column ICNEW.

Subroutine UPDATE. - UPDATE is a Fortran subroutine used for updating an existing data base at the start of a simulation.

Subroutine USE. - USE is used when a no-find on a data base name occurs. Several data bases are to be searched in a sequential circular manner. The data base names to be used come from the USE card which is processed by subroutine USE. It should be cautioned that loading/unloading data bases is a very high overhead item and should be kept to 9 minimum.

Subroutine VALIMG. - VALIMG is a Fortran subroutine for converting a value of arbitrary type stored in core to BCD format and placing it at a specified positional relationship in the image array.

Subroutine WRITBR. - This subroutine has three calling arguments, LU, INREC and NW. WRITBR writes on NW word record from the array INREC to the logical unit LU. The Fortran write functions are used in the Define File Format.

Subroutine WRITCR. - WRITCR has three calling arguments, LU, INREC and NW. The routine writes one NW word record from INREC to LU in binary coded format.

COMMON VARIABLES

DILOG Locations	Value	Local Name	Descriptions
1	'E'	ALFE	Integer word containing the character (E), left justified and blank filled.
2	'F'	ALFF	Integer word containing the character (F), left justified and blank filled.
3	-	BCD(20)	Integer array of BCD characters used for scratch purposes.
23	-	BCDLEN	The number of characters in the BCD array.
24		BCDNUM(10)	Integer array containing powers of ten in sequential order from 0 to 9.
34	' '	BLANK	Integer word containing blank characters.
35	','	COMMA	Integer word containing the character (,), left justified and blank filled.
36	.TRUE.	CONTIN	Logical variable set by option flag 'C'. If true, an end-of-record will signify the end of a command.
37	27	DBASE	Logical unit of the file containing the design data base.
38	0	ICOPY	A counter for the cumulation of input operations on the logical unit SI element.

DILOG Locations	Value	Local Name	Descriptions
39	(')	DELIM	Integer word containing the DIALOG delimiter used in the simulation input data, usually has a value of (').
40	0	ICONT	A counter for cumulating the number of output operations on the logical unit SO element.
41	'='	EQUAL	Integer word containing the character (=), left justified and blank filled.
42	8	MXCHAR	The maximum number of characters which can be used for interpreting a number in BCD format calculated as: MXCHAR = NWORD*NCAR-(LENEXP-1)
43	2	FIND	An integer word defining the FIND entry in RANDAC.
44	None	ICHAR(140)	Integer array containing the input image one character per word, left justified and blank filled.
184	None	IMAGE(36)	Integer array containing the input IMAGE.
219	1	INITAL	An integer word which defines the initialization entry in RANDAC.
220	80	INRECL	Maximum number of characters in the input record (IMAGE).

DILOG Locations	Value	Local NAME	Descriptions
221	3	INSTAL	An integer word which defines the installation entry in RANDAC.
223	None	IV(20)	An integer array containing the data base location and descriptive information for the current directory entry.
243	47	LD	Length of the directory in terms of number of entries.
244	1016	LDB	Length of the data base in terms of number of entries.
245	1009	LFDB	Last free data base location.
246	1	LK	Length of the data base directory key in terms of number of words.
247	'('	LPAREN	Integer word containing the character ((), left justified and blank filled.
248	8	LT	Length of the array containing the data base location and descriptive information for the current data base entry in computer words.
249	47	NCD	Maximum number of control directives.
250	.FALSE.	EDIT	Logical variable set by option character 'E'. If true, DLG requests and responses will be printed.

DILOG Locations	Value	Local Name	Descriptions
251	11	MAXINT	The number of characters representing the maximum size integer for the computer on which this program is installed.
252	0	ICNML	Integer word containing the record count from the NMLIST file.
253	0	ICNDB	Integer word containing the number of data base read and write requests.
254	None	MYEOF	Logical variable, if true an end-of-file has been encountered. MYEOF is set when a system end-of-file or users end-of-file (*EOF) is encountered.
255		NAME	Integer word containing the current data base name.
256	6	NCAR	The number of characters per computer word.
257	12	NCDBV	The number of characters per data base variable.
258	'-'	NEG	Integer word containing the character (-), left justified or blank filled.
260	14	NMLIST	Logical unit number for potential data base information. Also used for reading inserted files. See insert command.

DILOG Locations	Value	Local Name	Descriptions
261	15	NNUM	Maximum number of entries in the number directory.
262	∿	NOPER	Number of operations per dialog in the page array.
263	2	NWORD	Number of words per data base entry.
264	14	NWREC	Number of words per input record.
265	'.'	POINT	Integer word containing the character (.), left justified and blank filled.
266	'+'	POS	Integer word containing the character (+), left justified and blank filled.
267	')'	RPAREN	Integer word containing the character ()), left justified and blank filled.
268	None	VALUE	A real word containing the value of the current data base variable or result of an arithmetic operation.
269			Not used.
270			Not used.
271	4	DELET	Integer variable used for delete entry in RANDAC.
272	None	IDESC(20)	An integer array used for temporary storage of the current data base variable descriptive information.

DILOG Locations	Value	Local Name	Descriptions
292	.FALSE.	STORE	A logical variable set internally to .TRUE. if the option character 'B' is invoked. If store is true, all incoming data from the file NMLIST will be stored in the data base. Otherwise, only previously defined data will be stored.
293	∿	LENDES	Length of the descriptive information in IV and IDESC.
294		COMAND	Integer word containing the name of the current control directive (i.e., ADD,PRINT,...etc.)
295	0	ICNSRT	Integer word containing the record count of inserted records.
296	0	IRANDC	Integer word containing the number of collisions from RANDAC.
297	0	IRANDF	Integer word containing the number of RANDAC FIND requests.
298	0	IRANDE	Integer word containing the number of RANDAC entries.
299		NFCD	Next free control directive directory location.
300	FALSE	LISTI	Logical variable set to true by option character 'L'. If true, source input data will be listed.
301	FALSE	LISTO	Logical variable set to true by the option character 'O'. If true, the source output file will be listed.

DILOG Locations	Value	Local Name	Descriptions
302	None	COMSAV	Integer word used for saving the value in COMMA when COMMA is being used for storing alternate delimiter.
303	∿	CONDIR	Integer word containing the current control directive (i.e., CREATE,PRINT...).
304	FALSE	PGDUMP	Logical variable set to true by the option character 'D'. If true, card cracking information (IPAG array) will be printed.
305	FALSE	SPLITR	Logical variable set to true by the option character 'S'. If true, the list interrupt mode will be invoked.
306	'DBASE'	DDBASE	Integer word containing the name of the requested data base (DBASE), left justified and blank filled.
307	FALSE	TRACER	Logical variable set to true by the option character 'T'. If true, trace printout option will be invoked.
308	∿	DIRIN	Name of the directory (data base) which is currently in core.
309	0	FERROR	A counter for cumulating the number of fatal errors which have occurred since the start of execution.

DILOG Locations	Value	Local Name	Descriptions
310		IFILE	Integer word containing the name of the data base which is to be loaded, left justified and blank filled.
311	FALSE	INIT	Logical variable set by option character 'G' to specify a new data base file is being created.
312-331			Not used.
332	0	ICDLG	Integer variable for counting DLG input data.
333			Unknown usage.
334			Unknown usage.
335			Not used.
336			Not used.
337			Not used.
338	6	LUO	Logical unit number for the output file.
339			Not used.
340	FALSE	ANALY	Logical variable set by option character 'A'. If true, the current run analysis will be printed. This variable can also be set by the directive: 'ON A' or 'OFF A'
341	1	STCOLM	Start column for processing input records. Generally has a value

DILOG Locations	Value	Local Name	Descriptions
			of one but set to 2 for processing namelist data.
342			Not used.
344	20	MAXFER	The maximum number of fatal errors which can occur before execution is terminated.
345			Not used.
346		CRDFMT	An integer array containing the BCD definition of a card format.
347-350			Not used.
351-352			Not used.
353		NWPAGE	The number of words which define the width of a namelist record.
354			Not used.
355	*EOF	ENDATA	Integer word containing the character string '*EOF', left justified and blank filled, used to identify a user end-of-file.
356			Not used.
357		IOCONT	A counter used to accumulate the total number of input/output requests such as READ,WRITE, etc.
358-400			Not used.

OVERLAY STRUCTURE

```
TYPE CLRAFCM
LIB WORK
LIB LEC*UR.,MSC*LOCALIB.
.
SEG MAIN
   IN DLGDVR
   IN DIALEK,IF,DYNCOR
SEG A*,(MAIN)
  IN INMOD,RSPOND
SEG B*,A
  IN INITL,INITDM,OPTION,OPINIT,NUMNIT,CDINIT
SEG D*,(A)
  IN ADDER
SEG E*,D
  IN ATTACH
SEG F*,D
  IN CHANGE
SEG G*,D
  IN IGNORE
SEG H*,D
  IN COPY
SEG I*,D
  IN INITIZ
SEG J*,D
  IN CSF
SEG K*,D
  IN IDENT
SEG L*,D
  IN DELETE
SEG M*,D
  IN DETACH
SEG N*,D
  IN FORMAT
SEG O*,D
  IN INLINE
SEG P*,D
  IN INSRT
SEG Q*,D
  IN ONROFF
SEG R*,D
  IN CCDUMP
SEG S*,D
  IN SEARCH
SEG T*,D
  IN TIME
SEG U*,D
  IN USE
SEG V*,D
  IN UPDATE
SEG W*,D
  IN PRTT
  END
```

```
AFCM STATUS OF OUTPUT ELEMENT=CLRAFCM

ADDRESS LIMITS    001000 032377      13056 IBANK WORDS DECIMAL
                  040000 055742       7139 DBANK WORDS DECIMAL
SEGMENT LOAD TABLE       040000 040133
INDIRECT LOAD TABLE      040134 040400
STARTING ADDRESS  023452


             SEGMENT MAIN                001000 024316      040401 054124

BCNTL(COMMONBLOCK)                                          040401 041005

SYS$*RLIB$.NSWTC$/FOR69
                        $(1)   001000 001024
      EXTERNAL REFERENCES:  NTAB$, FNCTB$, IOCOD$, WRBLK$

SYS$*RLIB$.NRWND$/FOR68
                        $(1)   001025 001106      $(2)   041006 041017
      EXTERNAL REFERENCES:  NTAB$, NS11$, NHPFA$, IOCOD$, NFCHK$, WAIT$,
      NIDER$, MB$, DRAIN$, NRBF$, REW$, IO$, STREG$, PRINT$, NWALK$

SYS$*RLIB$.NRBLK$/FOR68
                        $(1)   001107 001131
      EXTERNAL REFERENCES:  NTAB$, UNIT$, WAIT$, NIDER$, R$, UPDDA$,
      IO$

SYS$*RLIB$.NWEF$/JSC69
                        $(1)   001132 001337      $(2)   041020 041037
      EXTERNAL REFERENCES:  NTAB$, NS11$, NHPFA$, IOCOD$, NFCHK$,
      NBFMG$, PACKT$, RDBLK$, UNIT$, UPDDA$, WAIT$, BS1BL$, DRAIN$,
      NBFGT$, NIDER$, NBFRL$, NSWTC$, NRBFA$, PUNCH$, PNCHA$, STREG$,
      PRINT$, NWALK$, CLOSE$, WEF$, IO$

SYS$*RLIB$.NBDCV$/FOR64
                        $(1)   001340 001465      $(2)   041040 041102
      EXTERNAL REFERENCES:  NC1UL0, NFDP$, NC1UL1

SYS$*RLIB$.NFTV$/FOR
                        $(1)   001466 001510

SYS$*RLIB$.NCNVT$/FOR68
                        $(1)   001511 001732      $(2)   041103 041177
      EXTERNAL REFERENCES:  STREG$, NSTSV$, NSTAT$, NCOM3$, NERCR$,
      NFTGL$, NCDOF$, NERCT$

SYS$*RLIB$.NCLOS$/FOR68
                        $(1)   001733 002123      $(2)   041200 041230
      EXTERNAL REFERENCES:  NTAB$, NS11$, UNIT$, CSF$, IOW$, MB$, NWEF$,
      WAIT$, NREW$, NRBF$, STREG$, NCEF$, PRINT$, NWALK$, NTBSZ$,
      NIDER$, W$, IO$
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
SYS$*RLIB$.NWBLK$/FOR68
                        $(1)  002124 002235
      EXTERNAL REFERENCES:  NTAB$, UNIT$, WAIT$, NIDER$, W$, UPDDA$,
      IO$

SYS$*RLIB$.NBSBL$/FOR68
                        $(1)  002236 002276
      EXTERNAL REFERENCES:  NTAB$, MB$, WAIT$, NIDER$, IOW$, UPDDA$

SYS$*RLIB$.NUPDA$/FOR68
                        $(1)  002277 002332
      EXTERNAL REFERENCES:  NTAB$, WAIT$, MB$

SYS$*RLIB$.NBF00$/FOR
                                                $(2)  041231 043432

SYS$*RLIB$.NFTCH$/FOR69
                        $(1)  002333 002615     $(2)  043433 043446
      EXTERNAL REFERENCES:  NTAB$, RDBLK$, WAIT$, NIDER$, IOCOD$,
      NBFRL$, NBFGT$, NBFMG$, R$, NFBY1$, NIDER$A, NBFRS$, NFRONF$, MB$,
      UNIT$, MF$, IOW$, FNCTB$, UPDDA$, STREG$, NSTAT$, NERCT$

SYS$*RLIB$.NININ$/FOR68
                        $(1)  002616 003006     $(2)  043447 043452
      EXTERNAL REFERENCES:  NTAB$, PACKT$, NFRH$, NREC$, NERU$, NRD$,
      NKLN$, NKL2$, NFRA$, NLLM$, NRTR$, NFTCB$, TEMP$, UNIT$, NFTCH$,
      NBCW$, NIIC$, NCSP$, NBIPA$, NEFCL$, READA$

SYS$*RLIB$.INFOR$/58
                        $(1)  003007 003367     $(0)  043453 043505
      EXTERNAL REFERENCES:  READ$, CSF$

SYS$*RLIB$.NOTIN$/FOR68
                        $(1)  003370 003664     $(2)  043506 043511
      EXTERNAL REFERENCES:  NTAB$, NFRJ$, NREC$, NFPC$, NSTSV$, PACKT$,
      NERU$, NPU$, NPR$, NKLN$, NKL2$, NFRA$, NOLM$, NTEND$, NBFMG$,
      NC1UL0, NBFGT$, NBFRS$, WAIT$, NIDER$, UPDDA$, BS1BL$, FNCTB$,
      PNCHA$, NEXIT$, NCCC$, PPP$, PRNTA$, NCSP$, TEMP$, DRAIN$, UNIT$,
      NBFRL$, NCJNI02$, CFE, NI02$

SYS$*RLIB$.NOUT$/FOR69
                        $(1)  003665 005041     $(2)  043512 043550
      EXTERNAL REFERENCES:  NCSP$, NFRJ$, NFPC$, IOCOD$, NPCT$, NR92$,
      NR93$, NRM92$, NFM96$, NFAR$, NFRZ$, NP91$, NBI$, NFNS1$, FMTOP,
      NFNS2$, NFNS3$, NDIG$, N3L$, NDOUT$, NIND$, NFGC$, NGC9$, NT10$,
      NFRA$, XFOR$, NR91$, NFMT$, PRNTA$, PRINT$, PUNCH$, NVEC$
```

```
SYS$*RLIB$.NINPT$/FOR69
                                    $(1)  005042 006050      $(2)  043551 043601
        EXTERNAL REFERENCES:  NNG90$, NFGT$, IOCOD$, NR92$, NR93$, NLLC$,
        NFM96$, NFAR$, NFRZ$, NPW2$, NP91$, STREG$, NSTSV$, NSTAT$,
        NCOM3$, NFTGL$, NERCR$, NFCI$, NCNV9$, NSF$, NFSG$, NFDB$, NDBFI$,
        NDBCV$, NFRC$, NFRH$, NEFCL$, NFCM$, NDBIN$, NGC9$, NPCT$, NT10$,
        NFGC$, NRTR$, NFRG$, NDBLT$, READ$, NCSP$, NVEC$

SYS$*RLIB$.NFMT$/FOR69
                                    $(1)  006051 006725      $(2)  043602 043656
        EXTERNAL REFERENCES:  NTAB$, NFRZ$, NFRZ$S, NFMTR$, NFTGL$,
        NID1V$, NFNID1$, NID3V$, NFNID1D$, NID3VA$, NDBI$, NAB7$, NAB0$,
        NAB4$, NAB2$, NAB5$, NAB3$, NAB1$, NAB6$, STREG$, NSTAT$, NERCR$,
        NFCS$, NDBCV$, NHVC$, NDBIN$, NXVC$, NAVC$, NFRG$, NRTR$, PRINT$,
        NFCA$, NVEC$, NI02$, IOCOD$, NCA$, NCHAR$, NSTSV$

SYS$*RLIB$.NFIND$/FOR68
                                    $(1)  006726 007075      $(2)  043657 043727
        EXTERNAL REFERENCES:  NTAB$, NS11$, IOCOD$, NBFRS$, NEIRN$,
        NBFMG$, R$, ID$, WAIT$, W$, IOW$, NIDER$, NERU$, NTBSZ$, UNIT$,
        PACKT$, STREG$, NBTOD$, NSTAT$, NERCT$

SYS$*RLIB$.NIDER$/FOR69
                                    $(1)  007076 007265      $(2)  043730 044066
        EXTERNAL REFERENCES:  NTAB$, STREG$, UNIT$, NLRT$, NLTB$, NSTAT$,
        NCJNI02$, NTEND$, NS11$, NRSF$, NSAD$, PRINT$, PACKT$, NWALK$

SYS$*RLIB$.NFCHK$/FOR69
                                    $(1)  007266 010253      $(2)  044067 044242
                                                             $(4)  044243 044314
        EXTERNAL REFERENCES:  NTAB$, NERU$, NTBSZ$, UNIT$, NBTOD$, FITEM$,
        PL$, BL$, PACKT$, IOCOD$, STREG$, NSTAT$, PRINT$, NWALK$, NS11$,
        CSF$, WAIT$, NIDER$, W$, IOW$, UPDDA$, BS1BL$, MB$, TEMP$, DRAIN$,
        WRBLK$, NC1UL0, NC1UL1, B2L$, B20$, B10$, B1L$, CLOSE$, EXIT

SYS$*RLIB$.NTAB$/JSC
                                                             $(2)  044315 044354

SYS$*RLIB$.NIBUF$/FOR68
                                    $(1)  010254 010313      $(2)  044355 044355
        EXTERNAL REFERENCES:  NTAB$, NHPFR$, NRSX$, IOCOD$, NFCHK$, NI02$,
        NID2V$, NR91$, FHS1$, FHS2$, NINI1$, NFMT$, NKLN$, NFRA$, NRTR$,
        NFRH$, NSTSV$, NNG90$

SYS$*RLIB$.PREPRM/63
                                    $(1)  010314 011016      $(0)  044356 044567
        EXTERNAL REFERENCES:  PARTBL, SCR$, ELT$, RINF$, SELT$, PFWL$,
        PFS$, IOW$, DUSE$, FACIL$, CSF$, PRINT$
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
SYS$*RLIB$.POSTPR$/54
                              $(1)  011017 011065      $(0)  044570 044602
      EXTERNAL REFERENCES:  PARTBL, CSF$, PRINT$

SYS$*RLIB$.SDFO
                              $(1)  011066 011200
      EXTERNAL REFERENCES:  WAIT$, IO$, IOW$

SYS$*RLIB$.SDFI/SYS69
                              $(1)  011201 011451
      EXTERNAL REFERENCES:  WAIT$, IO$, IOW$

SYS$*RLIB$.ERU$/SYS69

SYS$*RLIB$.NOBUF$/FOR68
                              $(1)  011452 011512
      EXTERNAL REFERENCES:  NTAB$, NHPFA$, NRSX$, IOCOD$, NFCHK$, NERU$,
      PACKT$, NIDER$A, NTST0$, NID2V$, NR91$, NBLNK$, FHS10$, FHS20$,
      NOTI1$, MB$, NFMT$, WAIT$

SYS$*RLIB$.NERR$/FOR69
                              $(1)  011513 012113      $(2)  044603 044773
      EXTERNAL REFERENCES:  PRINT$, NEE$, EABT$, NS11$

SYS$*RLIB$.NOSYM$/FOR69
                              $(1)  012114 012356      $(2)  044774 044775
      EXTERNAL REFERENCES:  NTAB$, NCH$, NHPFB$, ARRE, OUTCNT, ENDEC$,
      NCHAR$, NHPFA$, IOCOD$, NRSX$, PACKT$, NTST0$, NID2V$, NR91$,
      NBLNK$, FHS10$, FHS20$, NFRJ$, NREC$, NFPC$, NSTSV$, NID1$, NERU$,
      NPU$, NKLN$, NETF$, NPR$, NFRAS, NFMT$, FPC00, ARRM

SYS$*RLIB$.NWDA$/FOR69
                              $(1)  012357 013124      $(2)  044776 045001
      EXTERNAL REFERENCES:  NTAB$, IOCOD$, NHPFA$, NRSX$, NCDAF$,
      NDASCD$, NFIND$, WAIT$, NIDER$, NID1V$, NID2V$, NID3V$, NID1B$,
      NDT$, UNIT$, NRELD$, NDPFDL$, R$, IOW$, NDANW$, NNWDL$, NC1UL0,
      NID3$, NCSP$, NID2$, NEXIT$, W$, NR91$, NTST0$, NFRA$, NBLNK$,
      FHS10$, FHS20$, NFRJ$, NREC$, NFPC$, NSTSV$, NKLN$, NFMT$

SYS$*RLIB$.NIER$/FOR69
                              $(1)  013125 013306      $(2)  045002 045122
      EXTERNAL REFERENCES:  NR93$, NS11$

SYS$*RLIB$.NRDA$/FOR69
                              $(1)  013307 013736      $(2)  045123 045135
      EXTERNAL REFERENCES:  NTAB$, NHPFA$, 1OCOD$, NRSX$, NCDAF$,
      NDASCD$, NFIND$, WAIT$, NIDER$, NID1V$, NID2V$, NID3V$, NID1B$,
      NDT$, UNIT$, NRELD$, NDPFDL$, NDANW$, NNWDL$, NC1UL0, NID3$,
      NID2$, NFBY1$, NFRONF$, STREG$, NRCW$, NSTAT$, NERCT$, W$, IOW$,
      R$, NR91$, FHS1$, FHS2$, NFRH$, NREC$, NKLN$, NFRA$, NLLM$, NRTR$,
      NFMT$, NCSP$, NIIC$
```

```
SYS$*RLIB$.NDEF$/FOR69
                                  $(1)  013737 014474       $(2)  045136 045235
      EXTERNAL REFERENCES:  NTAB$, NS11$, IOCOD$, NFCHK$, NERU$, NC1UL0,
      TEMP$, NC1UL1, NFPKT$, NBTOD$, CSF$, NFAF$, UNIT$, NBFMG$, R$,
      IOW$, N1OER$, W$, STREG$, PRINT$, NWALK$

SYS$*RLIB$.IOL$/64
                                  $(1)  014475 014543
      EXTERNAL REFERENCES:  SLT$, LOAD$

MSC*LOCALIB.NERTRAN$
                                  $(1)  014544 014704       $(2)  045236 045344
      EXTERNAL REFERENCES:  ABORT$, ERR$, EXIT$, CSF$, SETC$, COND$,
      DATE$, NERR$, FIELD$, PRINT$

EX42-00002*WORK.EOFTST
                                  $(1)  014705 014732       $(0)  045345 045350
                                  $(3)  DILOG               $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NERR3$

EX42-00002*WORK.READCR
                                  $(1)  014733 015037       $(0)  045351 045365
                                  $(3)  DILOG               $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  EOFTST, NRDA$, NIO1$, NIO2$, NRDU$, NERR3$

EX42-00002*WORK.NWBLK
                                  $(1)  015040 015202       $(0)  045366 045374
                                  $(3)  MS                  $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  PACK7, RITBF, REDBF, UPACK7, NXTAD, NERR3$

EX42-00002*WORK.REDBF
                                  $(1)  015203 015322       $(0)  045375 045422
                                  $(3)  MS                  $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  UPACK7, PACK7, NRDA$, NIO1$, NIO2$, NERR3$

EX42-00002*WORK.PACK7
                                  $(1)  015323 015445       $(0)  045423 045447
                                                            $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NERR3$

EX42-00002*WORK.RITBF
                                  $(1)  015446 015531       $(0)  045450 045472
                                  $(3)  MS                  $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  UPACK7, PACK7, NWDA$, NIO1$, NIO2$, NERR3$

EX42-00002*WORK.UPACK7
                                  $(1)  015532 015646       $(0)  045473 045517
                                                            $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NERR3$
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
EX42-00002◆WORK.NINTR$/DLG
                        $(1)  015647 016171      $(2)  045520 045570
      EXTERNAL REFERENCES:  CEND$, FIELD$, PRINT$, NEE$, OPT$, INTRUP,
      IALL$

EX42-00002◆WORK.NSTOP$/JSC
                        $(1)  016172 016242      $(2)  045571 045631
      EXTERNAL REFERENCES:  COM$, EXIT$, NRSF$, REST$, COND$, EABT$,
      IALL$, ERR$, PRINT$

EX42-00002◆WORK.ANLSIS/DLG
                        $(1)  016243 016304      $(0)  045632 045672
                        $(3)  DILOG              $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NWDU$, NIO2$, NERR3$

EX42-00002◆WORK.CREATF/DLG
                        $(1)  016305 016352      $(0)  045673 045720
                                                 $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NNCOD$, NERTRN, NIO2$, NWDU$, NERR3$

EX42-00002◆WORK.NLADD/DLG
                        $(1)  016353 016532      $(0)  045721 045752
                        $(3)  DIRECT             $(2)  BLANK$COMMON
                                                 $(4)  DILOG
      EXTERNAL REFERENCES:  PAGE, CREATF, DISECT, PAGDMP, RANDAC, ADDER,
      RSPOND, NWDU$, NIO2$, NERR2$, NERR3$

EX42-00002◆WORK.PAGDMP/DLG
                        $(1)  016533 016671      $(0)  045753 046031
                        $(3)  DILOG              $(2)  BLANK$COMMON
                                                 $(4)  CARDBD
      EXTERNAL REFERENCES:  LOOP, NWDU$, NIO3$, NIO1$, NIO2$, NERR2$

EX42-00002◆WORK.PUT
                        $(1)  016672 016677      $(0)  046032 046043

EX42-00002◆WORK.CHARS/DLG
                        $(1)  016700 016774      $(0)  046044 046046

EX42-00002◆WORK.DISECT/DLG
                        $(1)  016775 017735      $(0)  046047 046120
                        $(3)  CARDBD             $(2)  BLANK$COMMON
                        $(5)  DILOG              $(4)  DIRECT
      EXTERNAL REFERENCES:  LOOP, MOVER, READCR, SIFEAD, UNPACK, RANDAC,
      PUT, NWDU$, NIO2$, NERR3$

EX42-00002◆WORK.LOOP
                        $(1)  017736 020010      $(0)  046121 046136
                        $(3)  CARDBD             $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NERR3$
```

```
EX42-00002*WORK.GET
                          $(1)   020011 020017      $(0)   046137 046150

EX42-00002*WORK.NXTAD
                          $(1)   020020 020066      $(0)   046151 046156
                          $(3)   MS                 $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  NDEF$, NERR3$

EX42-00002*WORK.DMAN
                          $(1)   020067 022260      $(0)   046157 046326
                          $(3)   MS                 $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  NXTAD, UPACK7, RITBF, PACK7, REDBF, NWBLK,
     NDEF$, NRDA$, NIO1$, NIO2$, NWDA$, NWDU$, NERR3$

EX42-00002*WORK.MOVER
                          $(1)   022261 022267

EX42-00002*WORK.RANDAC/DLG
                          $(1)   022270 022760      $(0)   046327 046442
                          $(3)   DILOG              $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  NERR2$, NWDU$, NIO2$, NIO1$, NERR3$

EX42-00002*WORK.DBLOAD/DLG
                          $(1)   022761 023416      $(0)   046443 046557
                          $(3)   UNITS              $(2)   BLANK$COMMON
                          $(5)   DILOG              $(4)   BCNTL
     EXTERNAL REFERENCES:  MOVER, DMAN, DYNCOR, NWDU$, NIO2$, NERR3$

EX42-00002*WORK.PAGE
                          $(1)   023417 023451      $(0)   046560 046566
                          $(3)   CARDBD             $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  LOCP, NERR3$
DIRECT(COMMONBLOCK)                                        046567 047240
CARDBD(COMMONBLOCK)                                        047241 050515
SEARH(COMMONBLOCK)                                         050516 050537
MS(COMMONBLOCK)                                            050540 050544
UNITS(COMMONBLOCK)                                         050545 051165
OPTDIR(COMMONBLOCK)                                        051166 051253
DILOG(COMMONBLOCK)                                         051254 052073
BLANK$COMMON(COMMONBLOCK)                                  052074 052100

EX42-00002*WORK.DLGDVR/DLG
                          $(1)   023452 023457      $(0)   052101 052101
                                                    $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  DIALEK, NINTR$, NSTOP$
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
EX42-00002*WORK.DIALEK/DLG
                              $(1)   023460 023644       $(0)   052102 052126
                              $(3)   UNITS               $(2)   BLANK$COMMON
                              $(5)   DIRECT              $(4)   MS
                              $(7)   CARDBD              $(6)   DILOG
      EXTERNAL REFERENCES:  PAGE, INITL, INITDM, OPTION, OPINIT, NUMNIT,
      CDINIT, DISECT, PAGDMP, PANDAC, INMOD, NLADD, RSPOND, ANLSIS,
      UNLOAD, DBEND, DONE, EXIT, NPRT$, NID2$, NWDU$, NERR2$, NERR6$,
      NERR3$

EX42-00002*WORK.IF
                              $(1)   023645 024130       $(0)   052127 054124
      EXTERNAL REFERENCES:  READ$, SDFI, SDFO, PRINT$, SDFIC, SDFOC,
      PFI$, POSTPR, PREPRM, SDFID, ERR$, PFWL$, SDFOD

EX42-00002*WORK.DYNCOR
                              $(1)   024131 024316
      EXTERNAL REFERENCES:  LASTD$, MCORE$, LCORE$, PRINT$, EABT$

             SEGMENT A*                024317 031102       054125 055164
             FOLLOWS SEGMENT MAIN


SYS$*RLIB$.NEXP1$/FOR68
                              $(1)   024317 024354       $(0)   054125 054125
      EXTERNAL REFERENCES:  NERRB$, NERRC$

SYS$*RLIB$.NEXP6$/FOR68
                              $(1)   024355 024551       $(2)   054126 054177
      EXTERNAL REFERENCES:  NERRA$, NERRB$, NERRC$

SYS$*RLIB$.NEXP5$/FOR68
                              $(1)   024552 024637       $(2)   054200 054207
      EXTERNAL REFERENCES:  NERRA$, NERRB$, NERRC$

EX42-00002*WORK.DECBCD/DLG
                              $(1)   024640 025061       $(0)   054210 054250
                              $(3)   DILOG               $(2)   BLANK$COMMON
      EXTERNAL REFERENCES:  MNCOD$, XPRI, NID1$, NID2$, NERR3$

EX42-00002*WORK.VALIMG/DLG
                              $(1)   025062 025340       $(0)   054251 054320
                              $(3)   DILOG               $(2)   BLANK$COMMON
      EXTERNAL REFERENCES:  PUT, MOVER, DECBCD, STRMOV, INTBCD, NERR2$,
      NWDU$, NID2$, NERR3$

EX42-00002*WORK.BCDVAL
                              $(1)   025341 025421       $(0)   054321 054325
                              $(3)   DILOG               $(2)   BLANK$COMMON
      EXTERNAL REFERENCES:  DECIDE, BCDDEC, BCDINT, NERR2$, NERR3$
```

```
EX42-00002◆WORK.FLDATA
                         $(1)  025422 025436     $(0)  054326 054327

EX42-00002◆WORK.CHRNUM/DLG
                         $(1)  025437 025521     $(0)  054330 054341
                                                 $(2)  BLANK$COMMON
     EXTERNAL REFERENCES:  NERR3$

EX42-00002◆WORK.SCALE
                         $(1)  025522 025756     $(0)  054342 054430
                         $(3)  DIRECT            $(2)  BLANK$COMMON
                                                 $(4)  DILOG
     EXTERNAL REFERENCES:  RANDAC, DECIDE, BCDDEC, NERR2$, XPRR, XPII,
     NWDU$, NIO3$, NIO1$, NIO2$, NERR3$

EX42-00002◆WORK.GETSUB/DLG
                         $(1)  025757 026123     $(0)  054431 054505
                         $(3)  DILOG             $(2)  BLANK$COMMON
     EXTERNAL REFERENCES:  MOVER, DECIDE, RANDAC, BCDDEC, BCDINT,
     NERR2$, NWDU$, NIO2$, NERR3$

EX42-00002◆WORK.RPLACE/DLG
                         $(1)  026124 027014     $(0)  054506 054706
                         $(3)  SEARH             $(2)  BLANK$COMMON
                         $(5)  CARDBD            $(4)  DILOG
     EXTERNAL REFERENCES:  LOCP, RANDAC, DBLOAD, GETSUB, SCALE, BCDVAL,
     WRITCR, PGMOUT, MOVER, VALIMG, PUT, NERR3$

EX42-00002◆WORK.IVCALC
                         $(1)  027015 027064     $(0)  054707 054716
                         $(3)  DILOG             $(2)  BLANK$COMMON
     EXTERNAL REFERENCES:  NERR3$

EX42-00002◆WORK.INTBCD/DLG
                         $(1)  027065 027212     $(0)  054717 054734
                         $(3)  DILOG             $(2)  BLANK$COMMON
     EXTERNAL REFERENCES:  INTFLD, GET, PUT, NERR3$

EX42-00002◆WORK.IVDESC
                         $(1)  027213 027251     $(0)  054735 054744
                         $(3)  DILOG             $(2)  BLANK$COMMON
     EXTERNAL REFERENCES:  NERR3$

EX42-00002◆WORK.STRMOV
                         $(1)  027252 027334     $(0)  054745 054755
                                                 $(2)  BLANK$COMMON
     EXTERNAL REFERENCES:  GET, PUT, NERR3$

EX42-00002◆WORK.WRITCR
                         $(1)  027335 027424     $(0)  054756 054767
                         $(3)  DILOG             $(2)  BLANK$COMMON
     EXTERNAL REFERENCES:  NWDA$, NIO1$, NIO2$, NWDU$, NERR3$
```

```
EX42-00002*WORK.BCDINT
                                   $(1)   027425 027602       $(0)   054770 055014
                                                              $(2)   BLANK$COMMON
       EXTERNAL REFERENCES:  CHRNUM, NERR3$

EX42-00002*WORK.BCDDEC/DLG
                                   $(1)   027603 030167       $(0)   055015 055047
                                                              $(2)   BLANK$COMMON
       EXTERNAL REFERENCES:  BCDINT, CHRNUM, XPRI, XPRR, NERR3$

EX42-00002*WORK.DECIDE/DLG
                                   $(1)   030170 030430       $(0)   055050 055074
                                   $(3)   DILOG               $(2)   BLANK$COMMON
       EXTERNAL REFERENCES:  STRIP6, CHRNUM, GET, NERR3$

EX42-00002*WORK.INMOD/DLG
                                   $(1)   030431 030660       $(0)   055075 055141
                                   $(3)   SEARH               $(2)   BLANK$COMMON
                                                              $(4)   DILOG
       EXTERNAL REFERENCES:  PAGE, RANDAC, DBLOAD, RPLACE, PGMOUT,
       WRITCR, NWDU$, NIO2$, NIO3$, NERR3$

EX42-00002*WORK.RSPOND/DLG
                                   $(1)   030661 031102       $(0)   055142 055164
                                   $(3)   DILOG               $(2)   BLANK$COMMON
                                   $(5)   BCNTL               $(4)   UNITS
       EXTERNAL REFERENCES:  PAGE, ADDER, ATTACH, CHANGE, IGNORE, COPY,
       INITIZ, CSF, IDENT, DELETE, DETACH, FORMAT, INLINE, INSRT, OFF,
       ON, DBLOAD, CCDUMP, SEARCH, TIME, USE, UPDATE, PRTT, NWDU$, NIO2$,
       NERR2$, NERR3$

                SEGMENT B*                024317 025306      054125 054556
                HAS THE SAME STARTING ADDRESS AS SEGMENT A


EX42-00002*WORK.IOPT
                                   $(1)   024317 024321
       EXTERNAL REFERENCES:  OPT$

EX42-00002*WORK.INITL/DLG
                                   $(1)   024322 024650       $(0)   054125 054216
                                   $(3)   SEARH               $(2)   BLANK$COMMON
                                   $(5)   DILOG               $(4)   CARDBD
       EXTERNAL REFERENCES:  NERTRN, NPRT$, NIO2$, NERR3$

EX42-00002*WORK.INITDM/DLG
                                   $(1)   024651 024674       $(0)   054217 054222
                                   $(3)   UNITS               $(2)   BLANK$COMMON
                                   $(5)   DILOG               $(4)   MS
       EXTERNAL RE  RENCES:  NERR3$
```

```
EX42-00002*WORK.OPTION/DLG
                        $(1)  024675 025010     $(0)  054223 054267
                        $(3)  OPTDIR            $(2)  BLANK$COMMON
                                                $(4)  DILOG
      EXTERNAL REFERENCES:  RANDAC, IOPT, GET, NERR3$

EX42-00002*WORK.OPINIT/DLG
                        $(1)  025011 025114     $(0)  054270 054330
                        $(3)  DIRECT            $(2)  BLANK$COMMON
                                                $(4)  DILOG
      EXTERNAL REFERENCES:  RANDAC, NERR3$

EX42-00002*WORK.NUMNIT
                        $(1)  025115 025211     $(0)  054331 054403
                        $(3)  DIRECT            $(2)  BLANK$COMMON
                                                $(4)  DILOG
      EXTERNAL REFERENCES:  RANDAC, NERR3$

EX42-00002*WORK.CDINIT/DLG
                        $(1)  025212 025306     $(0)  054404 054556
                        $(3)  DIRECT            $(2)  BLANK$COMMON
                                                $(4)  DILOG
      EXTERNAL REFERENCES:  RANDAC, NERR3$

             SEGMENT D*              031103 032377      055165 055412
             FOLLOWS SEGMENT A


EX42-00002*WORK.ADDONE/DLG
                        $(1)  031103 031511     $(0)  055165 055271
                        $(3)  DILOG             $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  RANDAC, IVCALC, MOVER, IVDESC, NWDU$, NIO1$,
      NIO2$, NERR6$, NERR3$

EX42-00002*WORK.ADDER/DLG
                        $(1)  031512 032377     $(0)  055272 055412
                        $(3)  CARDBD            $(2)  BLANK$COMMON
                                                $(4)  DILOG
      EXTERNAL REFERENCES:  PAGE, LOCP, DECIDE, RANDAC, GETSUB, BCDINT,
      SCALE, INTBCD, MOVER, DECBCD, ADDONE, NWDU$, NIO2$, NERR2$,
      NERR3$

             SEGMENT E*              031103 031120      055165 055177
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002*WORK.ATTACH/DLG
                        $(1)  031103 031120     $(0)  055165 055177
                        $(3)  DILOG             $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NWDU$, NIO2$, NERR3$
```

```
          SEGMENT F◆                  031103 031347      055165 055263
          HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002◆WORK.CHANGE/DLG
                            $(1)   031103 031347      $(0)   055165 055263
                            $(3)   DILOG              $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  PAGE, DECIDE, BCDINT, BCDDEC, NWDU$, NIO2$,
     NERR2$, NERR3$

          SEGMENT G◆                  031103 031355      055165 055221
          HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002◆WORK.IGNORE/DLG
                            $(1)   031103 031355      $(0)   055165 055221
                            $(3)   DILOG              $(2)   BLANK$COMMON
                                                      $(4)   CARDBD
     EXTERNAL REFERENCES:  PAGE, MOVER, STRMOV, PGMOUT, WRITCR, NWDU$,
     NIO2$, NERR3$

          SEGMENT H◆                  031103 031120      055165 055177
          HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002◆WORK.COPY/DLG
                            $(1)   031103 031120      $(0)   055165 055177
                            $(3)   DILOG              $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  NWDU$, NIO2$, NERR3$

          SEGMENT I◆                  031103 031553      055165 055321
          HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002◆WORK.DBINIT/DLG
                            $(1)   031103 031424      $(0)   055165 055267
                            $(3)   CARDBD             $(2)   BLANK$COMMON
                                                      $(4)   DILOG
     EXTERNAL REFERENCES:  PAGE, DECIDE, BCDDEC, BCDINT, DISECT, NWDU$,
     NIO2$, NERR2$, NID1$, NERR3$

EX42-00002◆WORK.INITIZ/DLG
                            $(1)   031425 031553      $(0)   055270 055321
                            $(3)   DILOG              $(2)   BLANK$COMMON
     EXTERNAL REFERENCES:  PAGE, DBINIT, DYNCOR, PANDAC, NWDU$, NIO2$,
     NERR3$

          SEGMENT J◆                  031103 031232      055165 055213
          HAS THE SAME STARTING ADDRESS AS SEGMENT D
```

```
EX42-00002*WORK.CSF/DLG
                             $(1)  031103 031232      $(0)  055165 055213
                             $(3)  DILOG              $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  PAGE, GET, PUT, NERTRN, NPRT$, NIO2$,
      NERR3$

             SEGMENT K*              031103 031472      055165 055250
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002*WORK.IDENT/DLG
                             $(1)  031103 031472      $(0)  055165 055250
                             $(3)  DILOG              $(2)  BLANK$COMMON
                                                      $(4)  CARDBD
      EXTERNAL REFERENCES:  PAGE, DECIDE, BCDDEC, BCDINT, RANDAC,
      IVDESC, MOVER, STRMOV, IVCALC, NWDU$, NIO2$, NERR3$

             SEGMENT L*              031103 031263      055165 055221
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002*WORK.DELETE/DLG
                             $(1)  031103 031263      $(0)  055165 055221
                             $(3)  CARDBD             $(2)  BLANK$COMMON
                                                      $(4)  DILOG
      EXTERNAL REFERENCES:  PAGE, RANDAC, IVDESC, MOVER, NWDU$, NIO2$,
      NERR3$

             SEGMENT M*              031103 031120      055165 055177
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002*WORK.DETACH/DLG
                             $(1)  031103 031120      $(0)  055165 055177
                             $(3)  DILOG              $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NWDU$, NIO2$, NERR3$

             SEGMENT N*              031103 031412      055165 055254
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002*WORK.FORMAT/DLG
                             $(1)  031103 031412      $(0)  055165 055254
                             $(3)  UNITS              $(2)  BLANK$COMMON
                             $(5)  DILOG              $(4)  BCNTL
      EXTERNAL REFERENCES:  MOVER, PAGE, STRIP6, BCDINT, STRMOV, DMAN,
      NNCOD$, PGMOUT, NPRT$, NIO2$, NIO1$, NIO3$, NERR3$

             SEGMENT O*              031103 031120      055165 055177
             HAS THE SAME STARTING ADDRESS AS SEGMENT D
```

```
EX42-00002*WORK.INLINE/DLG
                              $(1)  031103 031120      $(0)  055165 055177
                              $(3)  DILOG              $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NWDU$, NIO2$, NERR3$

             SEGMENT P*                031103 032265      055165 055402
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


SYS$*RLIB$.NFINP$/FOR69
                              $(1)  031103 031530      $(2)  055165 055250
      EXTERNAL REFERENCES:  NTAB$, NHPFA$, NRSX$, IOCOD$, NFCHK$, NERU$,
      NIO1V$, NIO2V$, NIO3V$, NFTCB$, NIO1B$, STREG$, UNIT$, NSTAT$,
      NERCT$, NDT$, NDPFDL$, NFBY1$, NIO2$, NBFPL$, WAIT$, NIOER$,
      NDANW$, NNWDL$, NC1UL0, NIO3$

EX42-00002*WORK.READBR
                              $(1)  031531 031633      $(0)  055251 055265
                              $(3)  DILOG              $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  EOFTST, NRDA$, NIO1$, NIO2$, NRBU$, NERR3$
BUFFER(COMMONBLOCK)                                          055266 055310

EX42-00002*WORK.INSRT/DLG
                              $(1)  031634 032265      $(0)  055311 055402
                              $(3)  BUFFER             $(2)  BLANK$COMMON
                              $(5)  CARDBD             $(4)  DILOG
      EXTERNAL REFERENCES:  PAGE, DECIDE, BCDDEC, BCDINT, CREATF,
      READBR, PGMOUT, WRITCR, NWDU$, NIO2$, NERR3$

             SEGMENT Q*                031103 031212      055165 055212
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002*WORK.ONPOFF/DLG
                              $(1)  031103 031212      $(0)  055165 055212
                              $(3)  OPTDIR             $(2)  BLANK$COMMON
                                                       $(4)  DILOG
      EXTERNAL REFERENCES:  PAGE, RANDAC, NWDU$, NIO1$, NIO2$, NERR3$

             SEGMENT R*                031103 031724      055165 055742
             HAS THE SAME STARTING ADDRESS AS SEGMENT D


EX42-00002*WORK.SHELL
                              $(1)  031103 031243      $(0)  055165 055215
                                                       $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NERR3$
```

```
EX42-00002*WORK.DBWRT
                         $(1)  031244 031516      $(0)  055216 055313
                         $(3)  DILOG              $(2)  BLANK$COMMON
    EXTERNAL REFERENCES:  INTBCD, SHELL, RANDAC, IVDESC, NWDU$, NID2$,
    NID1$, NERR3$

EX42-00002*WORK.CCDUMP/DLG
                         $(1)  031517 031724      $(0)  055314 055742
                         $(3)  DILOG              $(2)  BLANK$COMMON
    EXTERNAL REFERENCES:  DBWRT, NWDU$, NID2$, NERR3$

         SEGMENT S*                031103 031120      055165 055177
         HAS THE SAME STARTING ADDRESS AS SEGMENT D

EX42-00002*WORK.SEARCH/DLG
                         $(1)  031103 031120      $(0)  055165 055177
                         $(3)  DILOG              $(2)  BLANK$COMMON
    EXTERNAL REFERENCES:  NWDU$, NID2$, NERR3$

         SEGMENT T*                031103 031120      055165 055177
         HAS THE SAME STARTING ADDRESS AS SEGMENT D

EX42-00002*WORK.TIME/DLG
                         $(1)  031103 031120      $(0)  055165 055177
                         $(3)  DILOG              $(2)  BLANK$COMMON
    EXTERNAL REFERENCES:  NWDU$, NID2$, NERR3$

         SEGMENT U*                031103 031234      055165 055212
         HAS THE SAME STARTING ADDRESS AS SEGMENT D

EX42-00002*WORK.USE/DLG
                         $(1)  031103 031234      $(0)  055165 055212
                         $(3)  SEARH              $(2)  BLANK$COMMON
                         $(5)  DILOG              $(4)  CARDBD
    EXTERNAL REFERENCES:  PAGE, DBLOAD, NPRT$, NID2$, NID1$, NERR3$

         SEGMENT V*                031103 031157      055165 055214
         HAS THE SAME STARTING ADDRESS AS SEGMENT D

EX42-00002*WORK.UPDATE/DLG
                         $(1)  031103 031157      $(0)  055165 055214
                         $(3)  DILOG              $(2)  BLANK$COMMON
    EXTERNAL REFERENCES:  PAGE, DBLOAD, NWDU$, NID2$, NERR3$

         SEGMENT W*                031103 031477      055165 055251
         HAS THE SAME STARTING ADDRESS AS SEGMENT D
```

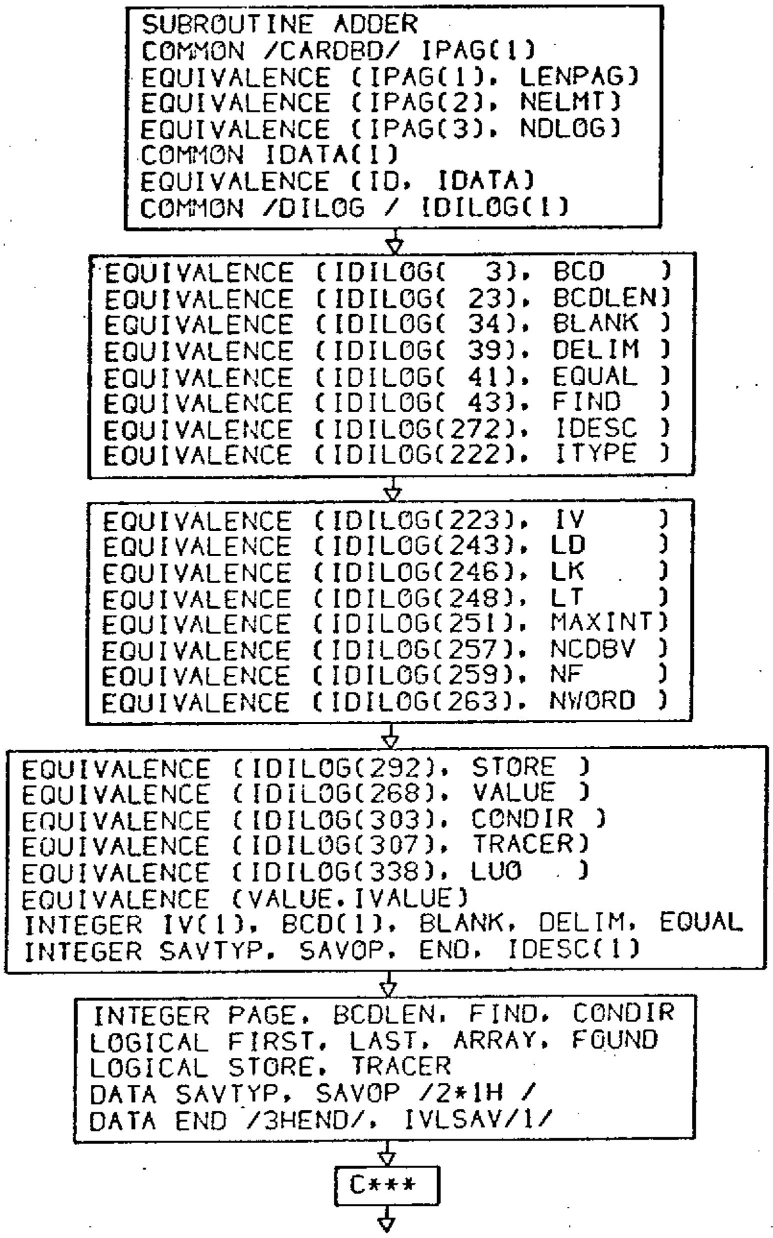
```
EX42-00002◆WORK.PRTT
                        $(1)  031103 031477     $(0)  055165 055251
                        $(3)  MS                $(2)  BLANK$COMMON
      EXTERNAL REFERENCES:  NXTAD, NDEF$, NRDA$, NID1$, NID2$, NWDA$,
      NPRT$, NERR3$


IBANK DRAWN TO SCALE:     200 WORDS DECIMAL PER DASH

MAIN (9935)
--------------------------------------------------
                                                  A◆ (2420)
                                                  ------------
                                                              D◆ (701)
                                                              ---
                                                              W◆ (253)
                                                              -
                                                              V◆ (45)
                                                              -
                                                              U◆ (90)
                                                              -
                                                              T◆ (14)
                                                              -
                                                              S◆ (14)
                                                              -
                                                              R◆ (402)
                                                              --
                                                              Q◆ (72)
                                                              -
                                                              P◆ (627)
                                                              ---
                                                              O◆ (14)
                                                              -
                                                              N◆ (200)
                                                              -
                                                              M◆ (14)
                                                              -
                                                              L◆ (113)
                                                              -
                                                              K◆ (248)
                                                              -
                                                              J◆ (88)
                                                              -
                                                              I◆ (297)
                                                              -
                                                              H◆ (14)
                                                              -
                                                              G◆ (171)
                                                              -
```

```
                                                               F◆ (165)
                                                               -
                                                               E◆ (14)
                                                               -
                                                   B◆ (504)
                                                   --
DBANK DRAWN TO SCALE:    100 WORDS DECIMAL PER DASH
   MAIN (5972)
   ------------------------------------------------------------
                                                               A◆ (544)
                                                               ------
                                                                     D◆ (
150)
                                                                     -
                                                                     W◆ (
53)
                                                                     -
                                                                     V◆ (
24)
                                                                     -
                                                                     U◆ (
22)
                                                                     -
                                                                     T◆ (
11)
                                                                     -
                                                                     S◆ (
11)
                                                                     -
                                                                     R◆ (
366)
                                                                     ---
                                                                     Q◆ (
22)
                                                                     -
                                                                     P◆ (
142)
                                                                     -
                                                                     O◆ (
11)
                                                                     -
                                                                     N◆ (
56)
                                                                     -
                                                                     M◆ (
11)
                                                                     -
                                                                     L◆ (
29)
```

```
—..
                                                                        —
                                                                        K◆ (
52)
                                                                        —
                                                                        J◆ (
23)
                                                                        —
                                                                        I◆ (
93)
                                                                        —
                                                                        H◆ (
11)
                                                                        —
                                                                        G◆ (
29)
                                                                        —
                                                                        F◆ (
63)
                                                                        —
                                                                        E◆ (
11)
                                                                        —
                                                                  B◆ (282)
                                                                  ———
```

```
INDIRECT LOAD TABLE
CALLS ON THE FOLLOWING IBANK ENTRY POINTS IN INDIRECT LOAD SEGMENTS ARE
ROUTED VIA THESE INDIRECT LOAD ADDRESSES, TO INSURE SEGMENTS ARE LOADED
ADDER         040134     ADDONE        040137     ATTACH        040142
BCDDEC        040145     BCDINT        040150     BCDVAL        040153
CCDUMP        040156     CDINIT        040161     CHANGE        040164
CHRNUM        040167     COPY          040172     CSF           040175
DBINIT        040200     DBWRT         040203     DECBCD        040206
DECIDE        040211     DELETE        040214     DETACH        040217
FORMAT        040222     GETSUB        040225     IDENT         040230
IGNORE        040233     INITDM        040236     INITIZ        040241
INITL         040244     INLINE        040247     INMOD         040252
INSRT         040255     INTBCD        040260     INTFLD        040263
IOPT          040266     IVCALC        040271     IVDESC        040274
NRBU$         040277     NUMNIT        040302     OFF           040305
ON            040310     OPINIT        040313     OPTION        040316
PRTT          040321     READBR        040324     RPLACE        040327
RSPOND        040332     SCALE         040335     SEARCH        040340
SHELL         040343     STRMOV        040346     TIME          040351
UPDATE        040354     USE           040357     VALIMG        040362
WRITCR        040365     XPII          040370     XPRI          040373
XPRR          040376
```

APPENDIX A - FLOW CHARTS OF SELECTED SUBROUTINES



```
SUBROUTINE ADDER
COMMON /CARDBD/ IPAG(1)
EQUIVALENCE (IPAG(1), LENPAG)
EQUIVALENCE (IPAG(2), NELMT)
EQUIVALENCE (IPAG(3), NDLOG)
COMMON IDATA(1)
EQUIVALENCE (ID, IDATA)
COMMON /DILOG / IDILOG(1)
```

```
EQUIVALENCE (IDILOG(  3), BCD    )
EQUIVALENCE (IDILOG( 23), BCDLEN)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 41), EQUAL )
EQUIVALENCE (IDILOG( 43), FIND  )
EQUIVALENCE (IDILOG(272), IDESC )
EQUIVALENCE (IDILOG(222), ITYPE )
```

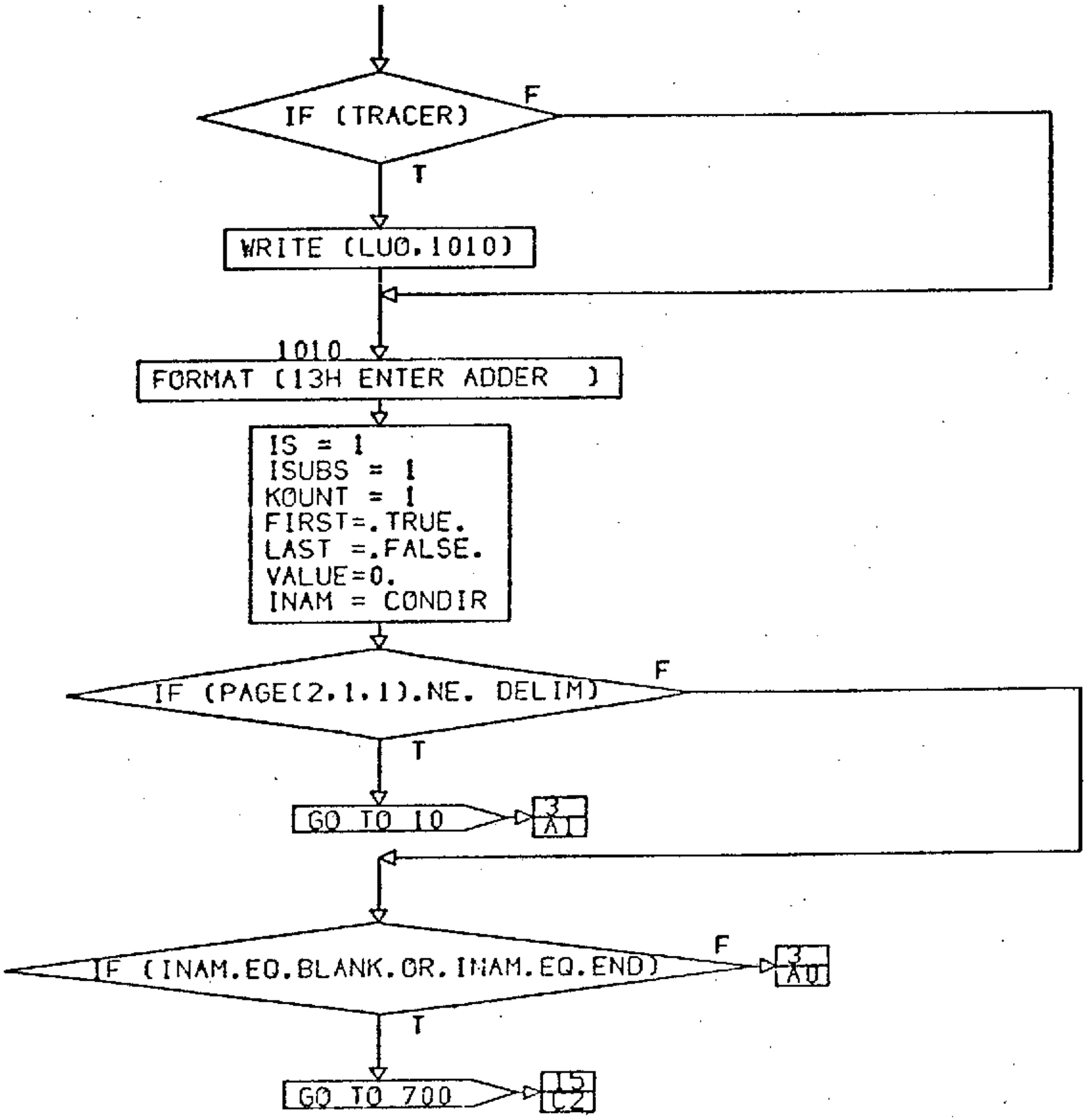
```
EQUIVALENCE (IDILOG(223), IV    )
EQUIVALENCE (IDILOG(243), LD    )
EQUIVALENCE (IDILOG(246), LK    )
EQUIVALENCE (IDILOG(248), LT    )
EQUIVALENCE (IDILOG(251), MAXINT)
EQUIVALENCE (IDILOG(257), NCDBV )
EQUIVALENCE (IDILOG(259), NF    )
EQUIVALENCE (IDILOG(263), NWORD )
```

```
EQUIVALENCE (IDILOG(292), STORE )
EQUIVALENCE (IDILOG(268), VALUE )
EQUIVALENCE (IDILOG(303), CONDIR )
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(338), LUO   )
EQUIVALENCE (VALUE,IVALUE)
INTEGER IV(1), BCD(1), BLANK, DELIM, EQUAL
INTEGER SAVTYP, SAVOP, END, IDESC(1)
```

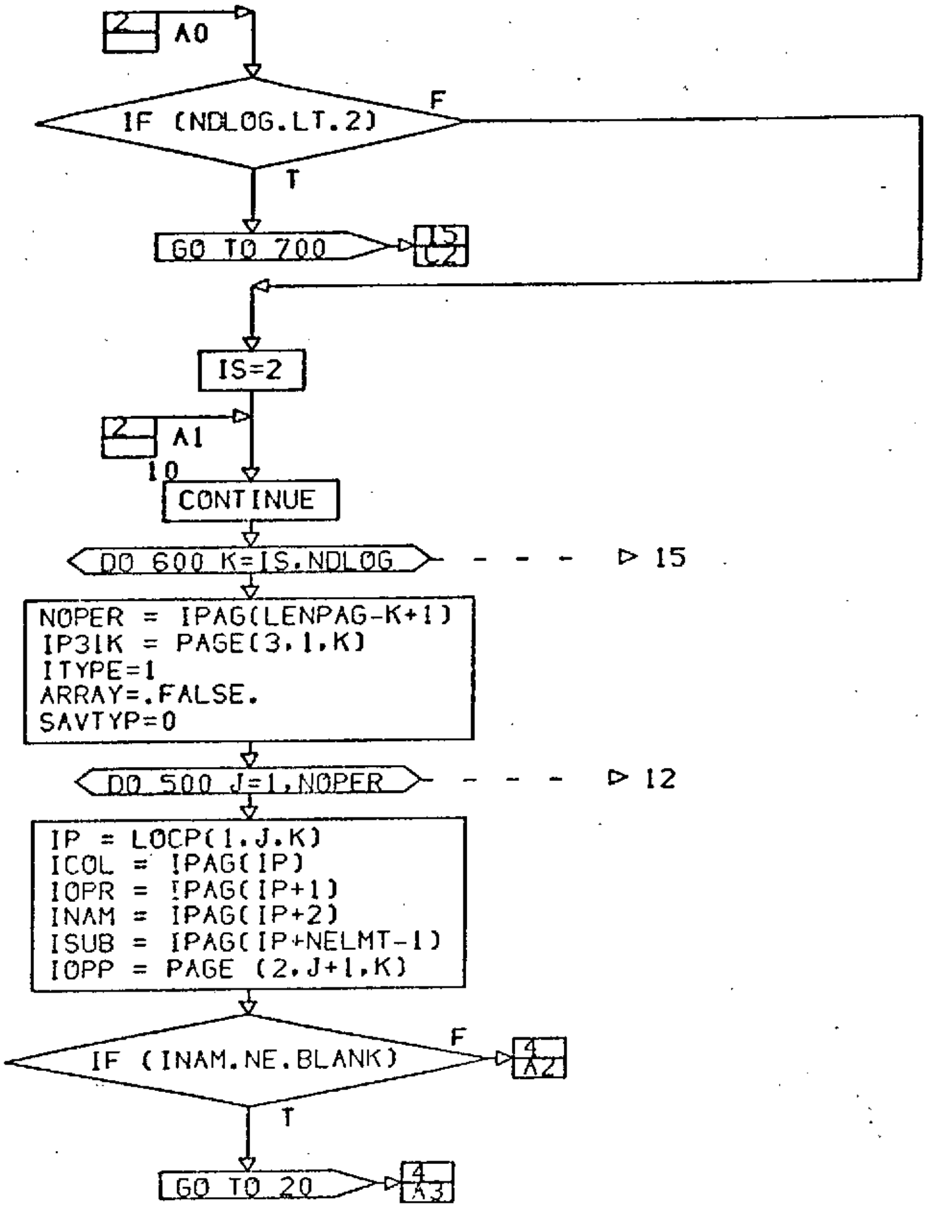
```
INTEGER PAGE, BCDLEN, FIND, CONDIR
LOGICAL FIRST, LAST, ARRAY, FOUND
LOGICAL STORE, TRACER
DATA SAVTYP, SAVOP /2*1H /
DATA END /3HEND/, IVLSAV/1/
```

```
C***
```

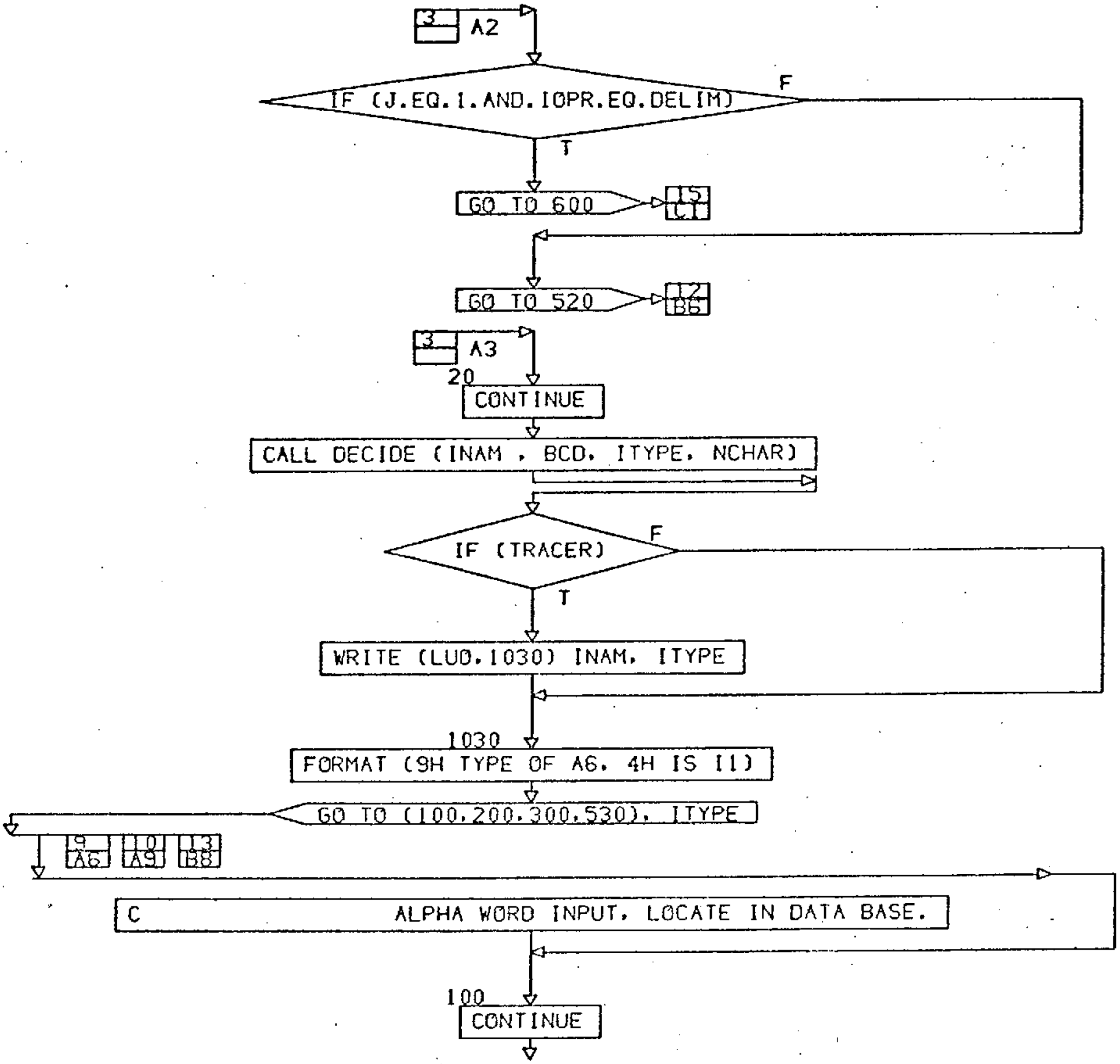
CONT. ON PG 2


IF (TRACER)
F
T
WRITE (LUO,1010)
1010
FORMAT (13H ENTER ADDER)
IS = 1
ISUBS = 1
KOUNT = 1
FIRST=.TRUE.
LAST =.FALSE.
VALUE=0.
INAM = CONDIR
IF (PAGE(2,1,1).NE. DELIM)
F
T
GO TO 10
3
A1
IF (INAM.EQ.BLANK.OR.INAM.EQ.END)
F
3
A0
T
GO TO 700
15
C2

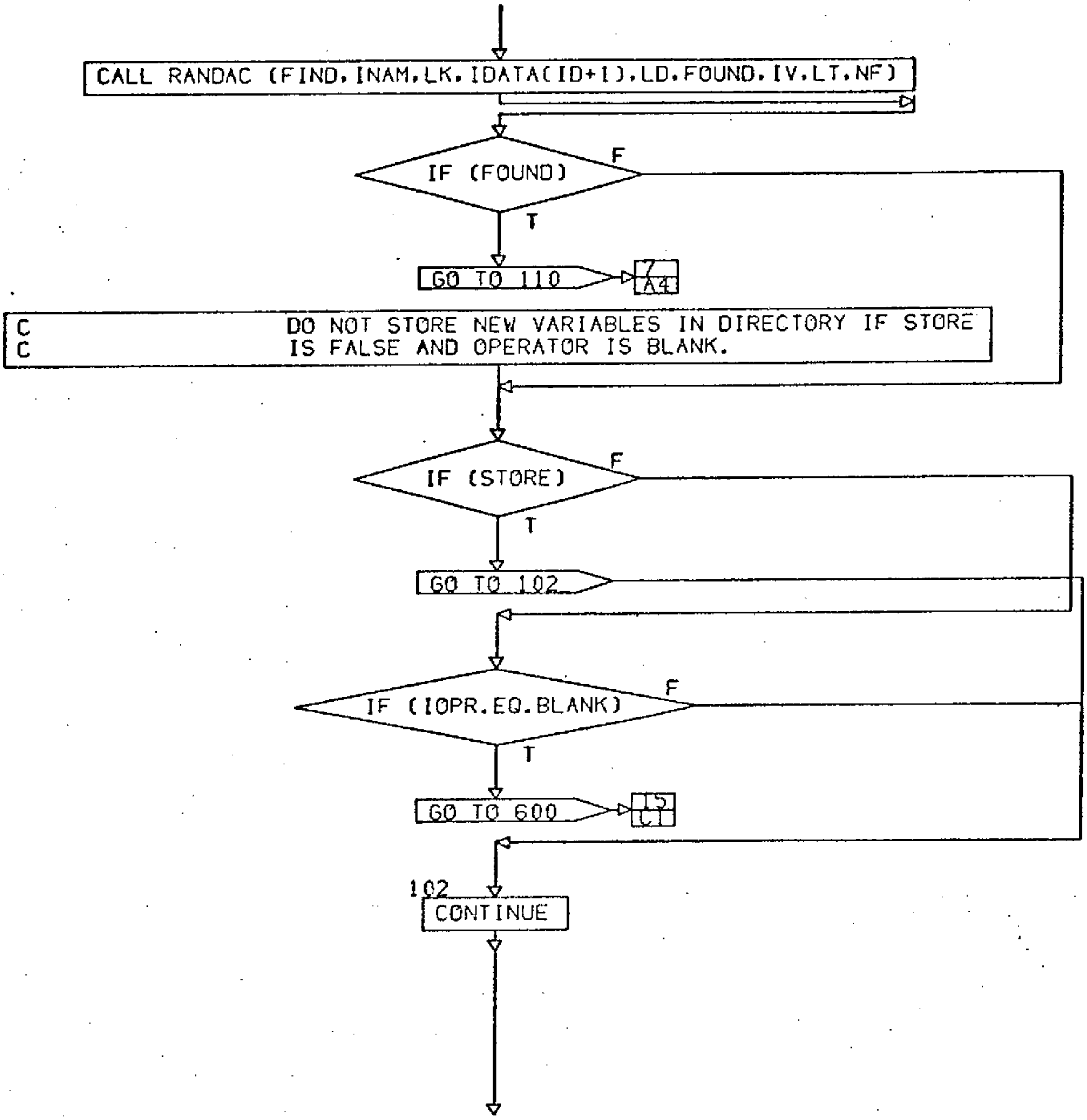
CONT. ON PG 3


2
A0
IF (NDLOG.LT.2)
F
T
GO TO 700
15
C2
IS=2
2
A1
10
CONTINUE
DO 600 K=IS,NDLOG
▷ 15
NOPER = IPAG(LENPAG-K+1)
IP31K = PAGE(3,1,K)
ITYPE=1
ARRAY=.FALSE.
SAVTYP=0
DO 500 J=1,NOPER
▷ 12
IP = LOCP(1,J,K)
ICOL = IPAG(IP)
IOPR = IPAG(IP+1)
INAM = IPAG(IP+2)
ISUB = IPAG(IP+NELMT-1)
IOPP = PAGE (2,J+1,K)
IF (INAM.NE.BLANK)
F
4
A2
T
GO TO 20
4
A3

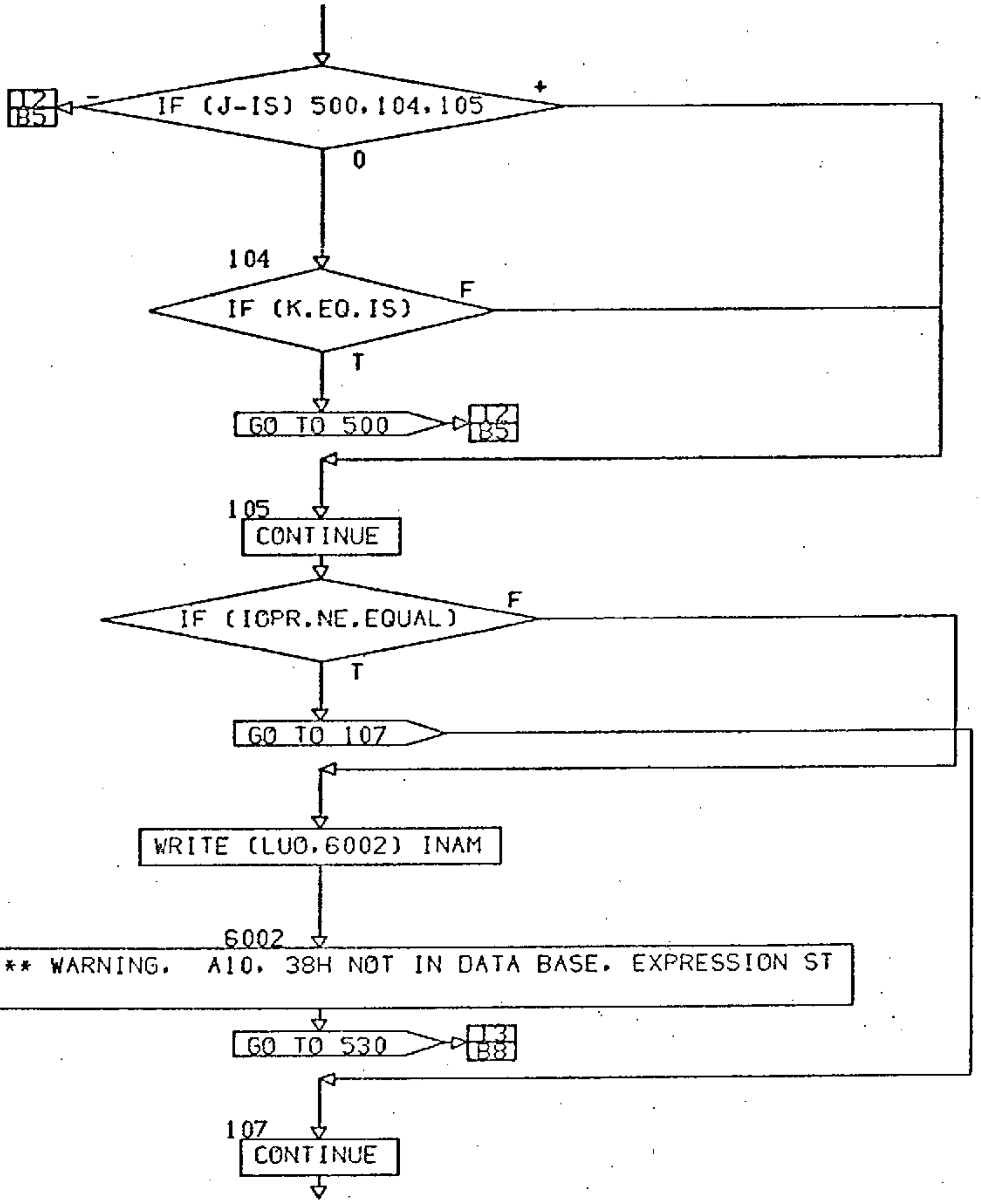
CONT. ON PG 4


3 A2
IF (J.EQ.1.AND.IOPR.EQ.DELIM)
F
T
GO TO 600
15 C1
GO TO 520
12 B6
3 A3
20
CONTINUE
CALL DECIDE (INAM , BCD, ITYPE, NCHAR)
IF (TRACER)
F
T
WRITE (LUO,1030) INAM, ITYPE
1030
FORMAT (9H TYPE OF A6, 4H IS I1)
GO TO (100,200,300,530), ITYPE
9 A6
10 A9
13 B8
C ALPHA WORD INPUT, LOCATE IN DATA BASE.
100
CONTINUE

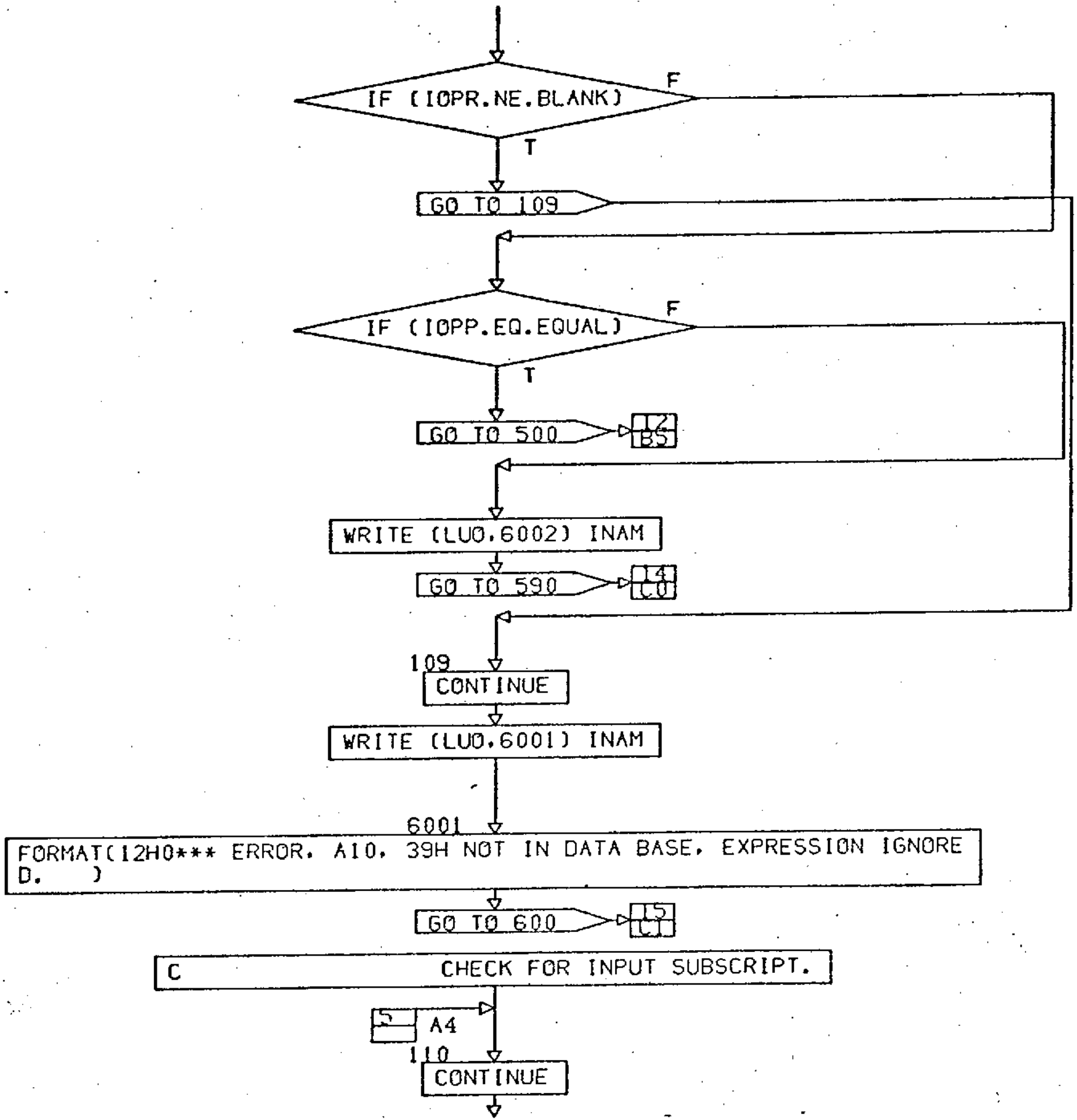
CONT. ON PG 5


CALL RANDAC (FIND, INAM, LK, IDATA(ID+1), LD, FOUND, IV, LT, NF)
IF (FOUND)
F
T
GO TO 110
7
A4
C DO NOT STORE NEW VARIABLES IN DIRECTORY IF STORE
C IS FALSE AND OPERATOR IS BLANK.
IF (STORE)
F
T
GO TO 102
IF (IOPR.EQ.BLANK)
F
T
GO TO 600
15
C1
102
CONTINUE

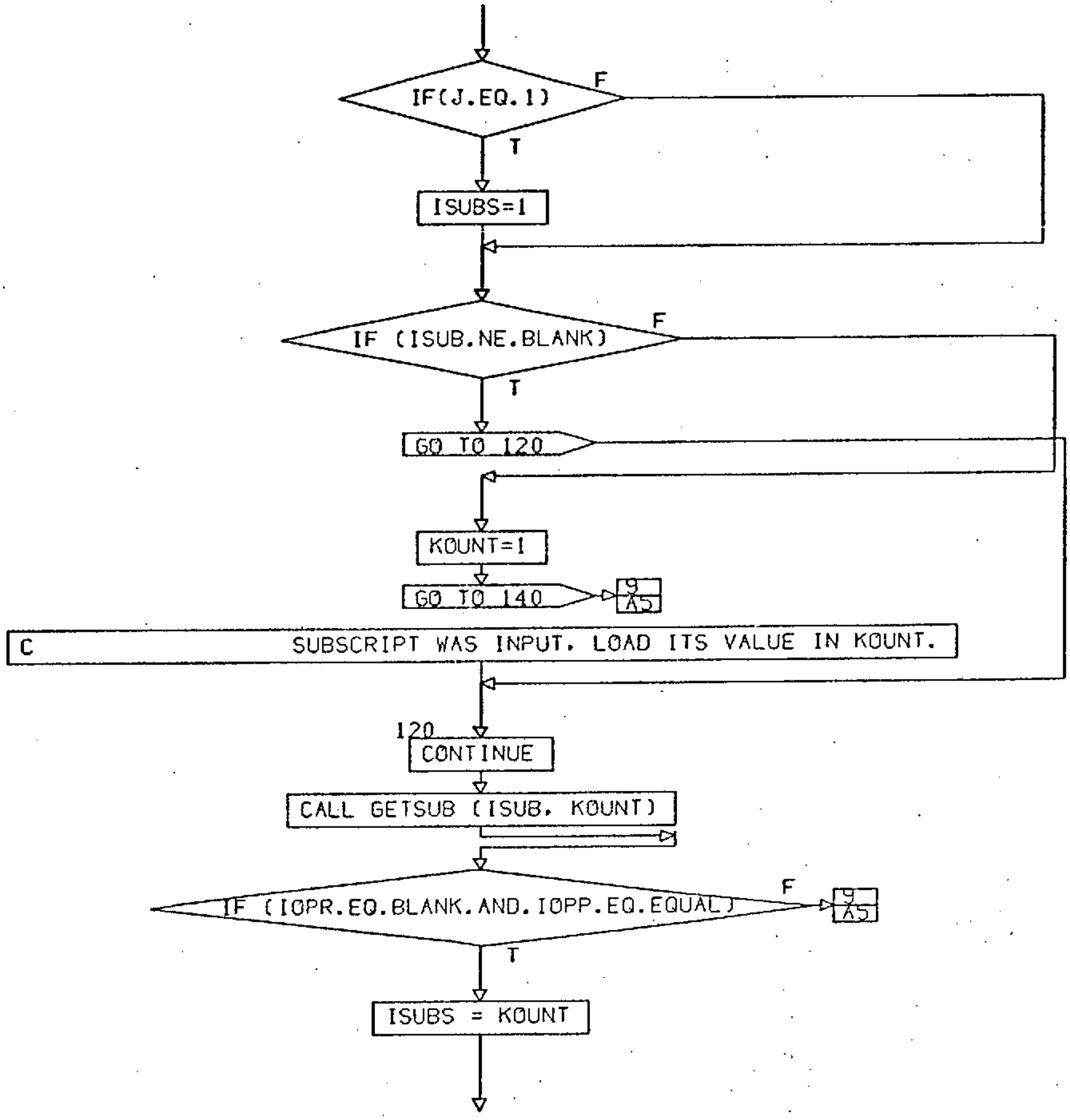
CONT. ON PG 6


I2
B5
-
IF (J-IS) 500,104,105
+
0
104
IF (K.EQ.IS)
F
T
GO TO 500
I2
B5
105
CONTINUE
IF (IOPR.NE.EQUAL)
F
T
GO TO 107
WRITE (LUO,6002) INAM
6002
FORMAT(14H0*** WARNING. A10, 38H NOT IN DATA BASE. EXPRESSION ST
ORED.   )
GO TO 530
I3
B8
107
CONTINUE

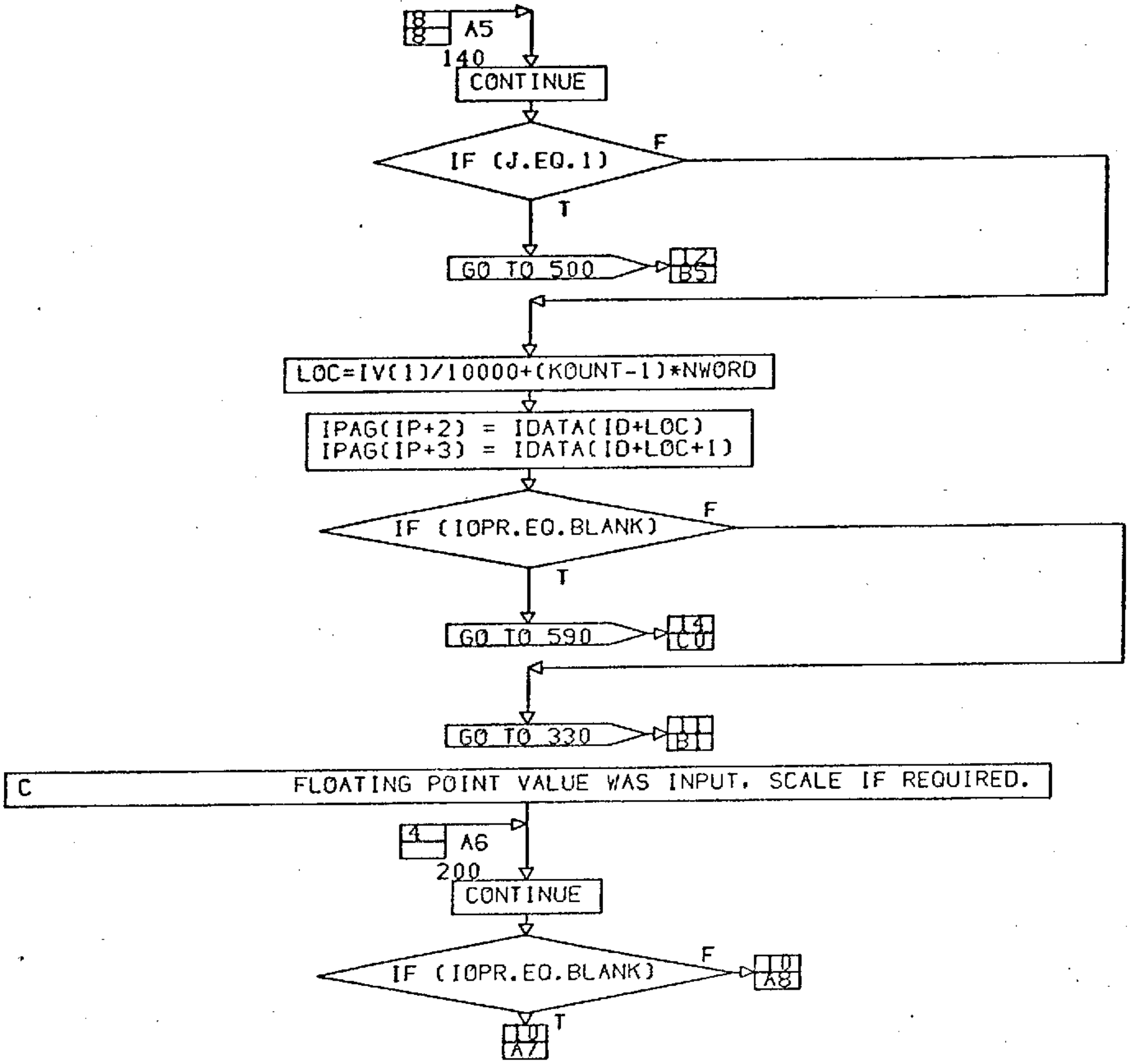
CONT. ON PG 7


IF (IOPR.NE.BLANK)
F
T
GO TO 109
IF (IOPP.EQ.EQUAL)
F
T
GO TO 500
12
B5
WRITE (LUO,6002) INAM
GO TO 590
14
C0
109
CONTINUE
WRITE (LUO,6001) INAM
6001
FORMAT(12H0*** ERROR, A10, 39H NOT IN DATA BASE, EXPRESSION IGNORE
D.)
GO TO 600
15
C1
C CHECK FOR INPUT SUBSCRIPT.
5
A4
110
CONTINUE

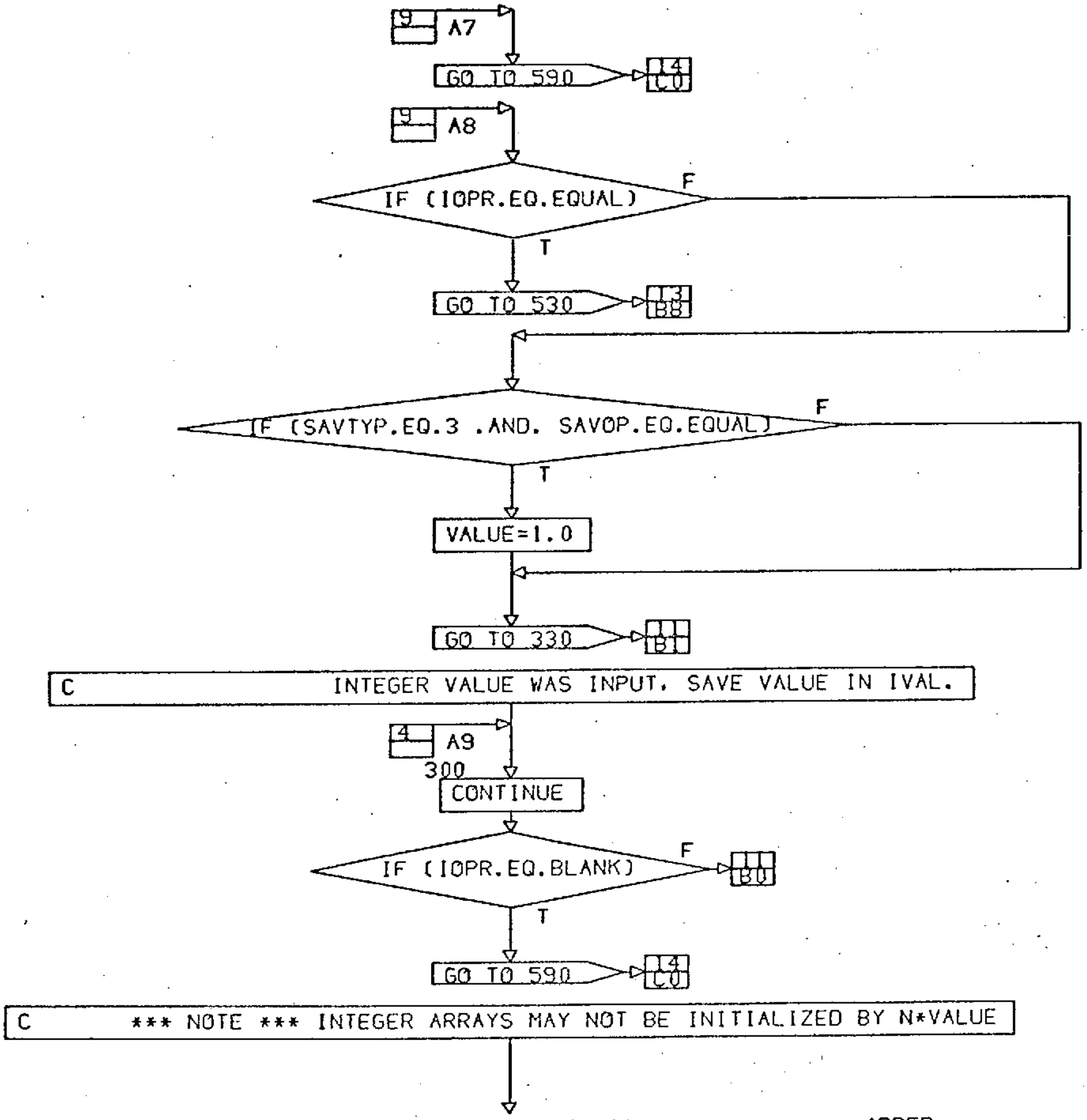
CONT. ON PG 8


IF(J.EQ.1)
F
T
ISUBS=1
IF (ISUB.NE.BLANK)
F
T
GO TO 120
KOUNT=1
GO TO 140
9
A5
C SUBSCRIPT WAS INPUT. LOAD ITS VALUE IN KOUNT.
120
CONTINUE
CALL GETSUB (ISUB, KOUNT)
IF (IOPR.EQ.BLANK.AND.IOPP.EQ.EQUAL)
F
9
A5
T
ISUBS = KOUNT

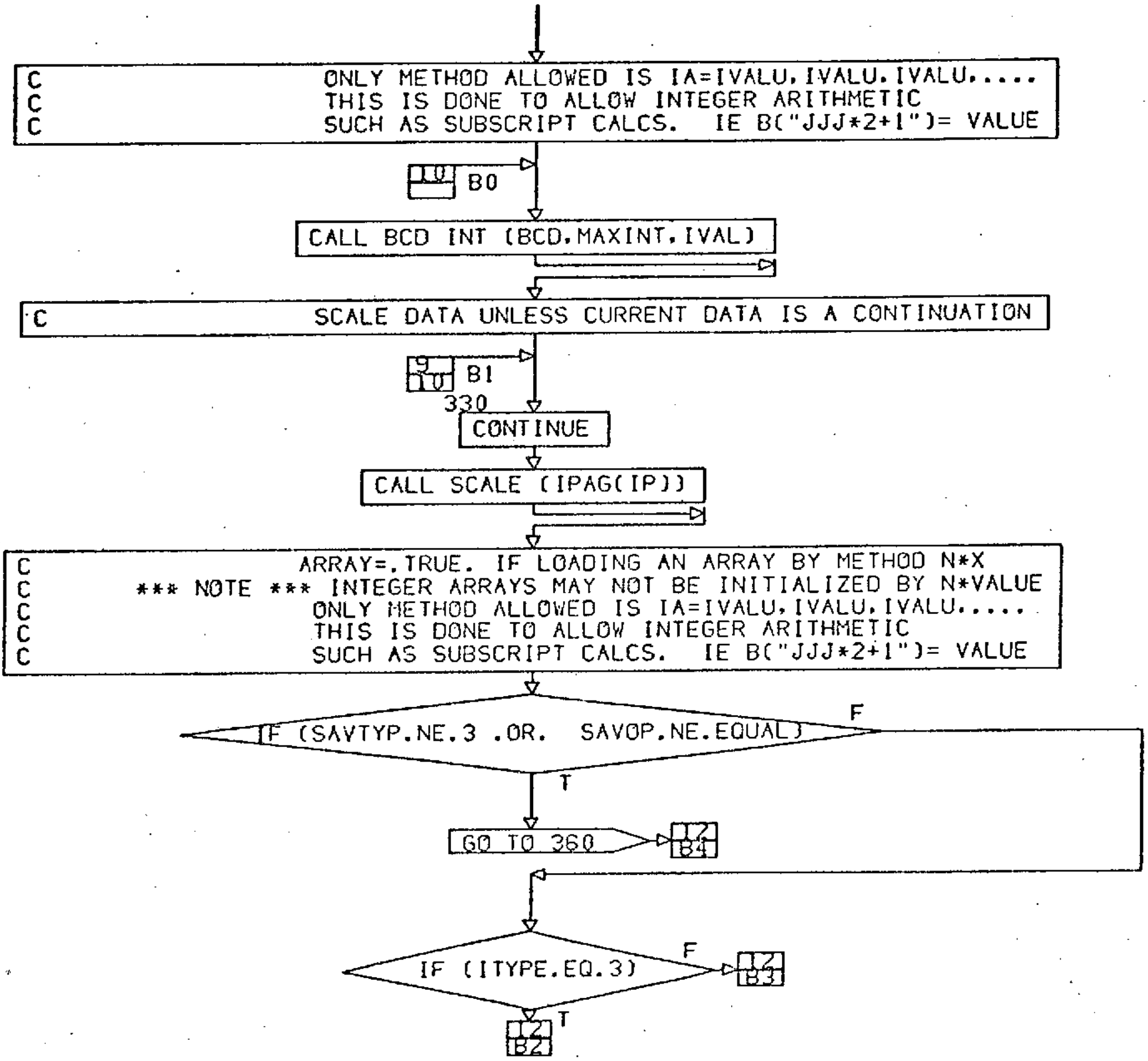
CONT. ON PG 9


8
8
A5
140
CONTINUE
IF (J.EQ.1)
F
T
GO TO 500
12
B5
LOC=IV(1)/10000+(KOUNT-1)*NWORD
IPAG(IP+2) = IDATA(ID+LOC)
IPAG(IP+3) = IDATA(ID+LOC+1)
IF (IOPR.EQ.BLANK)
F
T
GO TO 590
14
C0
GO TO 330
11
B1
C FLOATING POINT VALUE WAS INPUT, SCALE IF REQUIRED.
4
A6
200
CONTINUE
IF (IOPR.EQ.BLANK)
F
10
A8
T
10
A7

CONT. ON PG 10

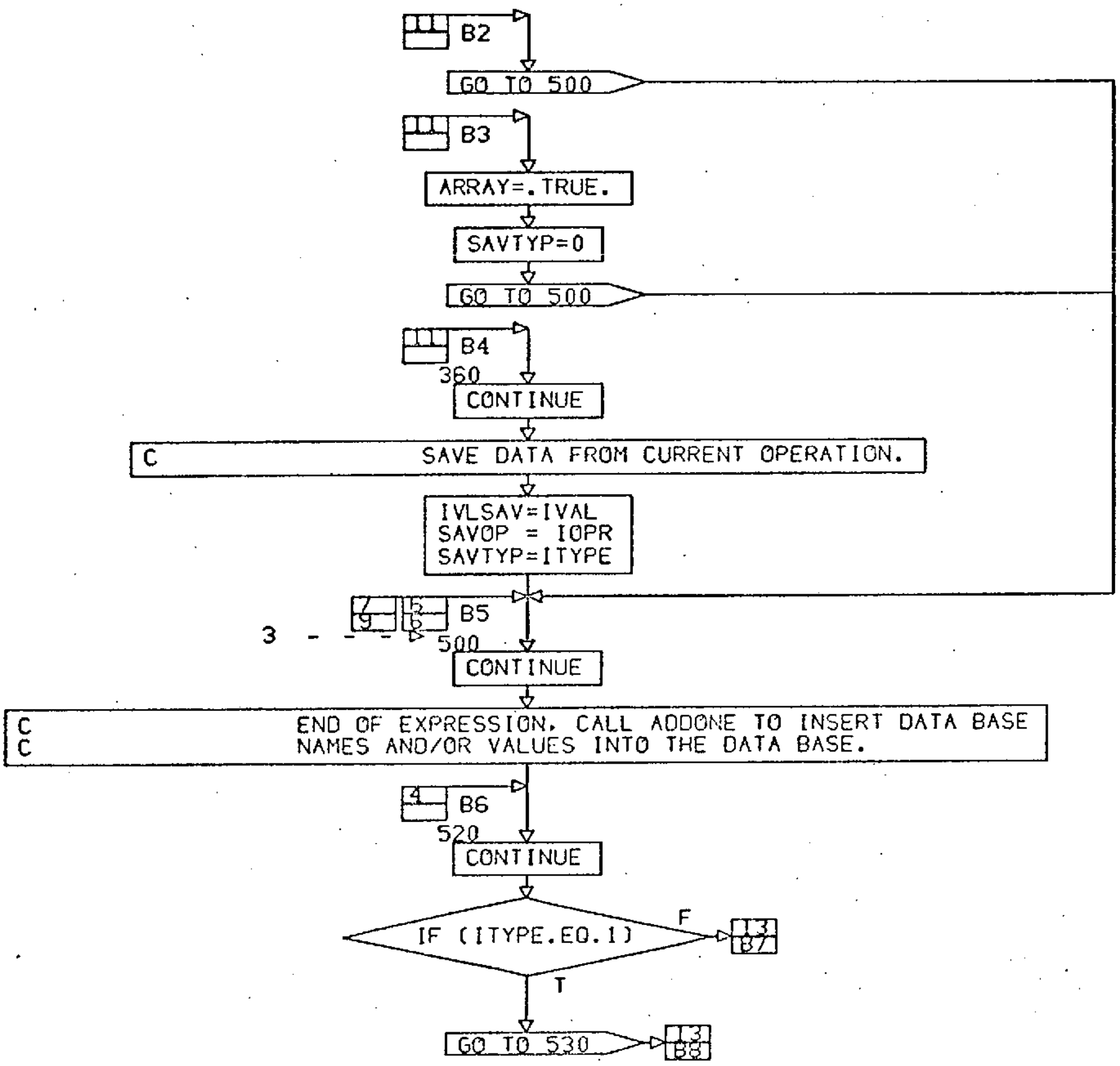

9
A7
GO TO 590
14
C0
9
A8
IF (IOPR.EQ.EQUAL)
F
T
GO TO 530
13
B8
IF (SAVTYP.EQ.3 .AND. SAVOP.EQ.EQUAL)
F
T
VALUE=1.0
GO TO 330
11
B1
C INTEGER VALUE WAS INPUT. SAVE VALUE IN IVAL.
4
A9
300
CONTINUE
IF (IOPR.EQ.BLANK)
F
11
B0
T
GO TO 590
14
C0
C *** NOTE *** INTEGER ARRAYS MAY NOT BE INITIALIZED BY N*VALUE

CONT. ON PG 11

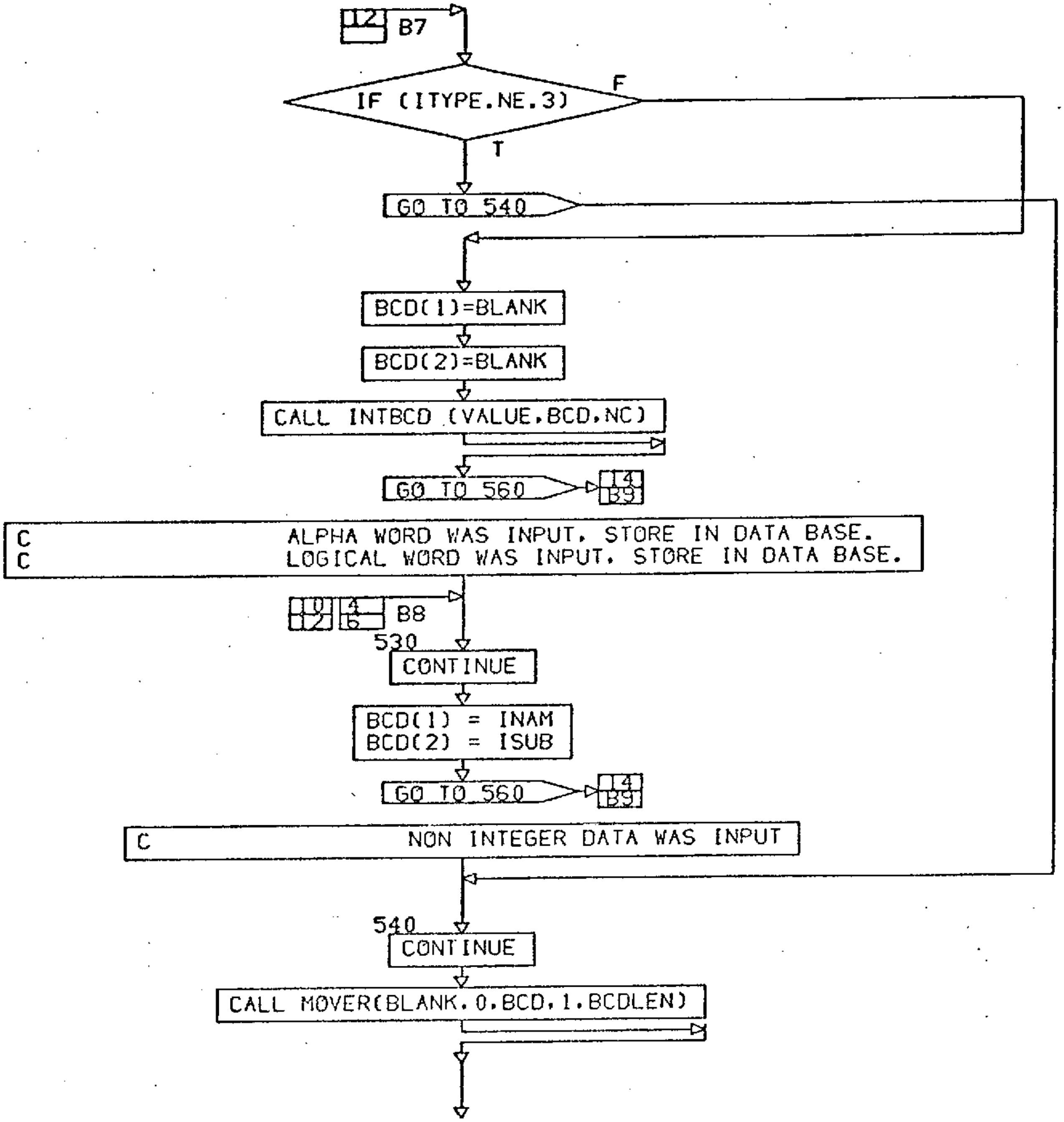


```
C                 ONLY METHOD ALLOWED IS IA=IVALU,IVALU,IVALU.....
C                 THIS IS DONE TO ALLOW INTEGER ARITHMETIC
C                 SUCH AS SUBSCRIPT CALCS.  IE B("JJJ*2+1")= VALUE

      CALL BCD INT (BCD,MAXINT,IVAL)

C                 SCALE DATA UNLESS CURRENT DATA IS A CONTINUATION

  330 CONTINUE

      CALL SCALE (IPAG(IP))

C                 ARRAY=.TRUE. IF LOADING AN ARRAY BY METHOD N*X
C       *** NOTE *** INTEGER ARRAYS MAY NOT BE INITIALIZED BY N*VALUE
C                 ONLY METHOD ALLOWED IS IA=IVALU,IVALU,IVALU.....
C                 THIS IS DONE TO ALLOW INTEGER ARITHMETIC
C                 SUCH AS SUBSCRIPT CALCS.  IE B("JJJ*2+1")= VALUE

      IF (SAVTYP.NE.3 .OR.  SAVOP.NE.EQUAL)

      GO TO 360

      IF (ITYPE.EQ.3)
```

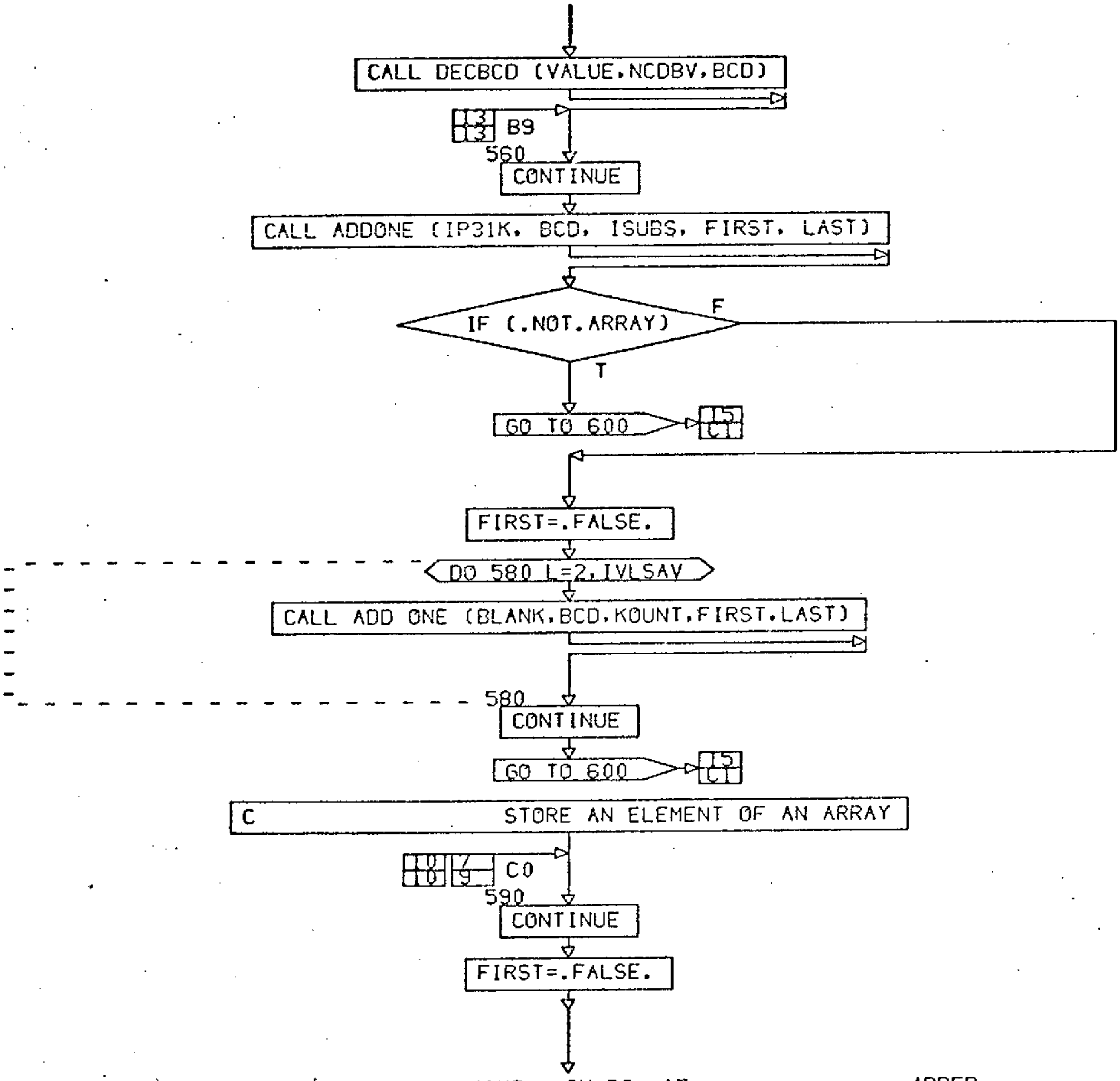
CONT. ON PG 12


B2
GO TO 500
B3
ARRAY=.TRUE.
SAVTYP=0
GO TO 500
B4
360
CONTINUE
C SAVE DATA FROM CURRENT OPERATION.
IVLSAV=IVAL
SAVOP = IOPR
SAVTYP=ITYPE
7
9
5
6
B5
3
500
CONTINUE
C END OF EXPRESSION. CALL ADDONE TO INSERT DATA BASE
C NAMES AND/OR VALUES INTO THE DATA BASE.
4
B6
520
CONTINUE
IF (ITYPE.EQ.1)
F
13
B7
T
GO TO 530
13
B8

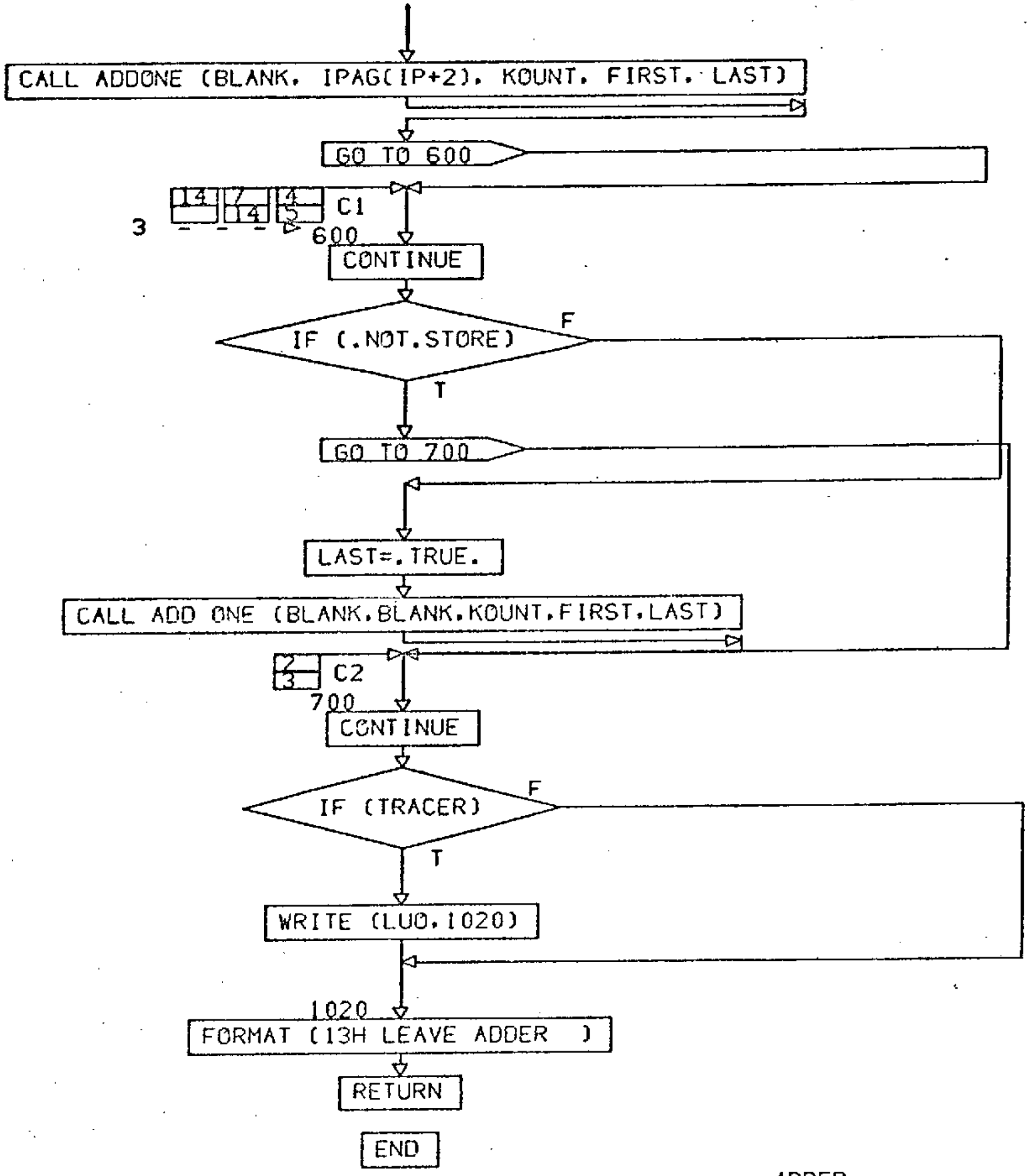
CONT. ON PG 13


12
B7
IF (ITYPE.NE.3)
F
T
GO TO 540
BCD(1)=BLANK
BCD(2)=BLANK
CALL INTBCD (VALUE,BCD,NC)
GO TO 560
14
B9
C ALPHA WORD WAS INPUT, STORE IN DATA BASE.
C LOGICAL WORD WAS INPUT, STORE IN DATA BASE.
10
12
4
6
B8
530
CONTINUE
BCD(1) = INAM
BCD(2) = ISUB
GO TO 560
14
B9
C NON INTEGER DATA WAS INPUT
540
CONTINUE
CALL MOVER(BLANK,0,BCD,1,BCDLEN)

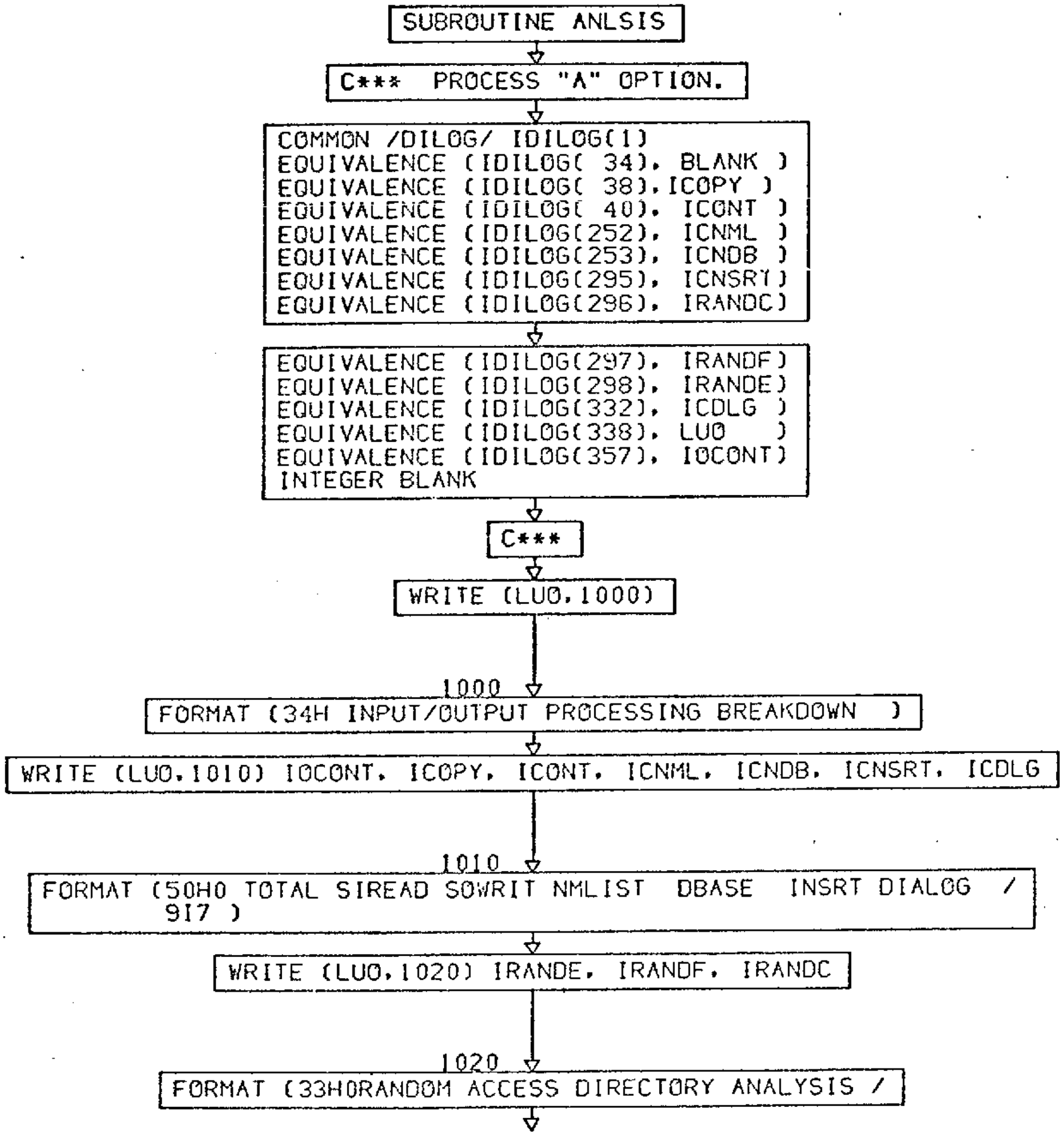
CONT. ON PG 14


CALL DECBCD (VALUE,NCDBV,BCD)
13
13
B9
560
CONTINUE
CALL ADDONE (IP31K, BCD, ISUBS, FIRST, LAST)
IF (.NOT.ARRAY)
F
T
GO TO 600
15
C1
FIRST=.FALSE.
DO 580 L=2,IVLSAV
CALL ADD ONE (BLANK,BCD,KOUNT,FIRST,LAST)
580
CONTINUE
GO TO 600
15
C1
C STORE AN ELEMENT OF AN ARRAY
10
10
7
9
C0
590
CONTINUE
FIRST=.FALSE.

CONT. ON PG 15

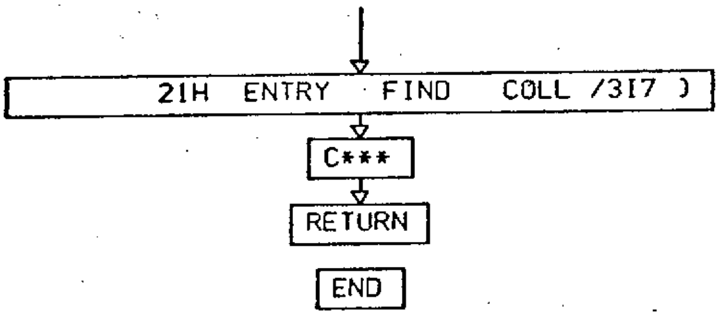


ADDER
PG 15 FINAL

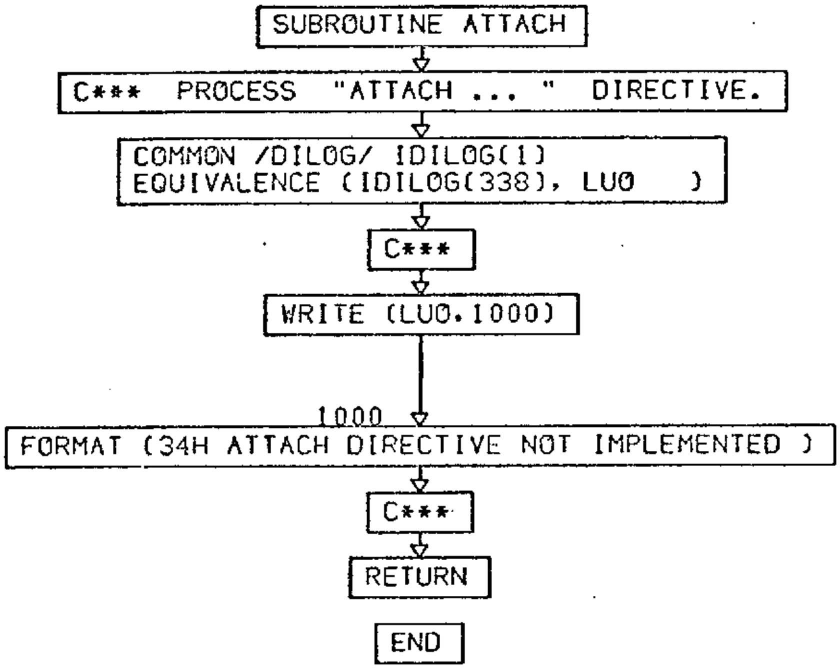


```
SUBROUTINE ANLSIS

C***  PROCESS "A" OPTION.

COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 38),ICOPY )
EQUIVALENCE (IDILOG( 40), ICONT )
EQUIVALENCE (IDILOG(252), ICNML )
EQUIVALENCE (IDILOG(253), ICNDB )
EQUIVALENCE (IDILOG(295), ICNSRT)
EQUIVALENCE (IDILOG(296), IRANDC)

EQUIVALENCE (IDILOG(297), IRANDF)
EQUIVALENCE (IDILOG(298), IRANDE)
EQUIVALENCE (IDILOG(332), ICDLG )
EQUIVALENCE (IDILOG(338), LUO    )
EQUIVALENCE (IDILOG(357), IOCONT)
INTEGER BLANK

C***

WRITE (LUO,1000)

1000
FORMAT (34H INPUT/OUTPUT PROCESSING BREAKDOWN  )

WRITE (LUO,1010) IOCONT, ICOPY, ICONT, ICNML, ICNDB, ICNSRT, ICDLG

1010
FORMAT (50H0 TOTAL SIREAD SOWRIT NMLIST  DBASE  INSRT DIALOG  /
       9I7 )

WRITE (LUO,1020) IRANDE, IRANDF, IRANDC

1020
FORMAT (33H0RANDOM ACCESS DIRECTORY ANALYSIS /
```

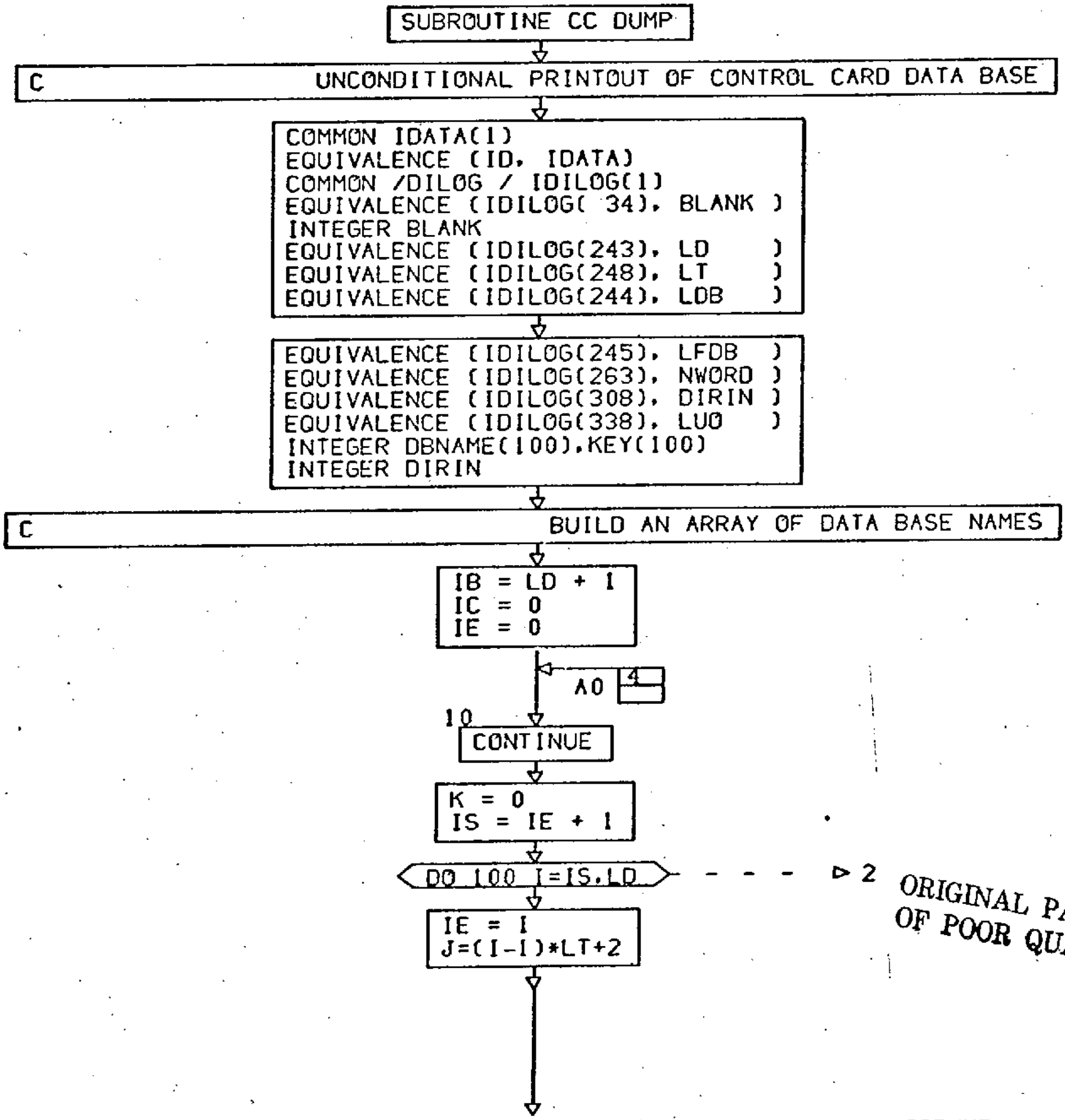
CONT. ON PG 2



ANLSIS
PG 2 FINAL

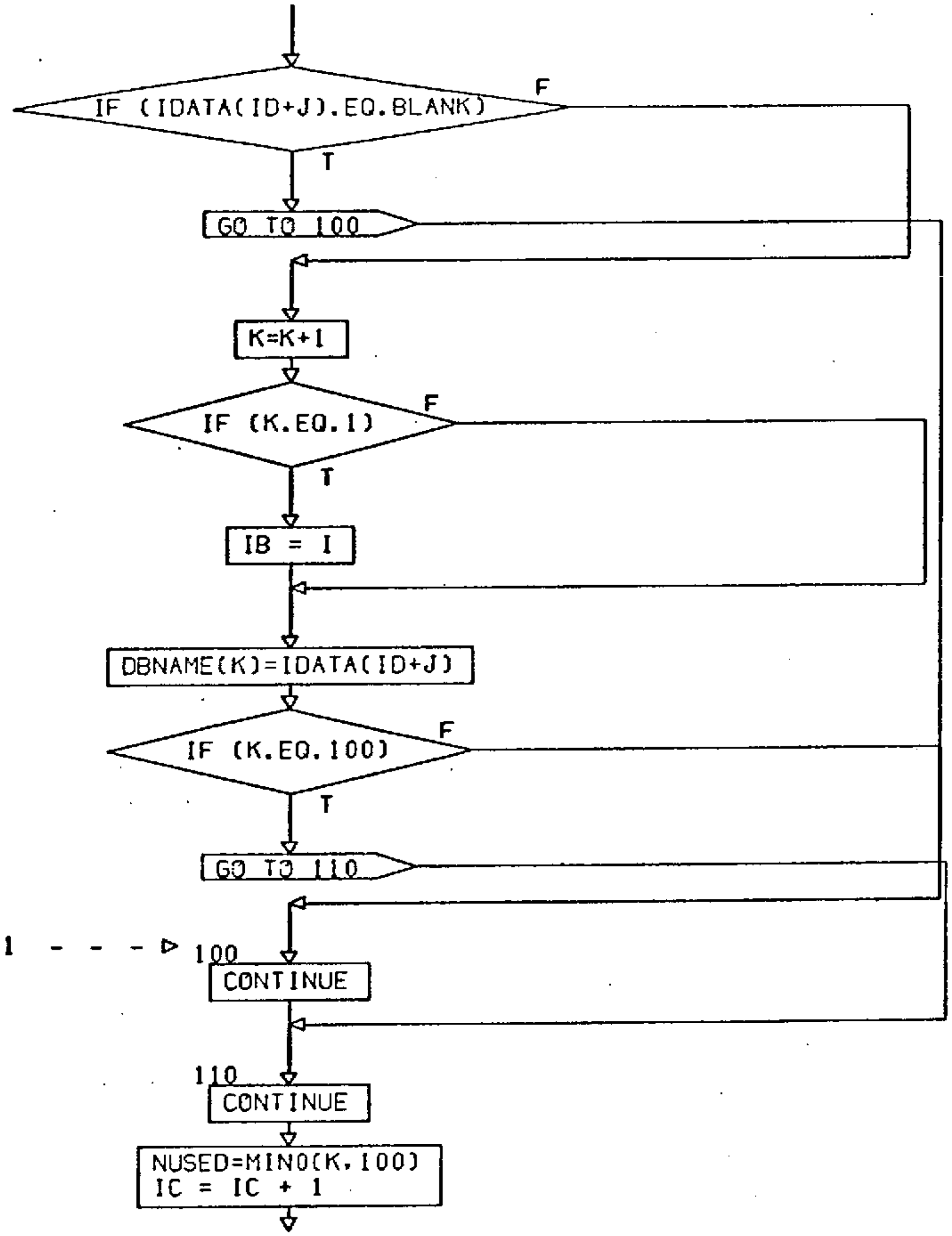


```
SUBROUTINE ATTACH
C***  PROCESS  "ATTACH ... "  DIRECTIVE.
COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG(338), LUO    )
C***
WRITE (LUO,1000)
1000
FORMAT (34H ATTACH DIRECTIVE NOT IMPLEMENTED )
C***
RETURN
END
```

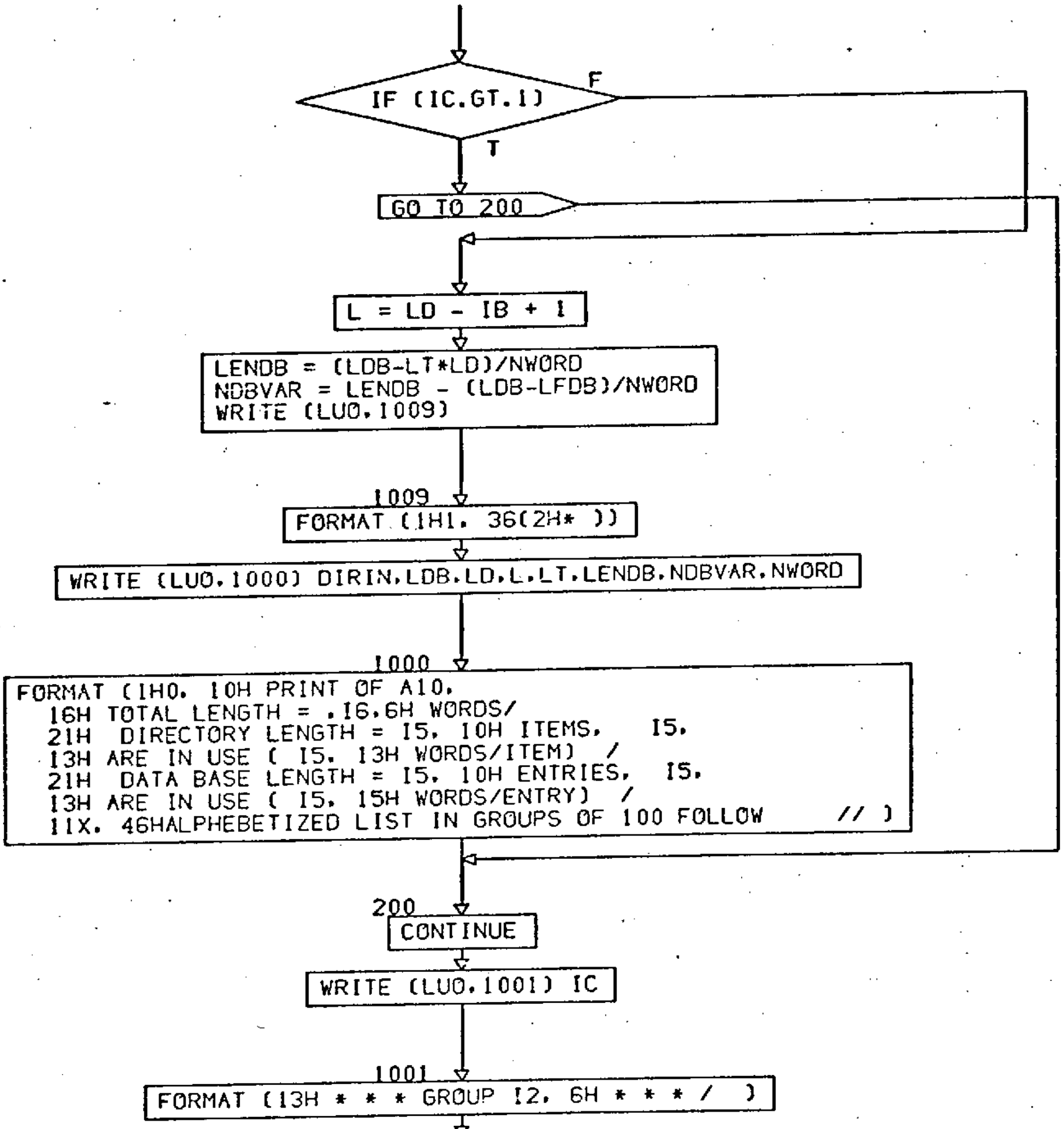


```
      SUBROUTINE CC DUMP
C                     UNCONDITIONAL PRINTOUT OF CONTROL CARD DATA BASE
      COMMON IDATA(1)
      EQUIVALENCE (ID, IDATA)
      COMMON /DILOG / IDILOG(1)
      EQUIVALENCE (IDILOG( 34), BLANK )
      INTEGER BLANK
      EQUIVALENCE (IDILOG(243), LD     )
      EQUIVALENCE (IDILOG(248), LT     )
      EQUIVALENCE (IDILOG(244), LDB    )
      EQUIVALENCE (IDILOG(245), LFDB   )
      EQUIVALENCE (IDILOG(263), NWORD  )
      EQUIVALENCE (IDILOG(308), DIRIN  )
      EQUIVALENCE (IDILOG(338), LUO    )
      INTEGER DBNAME(100),KEY(100)
      INTEGER DIRIN
C                                      BUILD AN ARRAY OF DATA BASE NAMES
      IB = LD + 1
      IC = 0
      IE = 0
   10 CONTINUE
      K = 0
      IS = IE + 1
      DO 100 I=IS,LD
      IE = I
      J=(I-1)*LT+2
```

CONT. ON PG 2

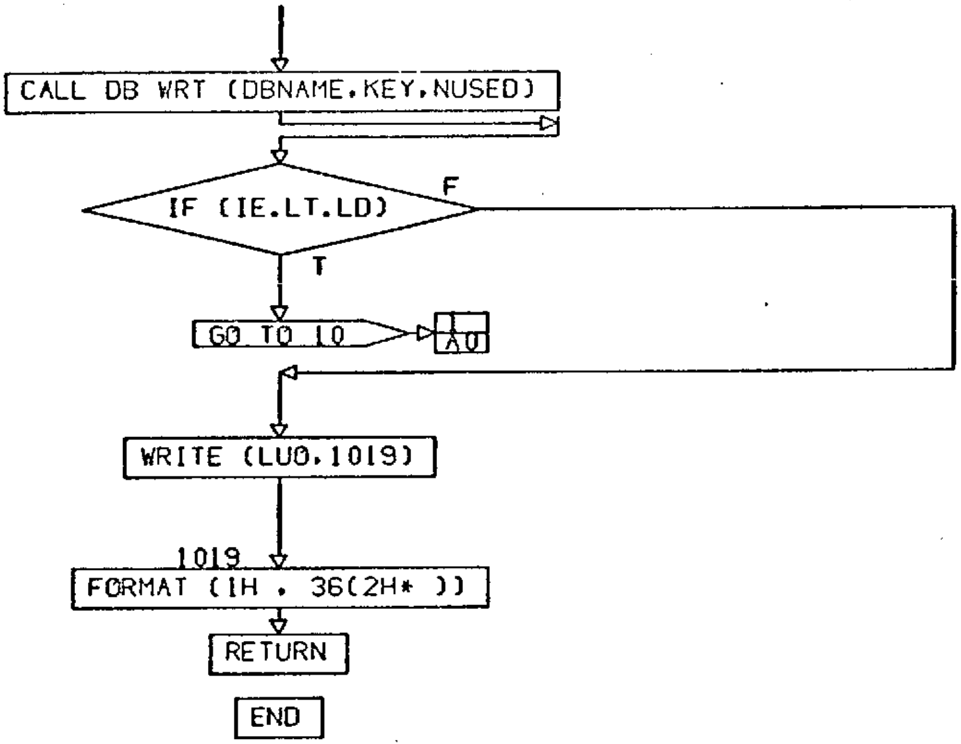

IF (IDATA(ID+J).EQ.BLANK)
F
T
GO TO 100
K=K+1
IF (K.EQ.1)
F
T
IB = I
DBNAME(K)=IDATA(ID+J)
IF (K.EQ.100)
F
T
GO TO 110
1
100
CONTINUE
110
CONTINUE
NUSED=MIN0(K,100)
IC = IC + 1

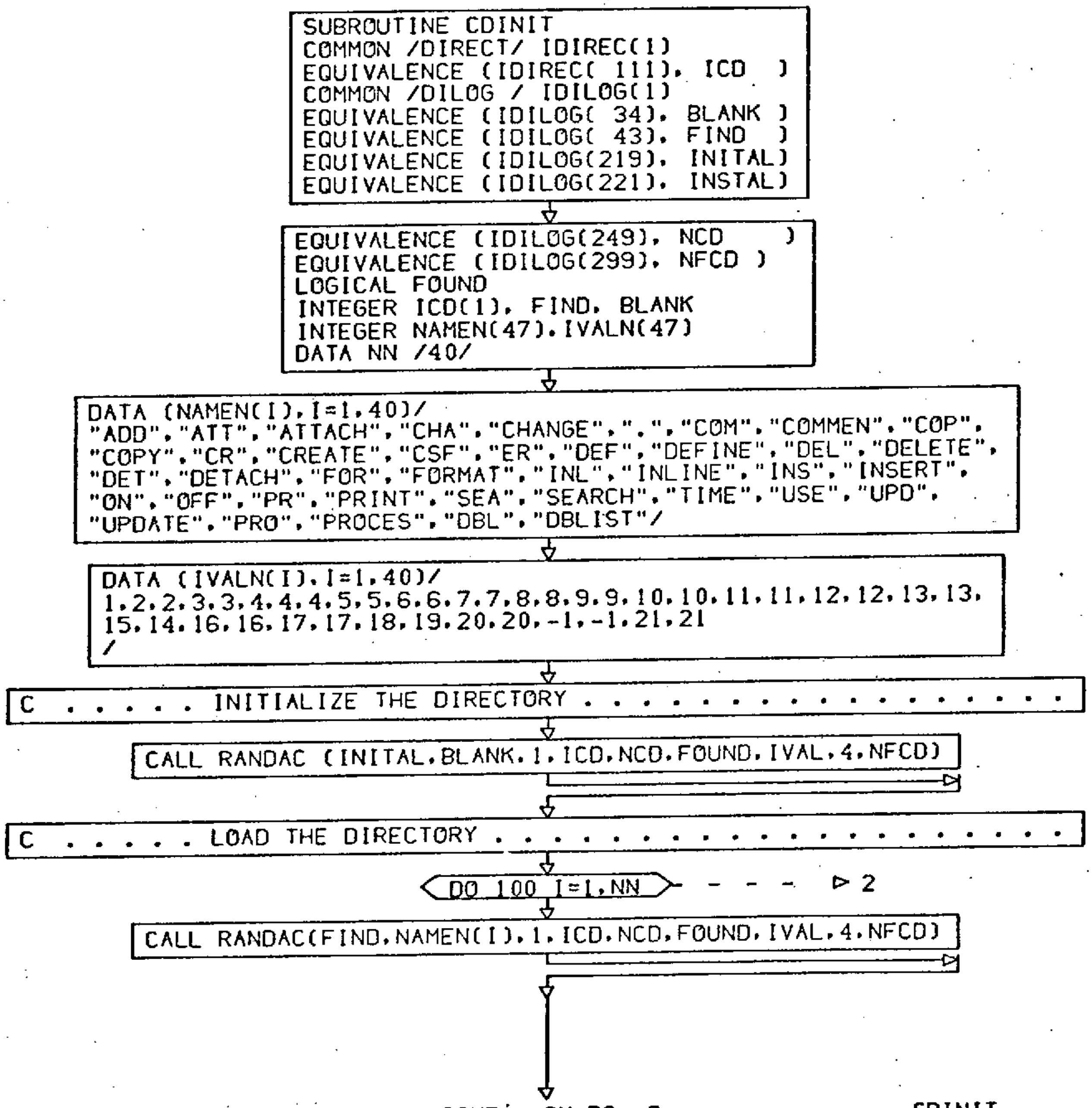
CONT. ON PG 3



```
IF (IC.GT.1)
GO TO 200
L = LD - IB + 1
LENDB = (LDB-LT*LD)/NWORD
NDBVAR = LENDB - (LDB-LFDB)/NWORD
WRITE (LUO,1009)
1009
FORMAT (1H1, 36(2H* ))
WRITE (LUO,1000) DIRIN,LDB,LD,L,LT,LENDB,NDBVAR,NWORD
1000
FORMAT (1H0, 10H PRINT OF A10,
  16H TOTAL LENGTH = ,I6,6H WORDS/
  21H  DIRECTORY LENGTH = I5, 10H ITEMS,   I5,
  13H ARE IN USE ( I5, 13H WORDS/ITEM)  /
  21H  DATA BASE LENGTH = I5, 10H ENTRIES,  I5,
  13H ARE IN USE ( I5, 15H WORDS/ENTRY)  /
  11X, 46HALPHEBETIZED LIST IN GROUPS OF 100 FOLLOW      // )
200
CONTINUE
WRITE (LUO,1001) IC
1001
FORMAT (13H * * * GROUP I2, 6H * * * /  )
```

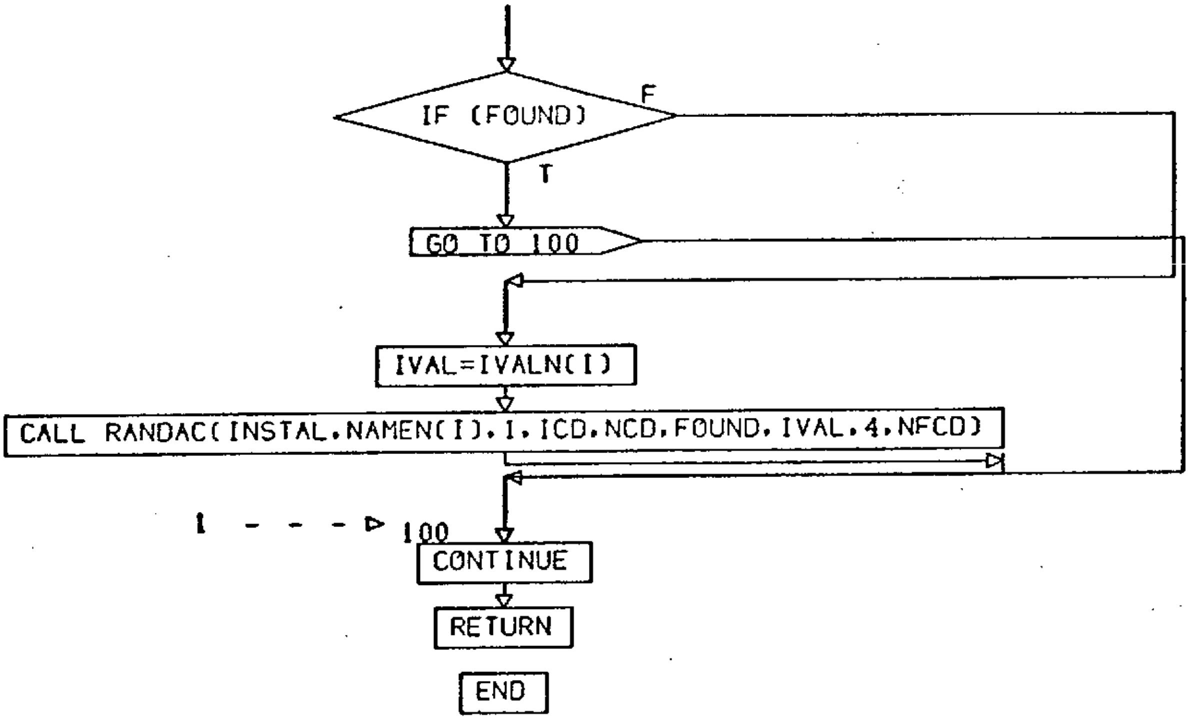
CONT. ON PG 4


CALL DB WRT (DBNAME,KEY,NUSED)
IF (IE.LT.LD)
F
T
GO TO 10
I
A0
WRITE (LU0,1019)
1019
FORMAT (1H , 36(2H*))
RETURN
END

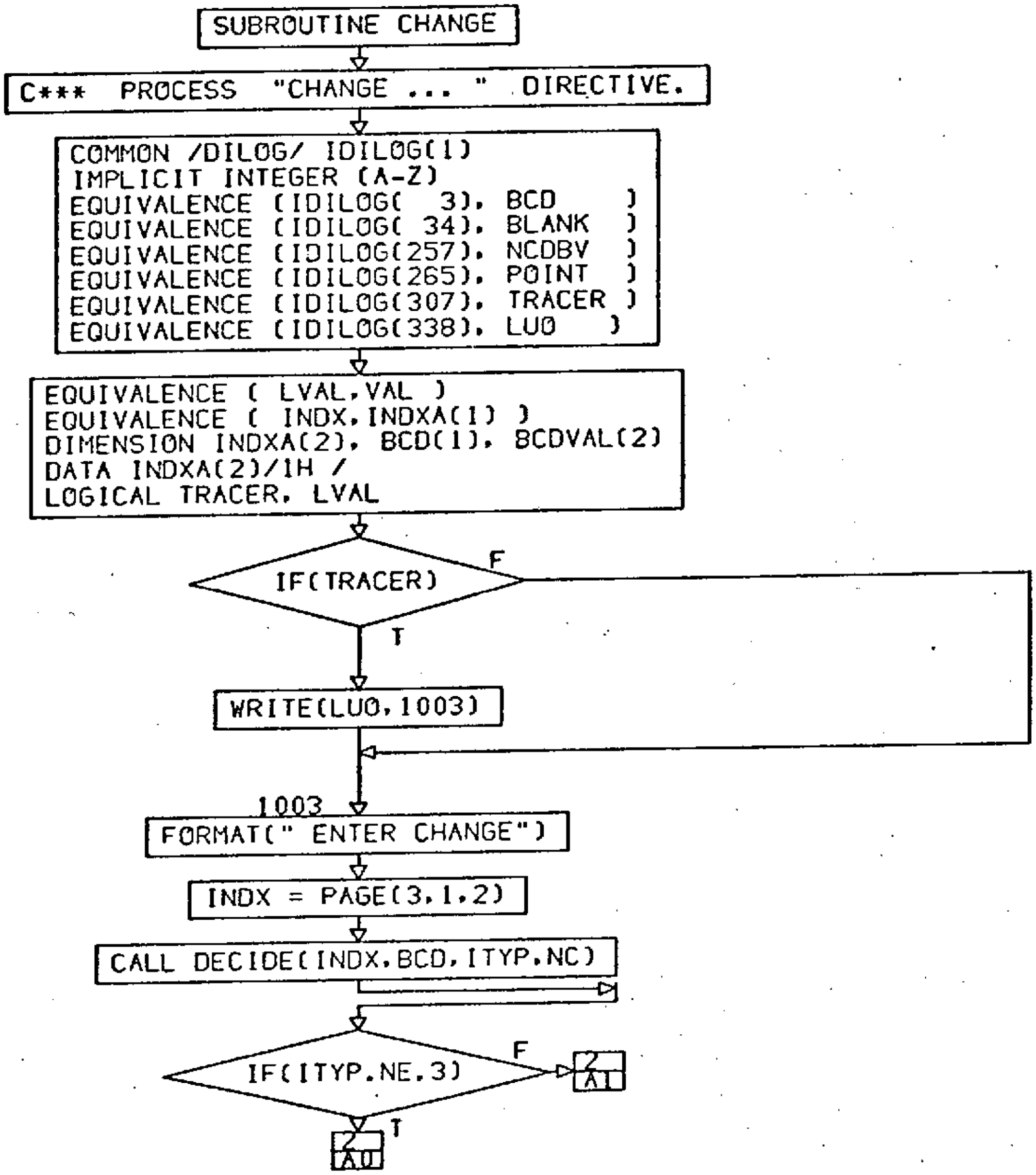


```
SUBROUTINE CDINIT
COMMON /DIRECT/ IDIREC(1)
EQUIVALENCE (IDIREC( 111), ICD  )
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 43), FIND  )
EQUIVALENCE (IDILOG(219), INITAL)
EQUIVALENCE (IDILOG(221), INSTAL)

EQUIVALENCE (IDILOG(249), NCD    )
EQUIVALENCE (IDILOG(299), NFCD )
LOGICAL FOUND
INTEGER ICD(1), FIND, BLANK
INTEGER NAMEN(47),IVALN(47)
DATA NN /40/

DATA (NAMEN(I),I=1,40)/
"ADD","ATT","ATTACH","CHA","CHANGE",".","COM","COMMEN","COP",
"COPY","CR","CREATE","CSF","ER","DEF","DEFINE","DEL","DELETE",
"DET","DETACH","FOR","FORMAT","INL","INLINE","INS","INSERT",
"ON","OFF","PR","PRINT","SEA","SEARCH","TIME","USE","UPD",
"UPDATE","PRO","PROCES","DBL","DBLIST"/

DATA (IVALN(I),I=1,40)/
1,2,2,3,3,4,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13,
15,14,16,16,17,17,18,19,20,20,-1,-1,21,21
/

C . . . . . INITIALIZE THE DIRECTORY . . . . . . . . . . . . . . . . .

CALL RANDAC (INITAL,BLANK,1,ICD,NCD,FOUND,IVAL,4,NFCD)

C . . . . . LOAD THE DIRECTORY . . . . . . . . . . . . . . . . . . . .

DO 100 I=1,NN  - - - -  ⊳ 2

CALL RANDAC(FIND,NAMEN(I),1,ICD,NCD,FOUND,IVAL,4,NFCD)
```

CONT. ON PG 2

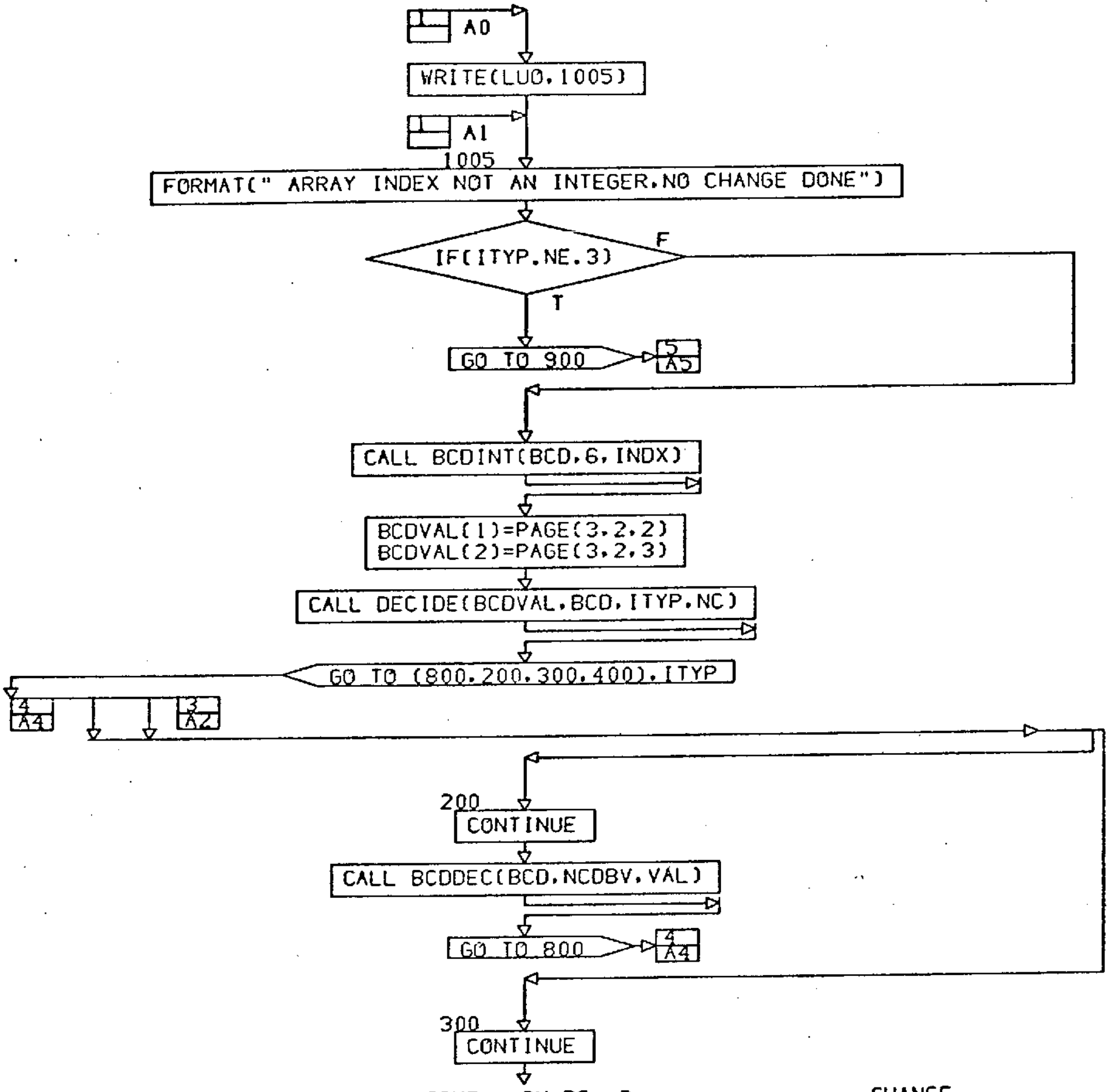


CDINIT
PG 2 FINAL

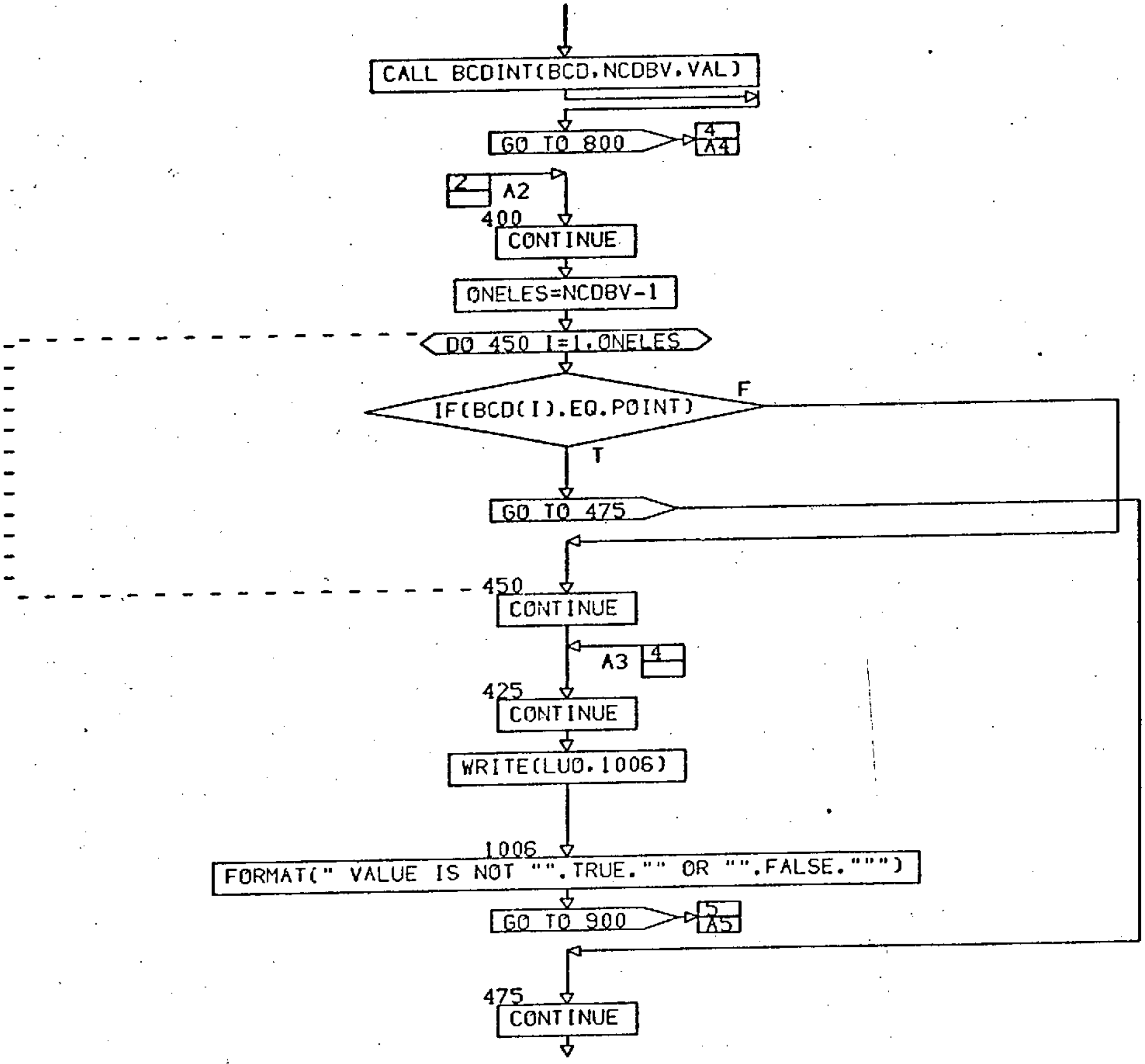


```
SUBROUTINE CHANGE
C***  PROCESS  "CHANGE ... "  DIRECTIVE.
COMMON /DILOG/ IDILOG(1)
IMPLICIT INTEGER (A-Z)
EQUIVALENCE (IDILOG(  3), BCD     )
EQUIVALENCE (IDILOG( 34), BLANK   )
EQUIVALENCE (IDILOG(257), NCDBV   )
EQUIVALENCE (IDILOG(265), POINT   )
EQUIVALENCE (IDILOG(307), TRACER  )
EQUIVALENCE (IDILOG(338), LUO    )
EQUIVALENCE ( LVAL,VAL )
EQUIVALENCE (  INDX,INDXA(1) )
DIMENSION INDXA(2), BCD(1), BCDVAL(2)
DATA INDXA(2)/1H /
LOGICAL TRACER, LVAL
IF(TRACER)
WRITE(LUO,1003)
1003
FORMAT(" ENTER CHANGE")
INDX = PAGE(3,1,2)
CALL DECIDE(INDX,BCD,ITYP,NC)
IF(ITYP.NE.3)
```

CONT. ON PG 2

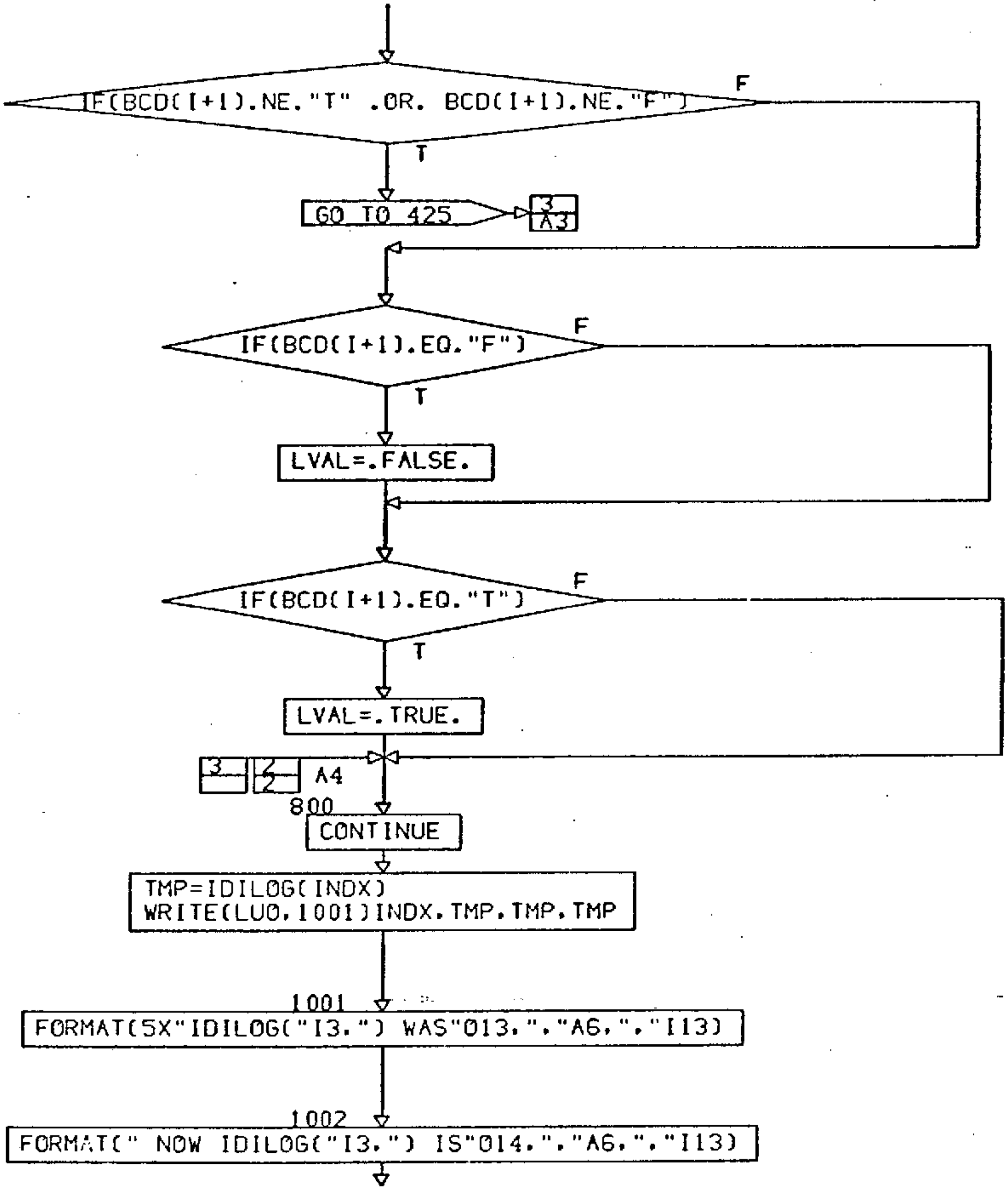

1
A0
WRITE(LUO,1005)
1
A1
1005
FORMAT(" ARRAY INDEX NOT AN INTEGER,NO CHANGE DONE")
IF(ITYP.NE.3)
F
T
GO TO 900
5
A5
CALL BCDINT(BCD,6,INDX)
BCDVAL(1)=PAGE(3,2,2)
BCDVAL(2)=PAGE(3,2,3)
CALL DECIDE(BCDVAL,BCD,ITYP,NC)
GO TO (800,200,300,400),ITYP
4
A4
3
A2
200
CONTINUE
CALL BCDDEC(BCD,NCDBV,VAL)
GO TO 800
4
A4
300
CONTINUE

CONT. ON PG 3



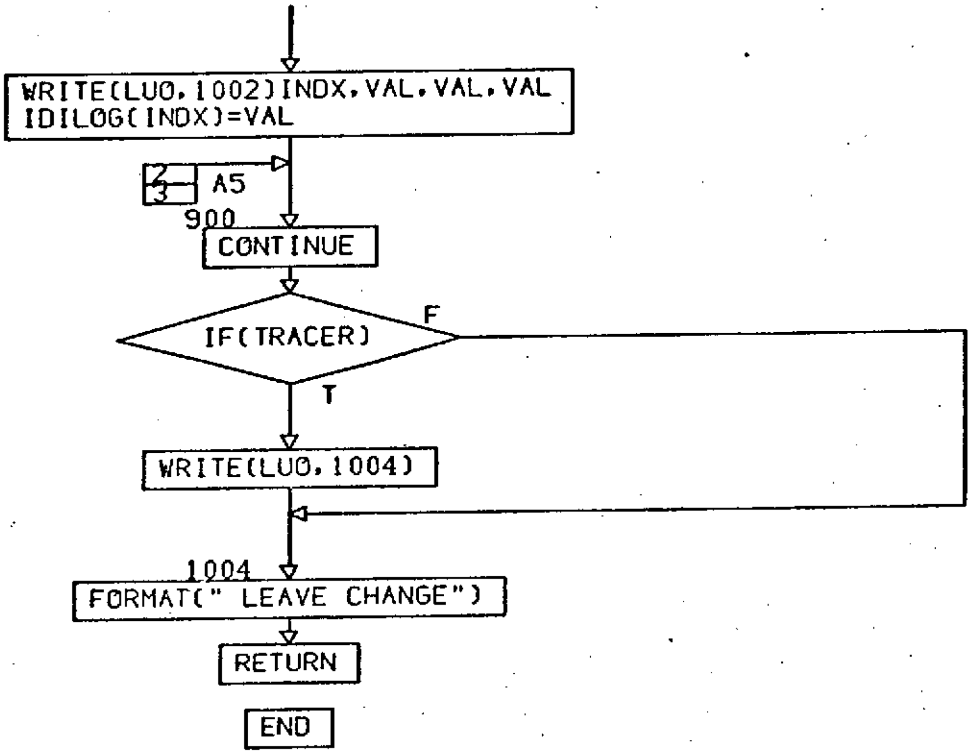
CONT. ON PG 4

CHANGE
PG 3 OF 5

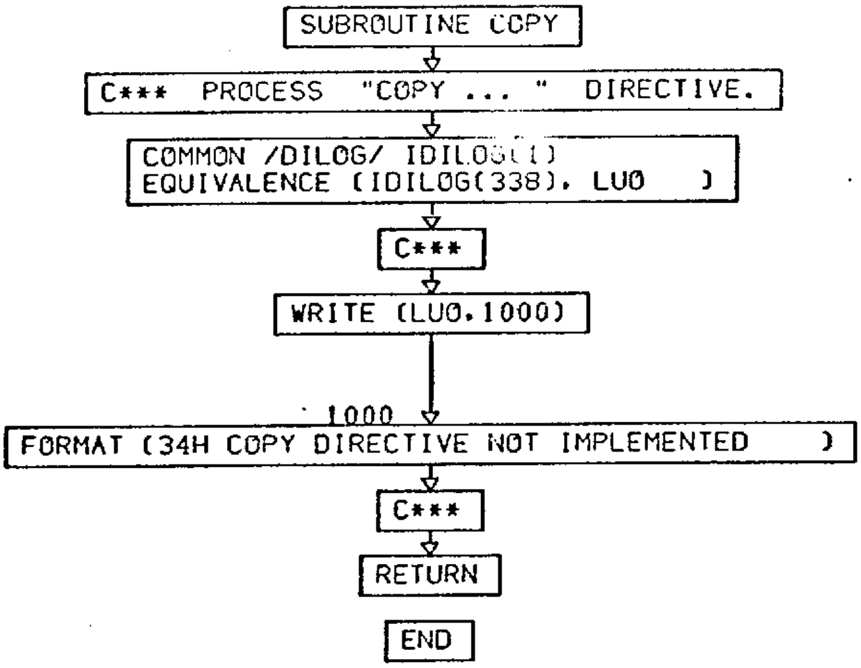

IF(BCD(I+1).NE."T" .OR. BCD(I+1).NE."F")
F
T
GO TO 425
3
A3
IF(BCD(I+1).EQ."F")
F
T
LVAL=.FALSE.
IF(BCD(I+1).EQ."T")
F
T
LVAL=.TRUE.
3
2
2
A4
800
CONTINUE
TMP=IDILOG(INDX)
WRITE(LUO,1001)INDX,TMP,TMP,TMP
1001
FORMAT(5X"IDILOG("I3,") WAS"O13,","A6,","I13)
1002
FORMAT(" NOW IDILOG("I3,") IS"O14,","A6,","I13)

CONT. ON PG 5

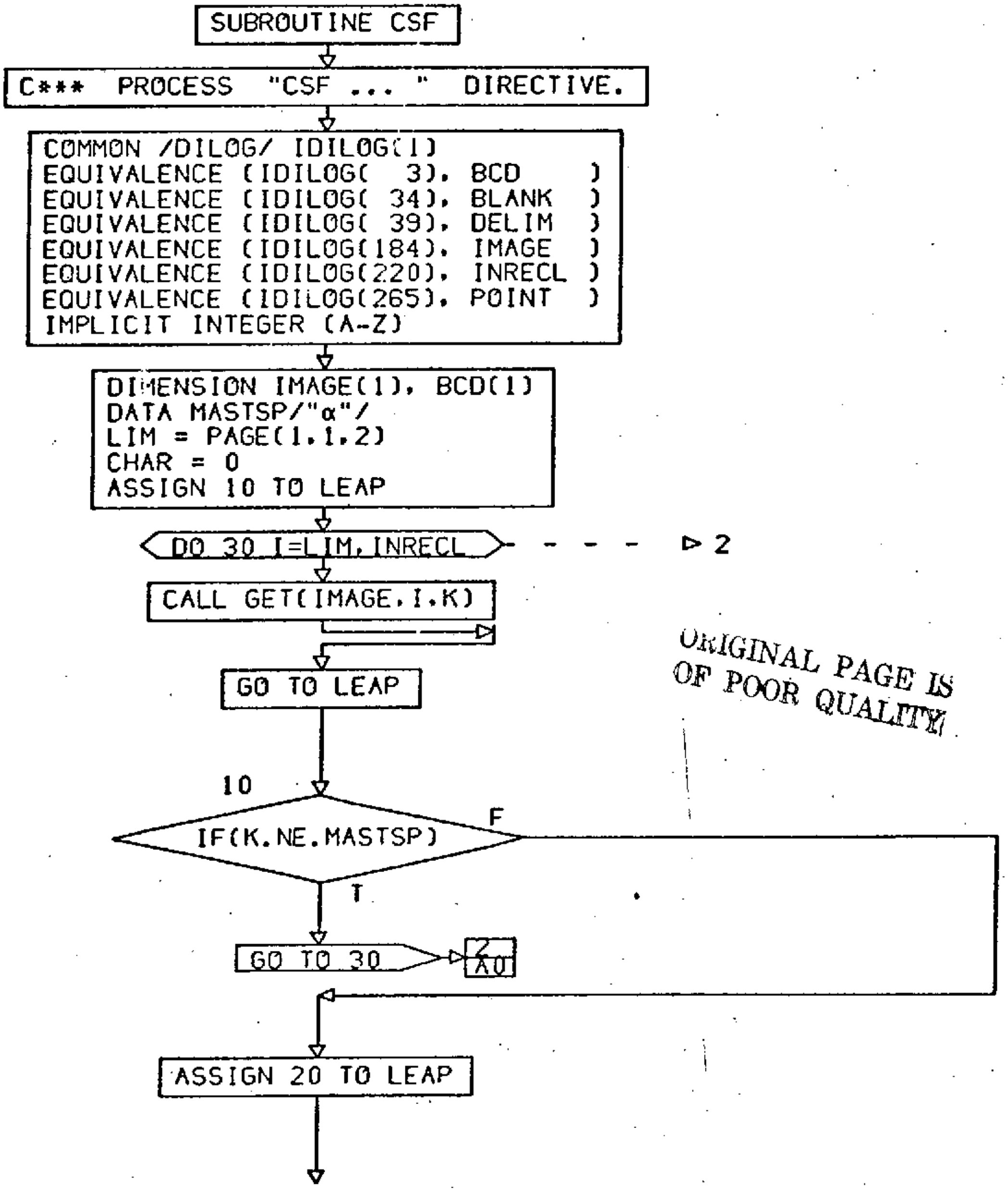
CHANGE
PG 4 OF 5



CHANGE
PG 5 FINAL

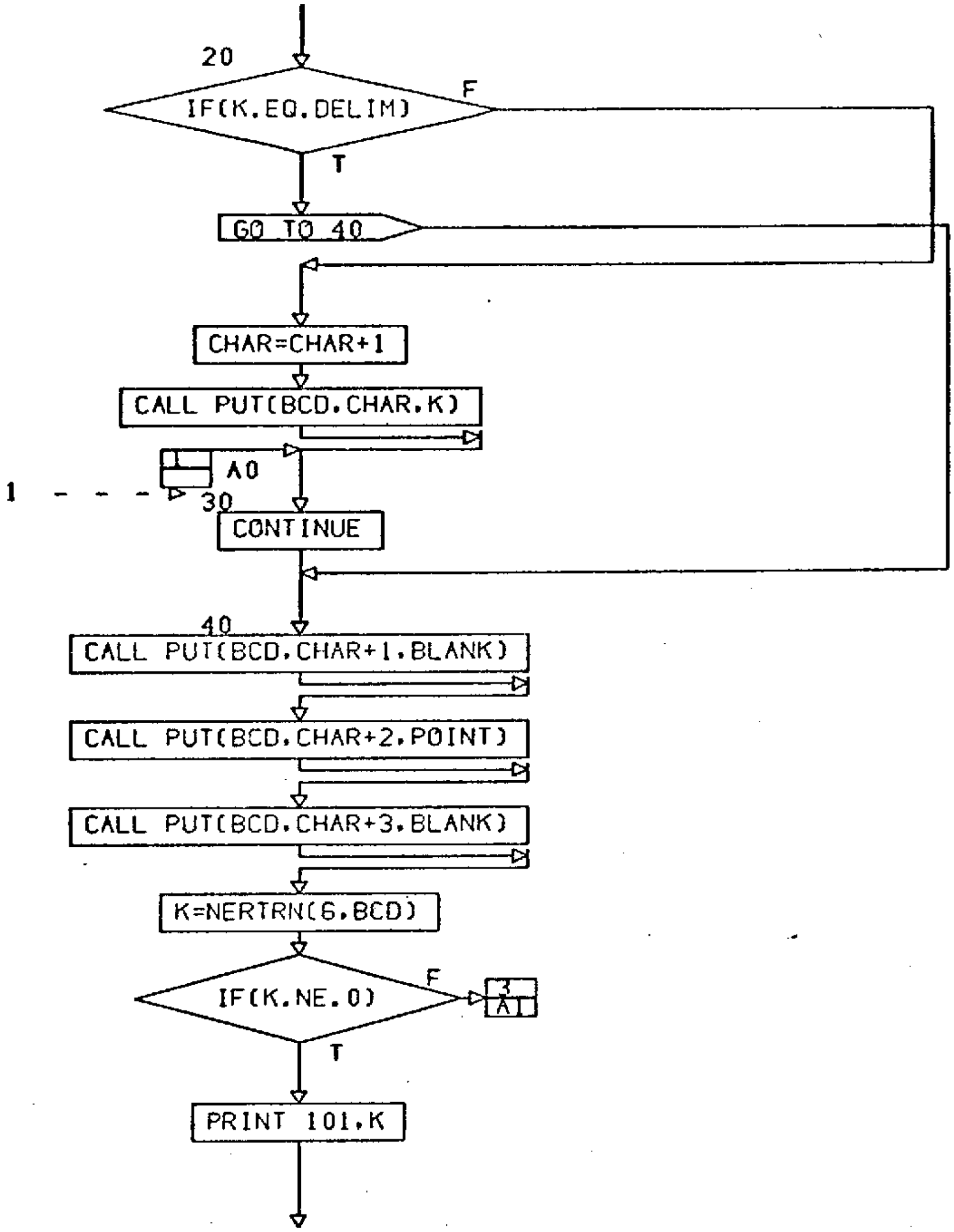


```
SUBROUTINE COPY
C***  PROCESS  "COPY ... "  DIRECTIVE.
COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG(338), LUO    )
C***
WRITE (LUO,1000)
1000
FORMAT (34H COPY DIRECTIVE NOT IMPLEMENTED      )
C***
RETURN
END
```

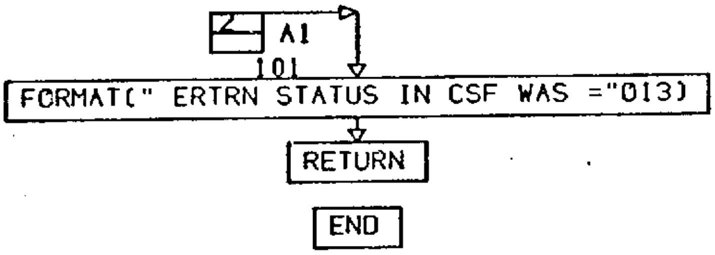


```
SUBROUTINE CSF

C***  PROCESS  "CSF ... "  DIRECTIVE.

COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG(  3), BCD     )
EQUIVALENCE (IDILOG( 34), BLANK   )
EQUIVALENCE (IDILOG( 39), DELIM   )
EQUIVALENCE (IDILOG(184), IMAGE   )
EQUIVALENCE (IDILOG(220), INRECL  )
EQUIVALENCE (IDILOG(265), POINT   )
IMPLICIT INTEGER (A-Z)

DIMENSION IMAGE(1), BCD(1)
DATA MASTSP/"α"/
LIM = PAGE(1,1,2)
CHAR = 0
ASSIGN 10 TO LEAP

DO 30 I=LIM,INRECL

CALL GET(IMAGE,I,K)

GO TO LEAP

10
IF(K.NE.MASTSP)

GO TO 30

ASSIGN 20 TO LEAP
```

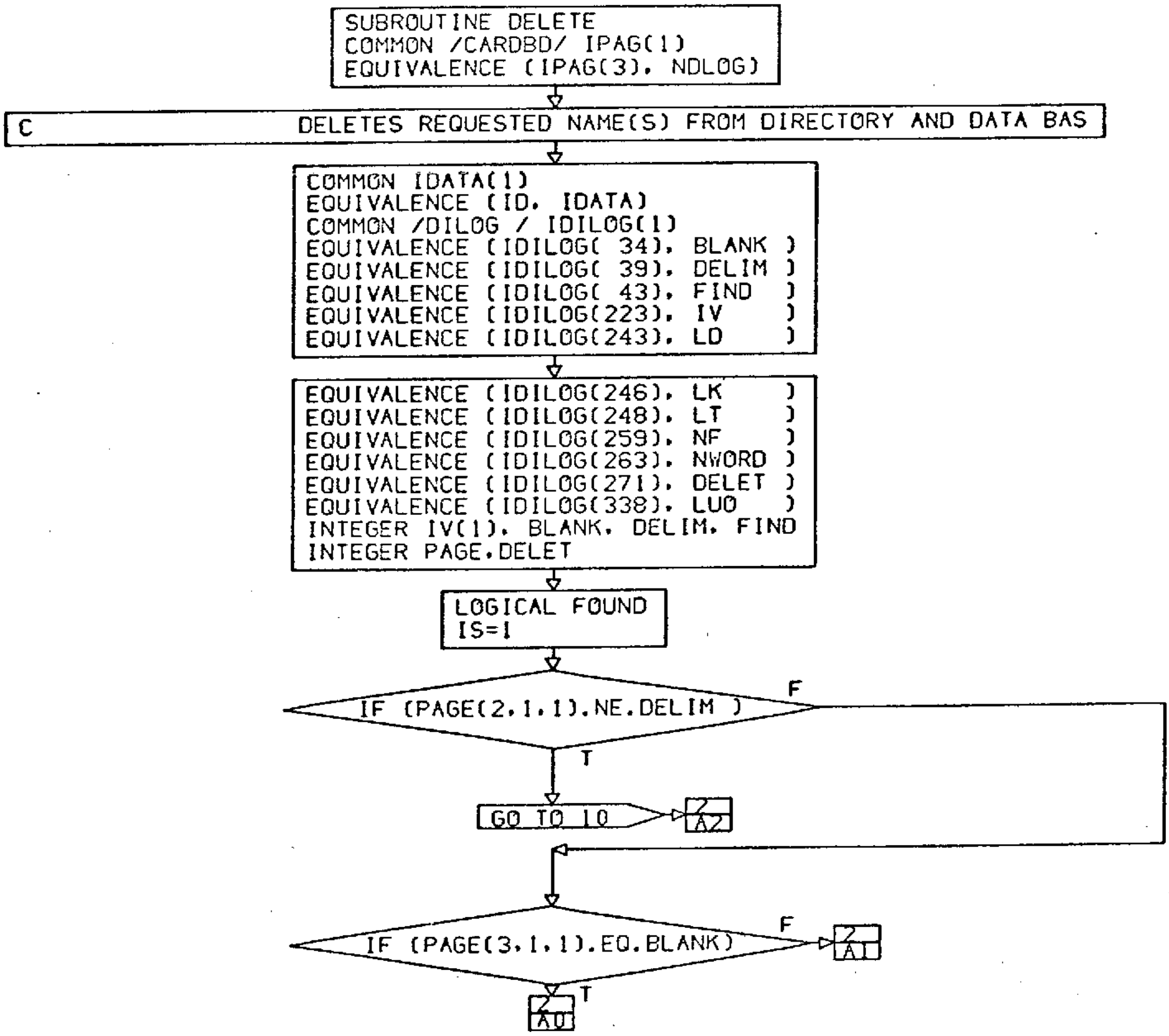
CONT. ON PG 2


20
IF(K.EQ.DELIM)
F
T
GO TO 40
CHAR=CHAR+1
CALL PUT(BCD.CHAR.K)
1
A0
1
30
CONTINUE
40
CALL PUT(BCD.CHAR+1.BLANK)
CALL PUT(BCD.CHAR+2.POINT)
CALL PUT(BCD.CHAR+3.BLANK)
K=NERTRN(6.BCD)
IF(K.NE.0)
F
3
A1
T
PRINT 101.K

CONT. ON PG 3

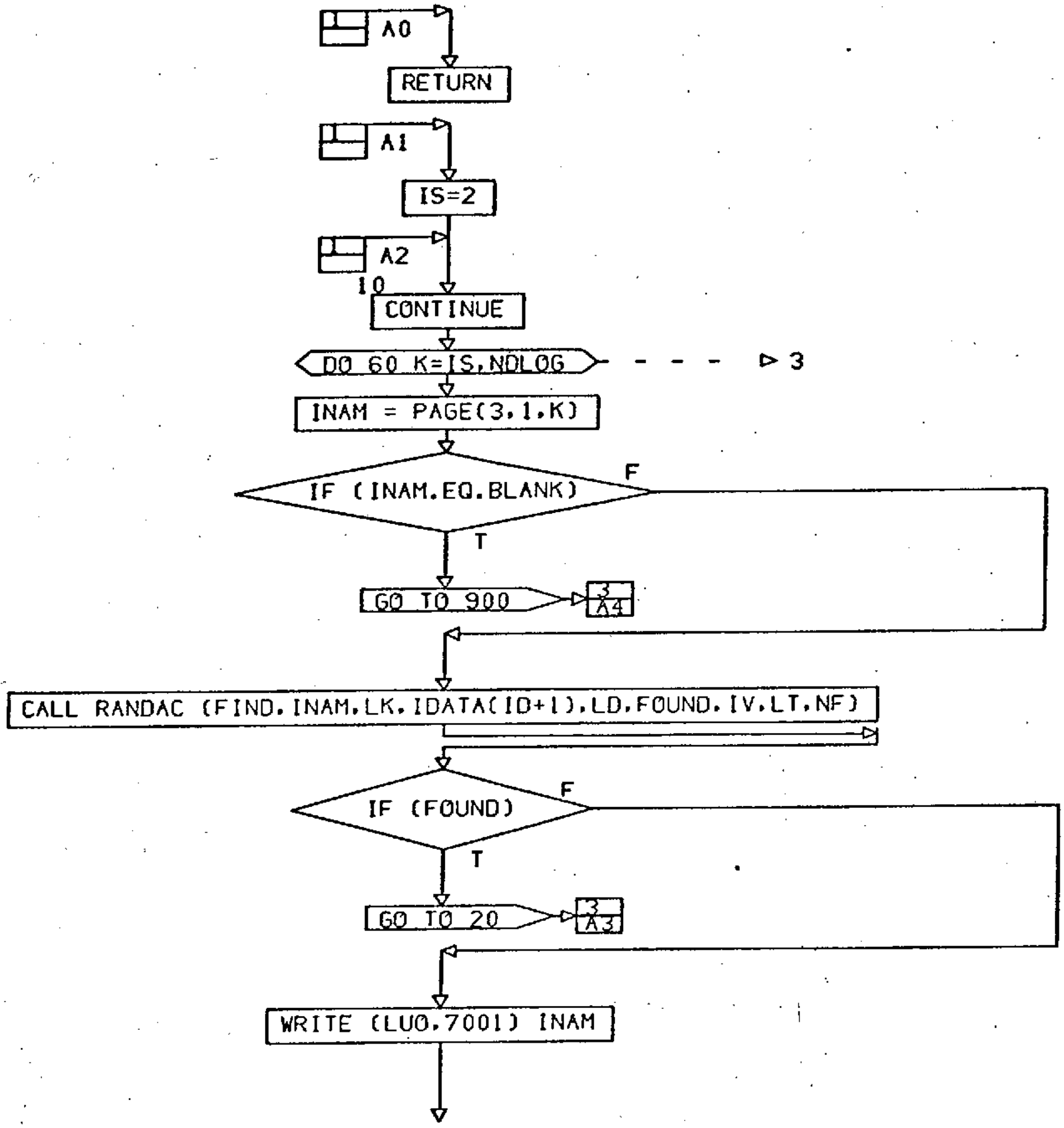


C-8

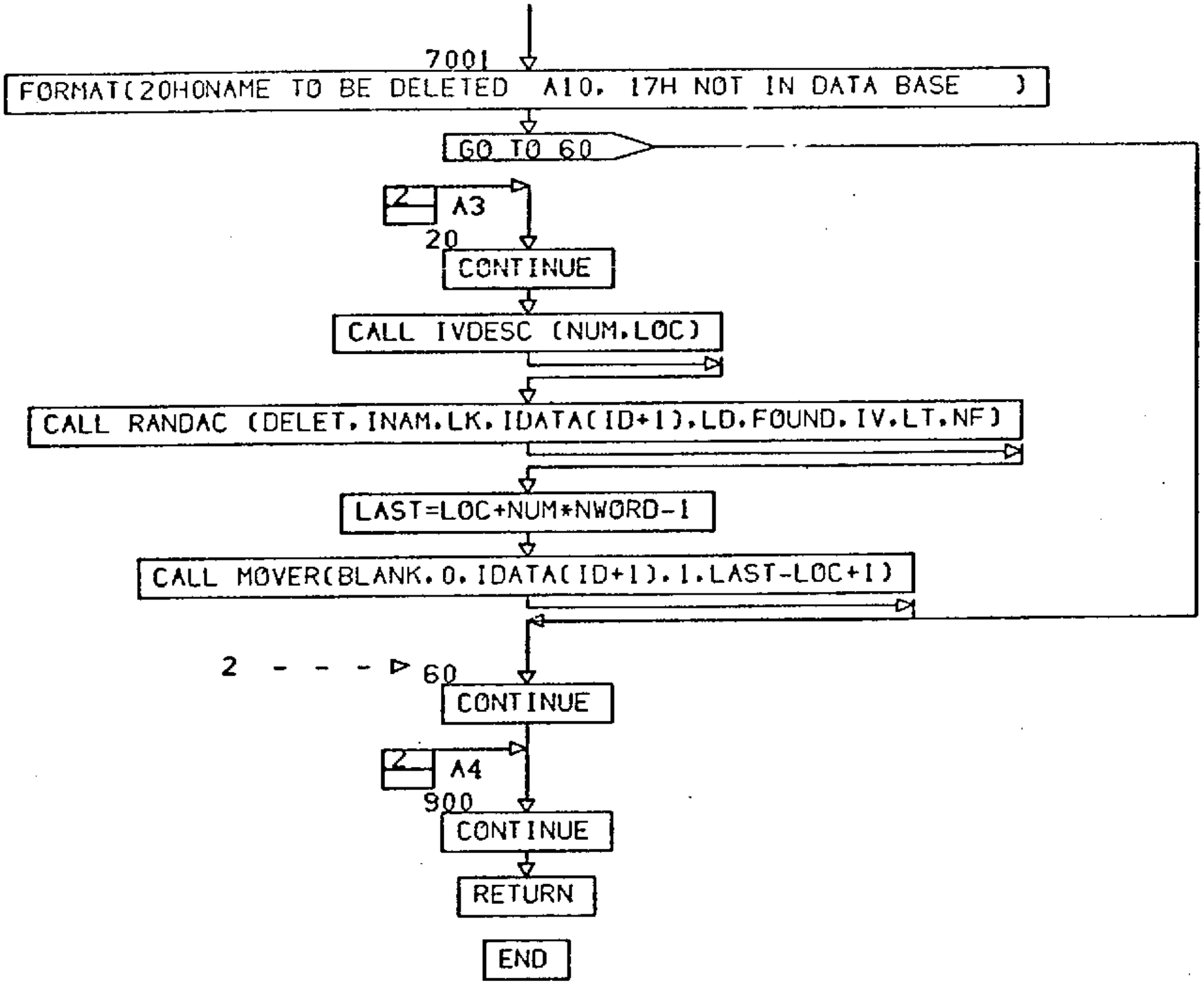


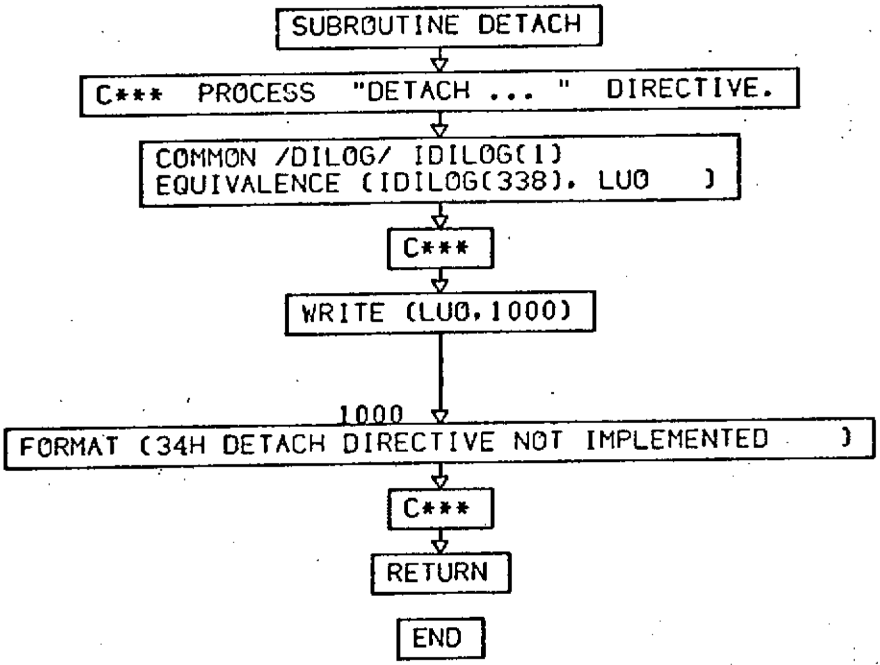
```
SUBROUTINE DELETE
COMMON /CARDBD/ IPAG(1)
EQUIVALENCE (IPAG(3), NDLOG)

C                    DELETES REQUESTED NAME(S) FROM DIRECTORY AND DATA BAS

COMMON IDATA(1)
EQUIVALENCE (ID, IDATA)
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 43), FIND  )
EQUIVALENCE (IDILOG(223), IV    )
EQUIVALENCE (IDILOG(243), LD    )

EQUIVALENCE (IDILOG(246), LK    )
EQUIVALENCE (IDILOG(248), LT    )
EQUIVALENCE (IDILOG(259), NF    )
EQUIVALENCE (IDILOG(263), NWORD )
EQUIVALENCE (IDILOG(271), DELET )
EQUIVALENCE (IDILOG(338), LUO   )
INTEGER IV(1), BLANK, DELIM, FIND
INTEGER PAGE,DELET

LOGICAL FOUND
IS=1

IF (PAGE(2,1,1).NE.DELIM )

GO TO 10

IF (PAGE(3,1,1).EQ.BLANK)
```

CONT. ON PG 2

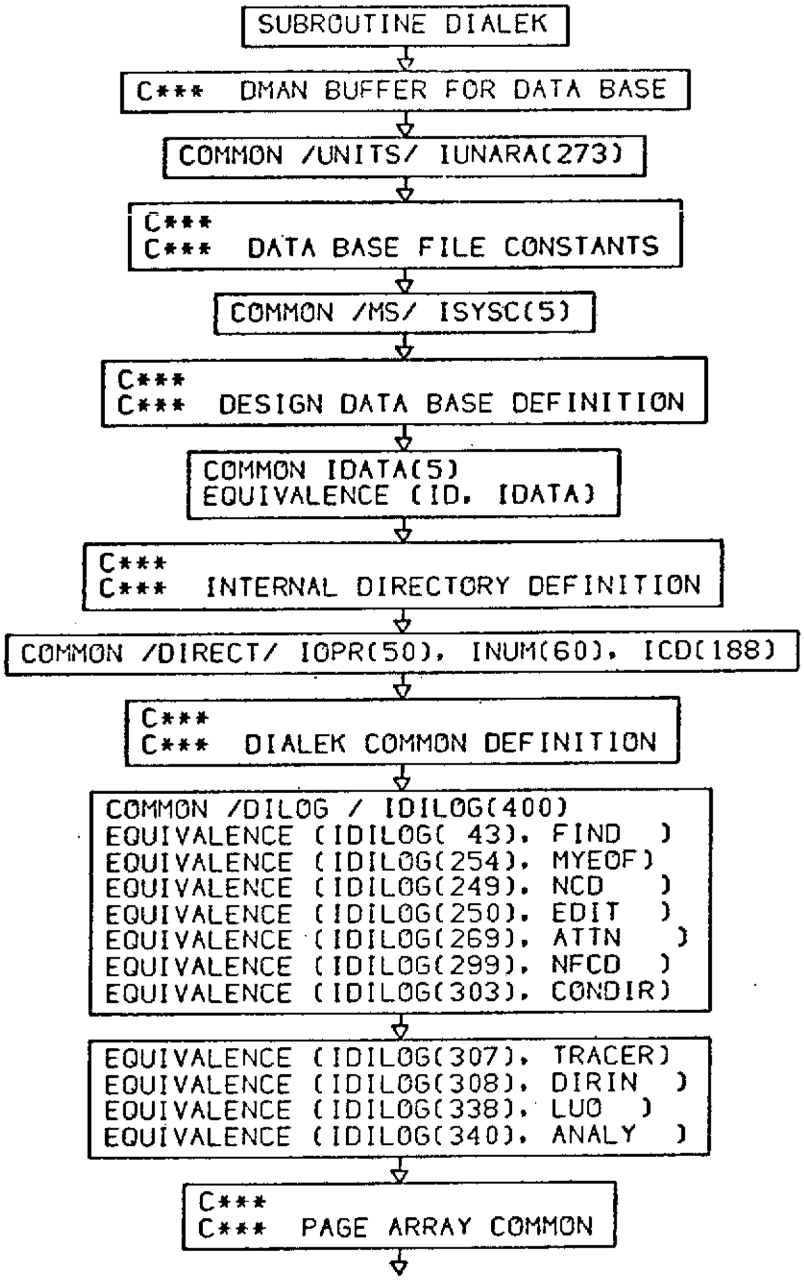

A0
RETURN
A1
IS=2
A2
10
CONTINUE
DO 60 K=IS,NDLOG
3
INAM = PAGE(3,1,K)
IF (INAM.EQ.BLANK)
F
T
GO TO 900
3
A4
CALL RANDAC (FIND,INAM,LK,IDATA(ID+1),LD,FOUND,IV,LT,NF)
IF (FOUND)
F
T
GO TO 20
3
A3
WRITE (LUO,7001) INAM

CONT. ON PG 3


7001
FORMAT(20HONAME TO BE DELETED A10, 17H NOT IN DATA BASE   )
GO TO 60
2
A3
20
CONTINUE
CALL IVDESC (NUM,LOC)
CALL RANDAC (DELET,INAM,LK,IDATA(ID+1),LD,FOUND,IV,LT,NF)
LAST=LOC+NUM*NWORD-1
CALL MOVER(BLANK,0,IDATA(ID+1),1,LAST-LOC+1)
2 - - - ▷ 60
CONTINUE
2
A4
900
CONTINUE
RETURN
END

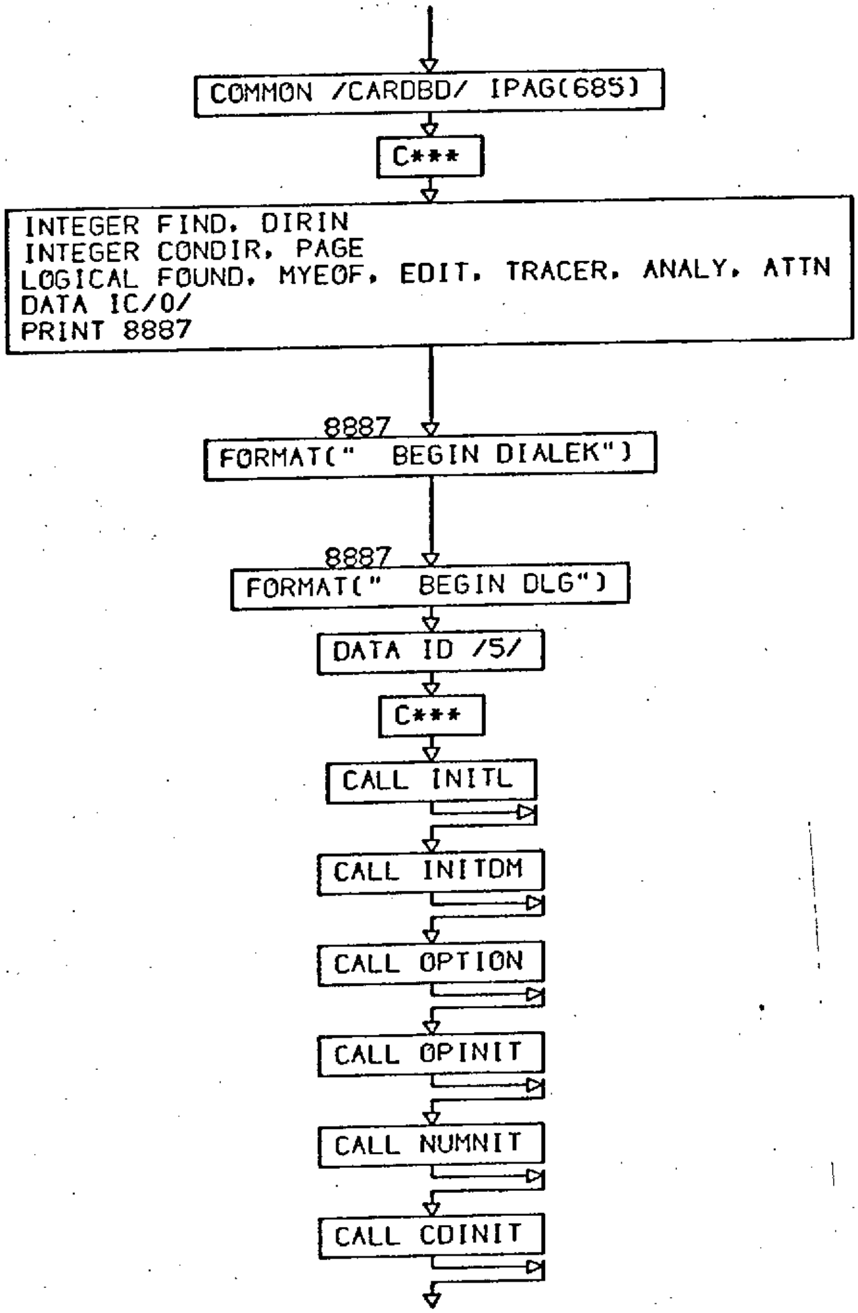


```
      SUBROUTINE DETACH
C***  PROCESS  "DETACH ... "  DIRECTIVE.
      COMMON /DILOG/ IDILOG(1)
      EQUIVALENCE (IDILOG(338), LUO    )
C***
      WRITE (LUO,1000)
 1000 FORMAT (34H DETACH DIRECTIVE NOT IMPLEMENTED      )
C***
      RETURN
      END
```



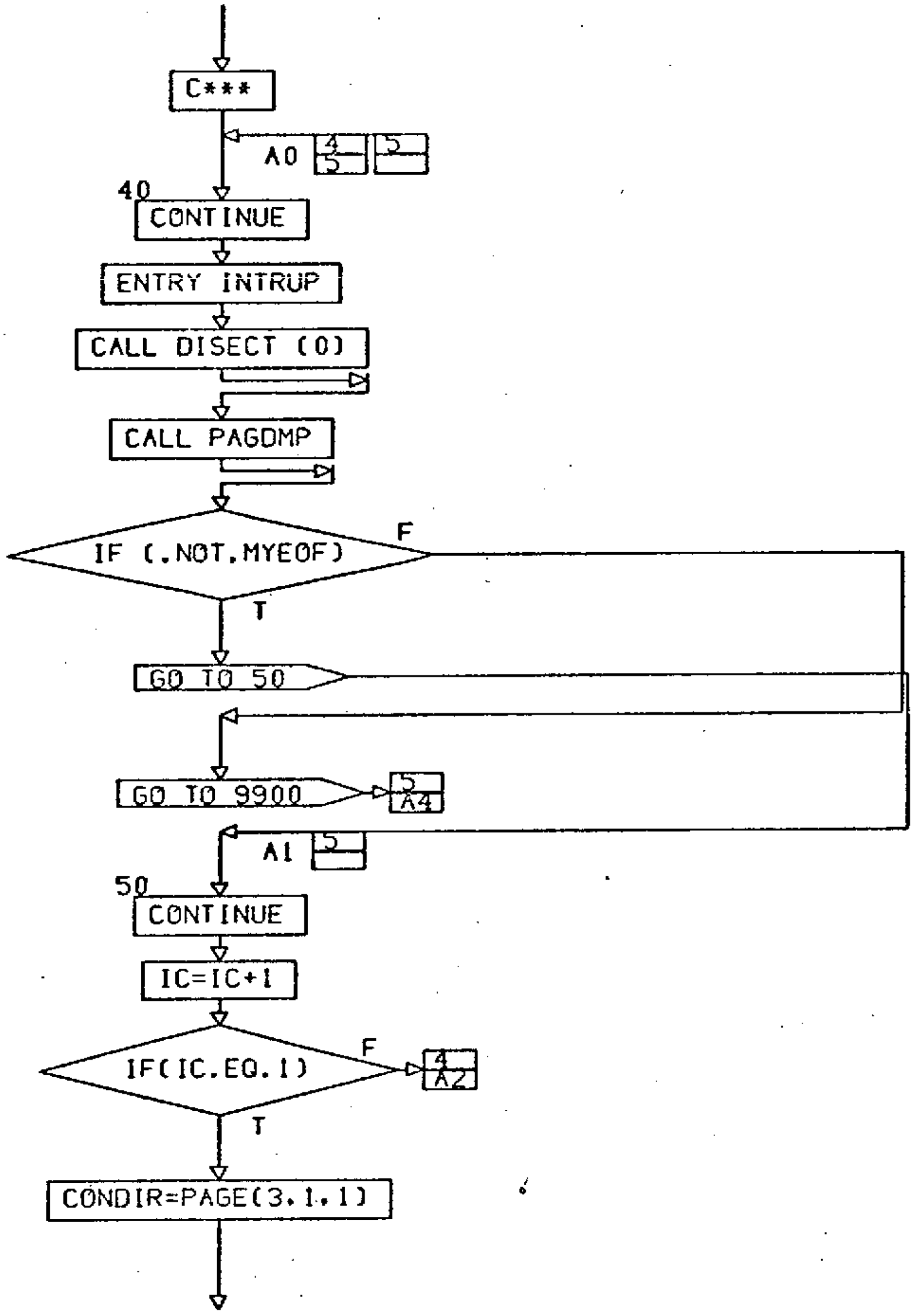
```
SUBROUTINE DIALEK
C***  DMAN BUFFER FOR DATA BASE
COMMON /UNITS/ IUNARA(273)
C***
C***  DATA BASE FILE CONSTANTS
COMMON /MS/ ISYSC(5)
C***
C***  DESIGN DATA BASE DEFINITION
COMMON IDATA(5)
EQUIVALENCE (ID, IDATA)
C***
C***  INTERNAL DIRECTORY DEFINITION
COMMON /DIRECT/ IOPR(50), INUM(60), ICD(188)
C***
C***  DIALEK COMMON DEFINITION
COMMON /DILOG / IDILOG(400)
EQUIVALENCE (IDILOG( 43), FIND  )
EQUIVALENCE (IDILOG(254), MYEOF)
EQUIVALENCE (IDILOG(249), NCD   )
EQUIVALENCE (IDILOG(250), EDIT  )
EQUIVALENCE (IDILOG(269), ATTN   )
EQUIVALENCE (IDILOG(299), NFCD  )
EQUIVALENCE (IDILOG(303), CONDIR)
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(308), DIRIN   )
EQUIVALENCE (IDILOG(338), LUO   )
EQUIVALENCE (IDILOG(340), ANALY   )
C***
C***  PAGE ARRAY COMMON
```

CONT. ON PG 2

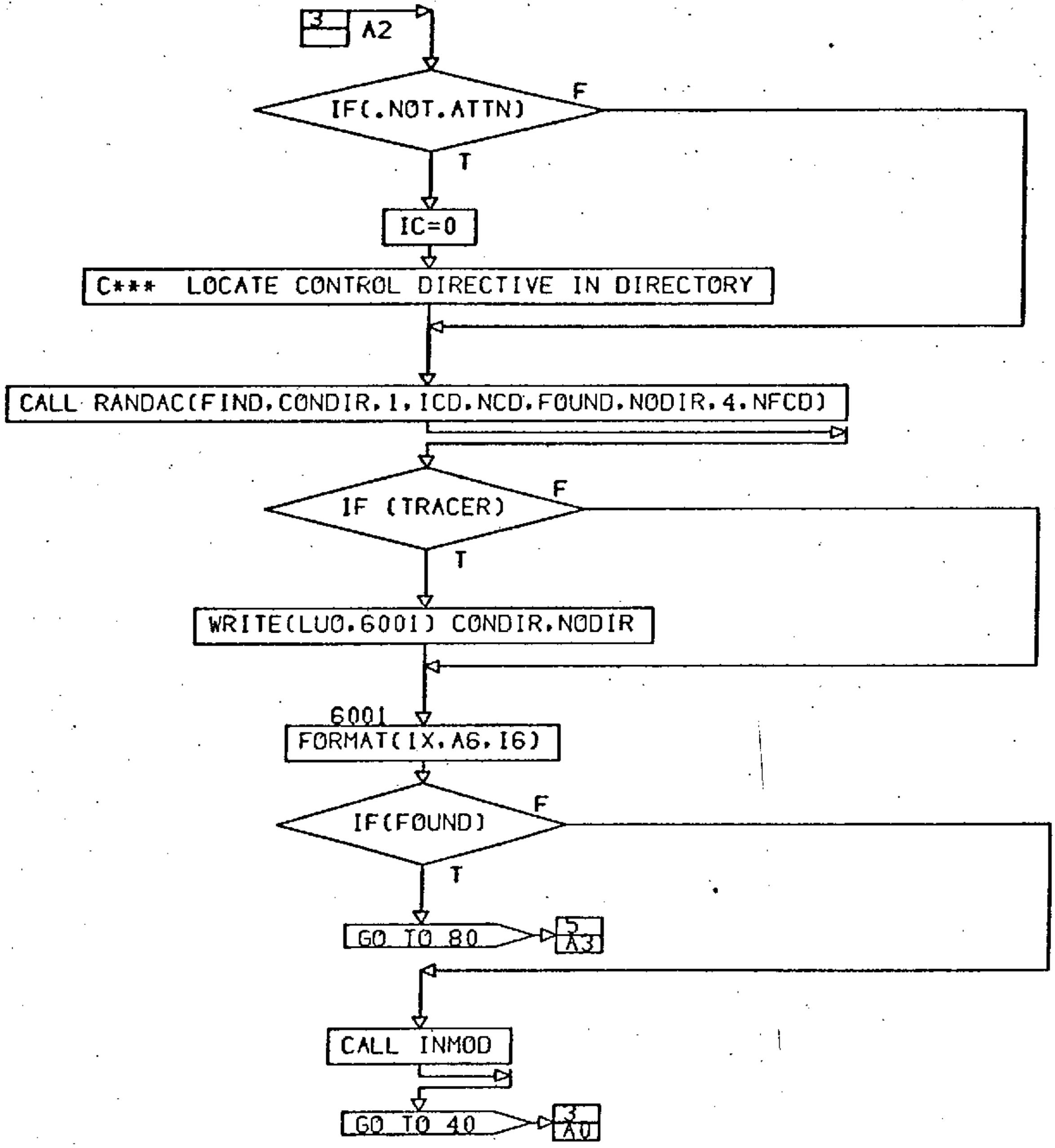


```
COMMON /CARDBD/ IPAG(685)
C***
INTEGER FIND, DIRIN
INTEGER CONDIR, PAGE
LOGICAL FOUND, MYEOF, EDIT, TRACER, ANALY, ATTN
DATA IC/0/
PRINT 8887
8887 FORMAT("  BEGIN DIALEK")
8887 FORMAT("  BEGIN DLG")
DATA ID /5/
C***
CALL INITL
CALL INITDM
CALL OPTION
CALL OPINIT
CALL NUMNIT
CALL CDINIT
```

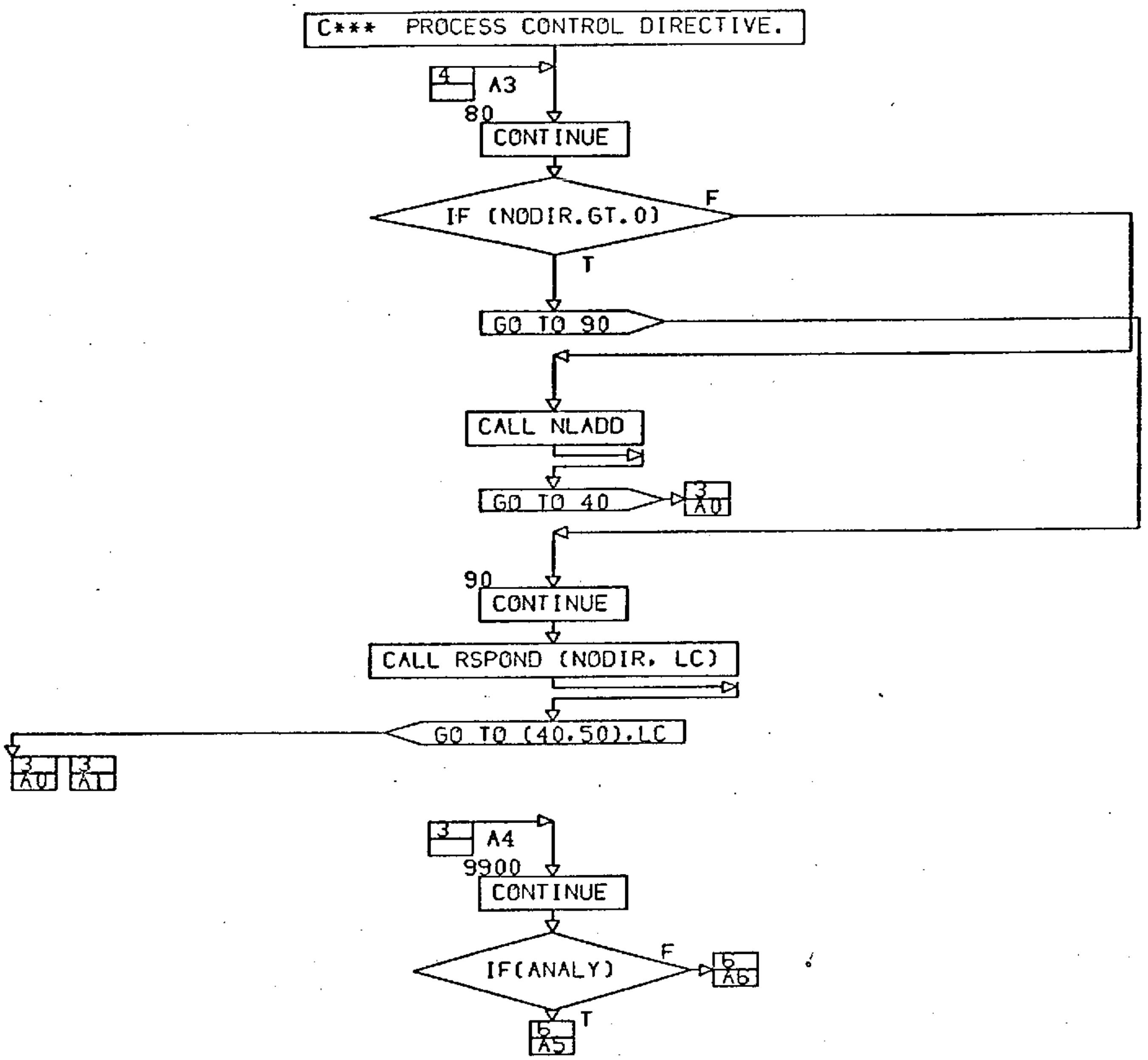
CONT. ON PG 3


C***
A0
4
5
5
40
CONTINUE
ENTRY INTRUP
CALL DISECT (0)
CALL PAGDMP
IF (.NOT.MYEOF)
F
T
GO TO 50
GO TO 9900
5
A4
A1
5
50
CONTINUE
IC=IC+1
IF(IC.EQ.1)
F
4
A2
T
CONDIR=PAGE(3,1,1)

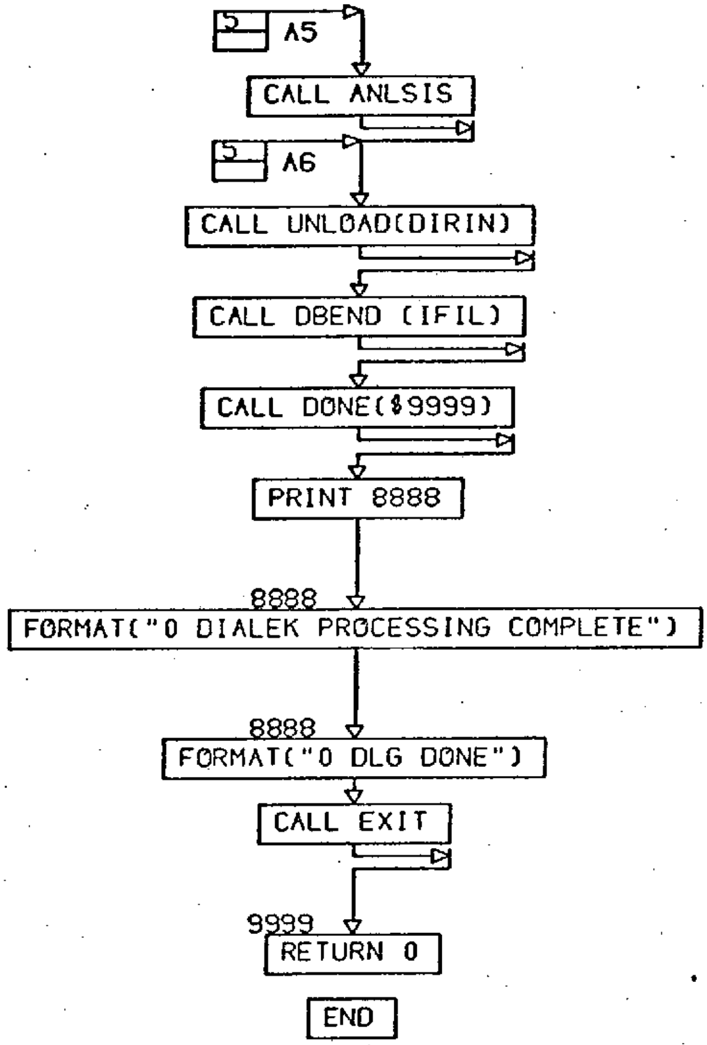
CONT. ON PG 4

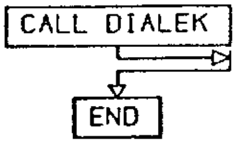

3
A2
IF(.NOT.ATTN)
F
T
IC=0
C*** LOCATE CONTROL DIRECTIVE IN DIRECTORY
CALL RANDAC(FIND,CONDIR,1,ICD,NCD,FOUND,NODIR,4,NFCD)
IF (TRACER)
F
T
WRITE(LUO,6001) CONDIR,NODIR
6001
FORMAT(1X,A6,I6)
IF(FOUND)
F
T
GO TO 80
5
A3
CALL INMOD
GO TO 40
3
A0

CONT. ON PG 5

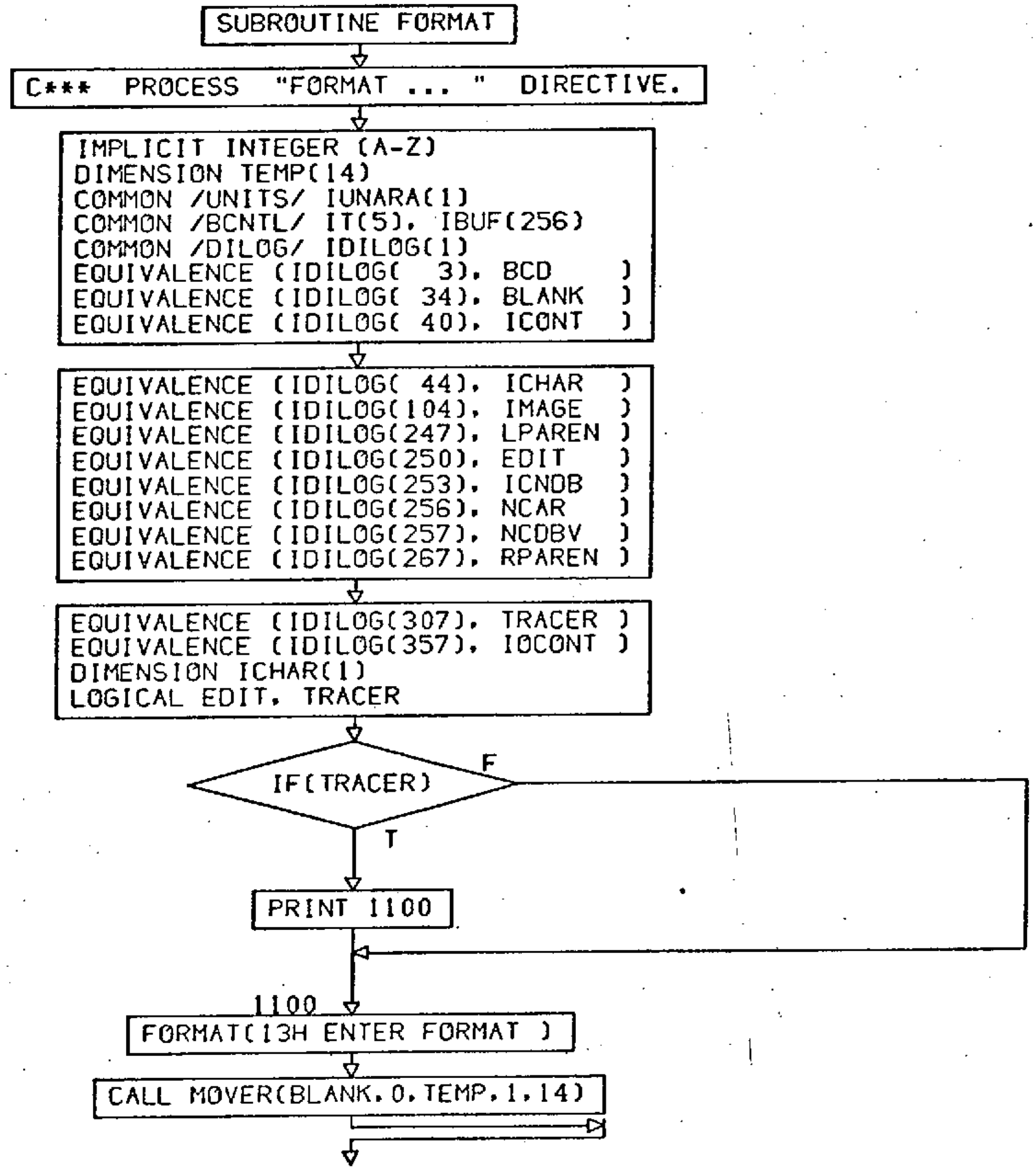

C*** PROCESS CONTROL DIRECTIVE.
4
A3
80
CONTINUE
IF (NODIR.GT.0)
F
T
GO TO 90
CALL NLADD
GO TO 40
3
A0
90
CONTINUE
CALL RSPOND (NODIR, LC)
GO TO (40,50),LC
3
A0
3
A1
3
A4
9900
CONTINUE
IF(ANALY)
F
6
A6
T
6
A5

CONT. ON PG 6


A5
CALL ANLSIS
A6
CALL UNLOAD(DIRIN)
CALL DBEND (IFIL)
CALL DONE($9999)
PRINT 8888
8888
FORMAT("0 DIALEK PROCESSING COMPLETE")
8888
FORMAT("0 DLG DONE")
CALL EXIT
9999
RETURN 0
END

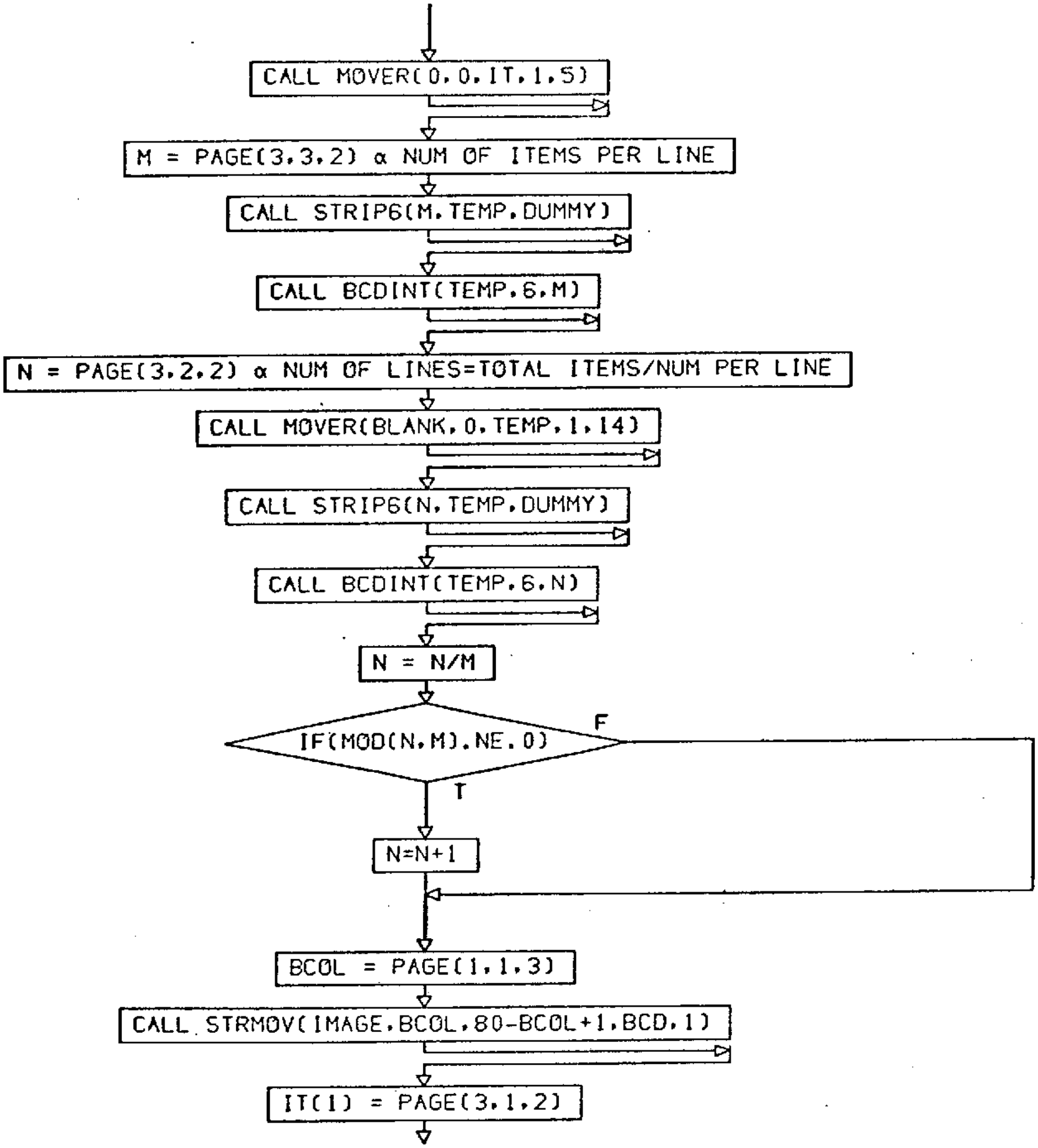


DLGDVR
PG 1 FINAL

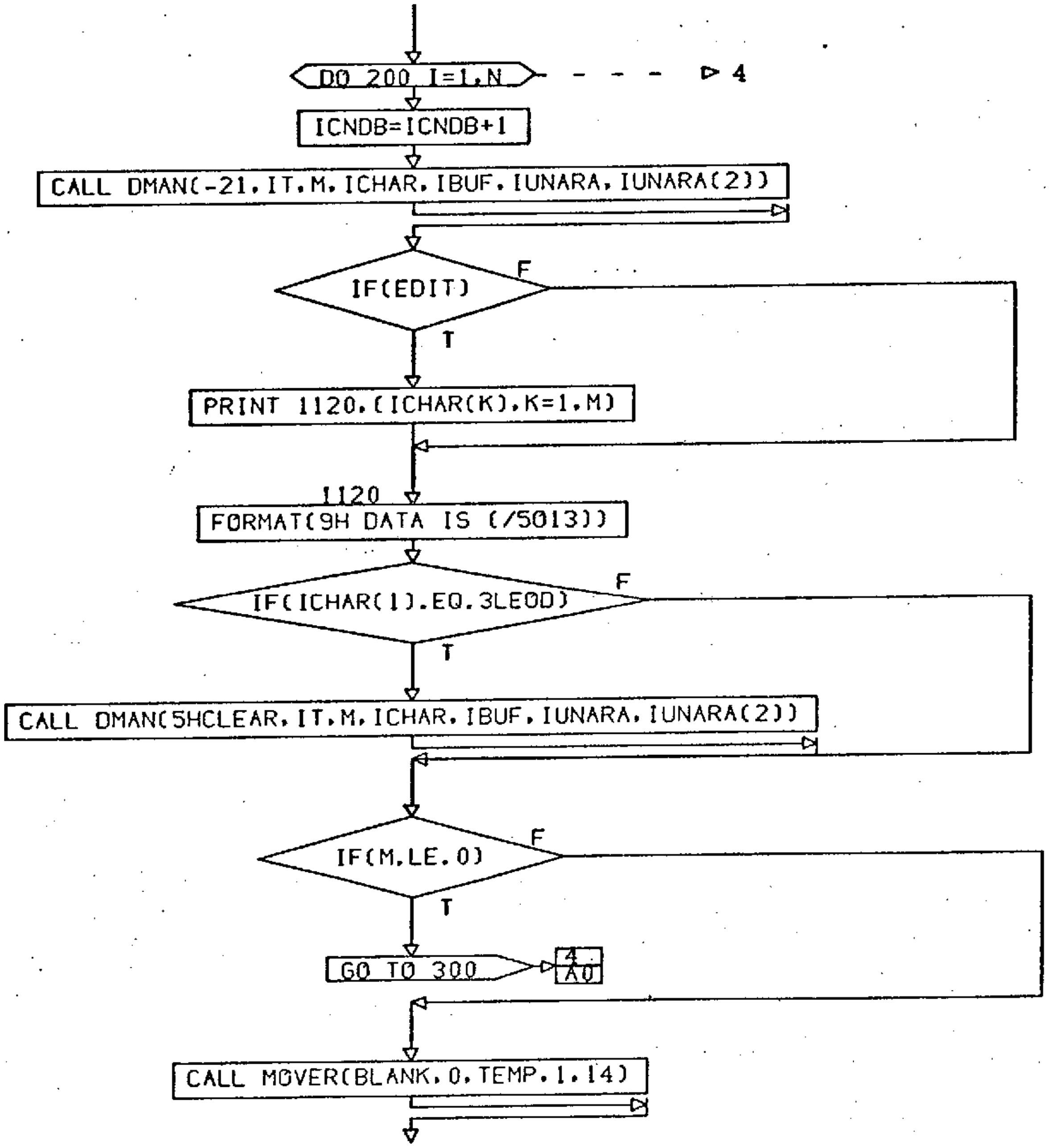


```
SUBROUTINE FORMAT
C***  PROCESS  "FORMAT ... "  DIRECTIVE.
IMPLICIT INTEGER (A-Z)
DIMENSION TEMP(14)
COMMON /UNITS/ IUNARA(1)
COMMON /BCNTL/ IT(5), IBUF(256)
COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG(  3), BCD     )
EQUIVALENCE (IDILOG( 34), BLANK   )
EQUIVALENCE (IDILOG( 40), ICONT   )
EQUIVALENCE (IDILOG( 44), ICHAR   )
EQUIVALENCE (IDILOG(104), IMAGE   )
EQUIVALENCE (IDILOG(247), LPAREN  )
EQUIVALENCE (IDILOG(250), EDIT    )
EQUIVALENCE (IDILOG(253), ICNDB   )
EQUIVALENCE (IDILOG(256), NCAR    )
EQUIVALENCE (IDILOG(257), NCDBV   )
EQUIVALENCE (IDILOG(267), RPAREN  )
EQUIVALENCE (IDILOG(307), TRACER  )
EQUIVALENCE (IDILOG(357), IOCONT  )
DIMENSION ICHAR(1)
LOGICAL EDIT, TRACER
IF(TRACER)
PRINT 1100
1100 FORMAT(13H ENTER FORMAT )
CALL MOVER(BLANK,0,TEMP,1,14)
```

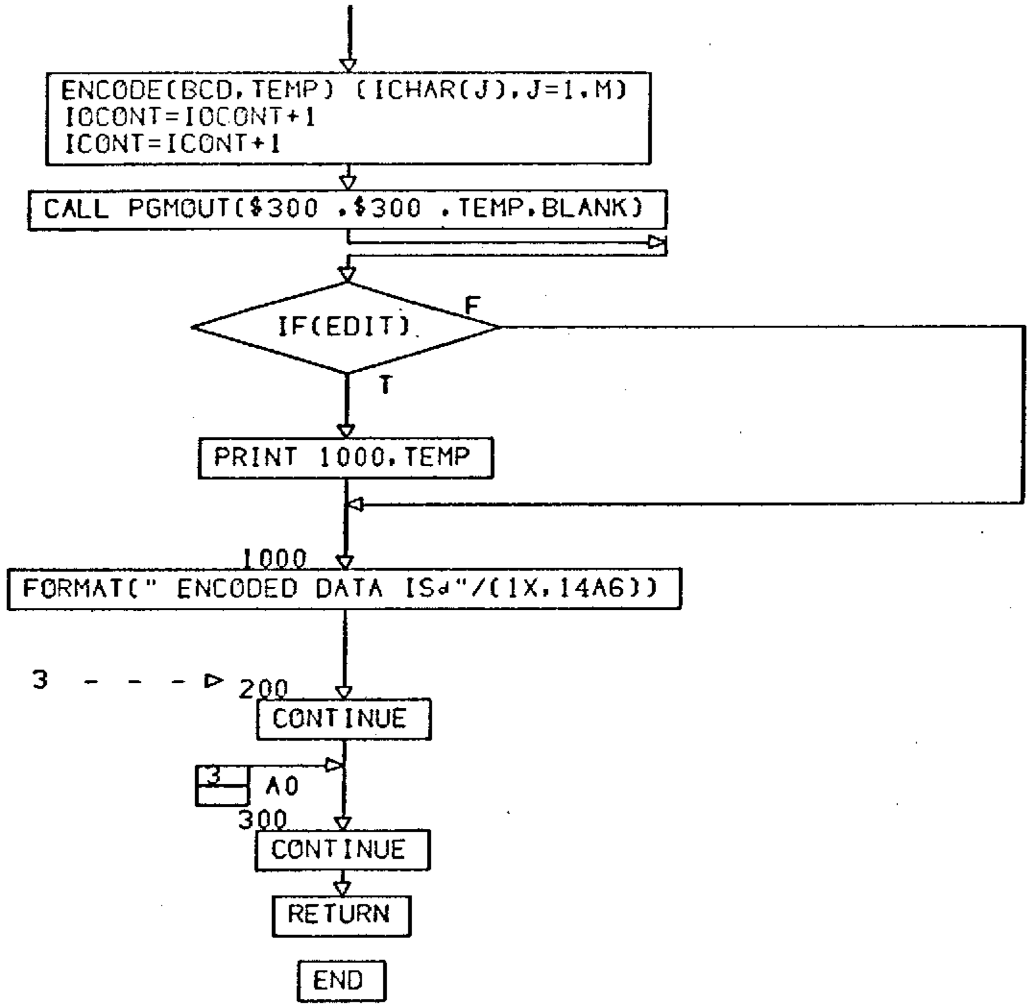
CONT. ON PG 2

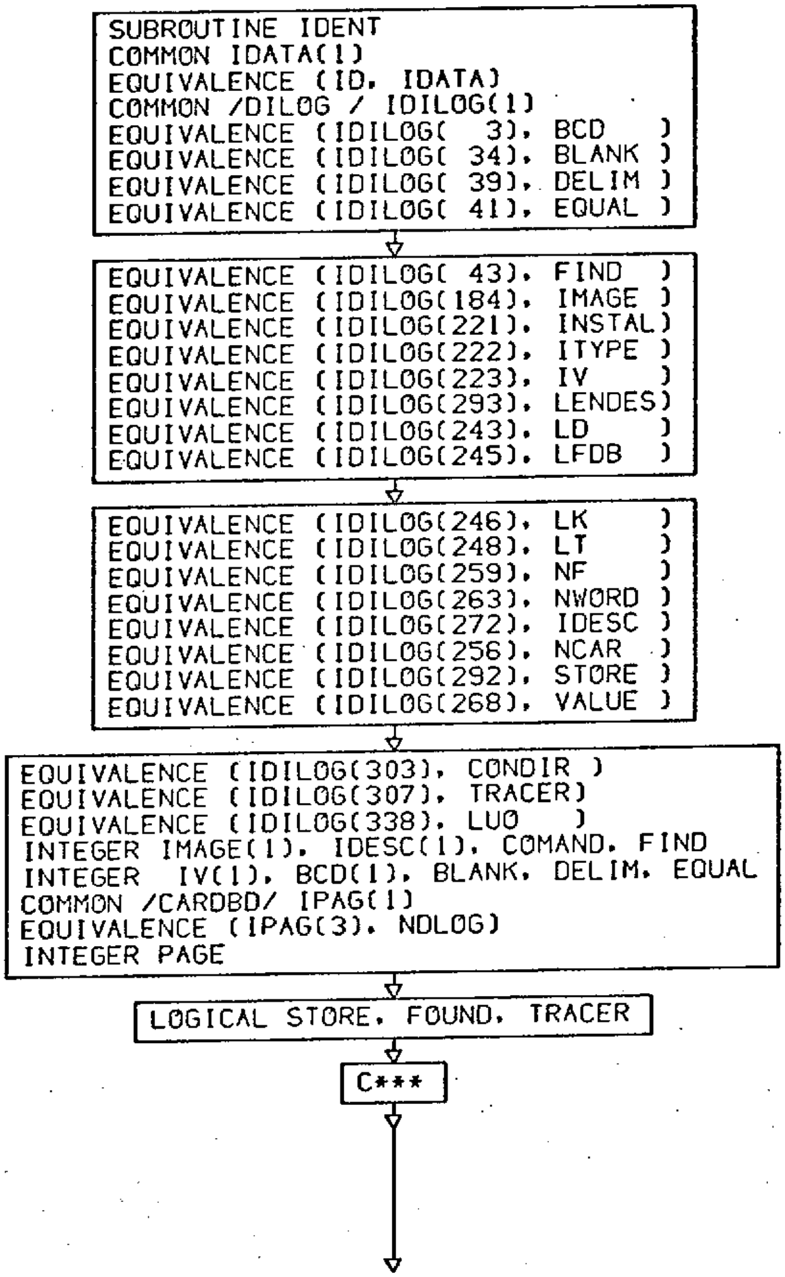

CALL MOVER(0,0,IT,1,5)
M = PAGE(3,3,2) α NUM OF ITEMS PER LINE
CALL STRIP6(M,TEMP,DUMMY)
CALL BCDINT(TEMP,6,M)
N = PAGE(3,2,2) α NUM OF LINES=TOTAL ITEMS/NUM PER LINE
CALL MOVER(BLANK,0,TEMP,1,14)
CALL STRIP6(N,TEMP,DUMMY)
CALL BCDINT(TEMP,6,N)
N = N/M
IF(MOD(N,M).NE.0)
F
T
N=N+1
BCOL = PAGE(1,1,3)
CALL STRMOV(IMAGE,BCOL,80-BCOL+1,BCD,1)
IT(1) = PAGE(3,1,2)

CONT. ON PG 3


DO 200 I=1,N
4
ICNDB=ICNDB+1
CALL DMAN(-21,IT,M,ICHAR,IBUF,IUNARA,IUNARA(2))
IF(EDIT)
F
T
PRINT 1120,(ICHAR(K),K=1,M)
1120
FORMAT(9H DATA IS (/5013))
IF(ICHAR(1).EQ.3LEOD)
F
T
CALL DMAN(5HCLEAR,IT,M,ICHAR,IBUF,IUNARA,IUNARA(2))
IF(M.LE.0)
F
T
GO TO 300
4
A0
CALL MOVER(BLANK,0,TEMP,1,14)

CONT. ON PG 4


ENCODE(BCD,TEMP) (ICHAR(J),J=1,M)
IOCONT=IOCONT+1
ICONT=ICONT+1
CALL PGMOUT($300 ,$300 ,TEMP,BLANK)
IF(EDIT)
F
T
PRINT 1000,TEMP
1000
FORMAT(" ENCODED DATA IS↲"/(1X,14A6))
3
200
CONTINUE
3
A0
300
CONTINUE
RETURN
END



```
SUBROUTINE IDENT
COMMON IDATA(1)
EQUIVALENCE (ID, IDATA)
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG(  3), BCD    )
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 41), EQUAL )
```

```
EQUIVALENCE (IDILOG( 43), FIND   )
EQUIVALENCE (IDILOG(184), IMAGE )
EQUIVALENCE (IDILOG(221), INSTAL)
EQUIVALENCE (IDILOG(222), ITYPE )
EQUIVALENCE (IDILOG(223), IV     )
EQUIVALENCE (IDILOG(293), LENDES)
EQUIVALENCE (IDILOG(243), LD     )
EQUIVALENCE (IDILOG(245), LFDB   )
```

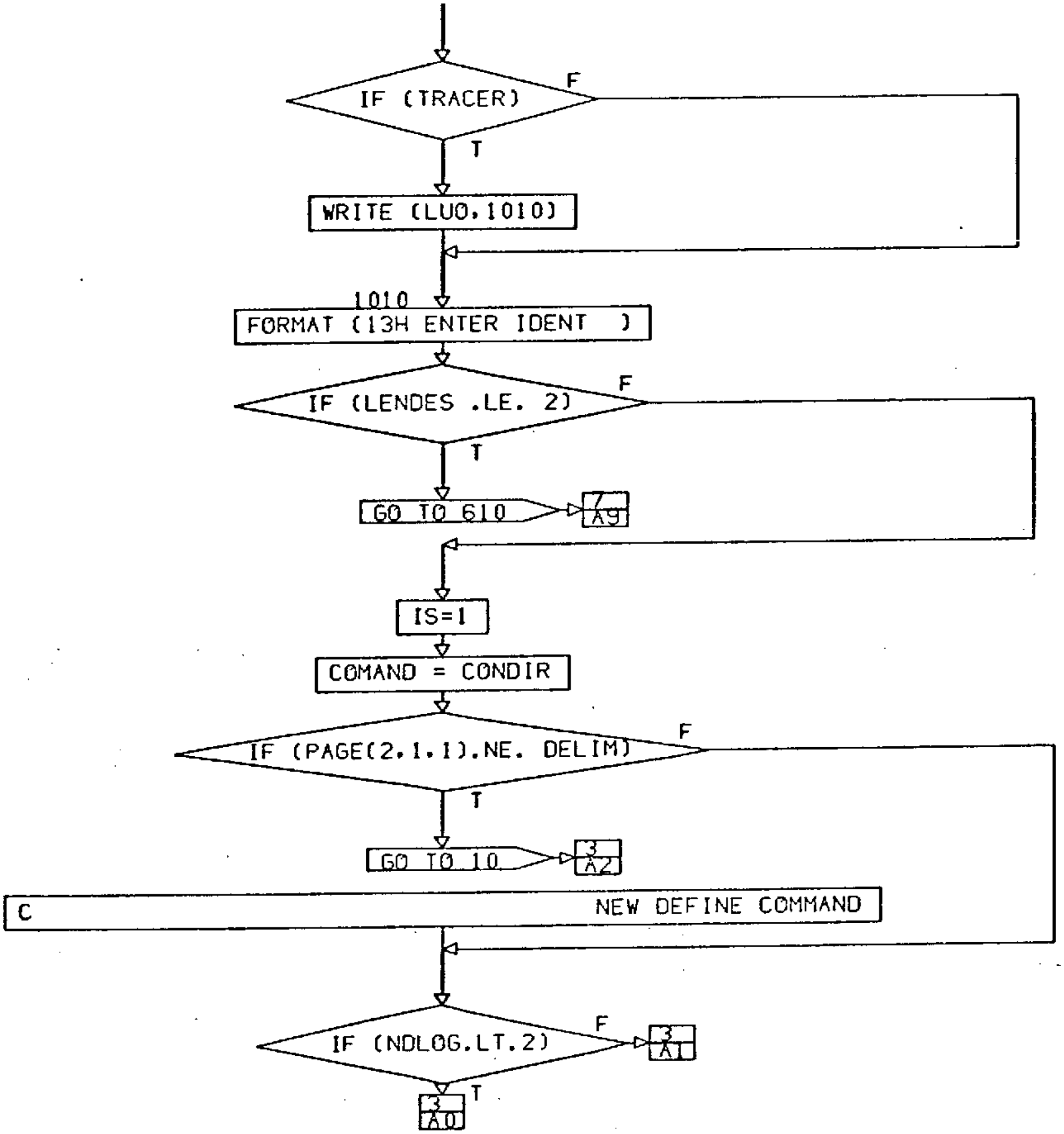
```
EQUIVALENCE (IDILOG(246), LK     )
EQUIVALENCE (IDILOG(248), LT     )
EQUIVALENCE (IDILOG(259), NF     )
EQUIVALENCE (IDILOG(263), NWORD )
EQUIVALENCE (IDILOG(272), IDESC )
EQUIVALENCE (IDILOG(256), NCAR  )
EQUIVALENCE (IDILOG(292), STORE )
EQUIVALENCE (IDILOG(268), VALUE )
```

```
EQUIVALENCE (IDILOG(303), CONDIR )
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(338), LUO    )
INTEGER IMAGE(1), IDESC(1), COMAND, FIND
INTEGER  IV(1), BCD(1), BLANK, DELIM, EQUAL
COMMON /CARDBD/ IPAG(1)
EQUIVALENCE (IPAG(3), NDLOG)
INTEGER PAGE
```

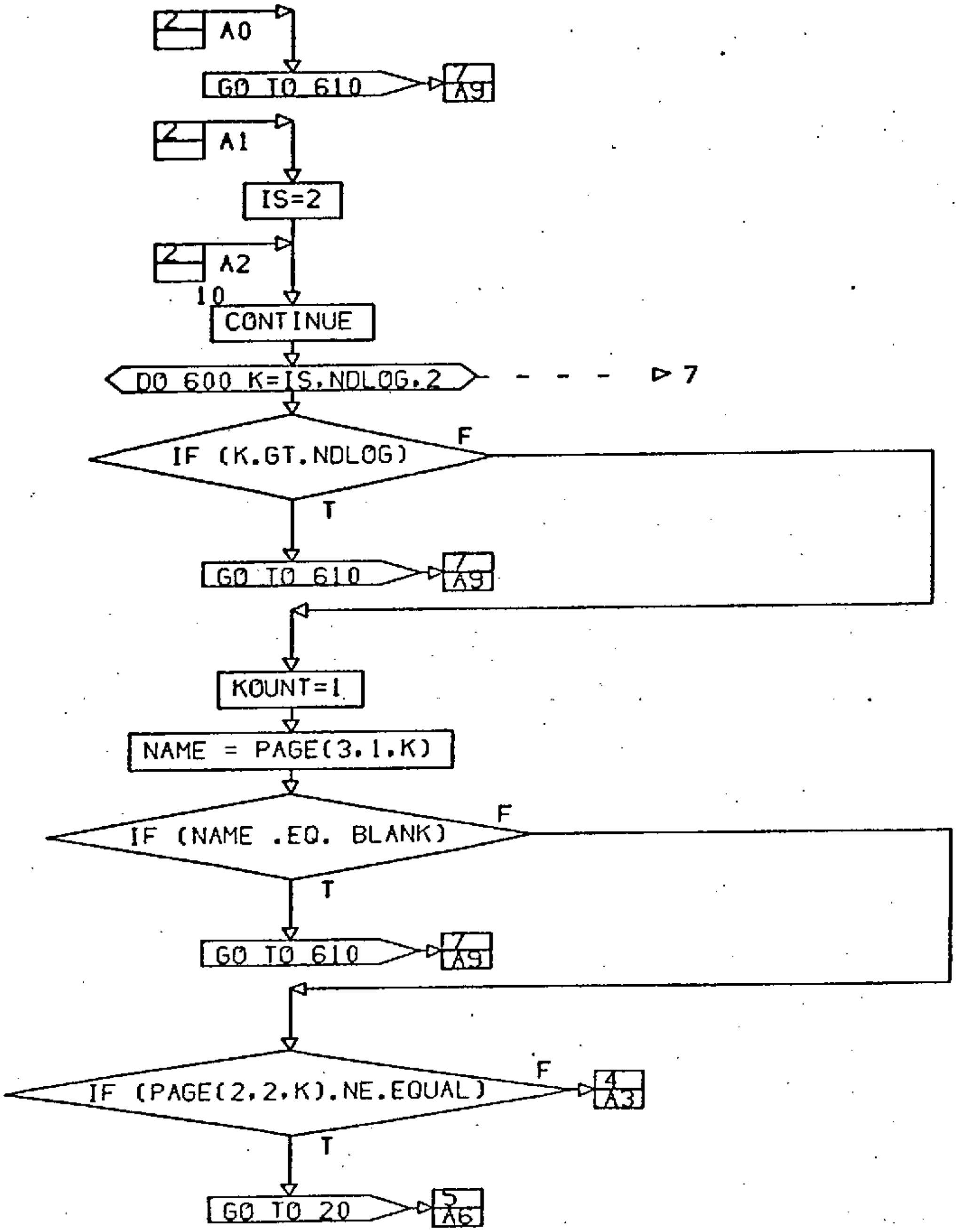
```
LOGICAL STORE, FOUND, TRACER
```

```
C***
```

CONT. ON PG 2

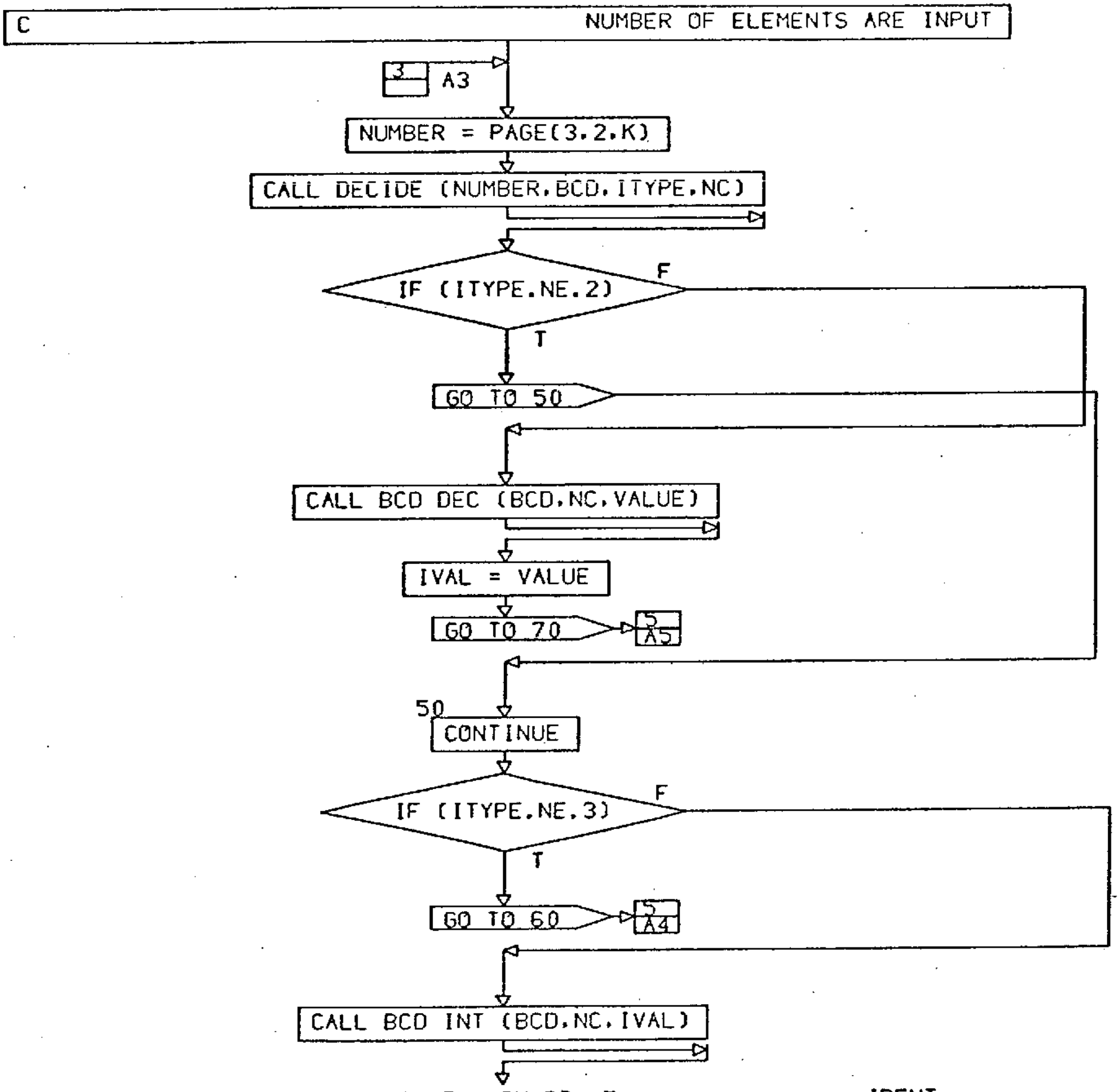

IF (TRACER)
F
T
WRITE (LUO,1010)
1010
FORMAT (13H ENTER IDENT)
IF (LENDES .LE. 2)
F
T
GO TO 610
7
A9
IS=1
COMAND = CONDIR
IF (PAGE(2,1,1).NE. DELIM)
F
T
GO TO 10
3
A2
C
NEW DEFINE COMMAND
IF (NDLOG.LT.2)
F
3
A1
T
3
A0

CONT. ON PG 3

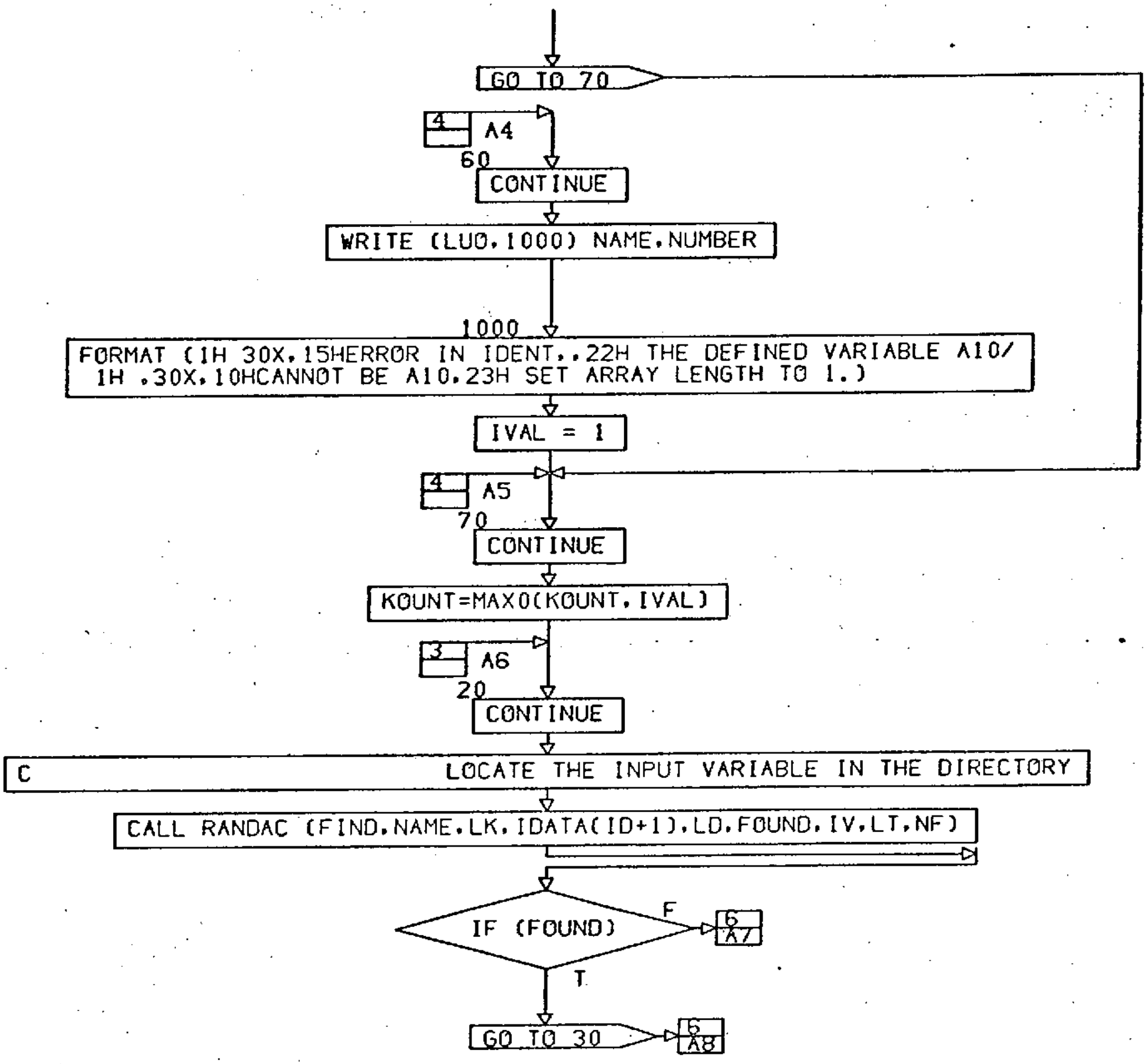


CONT. ON PG 4

IDENT
PG 3 OF 7

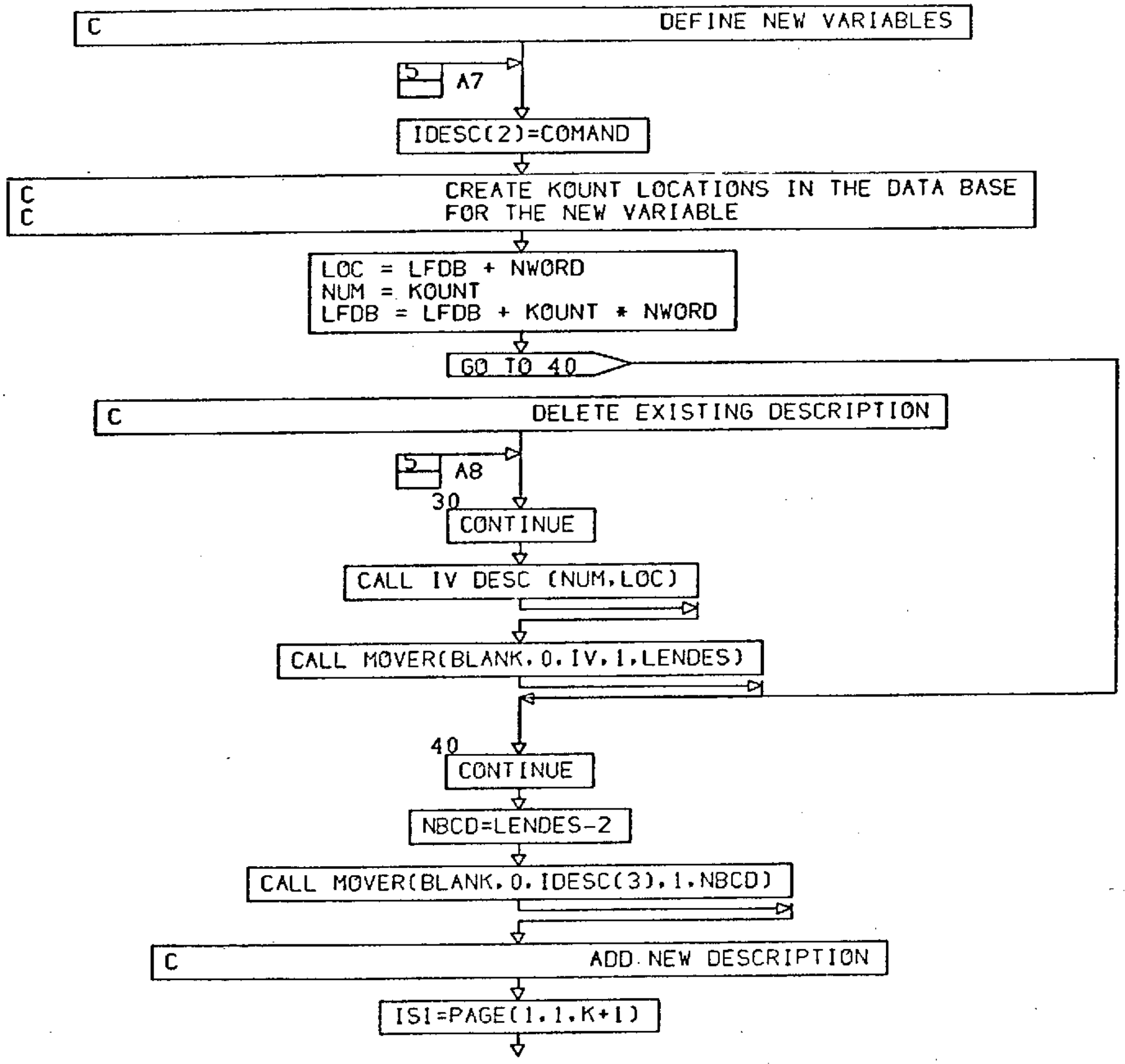

C NUMBER OF ELEMENTS ARE INPUT
3
A3
NUMBER = PAGE(3.2.K)
CALL DECIDE (NUMBER.BCD.ITYPE.NC)
IF (ITYPE.NE.2)
F
T
GO TO 50
CALL BCD DEC (BCD.NC.VALUE)
IVAL = VALUE
GO TO 70
5
A5
50
CONTINUE
IF (ITYPE.NE.3)
F
T
GO TO 60
5
A4
CALL BCD INT (BCD.NC.IVAL)

CONT. ON PG 5



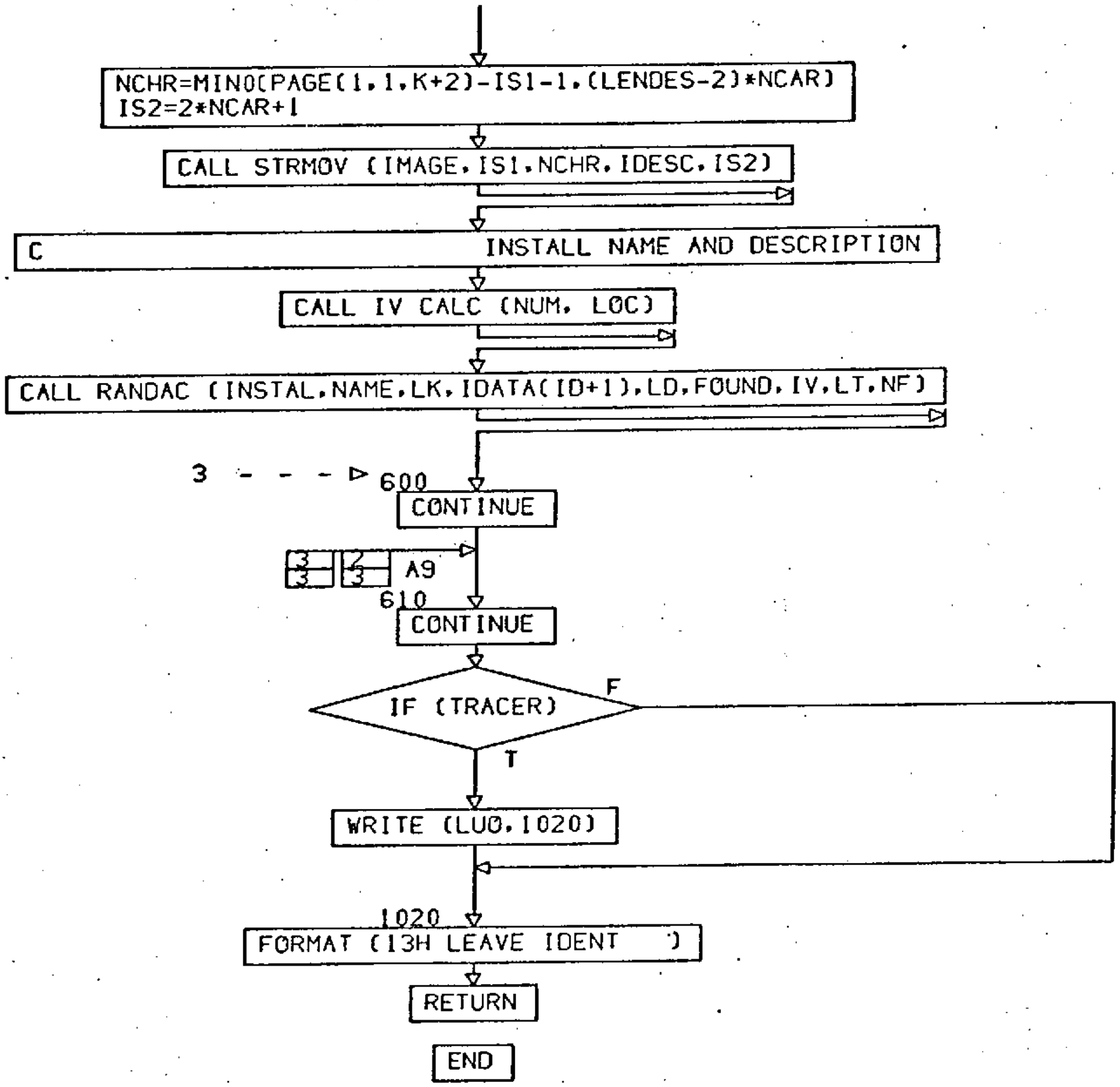
```
      GO TO 70
   60 CONTINUE
      WRITE (LUO,1000) NAME,NUMBER
 1000 FORMAT (1H 30X,15HERROR IN IDENT.,22H THE DEFINED VARIABLE A10/
       1H ,30X,10HCANNOT BE A10,23H SET ARRAY LENGTH TO 1.)
      IVAL = 1
   70 CONTINUE
      KOUNT=MAX0(KOUNT,IVAL)
   20 CONTINUE
C                                 LOCATE THE INPUT VARIABLE IN THE DIRECTORY
      CALL RANDAC (FIND,NAME,LK,IDATA(ID+1),LD,FOUND,IV,LT,NF)
      IF (FOUND)
      GO TO 30
```

CONT. ON PG 6

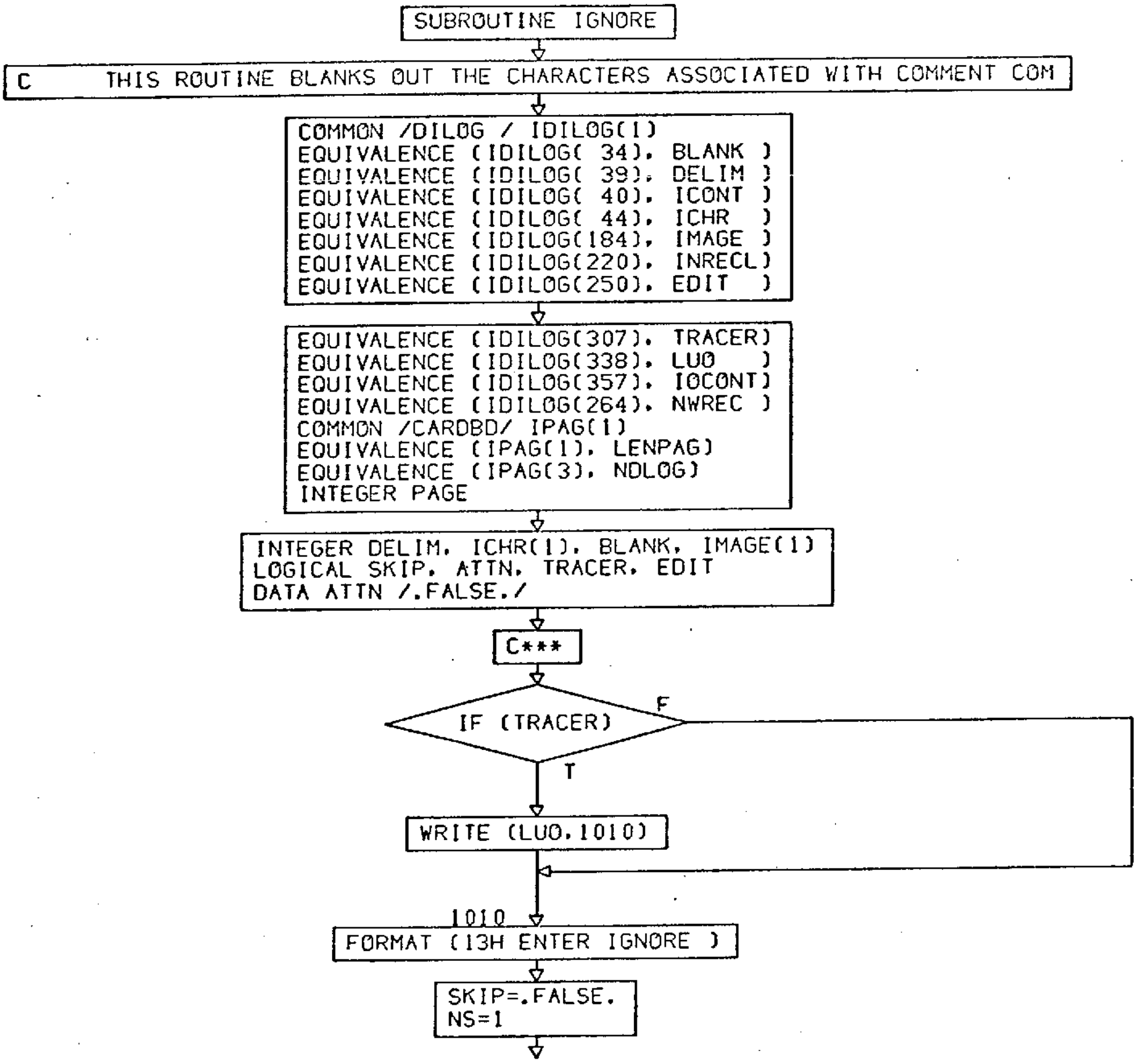


```
C                                         DEFINE NEW VARIABLES
      A7
      IDESC(2)=COMAND
C                             CREATE KOUNT LOCATIONS IN THE DATA BASE
C                             FOR THE NEW VARIABLE
      LOC = LFDB + NWORD
      NUM = KOUNT
      LFDB = LFDB + KOUNT * NWORD
      GO TO 40
C                                  DELETE EXISTING DESCRIPTION
      A8
   30 CONTINUE
      CALL IV DESC (NUM,LOC)
      CALL MOVER(BLANK,0,IV,1,LENDES)
   40 CONTINUE
      NBCD=LENDES-2
      CALL MOVER(BLANK,0,IDESC(3),1,NBCD)
C                                      ADD NEW DESCRIPTION
      ISI=PAGE(1,1,K+1)
```

CONT. ON PG 7

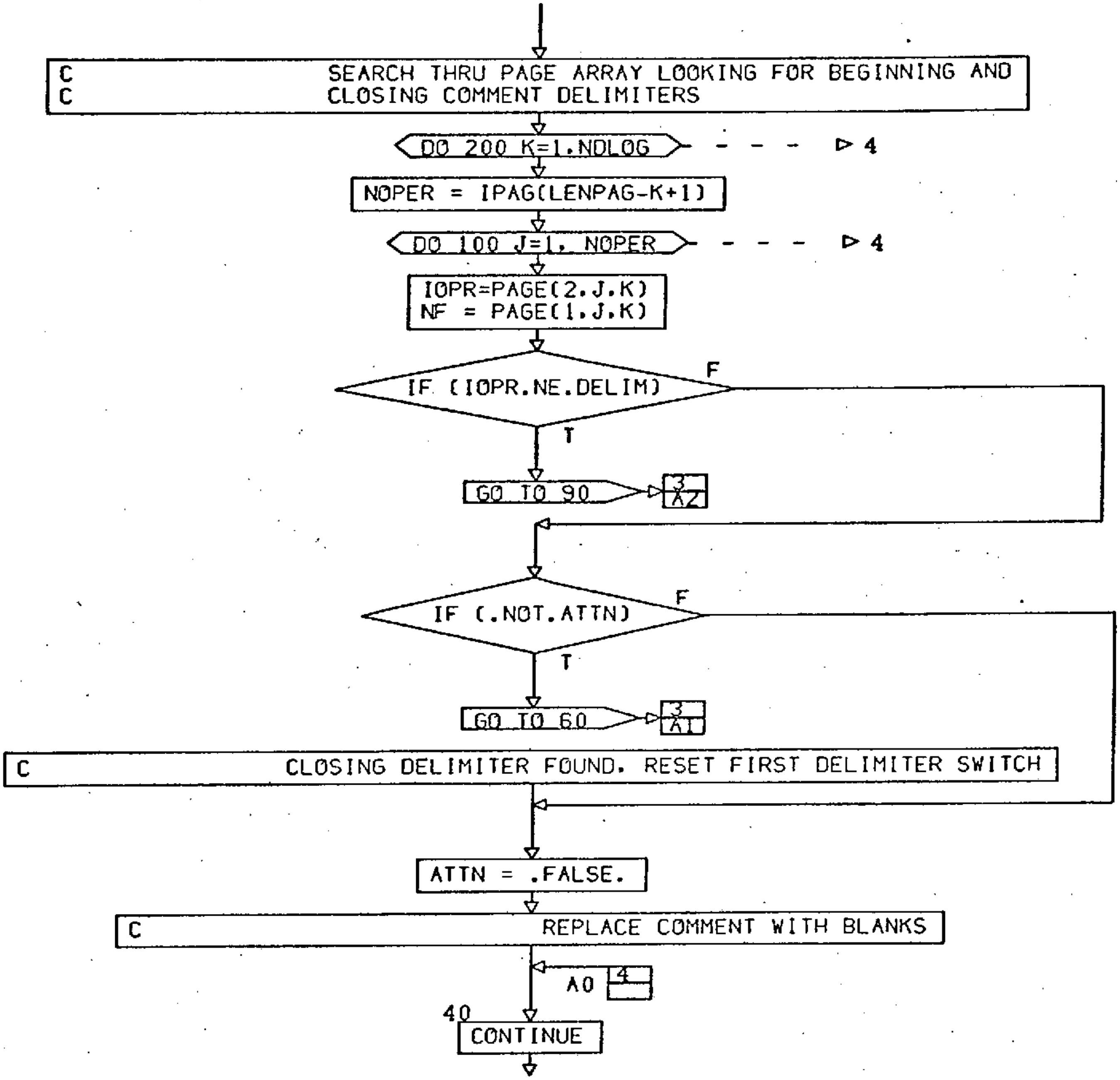


```
NCHR=MIN0(PAGE(1,1,K+2)-IS1-1,(LENDES-2)*NCAR)
IS2=2*NCAR+1
CALL STRMOV (IMAGE,IS1,NCHR,IDESC,IS2)
C                    INSTALL NAME AND DESCRIPTION
CALL IV CALC (NUM, LOC)
CALL RANDAC (INSTAL,NAME,LK,IDATA(ID+1),LD,FOUND,IV,LT,NF)
600 CONTINUE
610 CONTINUE
IF (TRACER)
WRITE (LUO,1020)
1020 FORMAT (13H LEAVE IDENT   )
RETURN
END
```

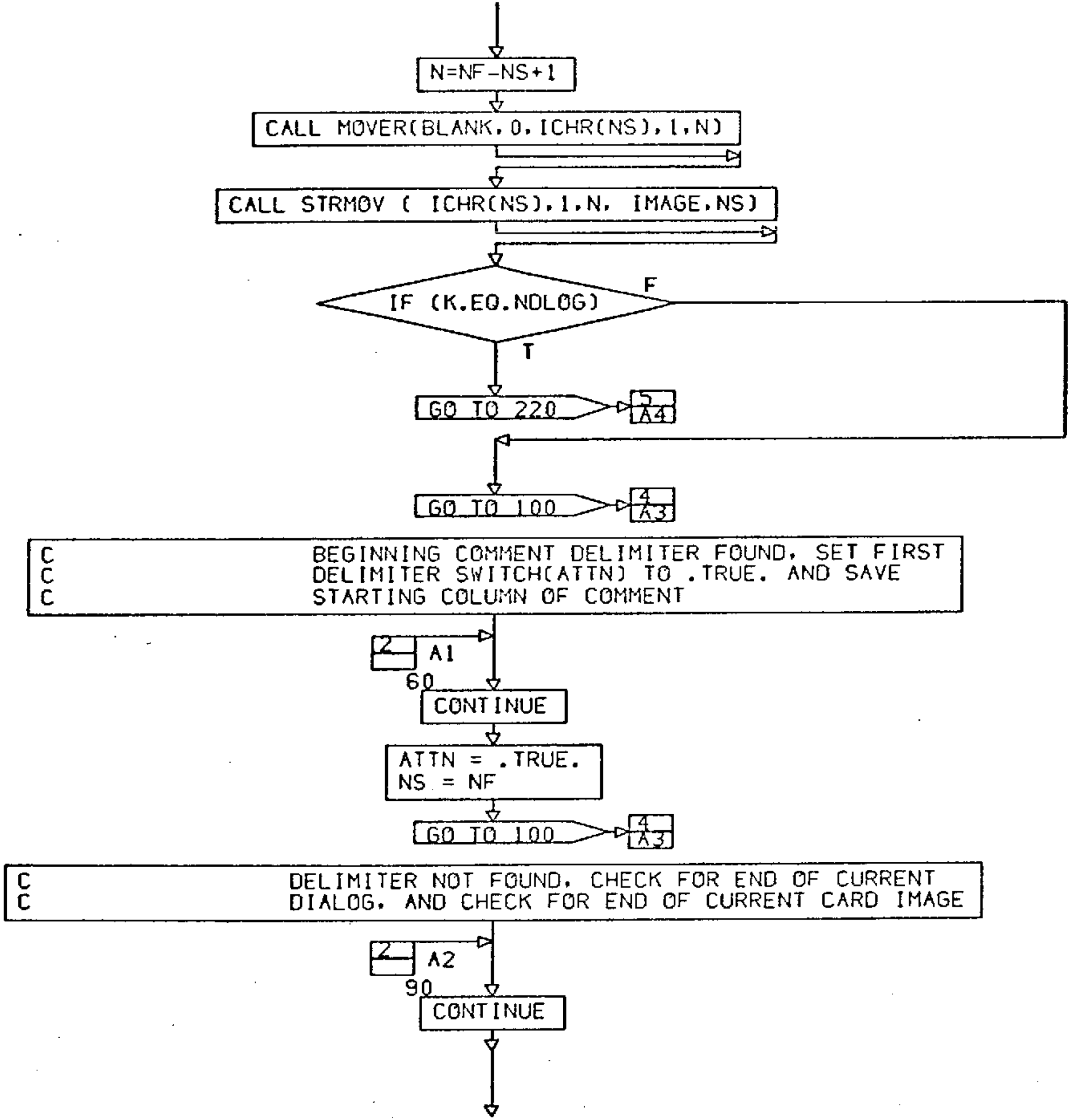


```
SUBROUTINE IGNORE
C     THIS ROUTINE BLANKS OUT THE CHARACTERS ASSOCIATED WITH COMMENT COM
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 40), ICONT )
EQUIVALENCE (IDILOG( 44), ICHR  )
EQUIVALENCE (IDILOG(184), IMAGE )
EQUIVALENCE (IDILOG(220), INRECL)
EQUIVALENCE (IDILOG(250), EDIT  )
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(338), LUO   )
EQUIVALENCE (IDILOG(357), IOCONT)
EQUIVALENCE (IDILOG(264), NWREC )
COMMON /CARDBD/ IPAG(1)
EQUIVALENCE (IPAG(1), LENPAG)
EQUIVALENCE (IPAG(3), NDLOG)
INTEGER PAGE
INTEGER DELIM, ICHR(1), BLANK, IMAGE(1)
LOGICAL SKIP, ATTN, TRACER, EDIT
DATA ATTN /.FALSE./
C***
IF (TRACER)
WRITE (LUO,1010)
1010 FORMAT (13H ENTER IGNORE )
SKIP=.FALSE.
NS=1
```

CONT. ON PG 2

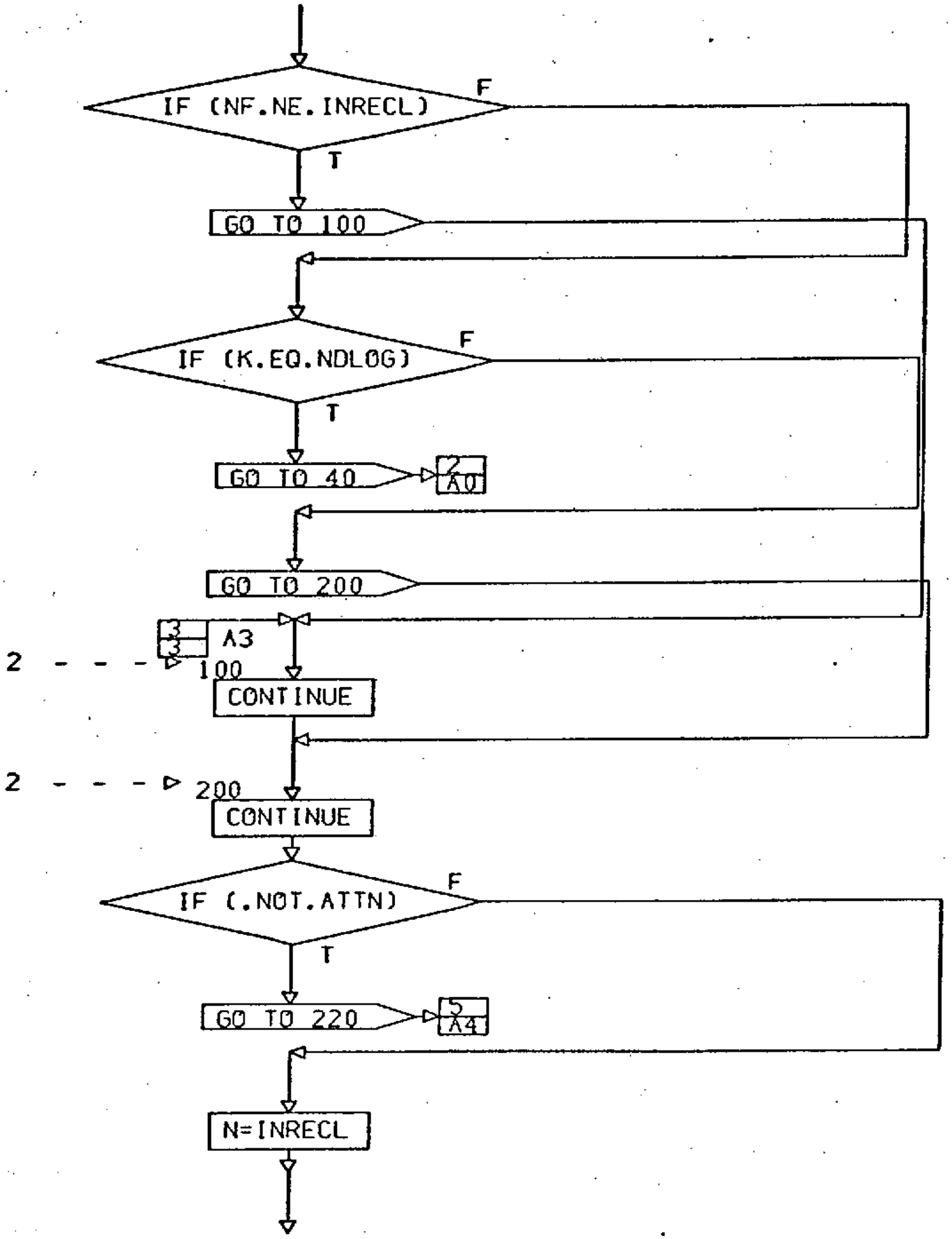

C SEARCH THRU PAGE ARRAY LOOKING FOR BEGINNING AND
C CLOSING COMMENT DELIMITERS
DO 200 K=1.NDLOG
▷ 4
NOPER = IPAG(LENPAG-K+1)
DO 100 J=1. NOPER
▷ 4
IOPR=PAGE(2.J.K)
NF = PAGE(1.J.K)
IF (IOPR.NE.DELIM)
F
T
GO TO 90
3
A2
IF (.NOT.ATTN)
F
T
GO TO 60
3
A1
C CLOSING DELIMITER FOUND. RESET FIRST DELIMITER SWITCH
ATTN = .FALSE.
C REPLACE COMMENT WITH BLANKS
A0
4
40
CONTINUE

CONT. ON PG 3

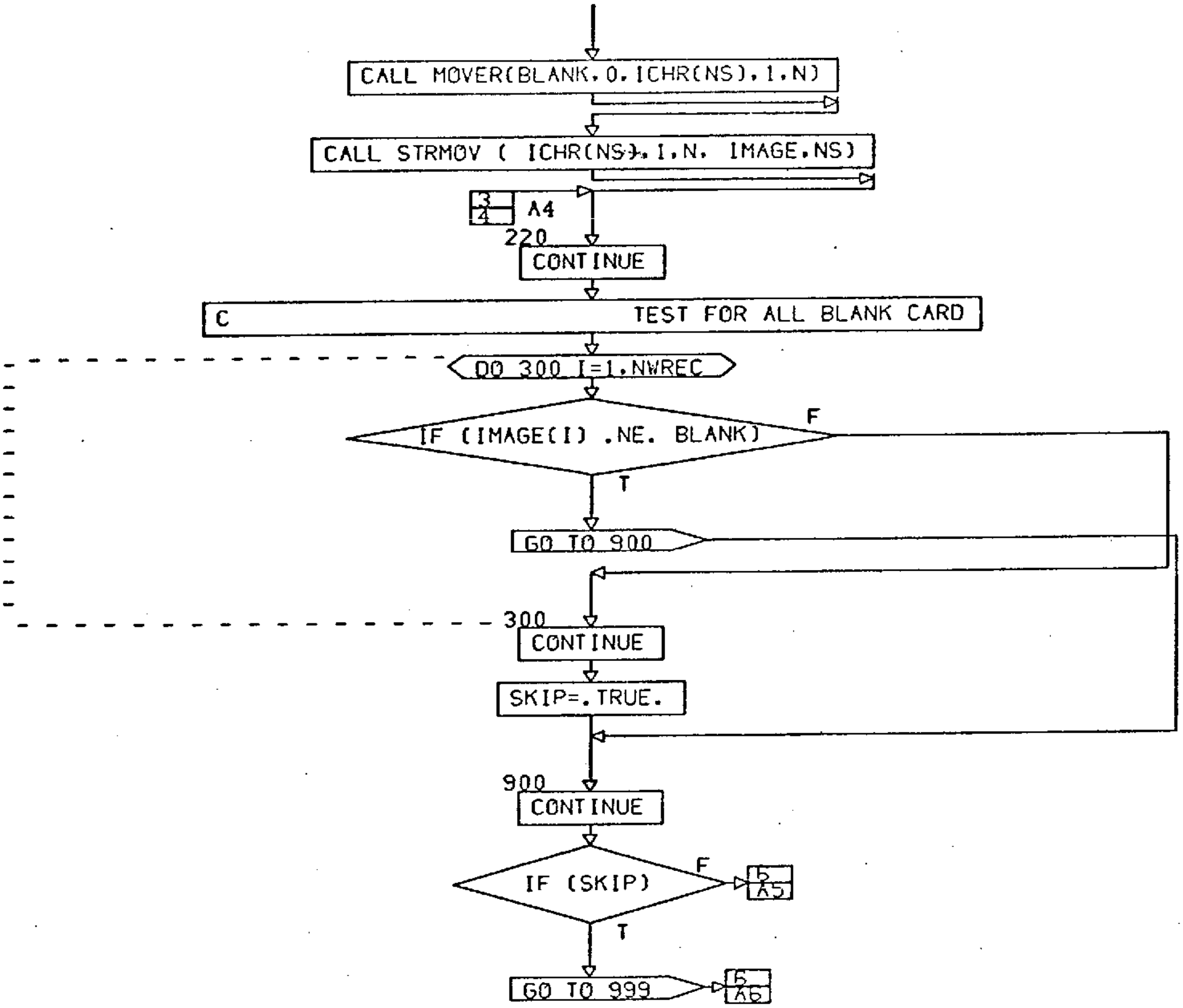

N=NF-NS+1
CALL MOVER(BLANK,0,ICHR(NS),1,N)
CALL STRMOV (ICHR(NS),1,N, IMAGE,NS)
IF (K.EQ.NDLOG)
F
T
GO TO 220
5
A4
GO TO 100
4
A3
C BEGINNING COMMENT DELIMITER FOUND, SET FIRST
C DELIMITER SWITCH(ATTN) TO .TRUE. AND SAVE
C STARTING COLUMN OF COMMENT
2
A1
60
CONTINUE
ATTN = .TRUE.
NS = NF
GO TO 100
4
A3
C DELIMITER NOT FOUND, CHECK FOR END OF CURRENT
C DIALOG, AND CHECK FOR END OF CURRENT CARD IMAGE
2
A2
90
CONTINUE

CONT. ON PG 4

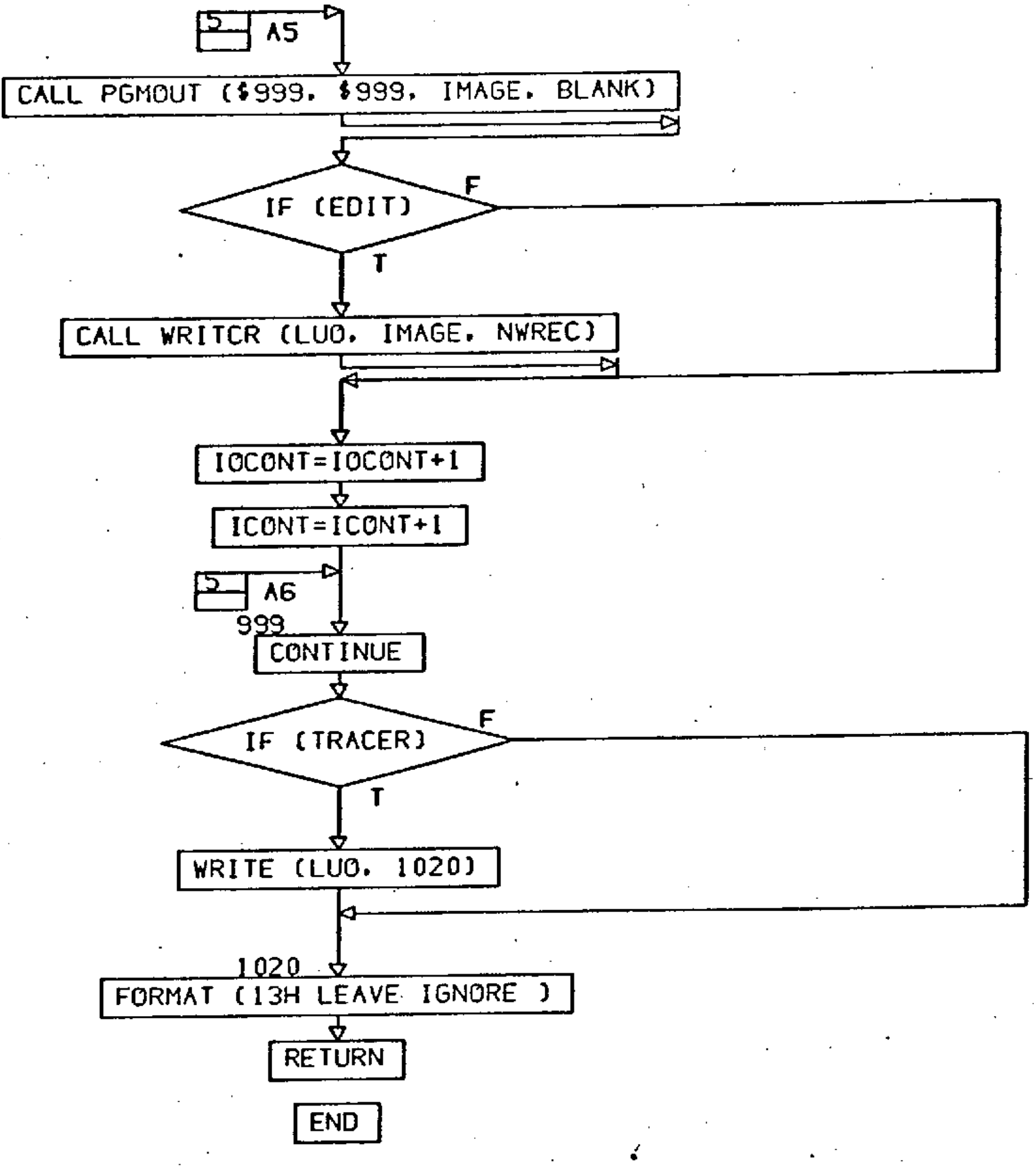
IGNORE
PG 3 OF 6


IF (NF.NE.INRECL)
F
T
GO TO 100
IF (K.EQ.NDLOG)
F
T
GO TO 40
2
A0
GO TO 200
3
3
A3
2
100
CONTINUE
2
200
CONTINUE
IF (.NOT.ATTN)
F
T
GO TO 220
5
A4
N=INRECL

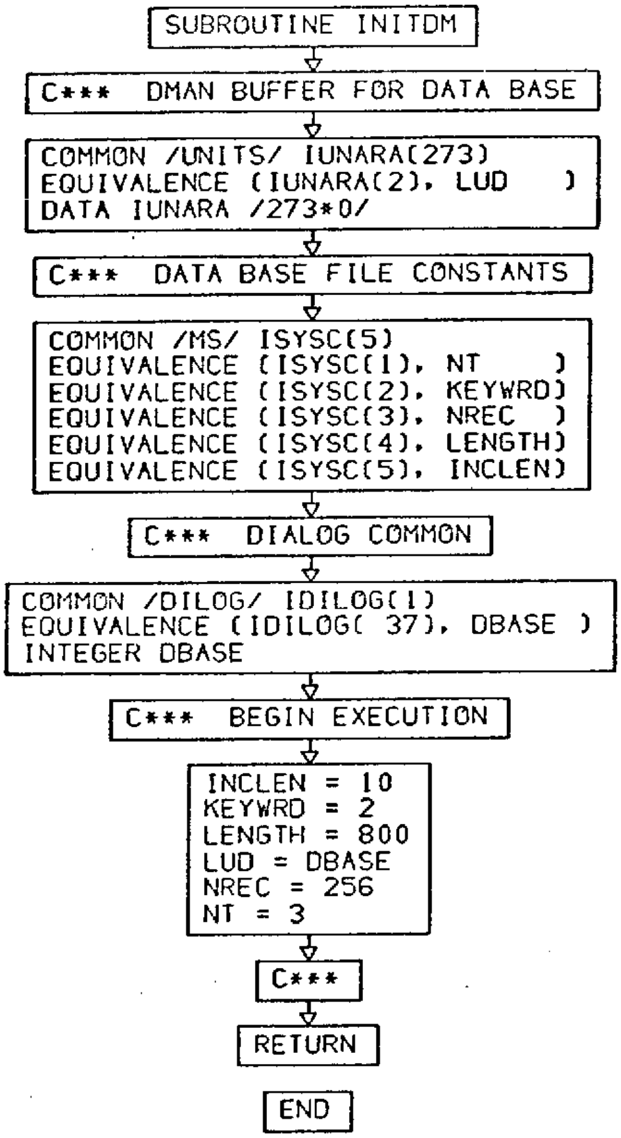
CONT. ON PG 5


CALL MOVER(BLANK,0,ICHR(NS),1,N)
CALL STRMOV (ICHR(NS~~)~~,1,N, IMAGE,NS)
3
4
A4
220
CONTINUE
C TEST FOR ALL BLANK CARD
DO 300 I=1,NWREC
IF (IMAGE(I) .NE. BLANK)
F
T
GO TO 900
300
CONTINUE
SKIP=.TRUE.
900
CONTINUE
IF (SKIP)
F
T
5
A5
GO TO 999
6
A6

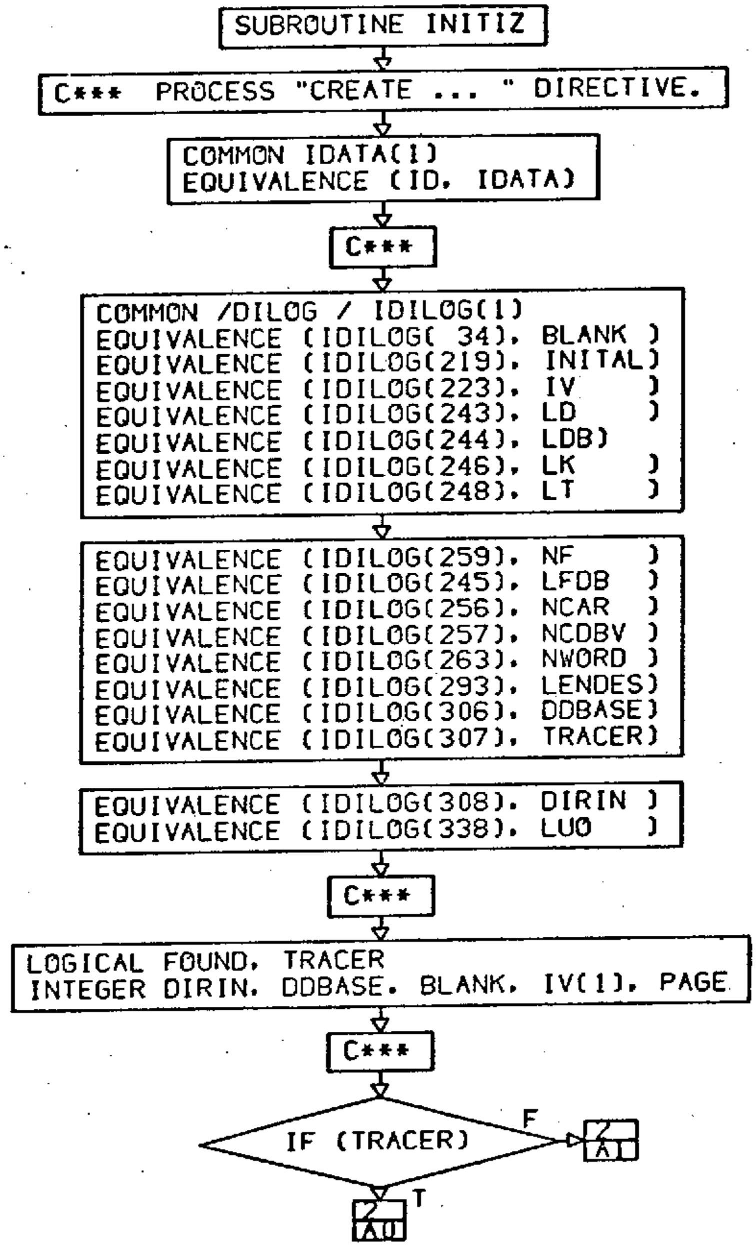
CONT. ON PG 6 .



IGNORE
PG 6 FINAL

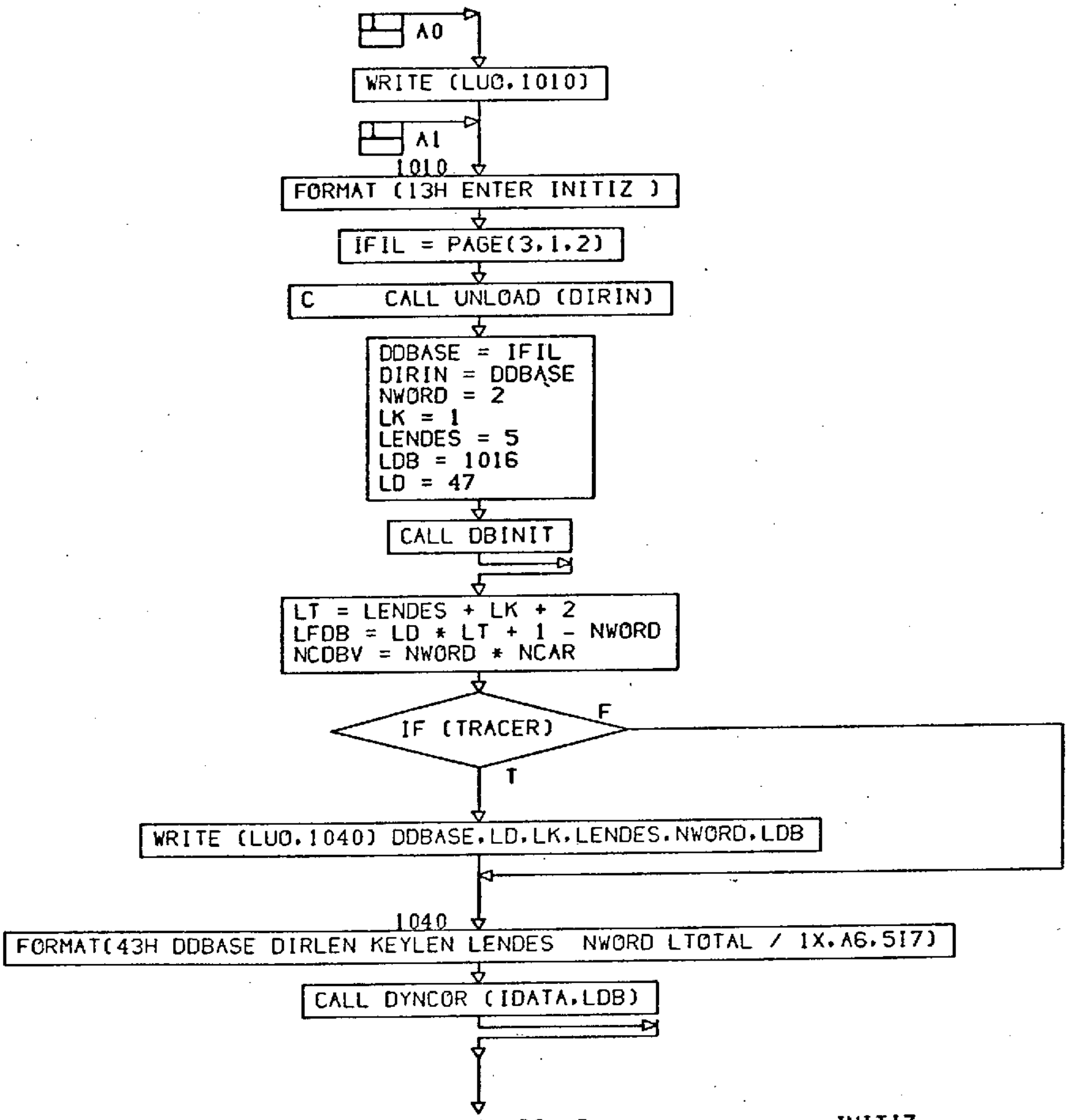


```
SUBROUTINE INITDM
C***  DMAN BUFFER FOR DATA BASE
COMMON /UNITS/ IUNARA(273)
EQUIVALENCE (IUNARA(2), LUD    )
DATA IUNARA /273*0/
C***  DATA BASE FILE CONSTANTS
COMMON /MS/ ISYSC(5)
EQUIVALENCE (ISYSC(1), NT     )
EQUIVALENCE (ISYSC(2), KEYWRD)
EQUIVALENCE (ISYSC(3), NREC   )
EQUIVALENCE (ISYSC(4), LENGTH)
EQUIVALENCE (ISYSC(5), INCLEN)
C***  DIALOG COMMON
COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG( 37), DBASE )
INTEGER DBASE
C***  BEGIN EXECUTION
INCLEN = 10
KEYWRD = 2
LENGTH = 800
LUD = DBASE
NREC = 256
NT = 3
C***
RETURN
END
```

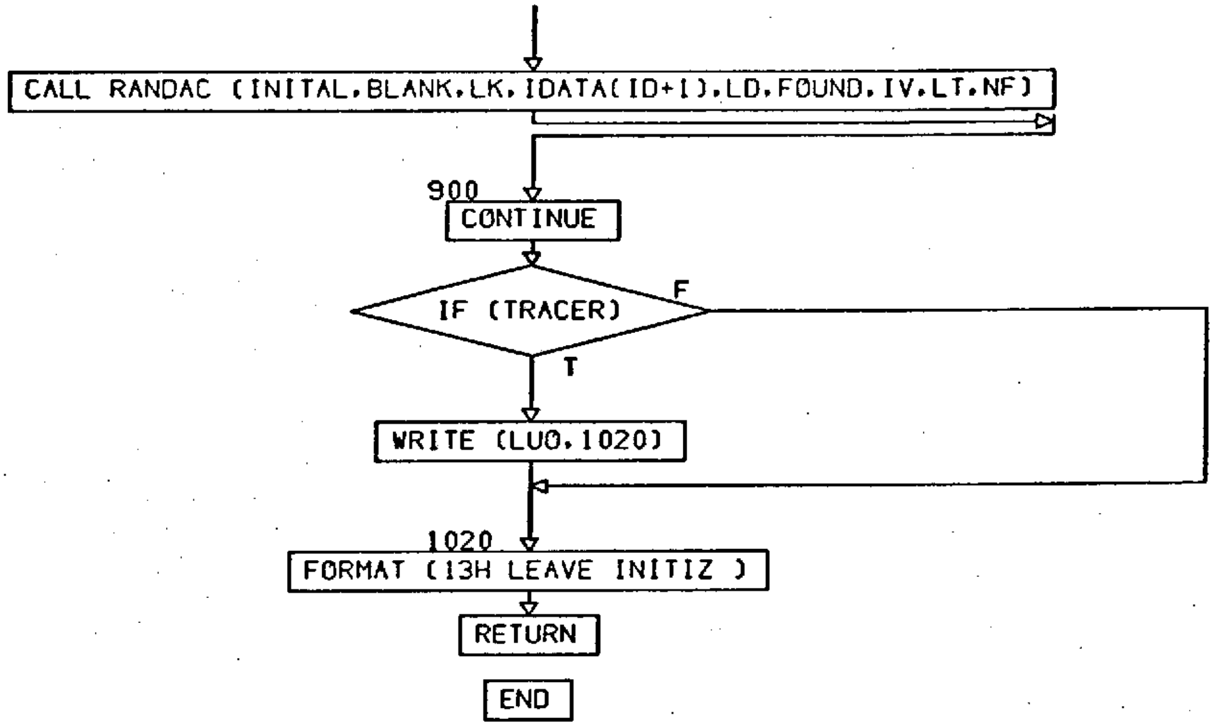


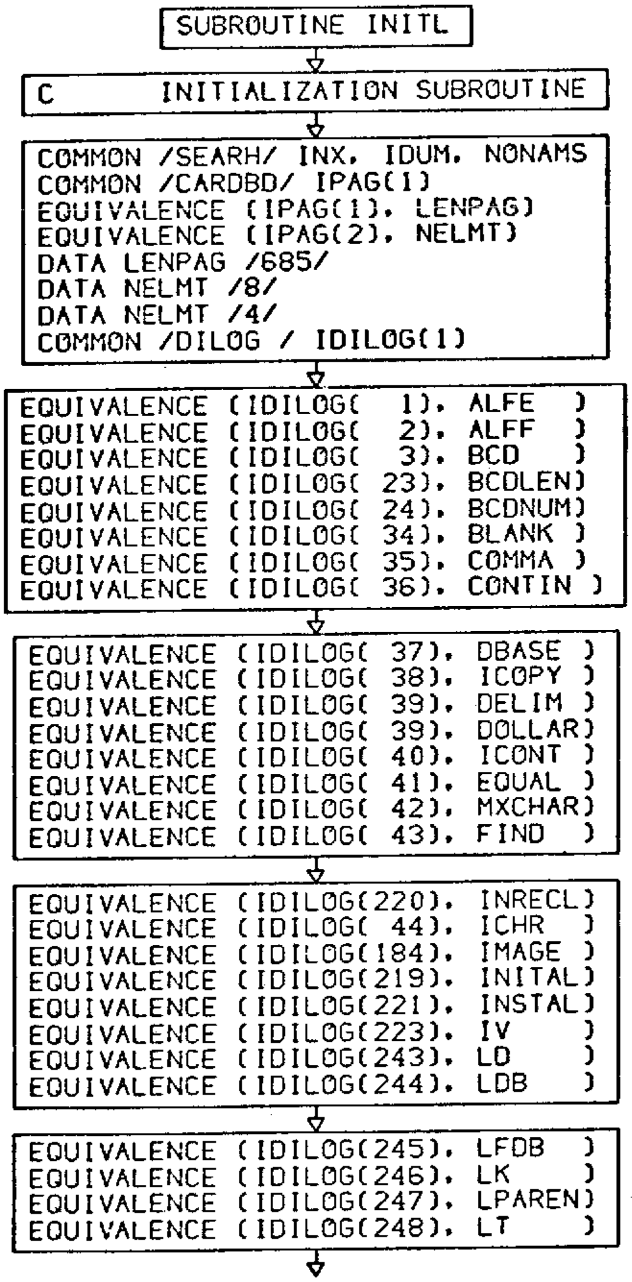
```
SUBROUTINE INITIZ
C***  PROCESS "CREATE ... " DIRECTIVE.
COMMON IDATA(1)
EQUIVALENCE (ID, IDATA)
C***
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG(219), INITAL)
EQUIVALENCE (IDILOG(223), IV    )
EQUIVALENCE (IDILOG(243), LD    )
EQUIVALENCE (IDILOG(244), LDB)
EQUIVALENCE (IDILOG(246), LK    )
EQUIVALENCE (IDILOG(248), LT    )
EQUIVALENCE (IDILOG(259), NF    )
EQUIVALENCE (IDILOG(245), LFDB  )
EQUIVALENCE (IDILOG(256), NCAR  )
EQUIVALENCE (IDILOG(257), NCDBV )
EQUIVALENCE (IDILOG(263), NWORD )
EQUIVALENCE (IDILOG(293), LENDES)
EQUIVALENCE (IDILOG(306), DDBASE)
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(308), DIRIN )
EQUIVALENCE (IDILOG(338), LUO   )
C***
LOGICAL FOUND, TRACER
INTEGER DIRIN, DDBASE, BLANK, IV(1), PAGE
C***
IF (TRACER)
```

CONT. ON PG 2


A0
WRITE (LUO.1010)
A1
1010
FORMAT (13H ENTER INITIZ)
IFIL = PAGE(3.1.2)
C CALL UNLOAD (DIRIN)
DDBASE = IFIL
DIRIN = DDBASE
NWORD = 2
LK = 1
LENDES = 5
LDB = 1016
LD = 47
CALL DBINIT
LT = LENDES + LK + 2
LFDB = LD * LT + 1 - NWORD
NCDBV = NWORD * NCAR
IF (TRACER)
F
T
WRITE (LUO.1040) DDBASE.LD.LK.LENDES.NWORD.LDB
1040
FORMAT(43H DDBASE DIRLEN KEYLEN LENDES NWORD LTOTAL / 1X.A6.5I7)
CALL DYNCOR (IDATA.LDB)

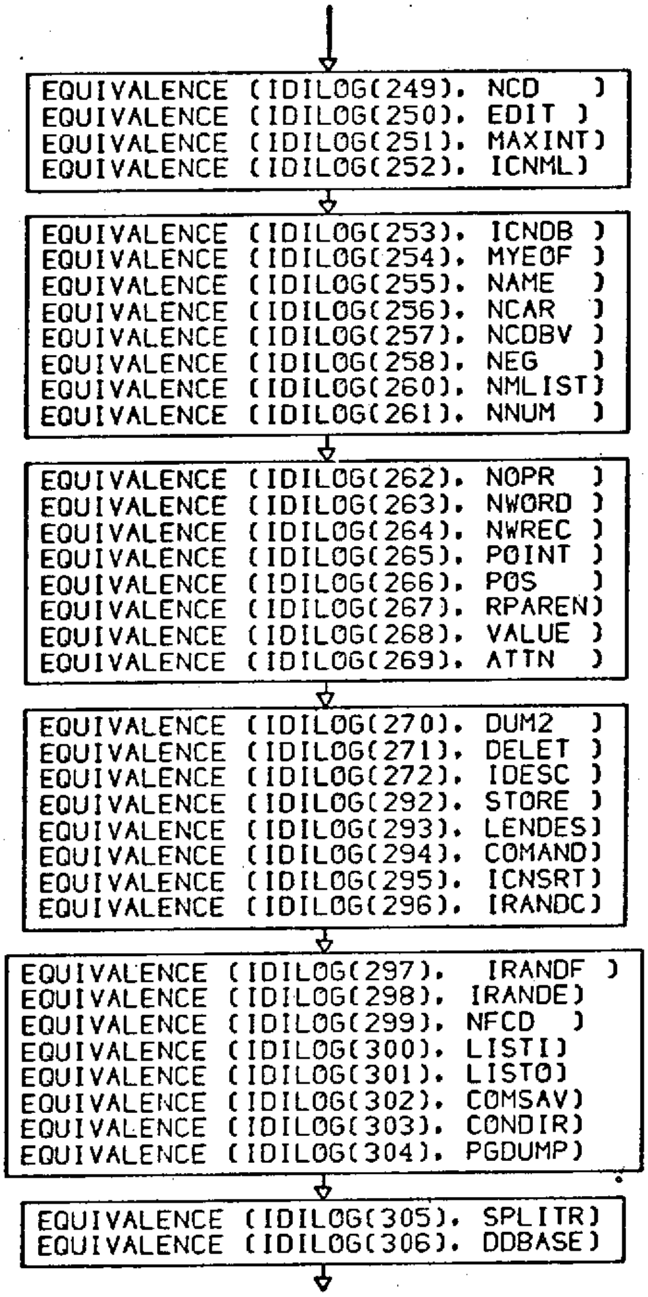
CONT. ON PG 3


CALL RANDAC (INITAL,BLANK,LK,IDATA(ID+I),LD,FOUND,IV,LT,NF)
900
CONTINUE
IF (TRACER)
F
T
WRITE (LUO,1020)
1020
FORMAT (13H LEAVE INITIZ)
RETURN
END



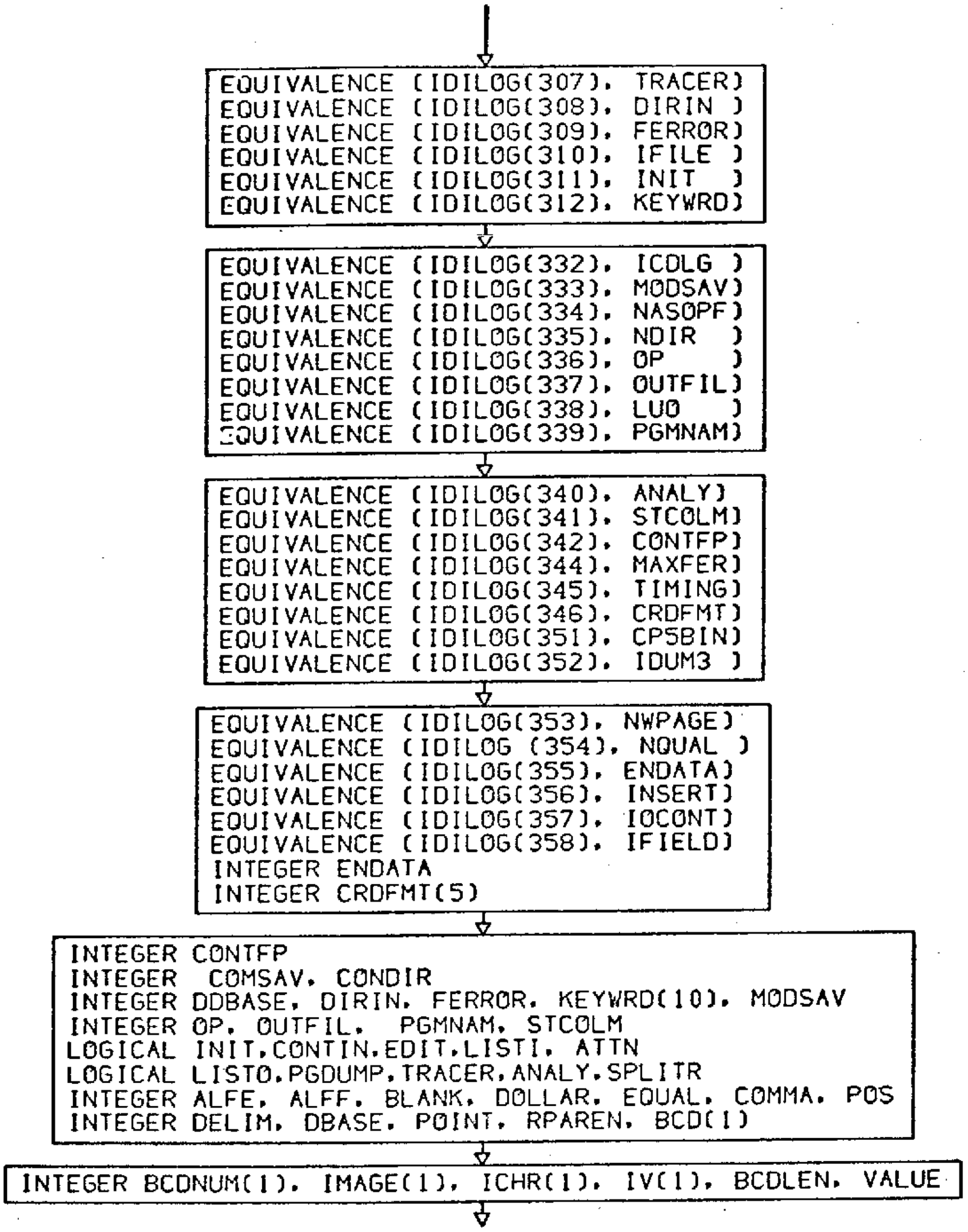
```
SUBROUTINE INITL

C       INITIALIZATION SUBROUTINE

COMMON /SEARH/ INX, IDUM, NONAMS
COMMON /CARDBD/ IPAG(1)
EQUIVALENCE (IPAG(1), LENPAG)
EQUIVALENCE (IPAG(2), NELMT)
DATA LENPAG /685/
DATA NELMT /8/
DATA NELMT /4/
COMMON /DILOG / IDILOG(1)

EQUIVALENCE (IDILOG(  1), ALFE  )
EQUIVALENCE (IDILOG(  2), ALFF  )
EQUIVALENCE (IDILOG(  3), BCD   )
EQUIVALENCE (IDILOG( 23), BCDLEN)
EQUIVALENCE (IDILOG( 24), BCDNUM)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 35), COMMA )
EQUIVALENCE (IDILOG( 36), CONTIN )

EQUIVALENCE (IDILOG( 37), DBASE )
EQUIVALENCE (IDILOG( 38), ICOPY )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 39), DOLLAR)
EQUIVALENCE (IDILOG( 40), ICONT )
EQUIVALENCE (IDILOG( 41), EQUAL )
EQUIVALENCE (IDILOG( 42), MXCHAR)
EQUIVALENCE (IDILOG( 43), FIND  )

EQUIVALENCE (IDILOG(220), INRECL)
EQUIVALENCE (IDILOG( 44), ICHR  )
EQUIVALENCE (IDILOG(184), IMAGE )
EQUIVALENCE (IDILOG(219), INITAL)
EQUIVALENCE (IDILOG(221), INSTAL)
EQUIVALENCE (IDILOG(223), IV    )
EQUIVALENCE (IDILOG(243), LD    )
EQUIVALENCE (IDILOG(244), LDB   )

EQUIVALENCE (IDILOG(245), LFDB  )
EQUIVALENCE (IDILOG(246), LK    )
EQUIVALENCE (IDILOG(247), LPAREN)
EQUIVALENCE (IDILOG(248), LT    )
```

CONT. ON PG 2



```
EQUIVALENCE (IDILOG(249). NCD    )
EQUIVALENCE (IDILOG(250). EDIT )
EQUIVALENCE (IDILOG(251). MAXINT)
EQUIVALENCE (IDILOG(252). ICNML)

EQUIVALENCE (IDILOG(253). ICNDB )
EQUIVALENCE (IDILOG(254). MYEOF )
EQUIVALENCE (IDILOG(255). NAME  )
EQUIVALENCE (IDILOG(256). NCAR  )
EQUIVALENCE (IDILOG(257). NCDBV )
EQUIVALENCE (IDILOG(258). NEG   )
EQUIVALENCE (IDILOG(260). NMLIST)
EQUIVALENCE (IDILOG(261). NNUM  )

EQUIVALENCE (IDILOG(262). NOPR  )
EQUIVALENCE (IDILOG(263). NWORD )
EQUIVALENCE (IDILOG(264). NWREC )
EQUIVALENCE (IDILOG(265). POINT )
EQUIVALENCE (IDILOG(266). POS   )
EQUIVALENCE (IDILOG(267). RPAREN)
EQUIVALENCE (IDILOG(268). VALUE )
EQUIVALENCE (IDILOG(269). ATTN  )

EQUIVALENCE (IDILOG(270). DUM2  )
EQUIVALENCE (IDILOG(271). DELET )
EQUIVALENCE (IDILOG(272). IDESC )
EQUIVALENCE (IDILOG(292). STORE )
EQUIVALENCE (IDILOG(293). LENDES)
EQUIVALENCE (IDILOG(294). COMAND)
EQUIVALENCE (IDILOG(295). ICNSRT)
EQUIVALENCE (IDILOG(296). IRANDC)

EQUIVALENCE (IDILOG(297).  IRANDF )
EQUIVALENCE (IDILOG(298). IRANDE)
EQUIVALENCE (IDILOG(299). NFCD   )
EQUIVALENCE (IDILOG(300). LISTI)
EQUIVALENCE (IDILOG(301). LISTO)
EQUIVALENCE (IDILOG(302). COMSAV)
EQUIVALENCE (IDILOG(303). CONDIR)
EQUIVALENCE (IDILOG(304). PGDUMP)

EQUIVALENCE (IDILOG(305). SPLITR)
EQUIVALENCE (IDILOG(306). DDBASE)
```

CONT. ON PG 3



```
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(308), DIRIN )
EQUIVALENCE (IDILOG(309), FERROR)
EQUIVALENCE (IDILOG(310), IFILE )
EQUIVALENCE (IDILOG(311), INIT   )
EQUIVALENCE (IDILOG(312), KEYWRD)
```

```
EQUIVALENCE (IDILOG(332), ICOLG )
EQUIVALENCE (IDILOG(333), MODSAV)
EQUIVALENCE (IDILOG(334), NASOPF)
EQUIVALENCE (IDILOG(335), NDIR   )
EQUIVALENCE (IDILOG(336), OP      )
EQUIVALENCE (IDILOG(337), OUTFIL)
EQUIVALENCE (IDILOG(338), LUO     )
EQUIVALENCE (IDILOG(339), PGMNAM)
```

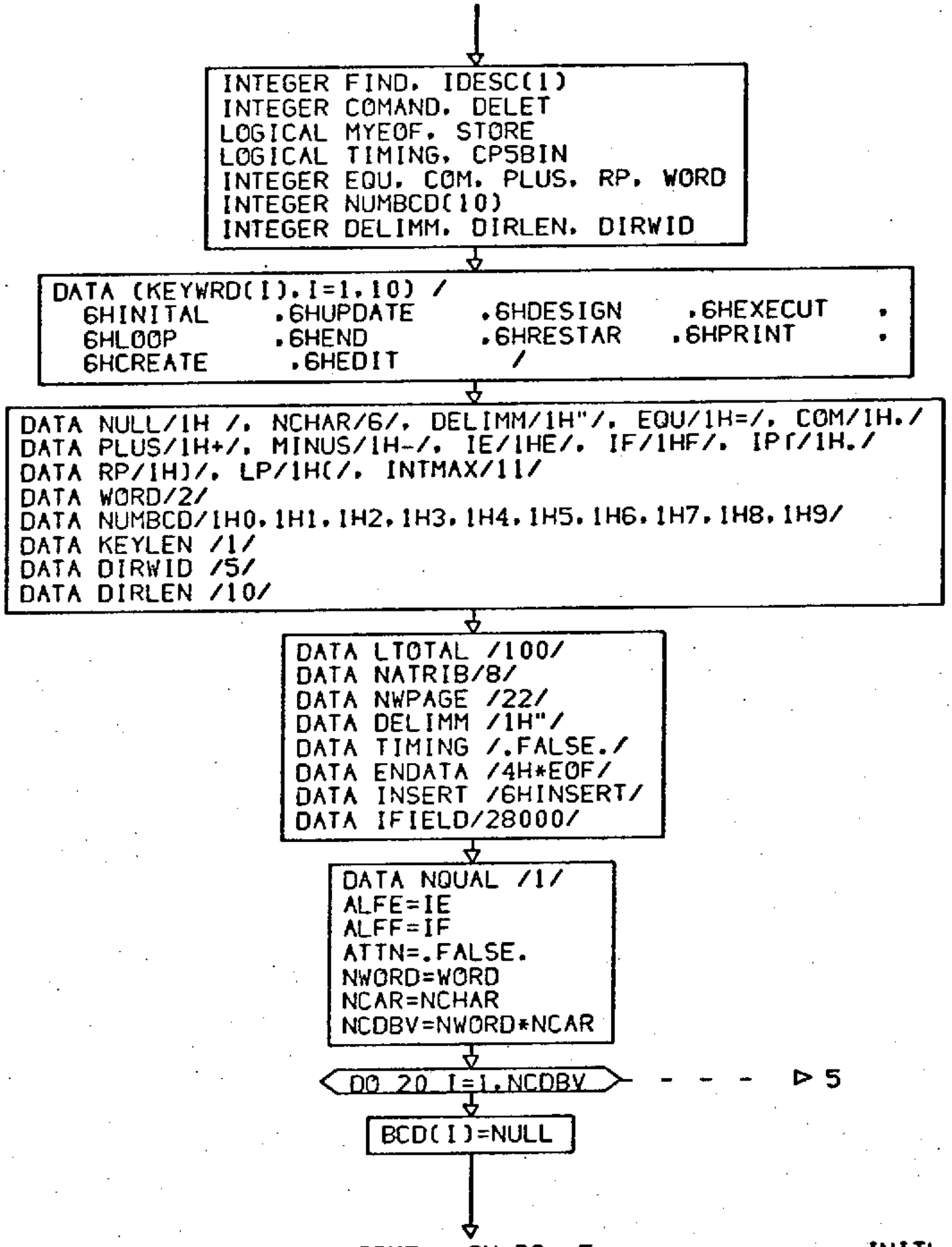
```
EQUIVALENCE (IDILOG(340), ANALY)
EQUIVALENCE (IDILOG(341), STCOLM)
EQUIVALENCE (IDILOG(342), CONTFP)
EQUIVALENCE (IDILOG(344), MAXFER)
EQUIVALENCE (IDILOG(345), TIMING)
EQUIVALENCE (IDILOG(346), CRDFMT)
EQUIVALENCE (IDILOG(351), CP5BIN)
EQUIVALENCE (IDILOG(352), IDUM3 )
```

```
EQUIVALENCE (IDILOG(353), NWPAGE)
EQUIVALENCE (IDILOG (354), NQUAL )
EQUIVALENCE (IDILOG(355), ENDATA)
EQUIVALENCE (IDILOG(356), INSERT)
EQUIVALENCE (IDILOG(357), IOCONT)
EQUIVALENCE (IDILOG(358), IFIELD)
INTEGER ENDATA
INTEGER CRDFMT(5)
```

```
 INTEGER CONTFP
 INTEGER  COMSAV, CONDIR
 INTEGER DDBASE, DIRIN, FERROR, KEYWRD(10), MODSAV
 INTEGER OP, OUTFIL,  PGMNAM, STCOLM
LOGICAL INIT,CONTIN,EDIT,LISTI, ATTN
LOGICAL LISTO,PGDUMP,TRACER,ANALY,SPLITR
 INTEGER ALFE, ALFF, BLANK, DOLLAR, EQUAL, COMMA, POS
 INTEGER DELIM, DBASE, POINT, RPAREN, BCD(1)
```

```
INTEGER BCDNUM(1), IMAGE(1), ICHR(1), IV(1), BCDLEN, VALUE
```

CONT. ON PG 4



```
INTEGER FIND, IDESC(1)
INTEGER COMAND, DELET
LOGICAL MYEOF, STORE
LOGICAL TIMING, CP5BIN
INTEGER EQU, COM, PLUS, RP, WORD
INTEGER NUMBCD(10)
INTEGER DELIMM, DIRLEN, DIRWID
```

```
DATA (KEYWRD(I),I=1,10) /
  6HINITAL    ,6HUPDATE    ,6HDESIGN    ,6HEXECUT   ,
  6HLOOP      ,6HEND       ,6HRESTAR   ,6HPRINT     ,
  6HCREATE     ,6HEDIT        /
```

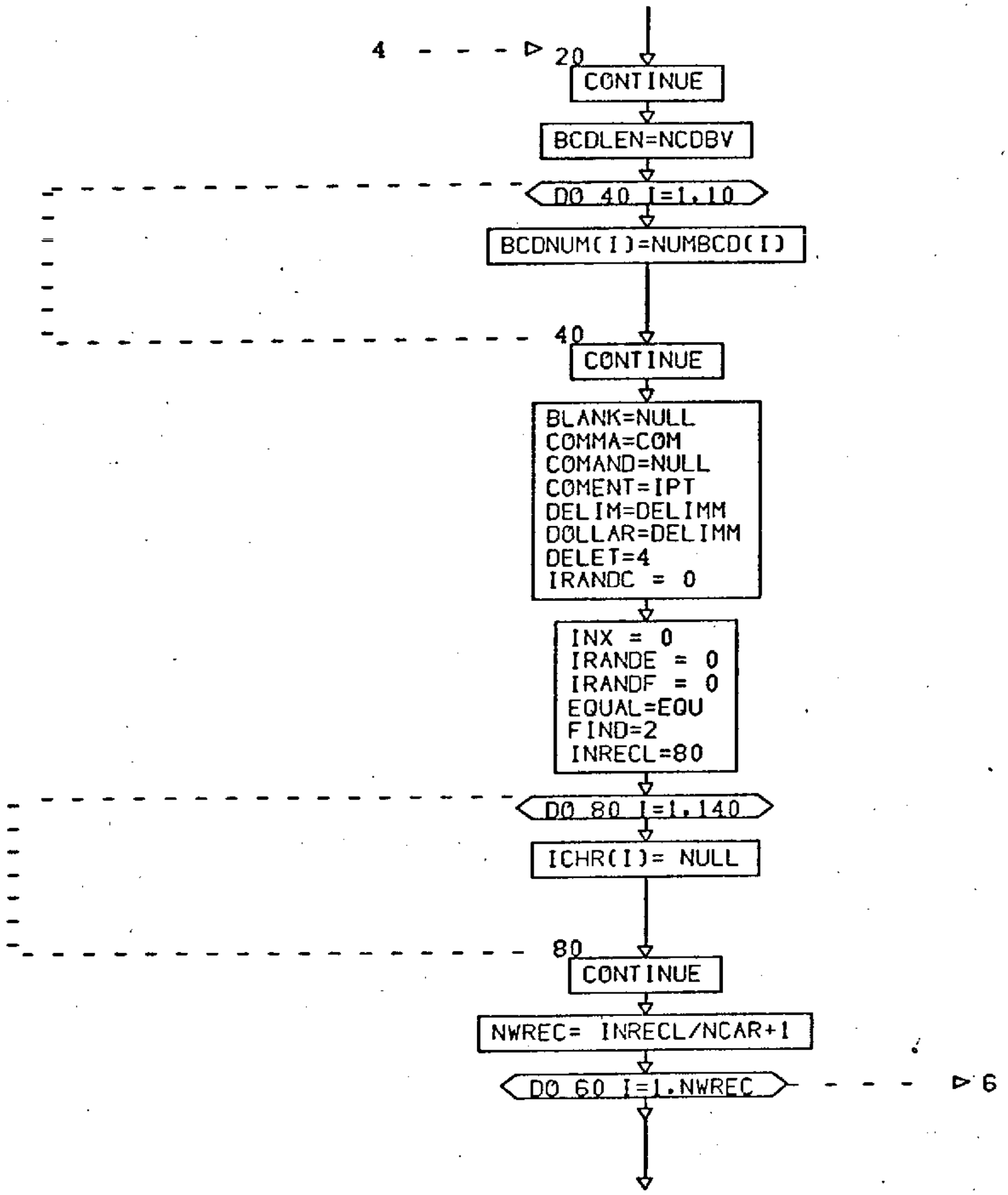
```
DATA NULL/1H /, NCHAR/6/, DELIMM/1H"/, EQU/1H=/, COM/1H,/
DATA PLUS/1H+/, MINUS/1H-/, IE/1HE/, IF/1HF/, IPT/1H./
DATA RP/1H)/, LP/1H(/, INTMAX/11/
DATA WORD/2/
DATA NUMBCD/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
DATA KEYLEN /1/
DATA DIRWID /5/
DATA DIRLEN /10/
```

```
DATA LTOTAL /100/
DATA NATRIB/8/
DATA NWPAGE /22/
DATA DELIMM /1H"/
DATA TIMING /.FALSE./
DATA ENDATA /4H*EOF/
DATA INSERT /6HINSERT/
DATA IFIELD/28000/
```

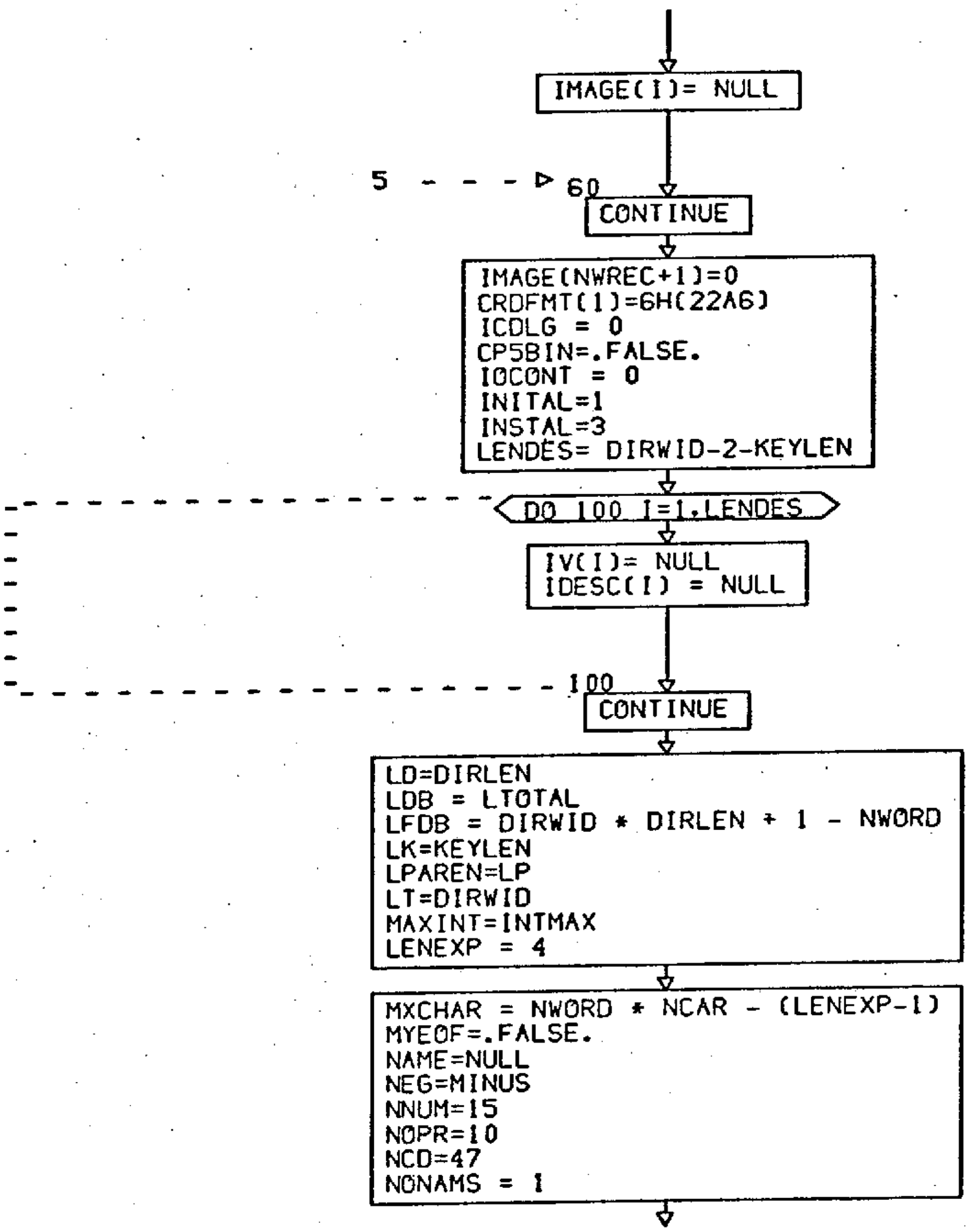
```
DATA NQUAL /1/
ALFE=IE
ALFF=IF
ATTN=.FALSE.
NWORD=WORD
NCAR=NCHAR
NCDBV=NWORD*NCAR
```

```
DO 20 I=1,NCDBV  - - - ▷ 5
BCD(I)=NULL
```

CONT. ON PG 5

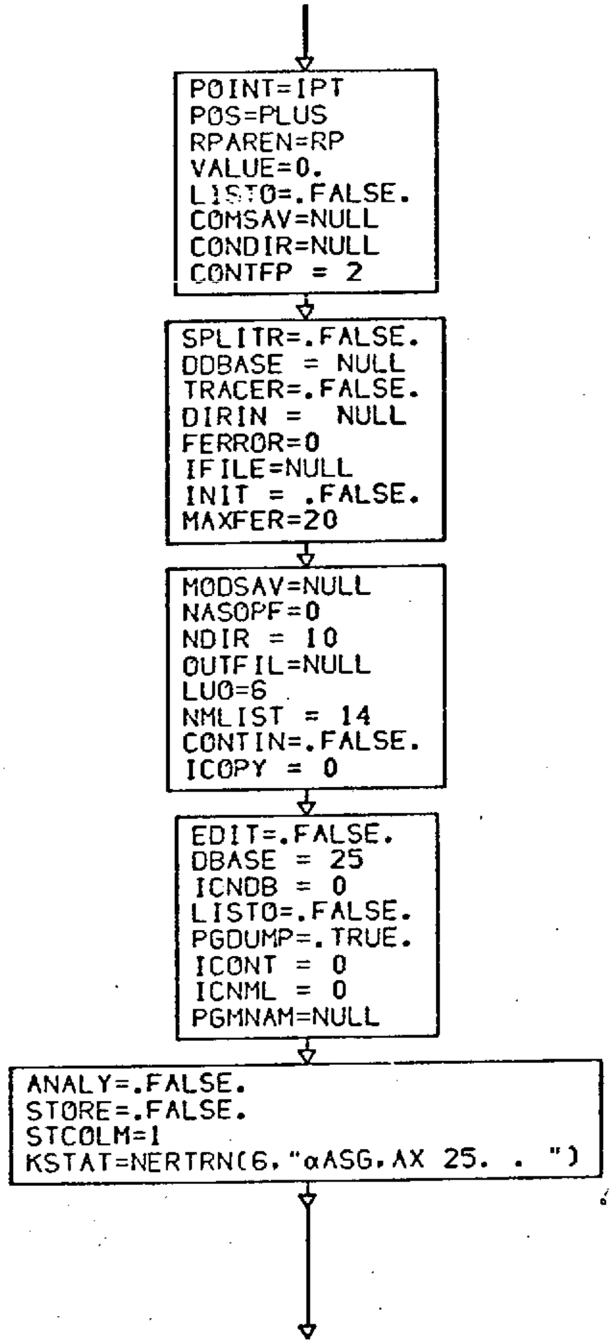

4
20
CONTINUE
BCDLEN=NCDBV
DO 40 I=1,10
BCDNUM(I)=NUMBCD(I)
40
CONTINUE
BLANK=NULL
COMMA=COM
COMAND=NULL
COMENT=IPT
DELIM=DELIMM
DOLLAR=DELIMM
DELET=4
IRANDC = 0
INX = 0
IRANDE = 0
IRANDF = 0
EQUAL=EQU
FIND=2
INRECL=80
DO 80 I=1,140
ICHR(I)= NULL
80
CONTINUE
NWREC= INRECL/NCAR+1
DO 60 I=1,NWREC
6

CONT. ON PG 6



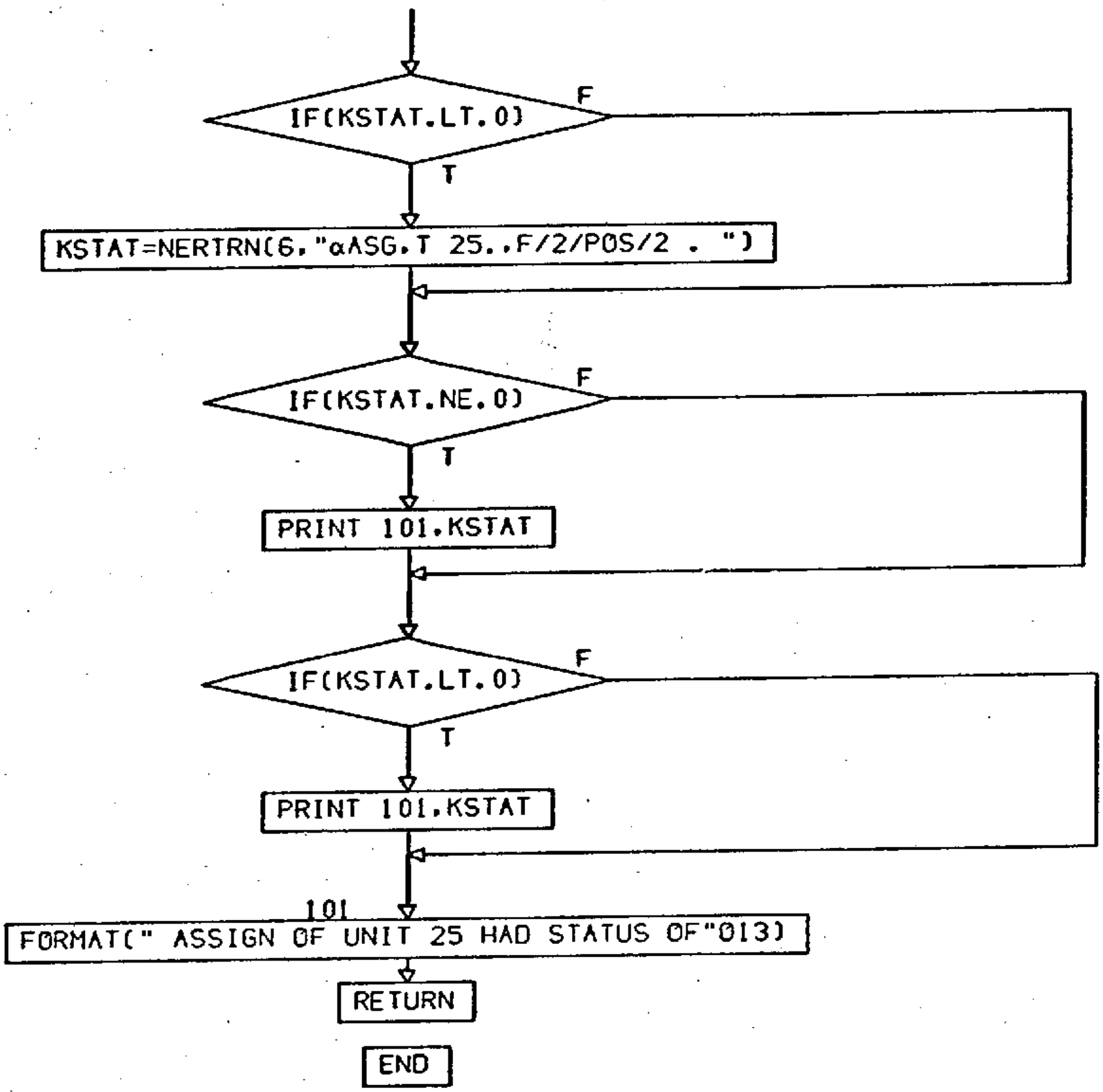
```
      IMAGE(I)= NULL

5 - - - -> 60
      CONTINUE

      IMAGE(NWREC+1)=0
      CRDFMT(1)=6H(22A6)
      ICDLG = 0
      CP5BIN=.FALSE.
      IOCONT = 0
      INITAL=1
      INSTAL=3
      LENDES= DIRWID-2-KEYLEN

      DO 100 I=1,LENDES

      IV(I)= NULL
      IDESC(I) = NULL

100
      CONTINUE

      LD=DIRLEN
      LDB = LTOTAL
      LFDB = DIRWID * DIRLEN + 1 - NWORD
      LK=KEYLEN
      LPAREN=LP
      LT=DIRWID
      MAXINT=INTMAX
      LENEXP = 4

      MXCHAR = NWORD * NCAR - (LENEXP-1)
      MYEOF=.FALSE.
      NAME=NULL
      NEG=MINUS
      NNUM=15
      NOPR=10
      NCD=47
      NONAMS = 1
```

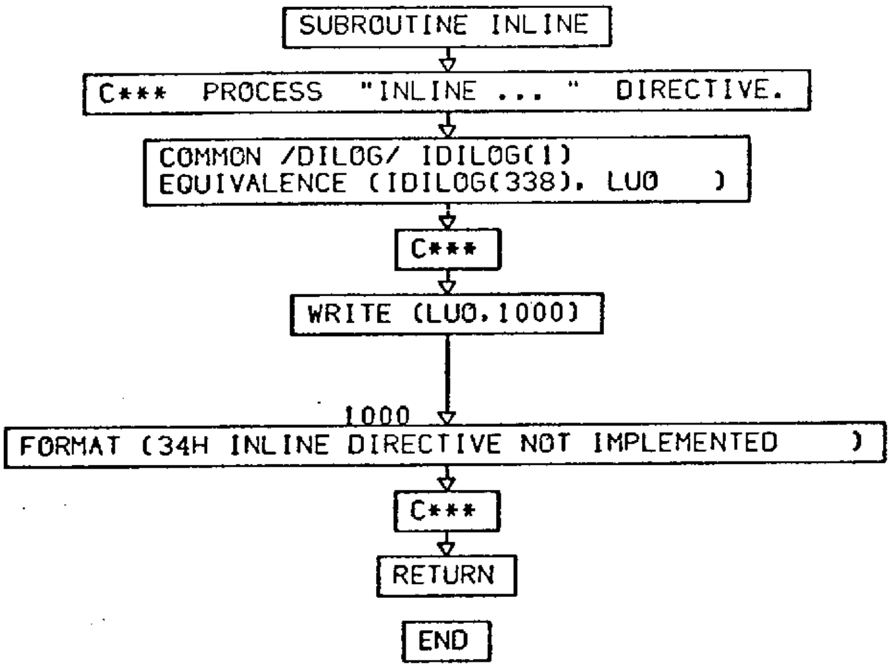
CONT. ON PG 7



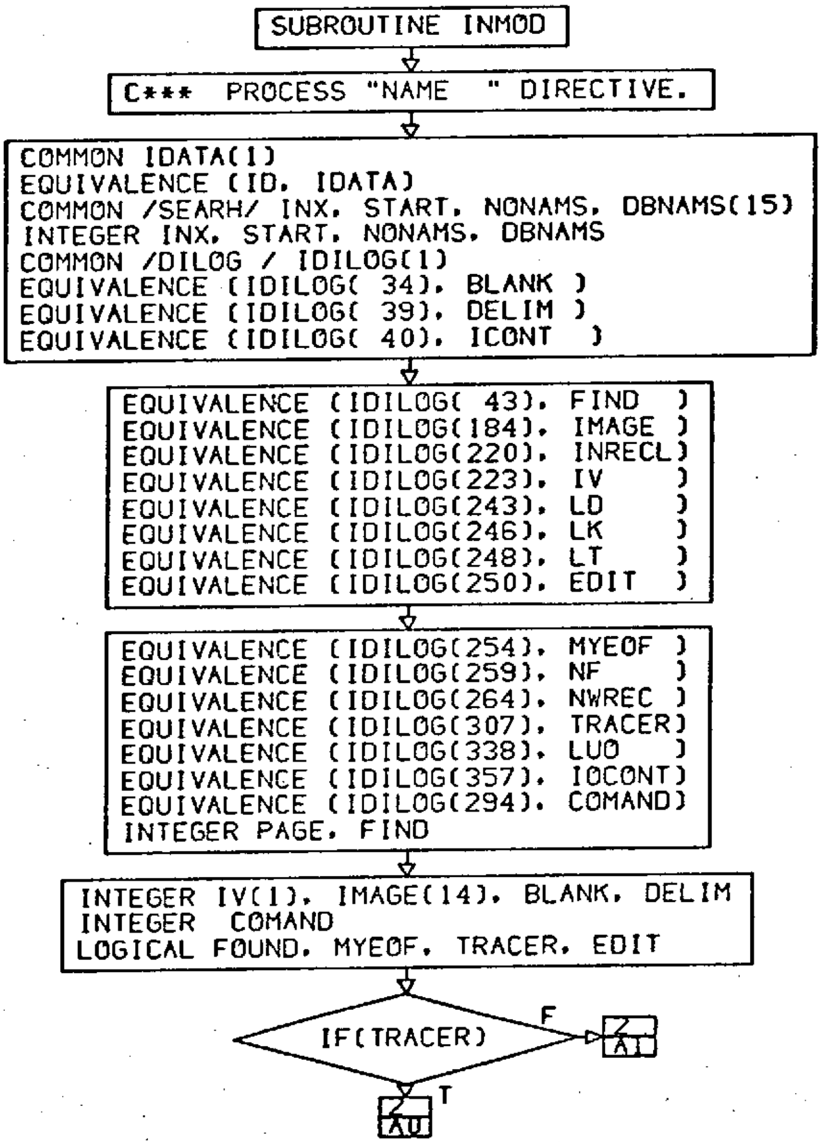
```
POINT=IPT
POS=PLUS
RPAREN=RP
VALUE=0.
LISTO=.FALSE.
COMSAV=NULL
CONDIR=NULL
CONTFP = 2

SPLITR=.FALSE.
DDBASE = NULL
TRACER=.FALSE.
DIRIN =  NULL
FERROR=0
IFILE=NULL
INIT = .FALSE.
MAXFER=20

MODSAV=NULL
NASOPF=0
NDIR = 10
OUTFIL=NULL
LUO=6
NMLIST = 14
CONTIN=.FALSE.
ICOPY = 0

EDIT=.FALSE.
DBASE = 25
ICNDB = 0
LISTO=.FALSE.
PGDUMP=.TRUE.
ICONT = 0
ICNML = 0
PGMNAM=NULL

ANALY=.FALSE.
STORE=.FALSE.
STCOLM=1
KSTAT=NERTRN(6,"αASG,AX 25. . ")
```

CONT. ON PG 8


IF(KSTAT.LT.0)
F
T
KSTAT=NERTRN(6,"αASG.T 25.,F/2/POS/2 . ")
IF(KSTAT.NE.0)
F
T
PRINT 101,KSTAT
IF(KSTAT.LT.0)
F
T
PRINT 101,KSTAT
101
FORMAT(" ASSIGN OF UNIT 25 HAD STATUS OF"O13)
RETURN
END

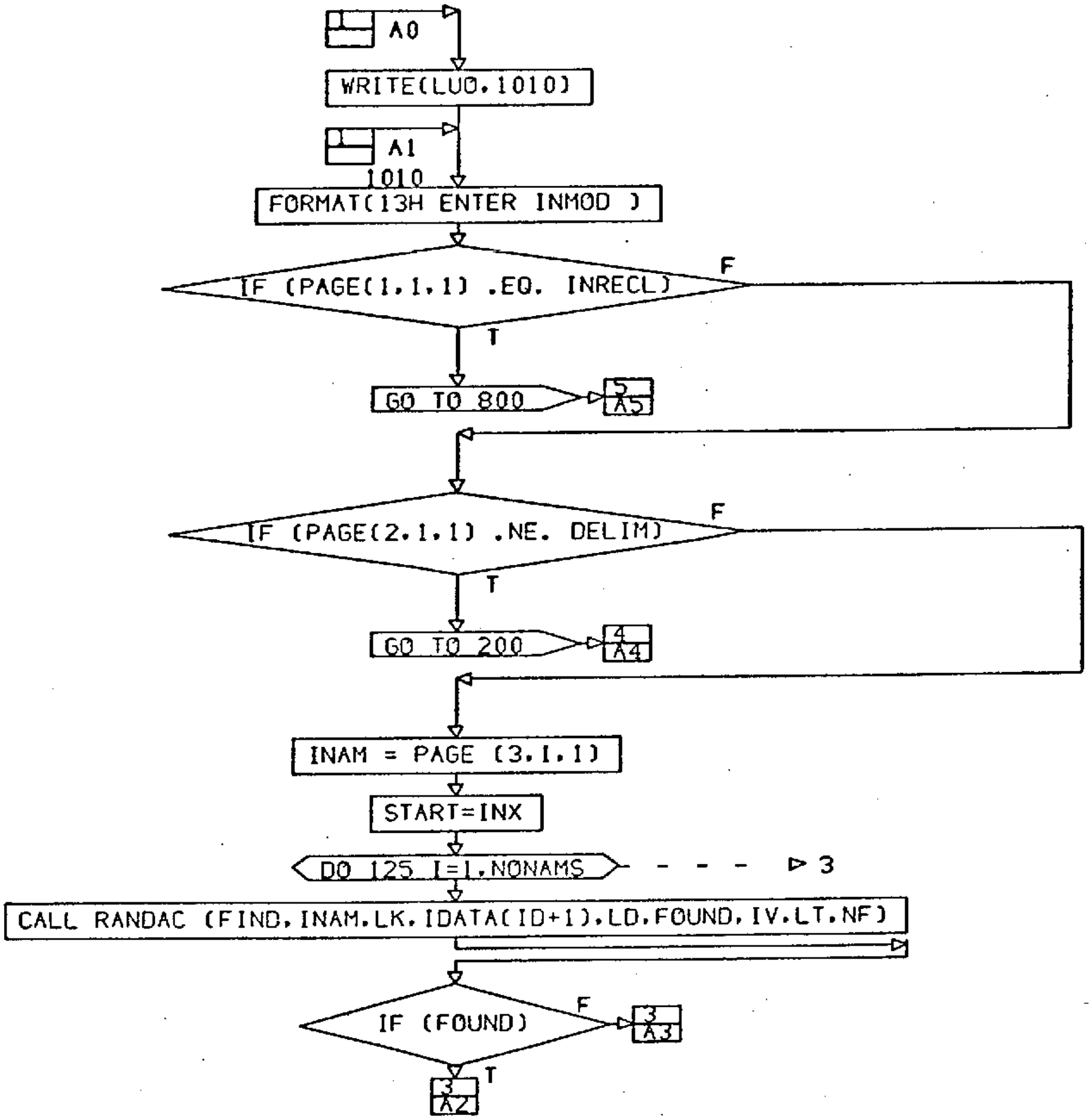


```
SUBROUTINE INLINE
C***  PROCESS  "INLINE ... "  DIRECTIVE.
COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG(338), LUO    )
C***
WRITE (LUO,1000)
1000
FORMAT (34H INLINE DIRECTIVE NOT IMPLEMENTED      )
C***
RETURN
END
```

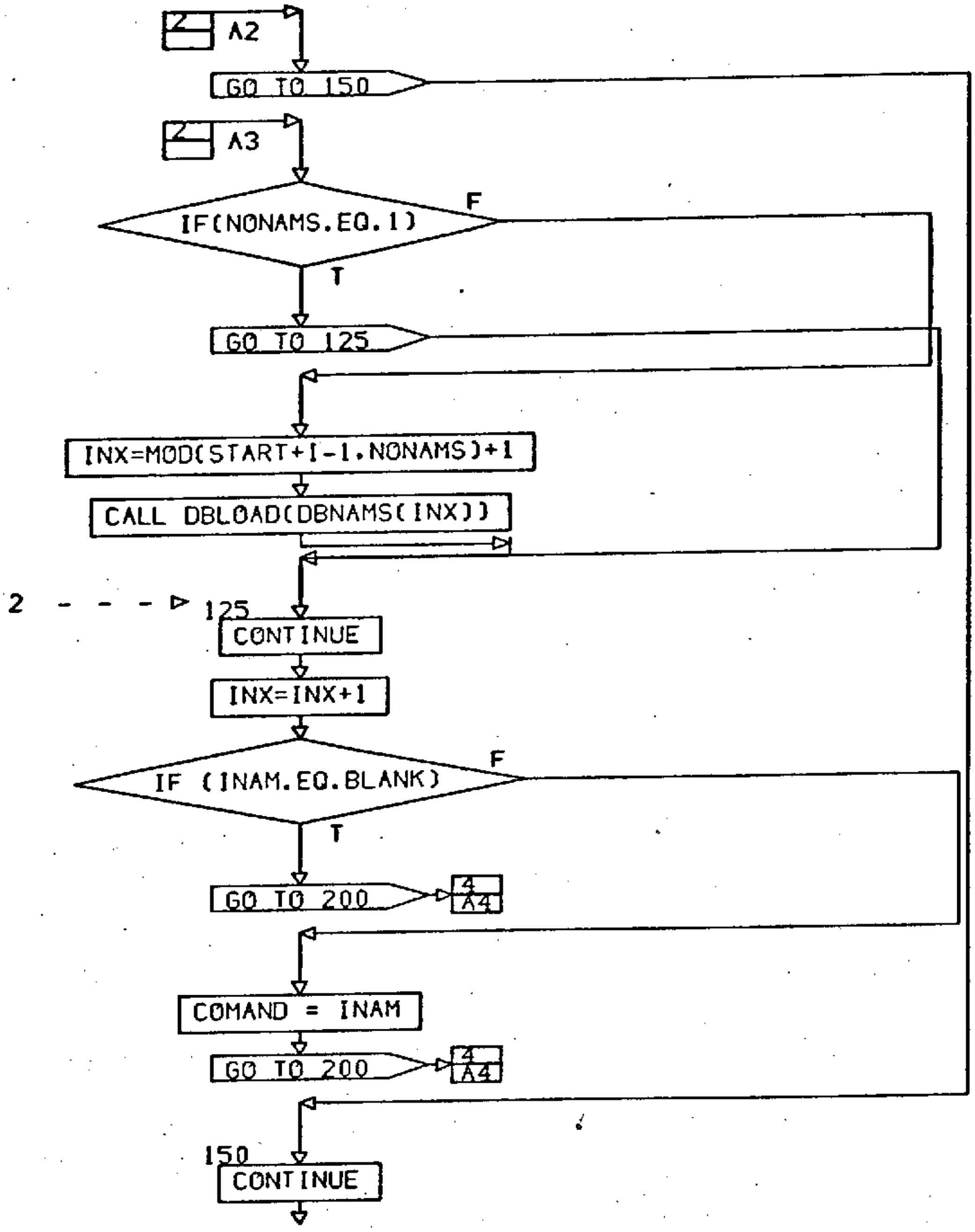


```
SUBROUTINE INMOD

C***  PROCESS "NAME  " DIRECTIVE.

COMMON IDATA(1)
EQUIVALENCE (ID, IDATA)
COMMON /SEARH/ INX, START, NONAMS, DBNAMS(15)
INTEGER INX, START, NONAMS, DBNAMS
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 40), ICONT  )

EQUIVALENCE (IDILOG( 43), FIND   )
EQUIVALENCE (IDILOG(184), IMAGE  )
EQUIVALENCE (IDILOG(220), INRECL)
EQUIVALENCE (IDILOG(223), IV     )
EQUIVALENCE (IDILOG(243), LD     )
EQUIVALENCE (IDILOG(246), LK     )
EQUIVALENCE (IDILOG(248), LT     )
EQUIVALENCE (IDILOG(250), EDIT   )

EQUIVALENCE (IDILOG(254), MYEOF )
EQUIVALENCE (IDILOG(259), NF     )
EQUIVALENCE (IDILOG(264), NWREC )
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(338), LUO    )
EQUIVALENCE (IDILOG(357), IOCONT)
EQUIVALENCE (IDILOG(294), COMAND)
INTEGER PAGE, FIND

INTEGER IV(1), IMAGE(14), BLANK, DELIM
INTEGER  COMAND
LOGICAL FOUND, MYEOF, TRACER, EDIT

IF(TRACER)
```

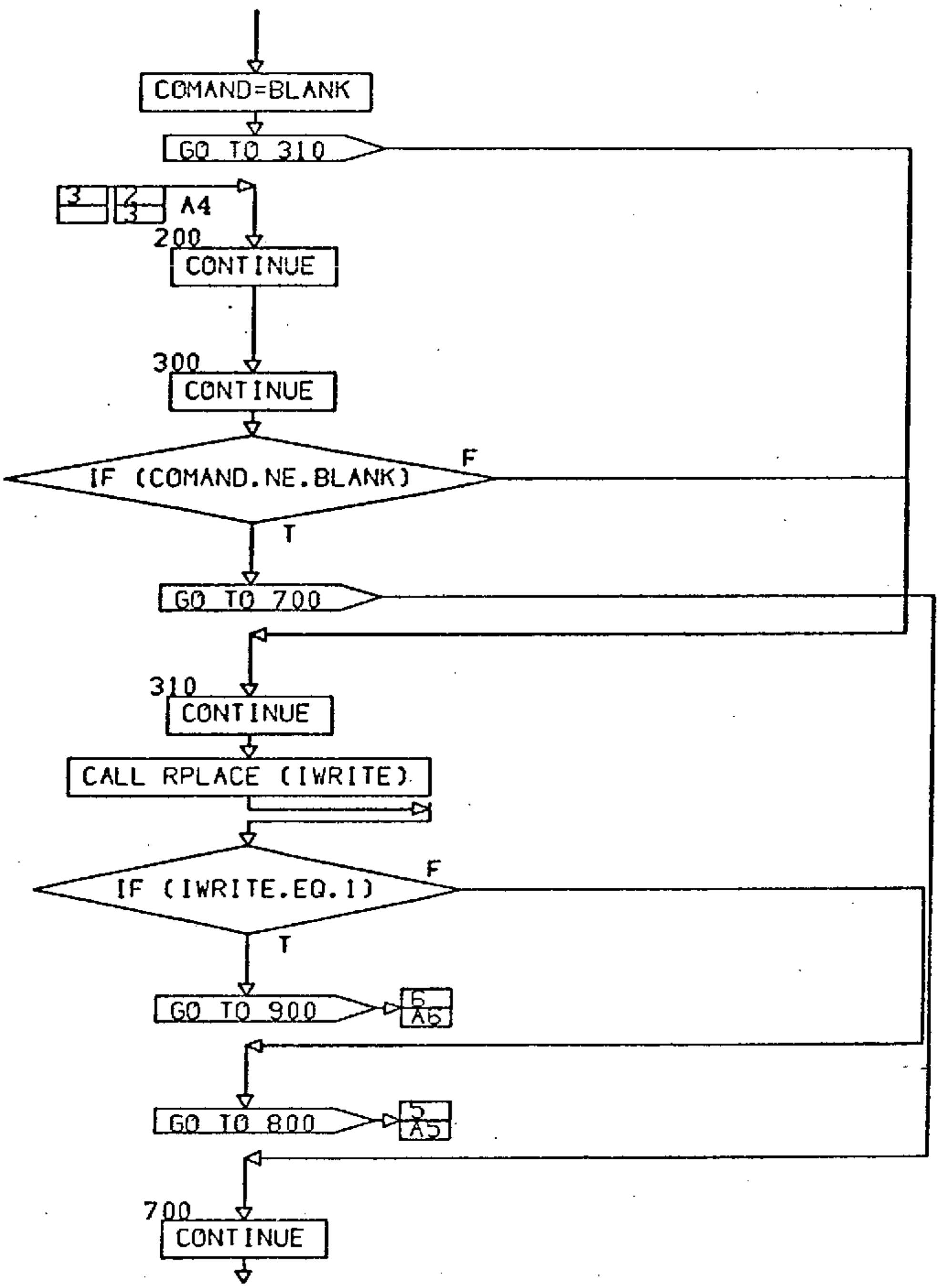
CONT. ON PG 2


A0
WRITE(LUO,1010)
A1
1010
FORMAT(13H ENTER INMOD)
IF (PAGE(1,1,1) .EQ. INRECL)
F
T
GO TO 800
5
A5
IF (PAGE(2,1,1) .NE. DELIM)
F
T
GO TO 200
4
A4
INAM = PAGE (3,I,1)
START=INX
DO 125 I=1,NONAMS
3
CALL RANDAC (FIND,INAM,LK,IDATA(ID+1),LD,FOUND,IV,LT,NF)
IF (FOUND)
F
3
A3
T
3
A2

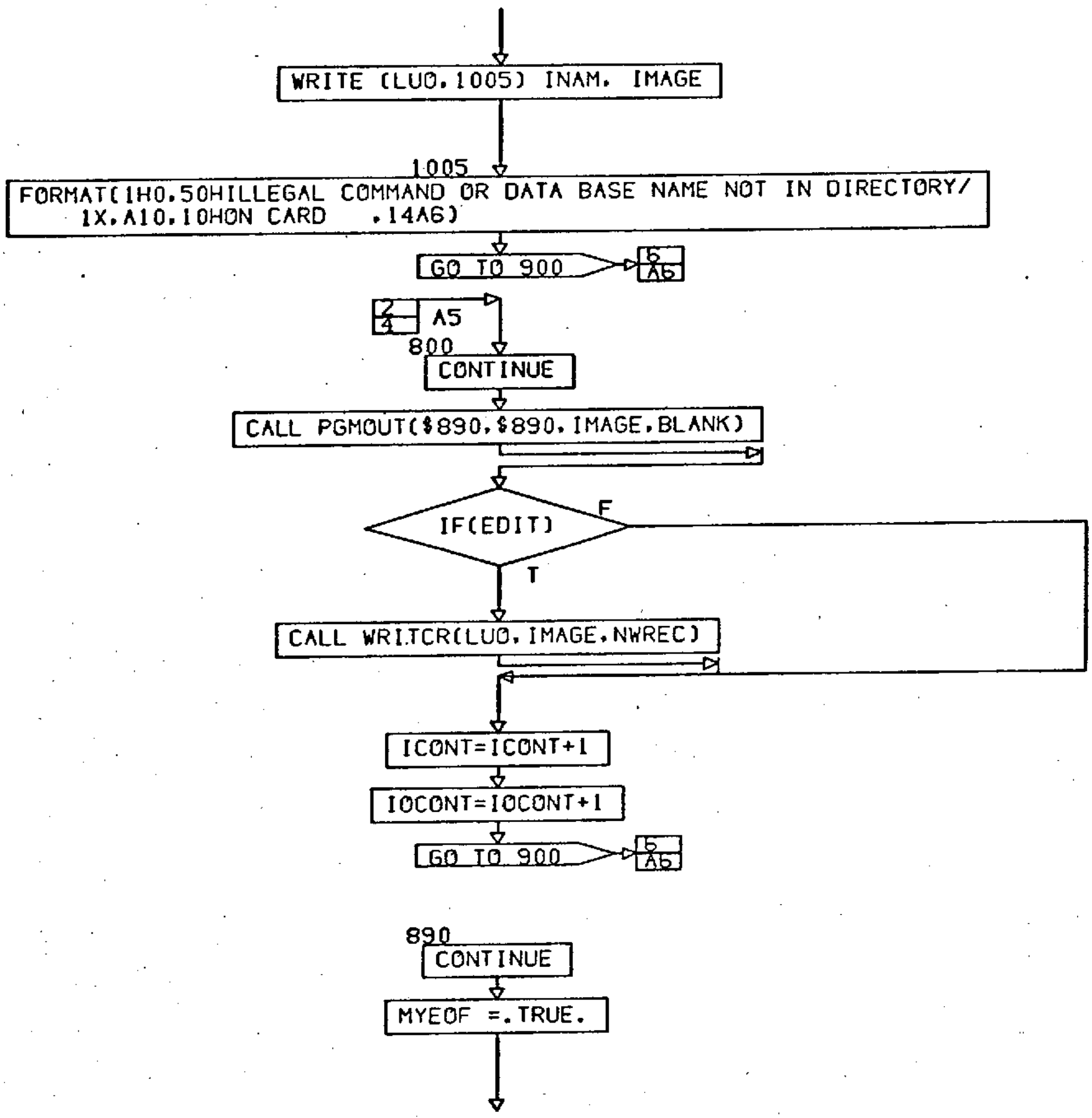
CONT. ON PG 3


2 A2
GO TO 150
2 A3
IF(NONAMS.EQ.1)
F
T
GO TO 125
INX=MOD(START+I-1,NONAMS)+1
CALL DBLOAD(DBNAMS(INX))
2
125
CONTINUE
INX=INX+1
IF (INAM.EQ.BLANK)
F
T
GO TO 200
4 A4
COMAND = INAM
GO TO 200
4 A4
150
CONTINUE

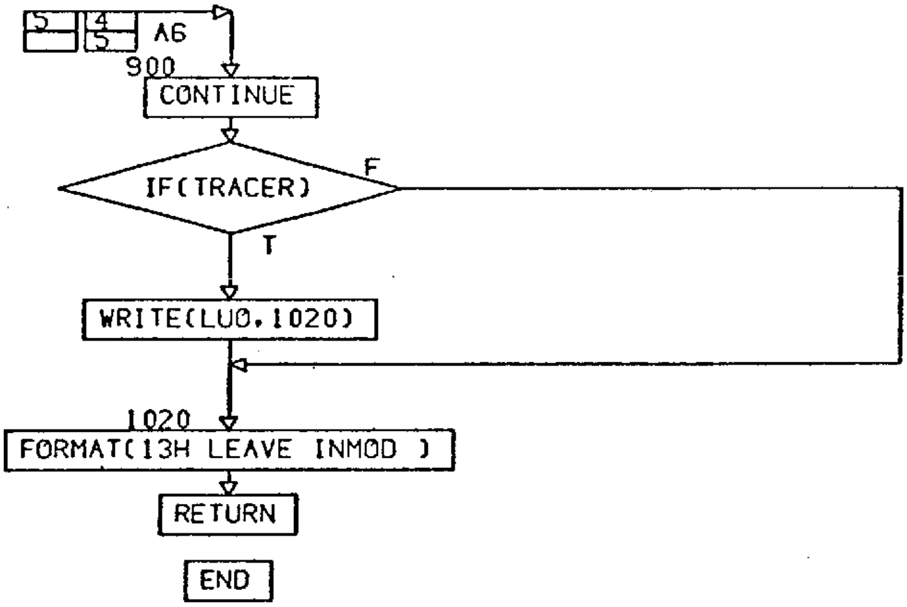
CONT. ON PG 4

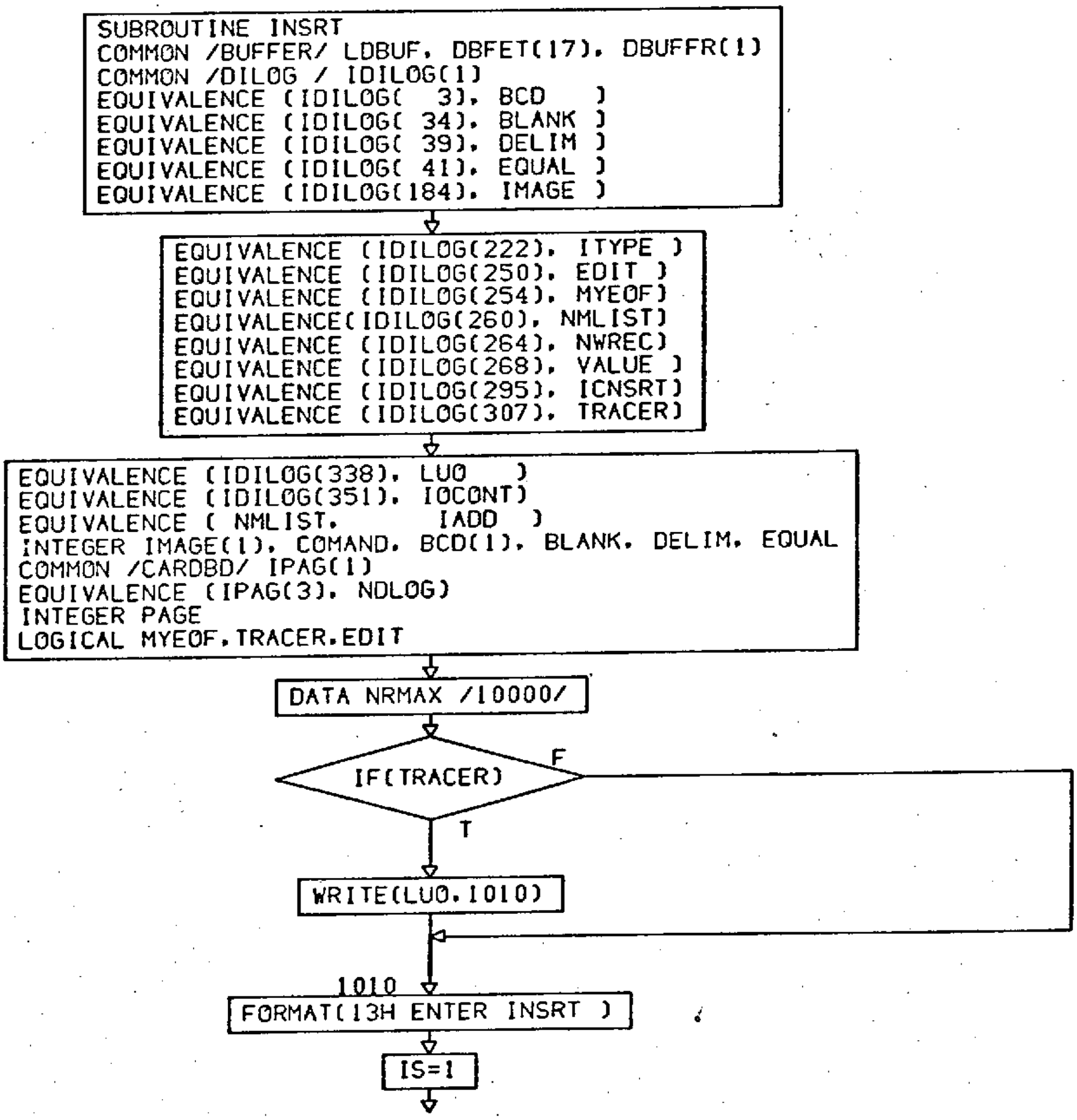

COMAND=BLANK
GO TO 310
3
2
3
A4
200
CONTINUE
300
CONTINUE
IF (COMAND.NE.BLANK)
F
T
GO TO 700
310
CONTINUE
CALL RPLACE (IWRITE)
IF (IWRITE.EQ.1)
F
T
GO TO 900
6
A6
GO TO 800
5
A5
700
CONTINUE

CONT. ON PG 5


WRITE (LUO,1005) INAM, IMAGE
1005
FORMAT(1H0,50HILLEGAL COMMAND OR DATA BASE NAME NOT IN DIRECTORY/
1X,A10,10HON CARD ,14A6)
GO TO 900
5
A6
2
4
A5
800
CONTINUE
CALL PGMOUT($890,$890,IMAGE,BLANK)
IF(EDIT)
F
T
CALL WRITCR(LUO,IMAGE,NWREC)
ICONT=ICONT+1
IOCONT=IOCONT+1
GO TO 900
5
A6
890
CONTINUE
MYEOF =.TRUE.

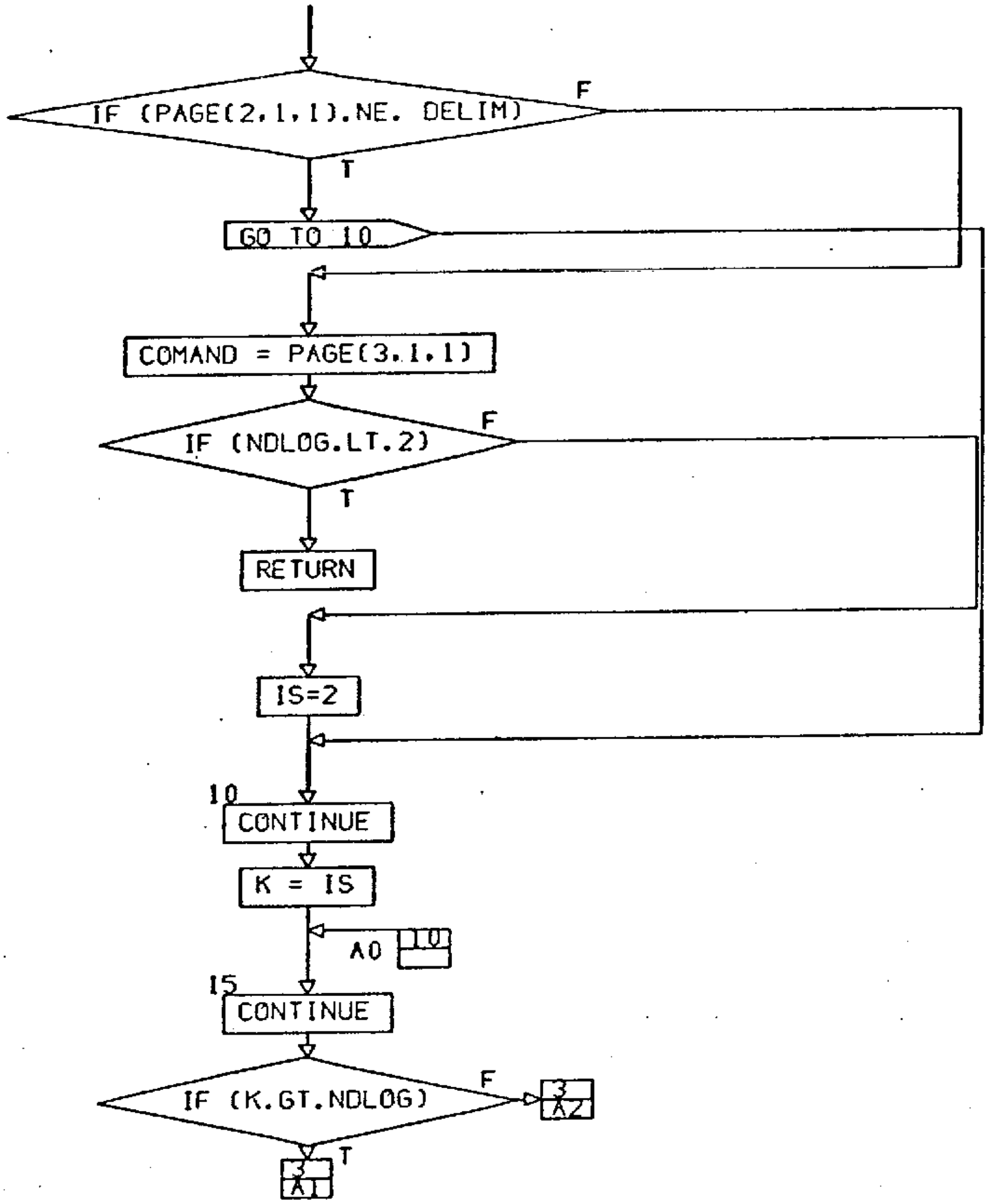
CONT. ON PG 6


5
4
5
A6
900
CONTINUE
IF(TRACER)
F
T
WRITE(LUO,1020)
1020
FORMAT(13H LEAVE INMOD)
RETURN
END

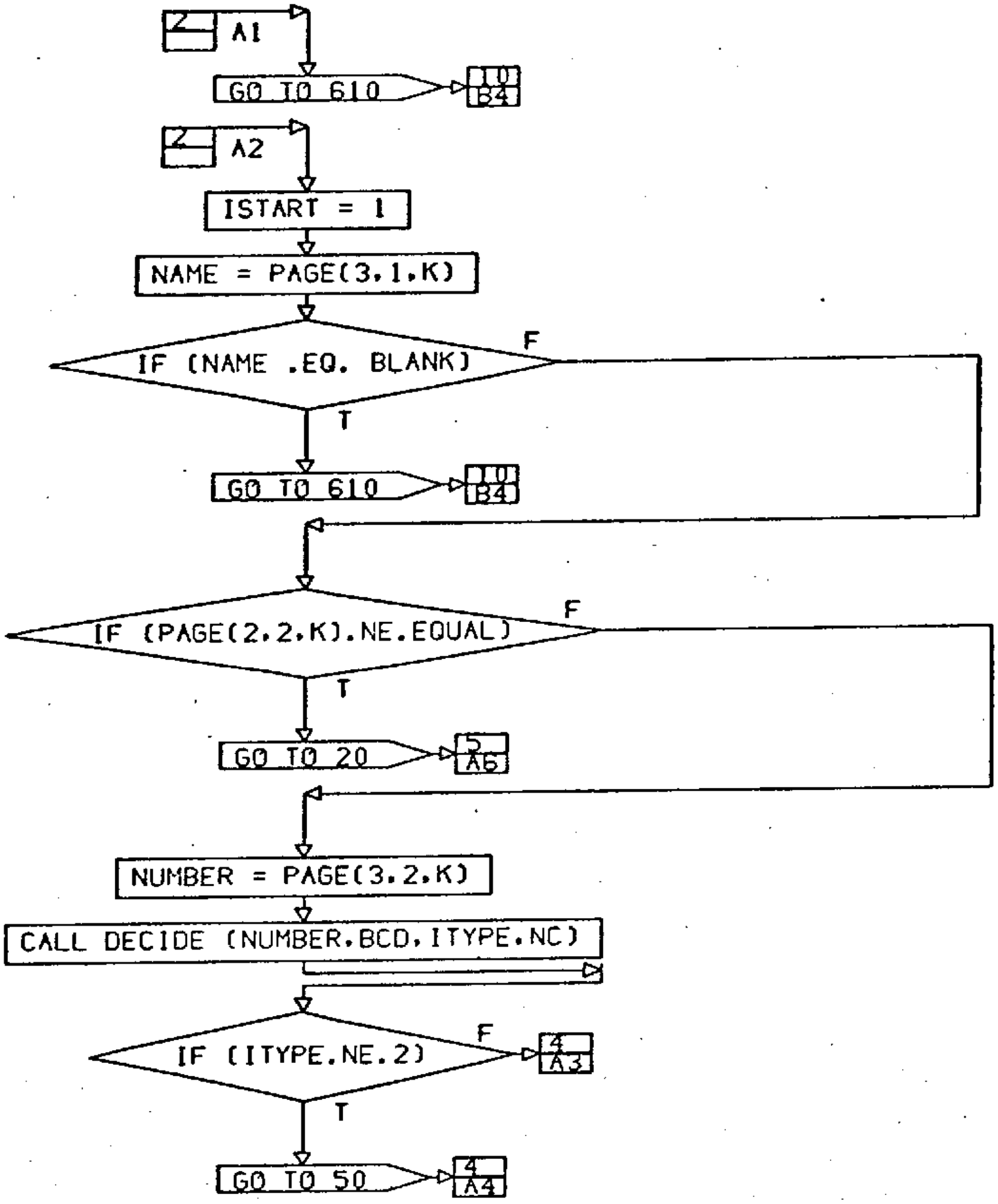


```
SUBROUTINE INSRT
COMMON /BUFFER/ LDBUF, DBFET(17), DBUFFR(1)
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG(  3), BCD    )
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 41), EQUAL )
EQUIVALENCE (IDILOG(184), IMAGE )

EQUIVALENCE (IDILOG(222), ITYPE )
EQUIVALENCE (IDILOG(250), EDIT )
EQUIVALENCE (IDILOG(254), MYEOF)
EQUIVALENCE(IDILOG(260), NMLIST)
EQUIVALENCE (IDILOG(264), NWREC)
EQUIVALENCE (IDILOG(268), VALUE )
EQUIVALENCE (IDILOG(295), ICNSRT)
EQUIVALENCE (IDILOG(307), TRACER)

EQUIVALENCE (IDILOG(338), LUO    )
EQUIVALENCE (IDILOG(351), IOCONT)
EQUIVALENCE ( NMLIST,       IADD   )
INTEGER IMAGE(1), COMAND, BCD(1), BLANK, DELIM, EQUAL
COMMON /CARDBD/ IPAG(1)
EQUIVALENCE (IPAG(3), NDLOG)
INTEGER PAGE
LOGICAL MYEOF,TRACER,EDIT

DATA NRMAX /10000/

IF(TRACER)

WRITE(LUO,1010)

1010
FORMAT(13H ENTER INSRT )

IS=1
```

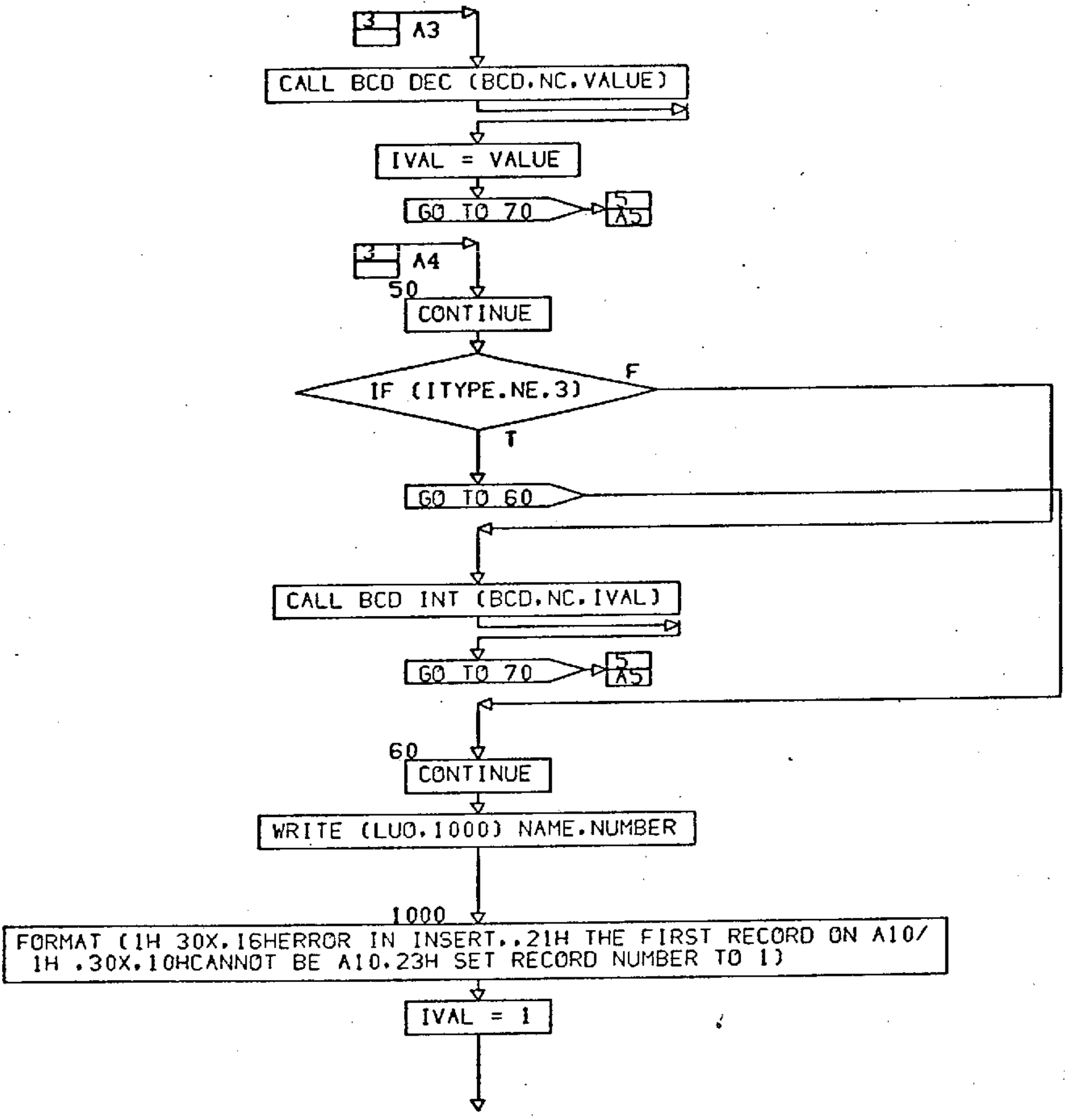
CONT. ON PG 2


IF (PAGE(2,1,1).NE. DELIM)
F
T
GO TO 10
COMAND = PAGE(3,1,1)
IF (NDLOG.LT.2)
F
T
RETURN
IS=2
10
CONTINUE
K = IS
A0
10
15
CONTINUE
IF (K.GT.NDLOG)
F
3
A2
T
3
A1

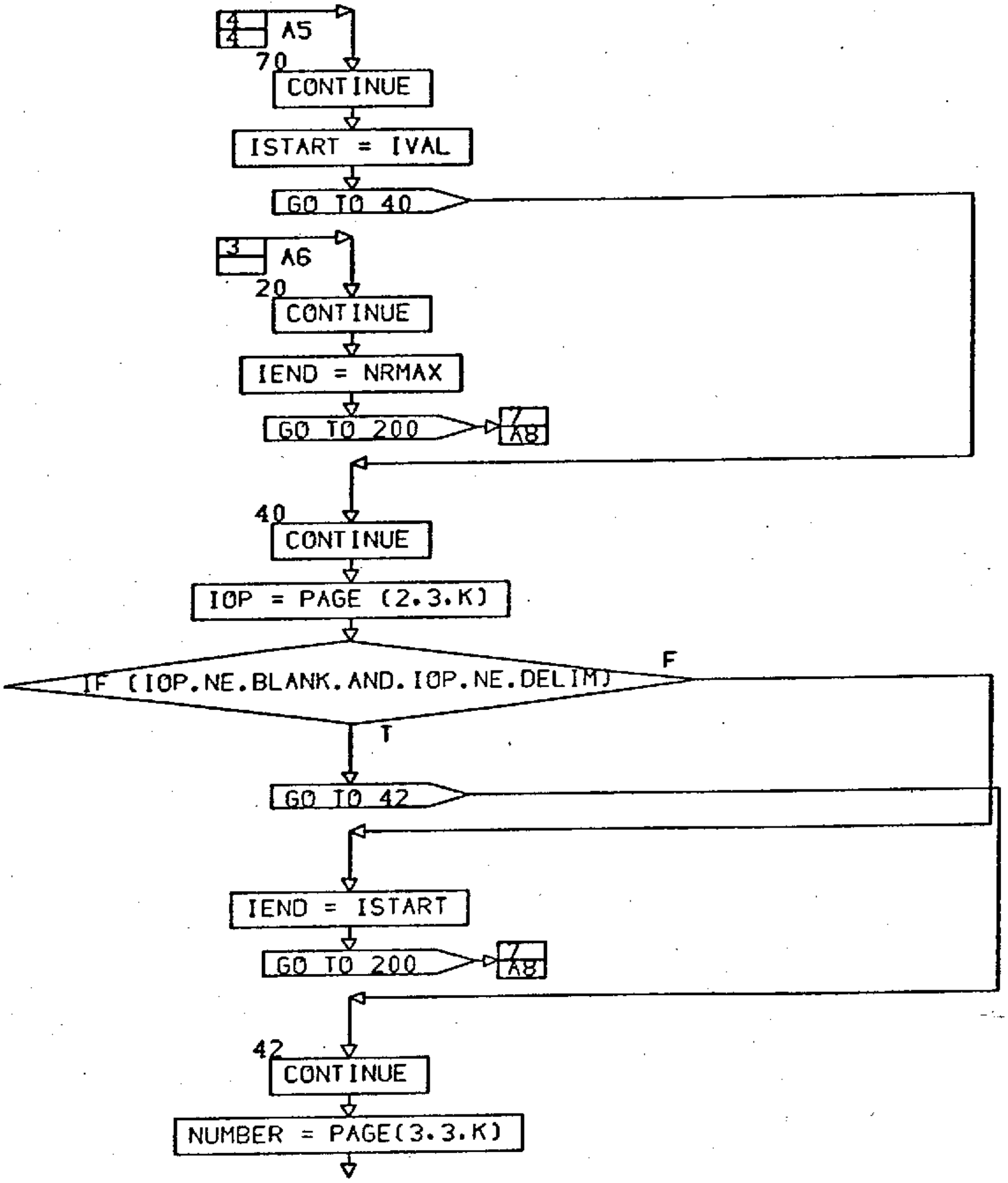
CONT. ON PG 3


2
A1
GO TO 610
10
B4
2
A2
ISTART = 1
NAME = PAGE(3,1,K)
IF (NAME .EQ. BLANK)
F
T
GO TO 610
10
B4
IF (PAGE(2,2,K).NE.EQUAL)
F
T
GO TO 20
5
A6
NUMBER = PAGE(3,2,K)
CALL DECIDE (NUMBER,BCD,ITYPE,NC)
IF (ITYPE.NE.2)
F
4
A3
T
GO TO 50
4
A4

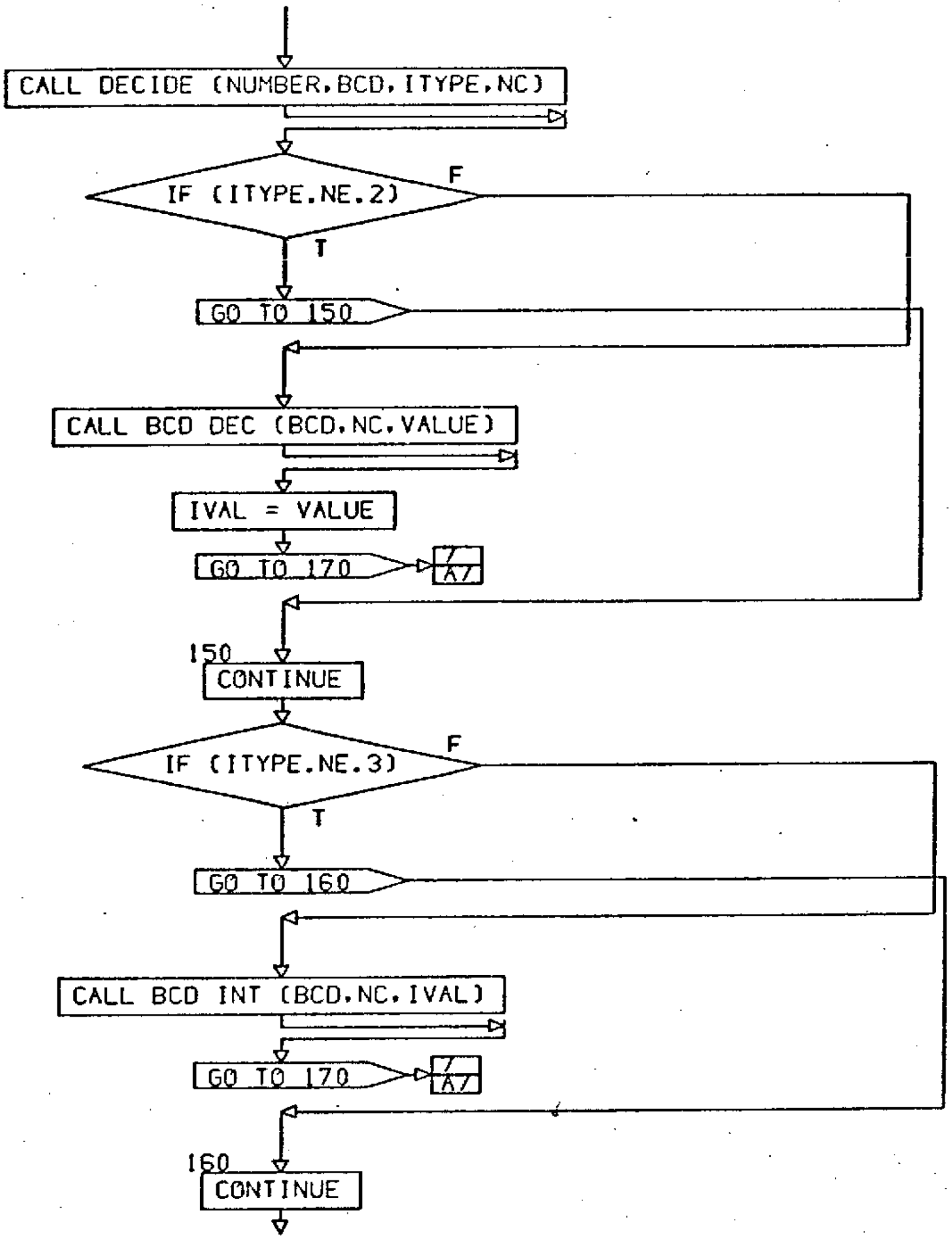
CONT. ON PG 4


3
A3
CALL BCD DEC (BCD.NC.VALUE)
IVAL = VALUE
GO TO 70
5
A5
3
A4
50
CONTINUE
IF (ITYPE.NE.3)
F
T
GO TO 60
CALL BCD INT (BCD.NC.IVAL)
GO TO 70
5
A5
60
CONTINUE
WRITE (LUO.1000) NAME.NUMBER
1000
FORMAT (1H 30X.16HERROR IN INSERT..21H THE FIRST RECORD ON A10/
1H .30X.10HCANNOT BE A10.23H SET RECORD NUMBER TO 1)
IVAL = 1

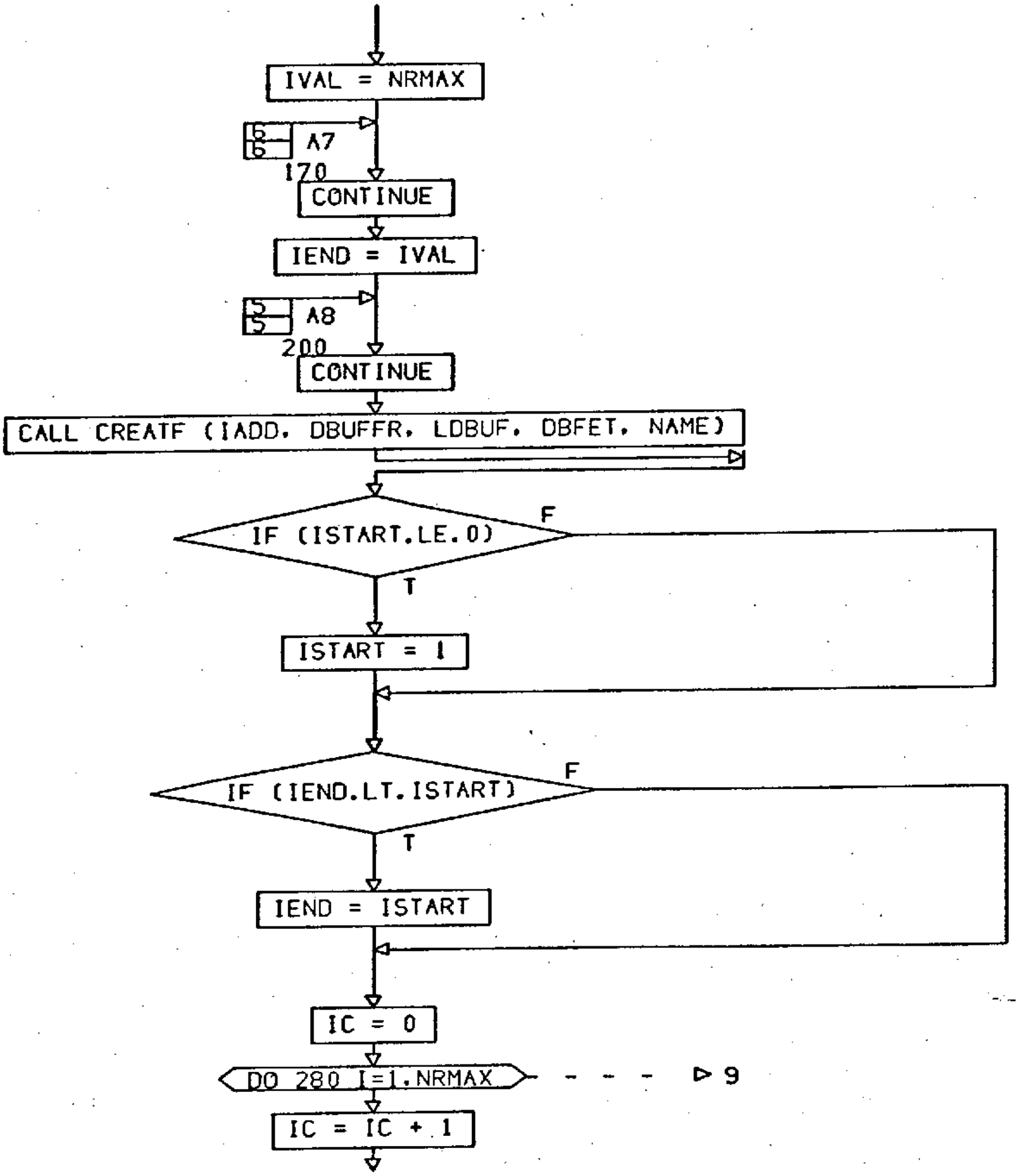
CONT. ON PG 5


4
4
A5
70
CONTINUE
ISTART = IVAL
GO TO 40
3
A6
20
CONTINUE
IEND = NRMAX
GO TO 200
7
A8
40
CONTINUE
IOP = PAGE (2.3.K)
IF (IOP.NE.BLANK.AND.IOP.NE.DELIM)
F
T
GO TO 42
IEND = ISTART
GO TO 200
7
A8
42
CONTINUE
NUMBER = PAGE(3.3.K)

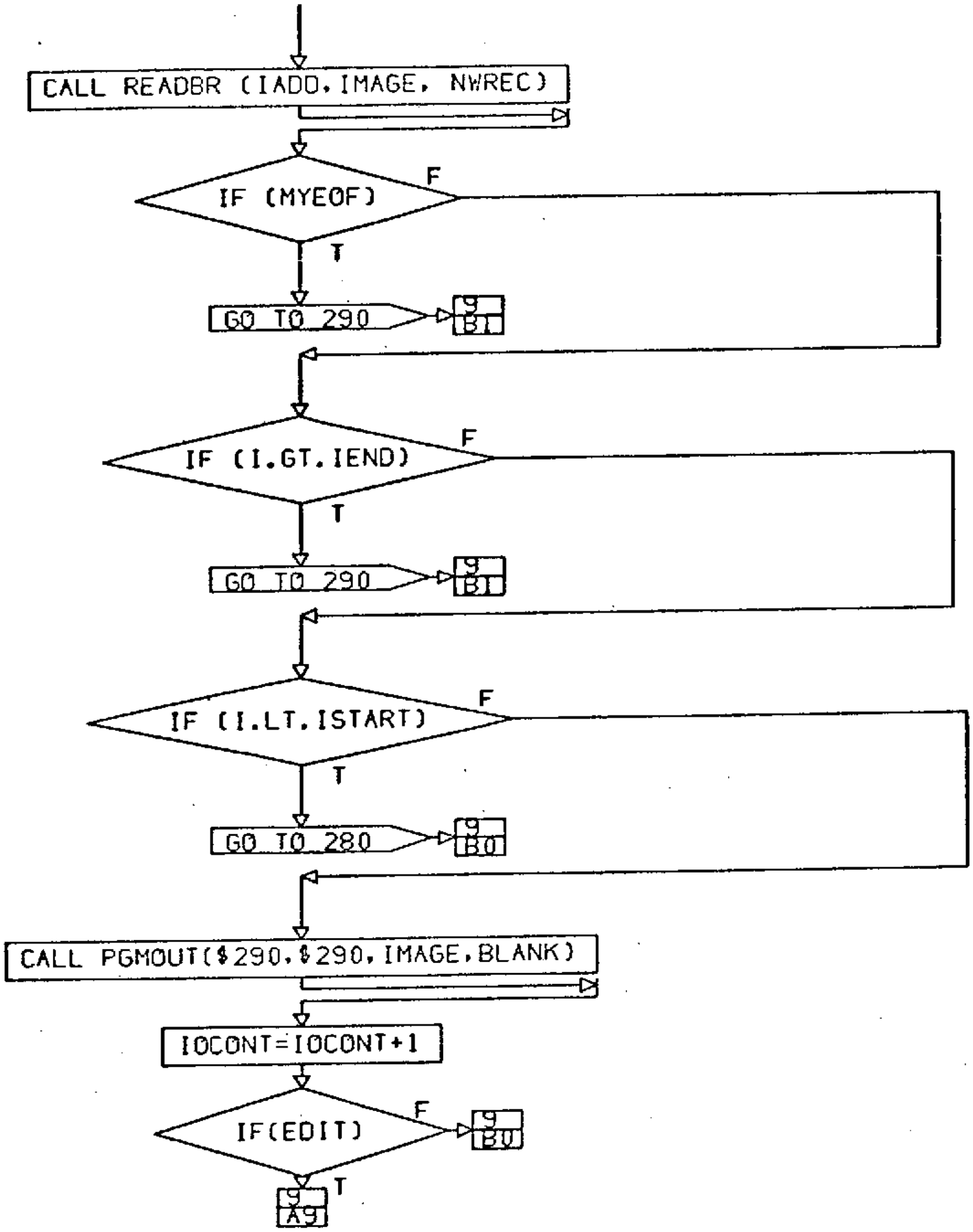
CONT. ON PG 6


CALL DECIDE (NUMBER,BCD,ITYPE,NC)
IF (ITYPE.NE.2)
F
T
GO TO 150
CALL BCD DEC (BCD,NC,VALUE)
IVAL = VALUE
GO TO 170
7
A7
150
CONTINUE
IF (ITYPE.NE.3)
F
T
GO TO 160
CALL BCD INT (BCD,NC,IVAL)
GO TO 170
7
A7
160
CONTINUE

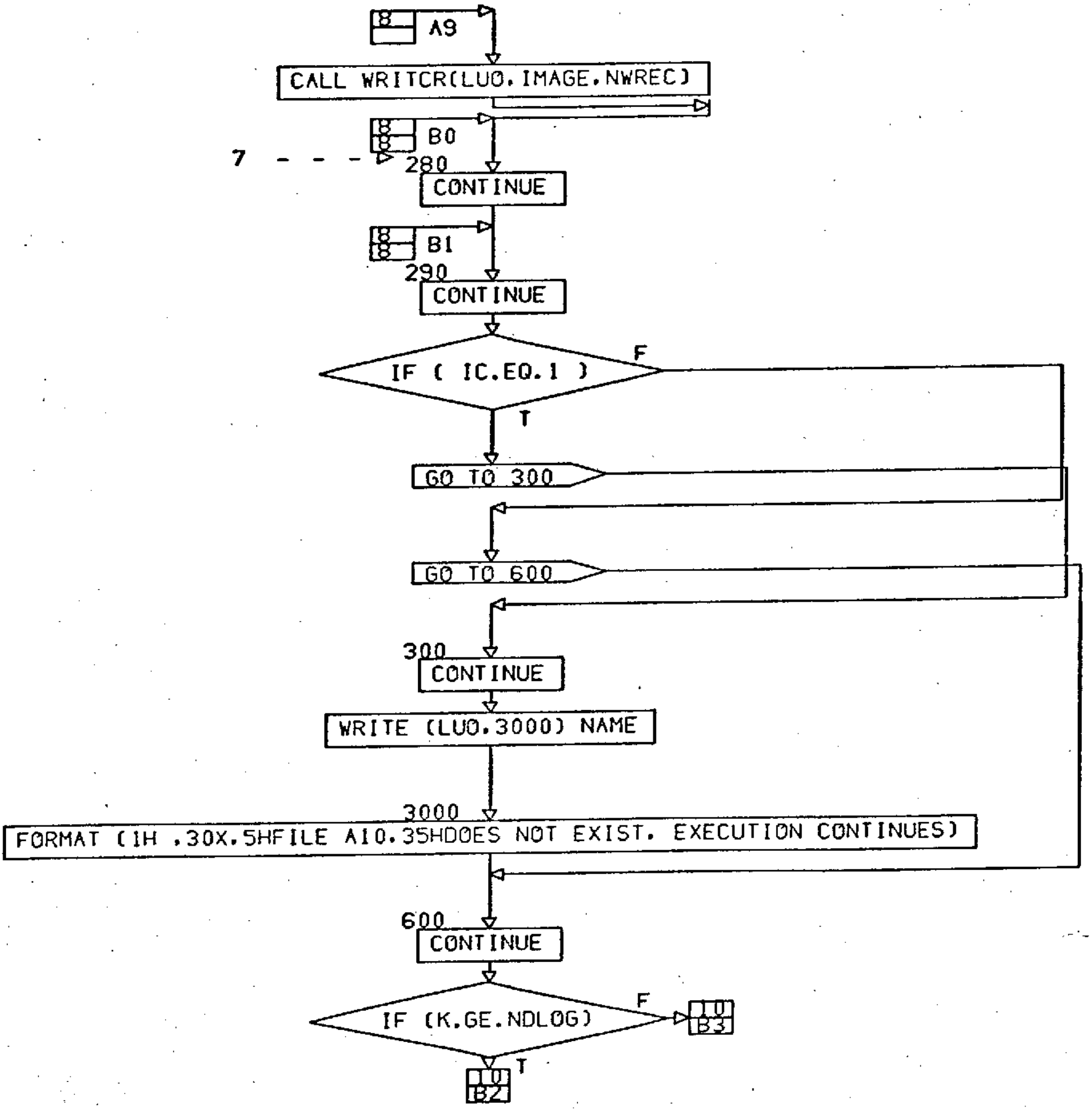
CONT. ON PG 7


IVAL = NRMAX
6
6
A7
170
CONTINUE
IEND = IVAL
5
5
A8
200
CONTINUE
CALL CREATF (IADD, DBUFFR, LDBUF, DBFET, NAME)
IF (ISTART.LE.0)
F
T
ISTART = 1
IF (IEND.LT.ISTART)
F
T
IEND = ISTART
IC = 0
DO 280 I=1,NRMAX
▷ 9
IC = IC + 1

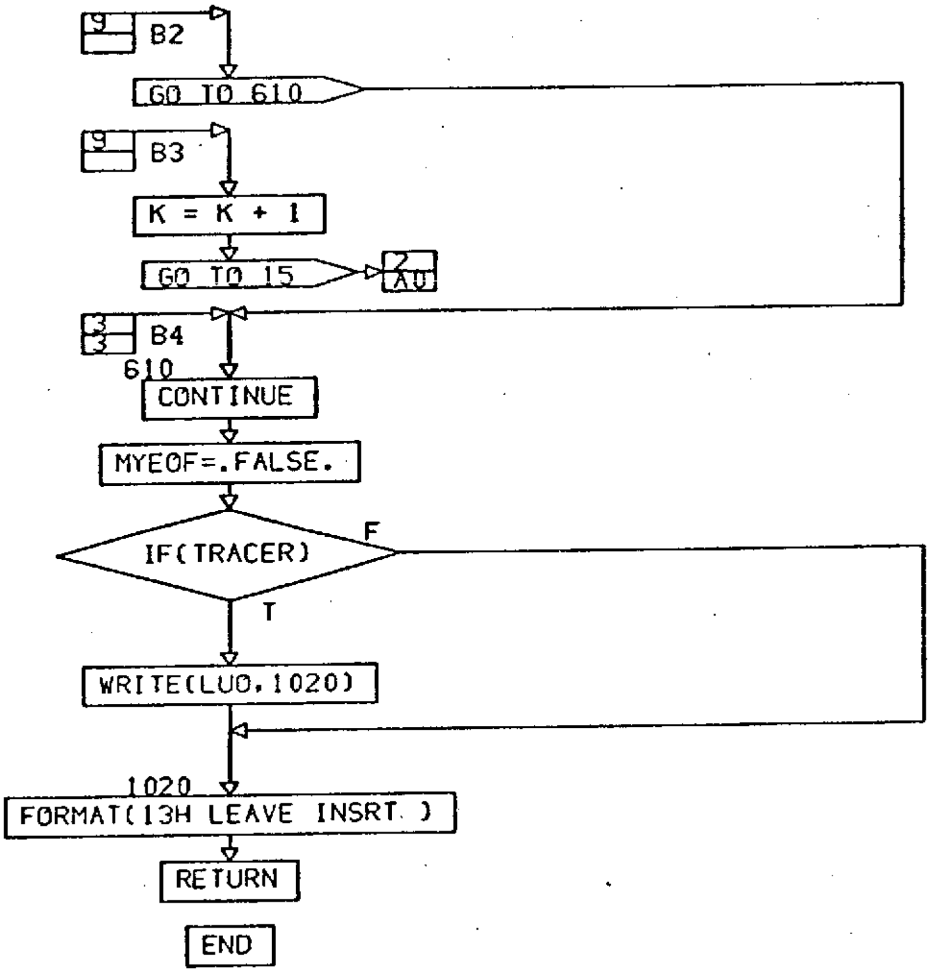
CONT. ON PG 8

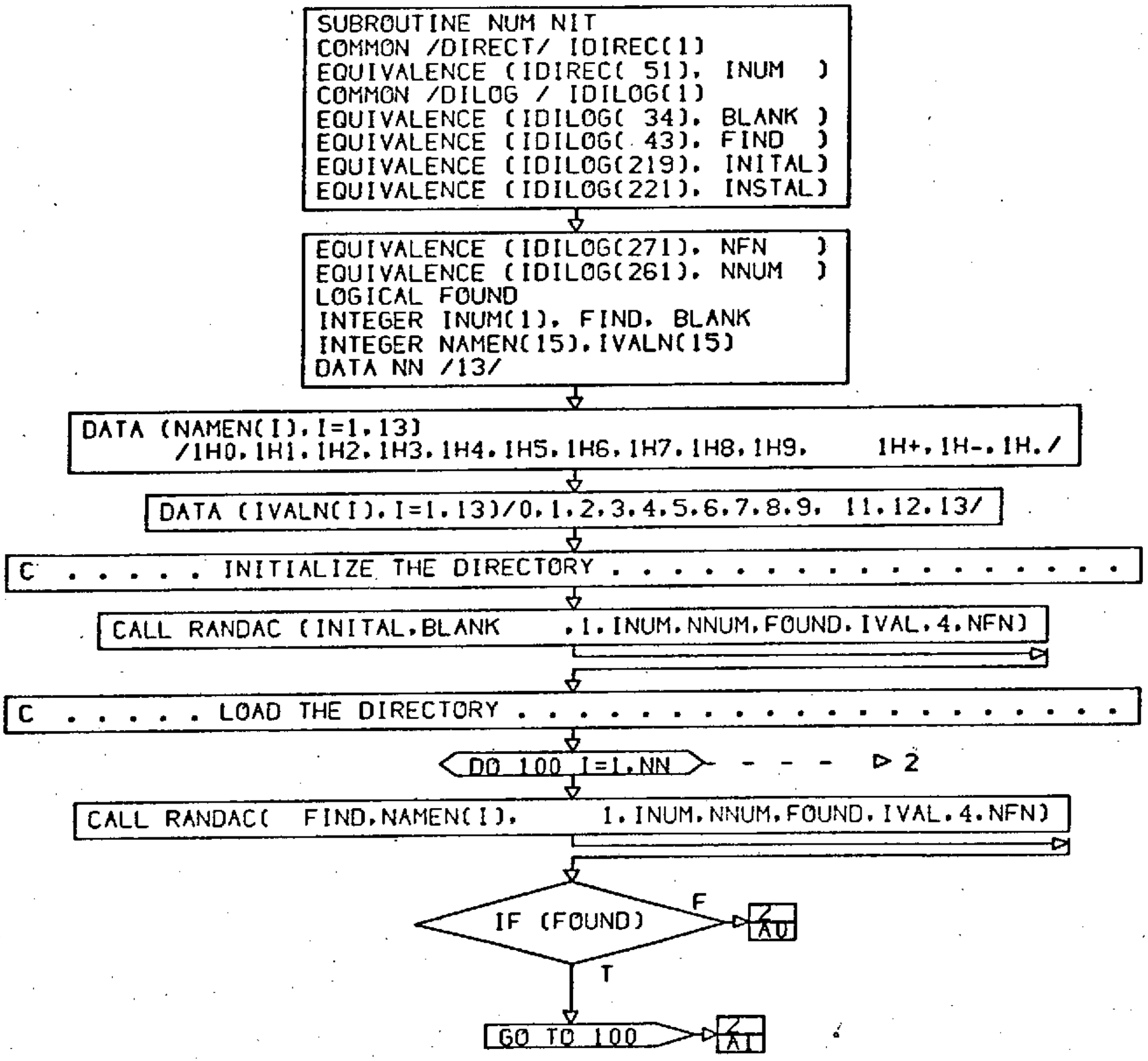

CALL READBR (IADD,IMAGE, NWREC)
IF (MYEOF)
F
T
GO TO 290
9
B1
IF (I.GT.IEND)
F
T
GO TO 290
9
B1
IF (I.LT.ISTART)
F
T
GO TO 280
9
B0
CALL PGMOUT($290,$290,IMAGE,BLANK)
IOCONT=IOCONT+1
IF(EDIT)
F
9
B0
T
9
A9

CONT. ON PG 9


8 A9
CALL WRITCR(LUO.IMAGE.NWREC)
8 8 B0
7
280
CONTINUE
8 8 B1
290
CONTINUE
IF (IC.EQ.1)
F
T
GO TO 300
GO TO 600
300
CONTINUE
WRITE (LUO.3000) NAME
3000
FORMAT (1H .30X.5HFILE A10.35HDOES NOT EXIST. EXECUTION CONTINUES)
600
CONTINUE
IF (K.GE.NDLOG)
F
10 B3
T
10 B2

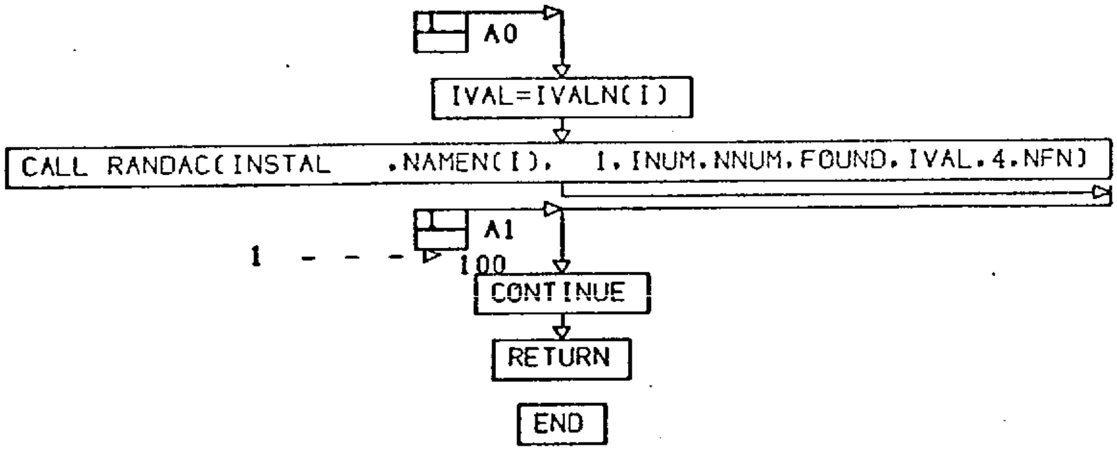
CONT. ON PG 10

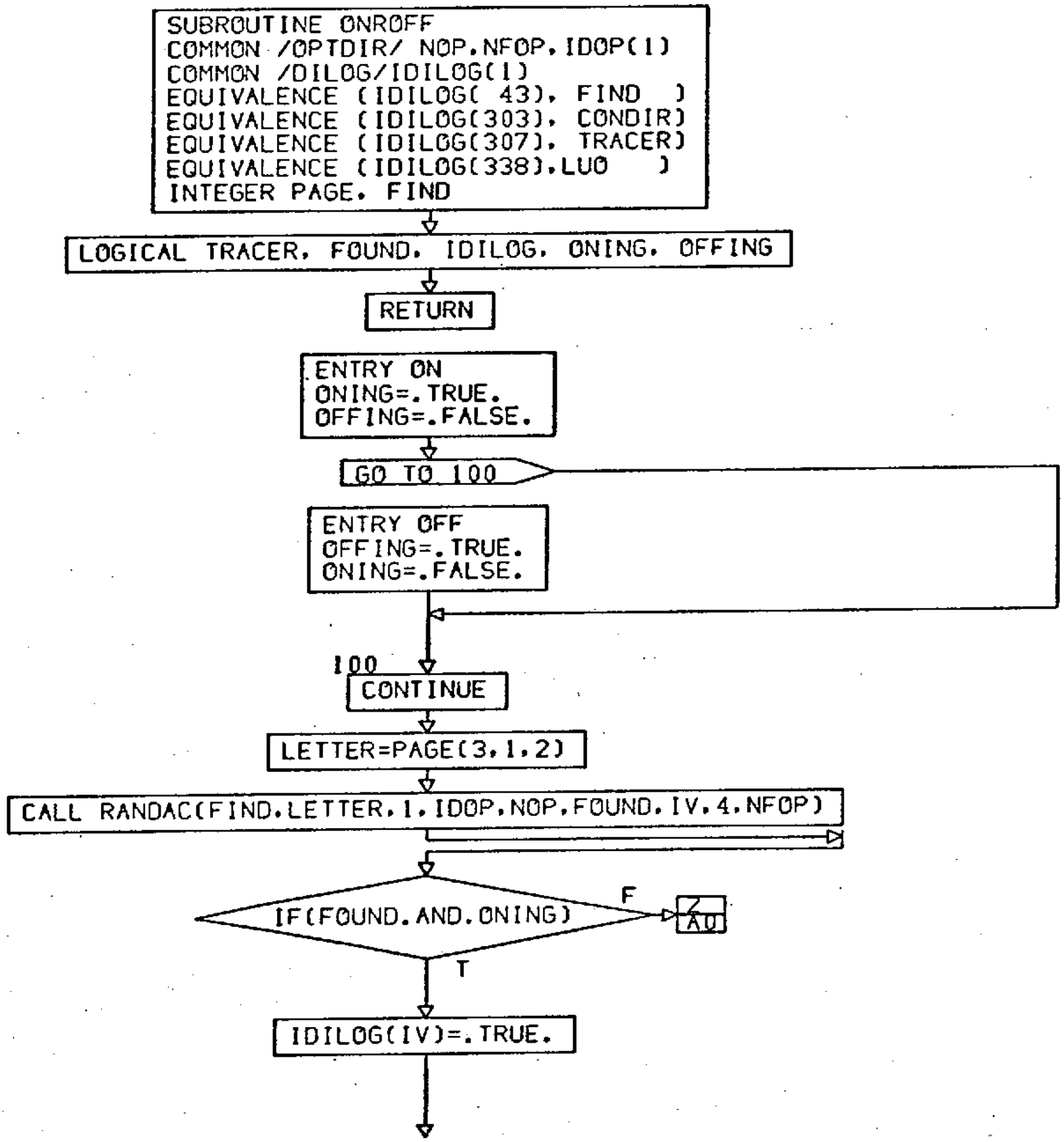

9
B2
GO TO 610
9
B3
K = K + 1
GO TO 15
2
A0
3
3
B4
610
CONTINUE
MYEOF=.FALSE.
IF(TRACER)
F
T
WRITE(LUO,1020)
1020
FORMAT(13H LEAVE INSRT)
RETURN
END



```
SUBROUTINE NUM NIT
COMMON /DIRECT/ IDIREC(1)
EQUIVALENCE (IDIREC( 51), INUM  )
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 43), FIND  )
EQUIVALENCE (IDILOG(219), INITAL)
EQUIVALENCE (IDILOG(221), INSTAL)

EQUIVALENCE (IDILOG(271), NFN   )
EQUIVALENCE (IDILOG(261), NNUM  )
LOGICAL FOUND
INTEGER INUM(1), FIND, BLANK
INTEGER NAMEN(15),IVALN(15)
DATA NN /13/

DATA (NAMEN(I),I=1,13)
      /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,     1H+,1H-,1H./

DATA (IVALN(I),I=1,13)/0,1,2,3,4,5,6,7,8,9, 11,12,13/

C  . . . . . INITIALIZE THE DIRECTORY . . . . . . . . . . . . . . . . . . .

CALL RANDAC (INITAL,BLANK     ,1,INUM,NNUM,FOUND,IVAL,4,NFN)

C  . . . . . LOAD THE DIRECTORY . . . . . . . . . . . . . . . . . . . . . .

DO 100 I=1,NN

CALL RANDAC(  FIND,NAMEN(I),       1,INUM,NNUM,FOUND,IVAL,4,NFN)

IF (FOUND)

GO TO 100
```

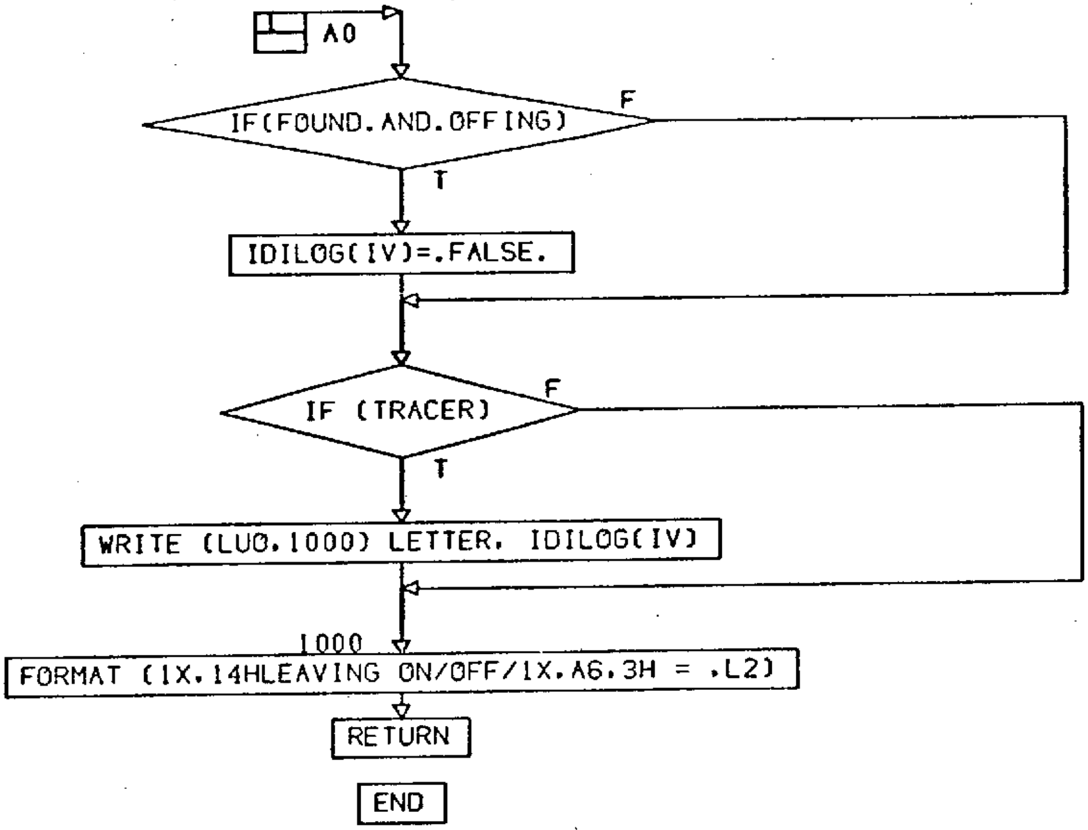
CONT. ON PG 2


A0
IVAL=IVALN(I)
CALL RANDAC(INSTAL   ,NAMEN(I),   I,INUM,NNUM,FOUND,IVAL,4,NFN)
A1
1
100
CONTINUE
RETURN
END

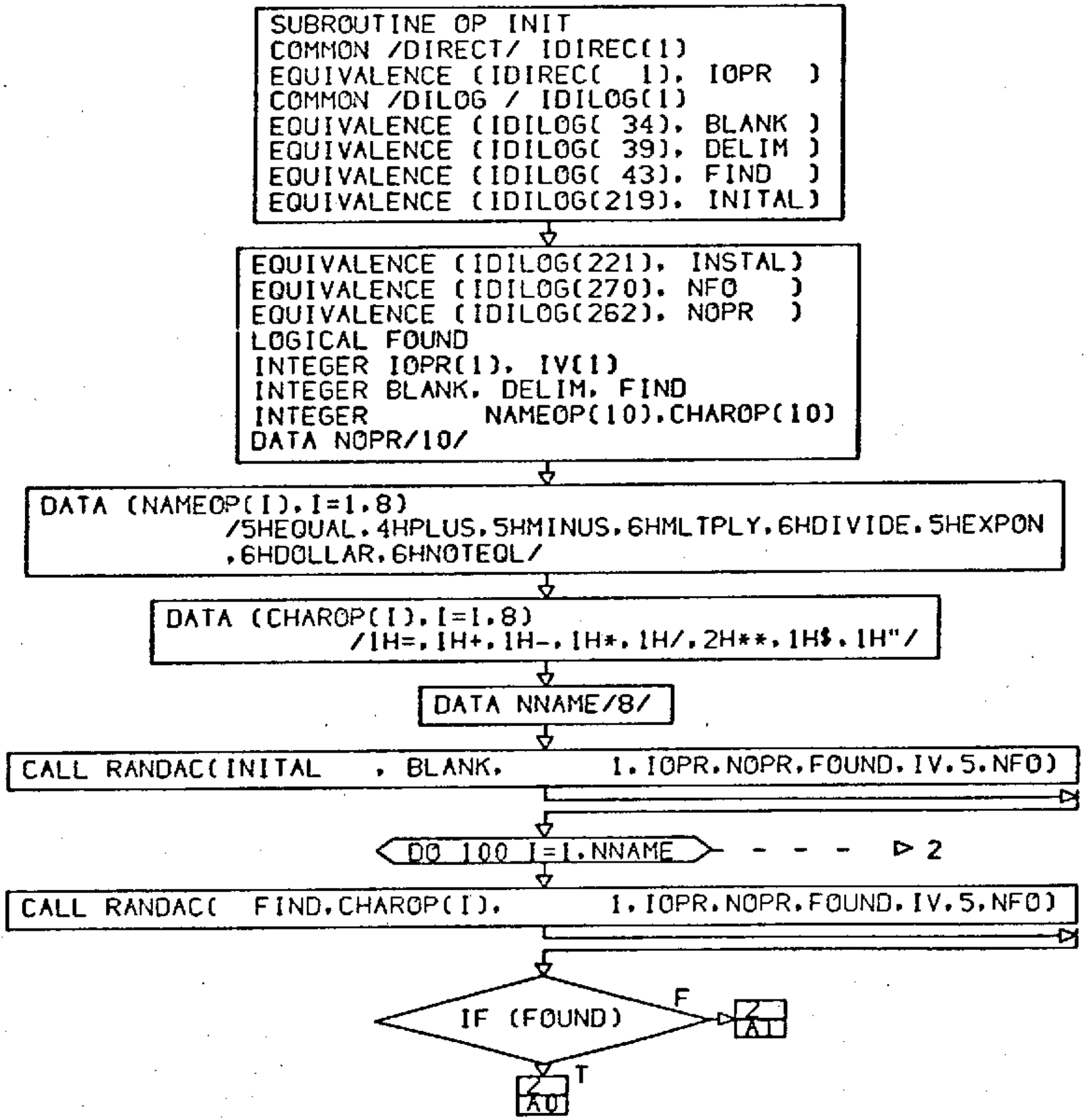


```
SUBROUTINE ONROFF
COMMON /OPTDIR/ NOP.NFOP.IDOP(1)
COMMON /DILOG/IDILOG(1)
EQUIVALENCE (IDILOG( 43), FIND   )
EQUIVALENCE (IDILOG(303), CONDIR)
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(338),LUO    )
INTEGER PAGE, FIND
LOGICAL TRACER, FOUND, IDILOG, ONING, OFFING
RETURN
ENTRY ON
ONING=.TRUE.
OFFING=.FALSE.
GO TO 100
ENTRY OFF
OFFING=.TRUE.
ONING=.FALSE.
100 CONTINUE
LETTER=PAGE(3,1,2)
CALL RANDAC(FIND.LETTER,1,IDOP,NOP,FOUND,IV,4,NFOP)
IF(FOUND.AND.ONING)
IDILOG(IV)=.TRUE.
```

CONT. ON PG 2

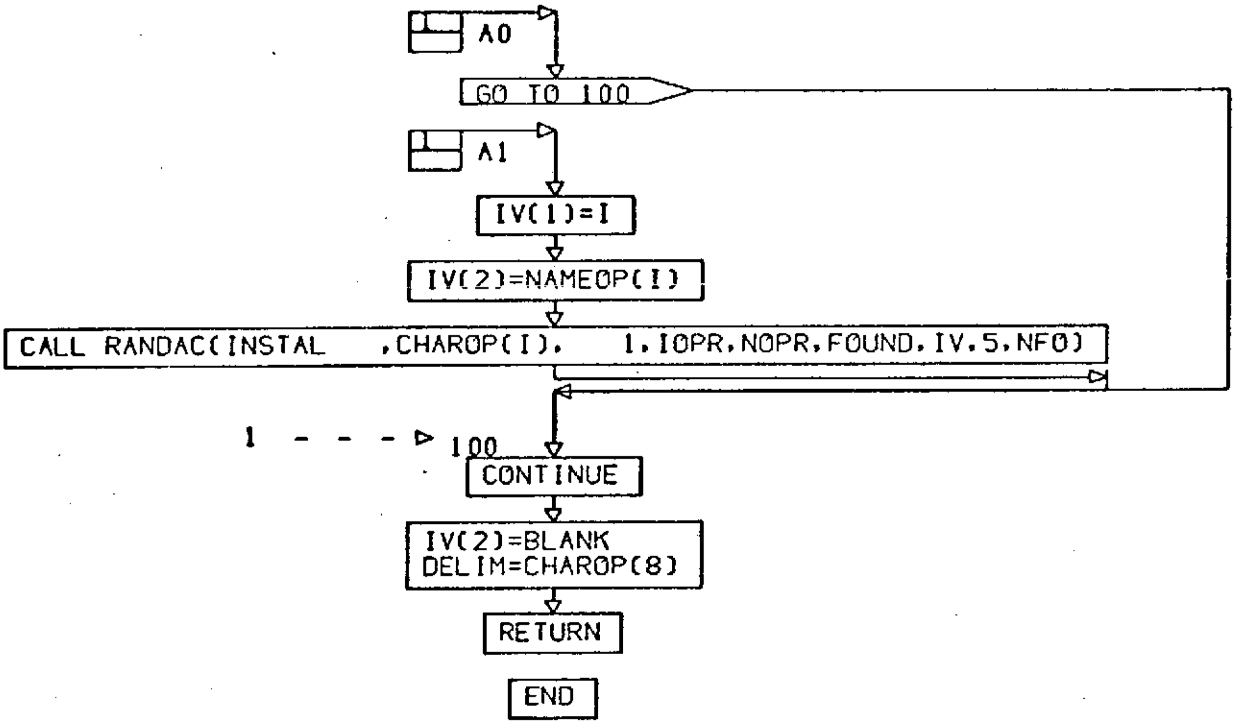

A0
IF(FOUND.AND.OFFING)
F
T
IDILOG(IV)=.FALSE.
IF (TRACER)
F
T
WRITE (LUO,1000) LETTER, IDILOG(IV)
1000
FORMAT (1X,14HLEAVING ON/OFF/1X,A6,3H = ,L2)
RETURN
END

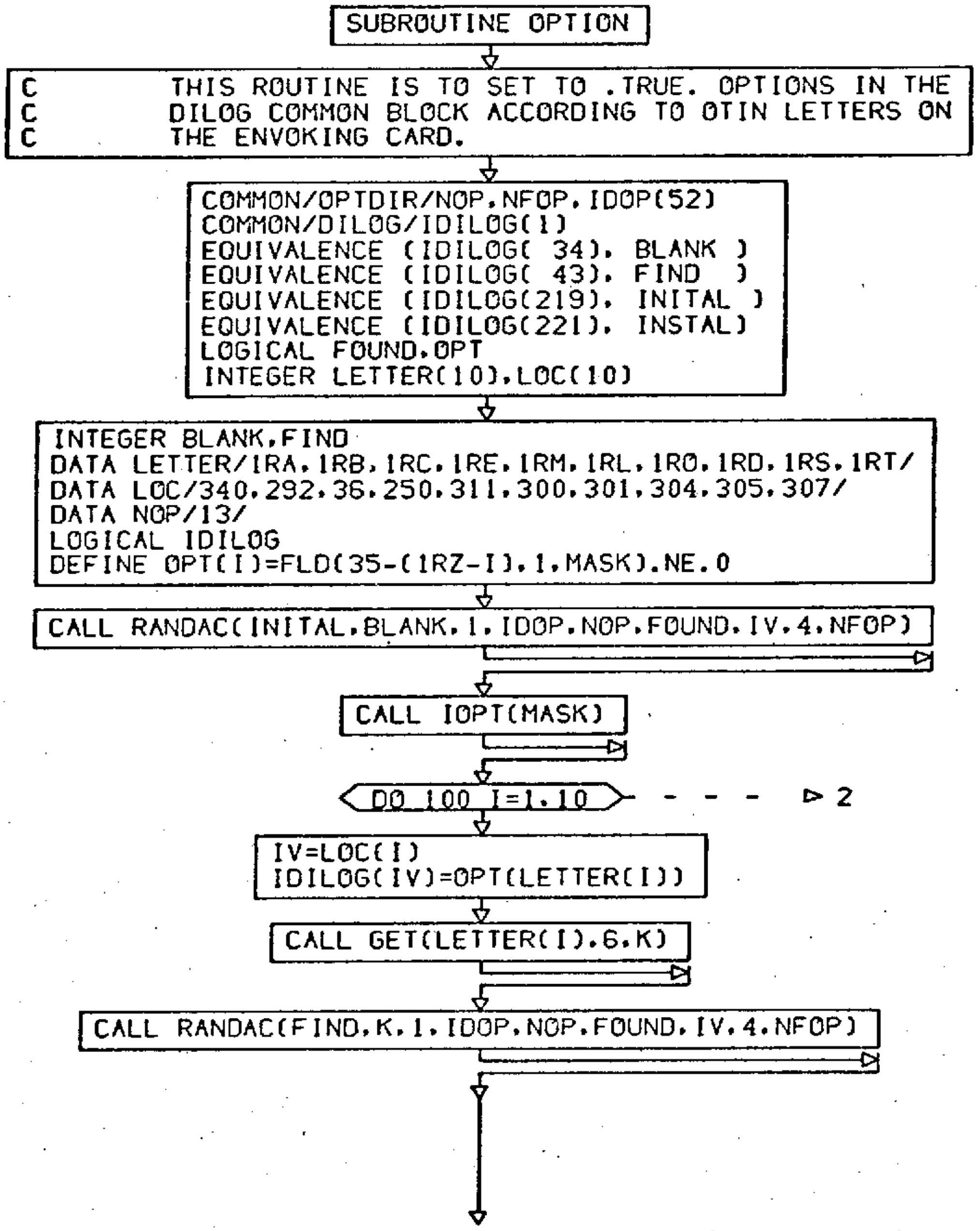
ONROFF
PG 2 FINAL



```
SUBROUTINE OP INIT
COMMON /DIRECT/ IDIREC(1)
EQUIVALENCE (IDIREC(   1), IOPR   )
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 39), DELIM )
EQUIVALENCE (IDILOG( 43), FIND  )
EQUIVALENCE (IDILOG(219), INITAL)

EQUIVALENCE (IDILOG(221), INSTAL)
EQUIVALENCE (IDILOG(270), NFO   )
EQUIVALENCE (IDILOG(262), NOPR  )
LOGICAL FOUND
INTEGER IOPR(1), IV(1)
INTEGER BLANK, DELIM, FIND
INTEGER        NAMEOP(10),CHAROP(10)
DATA NOPR/10/

DATA (NAMEOP(I),I=1,8)
           /5HEQUAL,4HPLUS,5HMINUS,6HMLTPLY,6HDIVIDE,5HEXPON
           ,6HDOLLAR,6HNOTEQL/

DATA (CHAROP(I),I=1,8)
           /1H=,1H+,1H-,1H*,1H/,2H**,1H$,1H"/

DATA NNAME/8/

CALL RANDAC(INITAL    , BLANK,          1,IOPR,NOPR,FOUND,IV,5,NFO)

DO 100 I=1,NNAME

CALL RANDAC(  FIND,CHAROP(I),          1,IOPR,NOPR,FOUND,IV,5,NFO)

IF (FOUND)
```

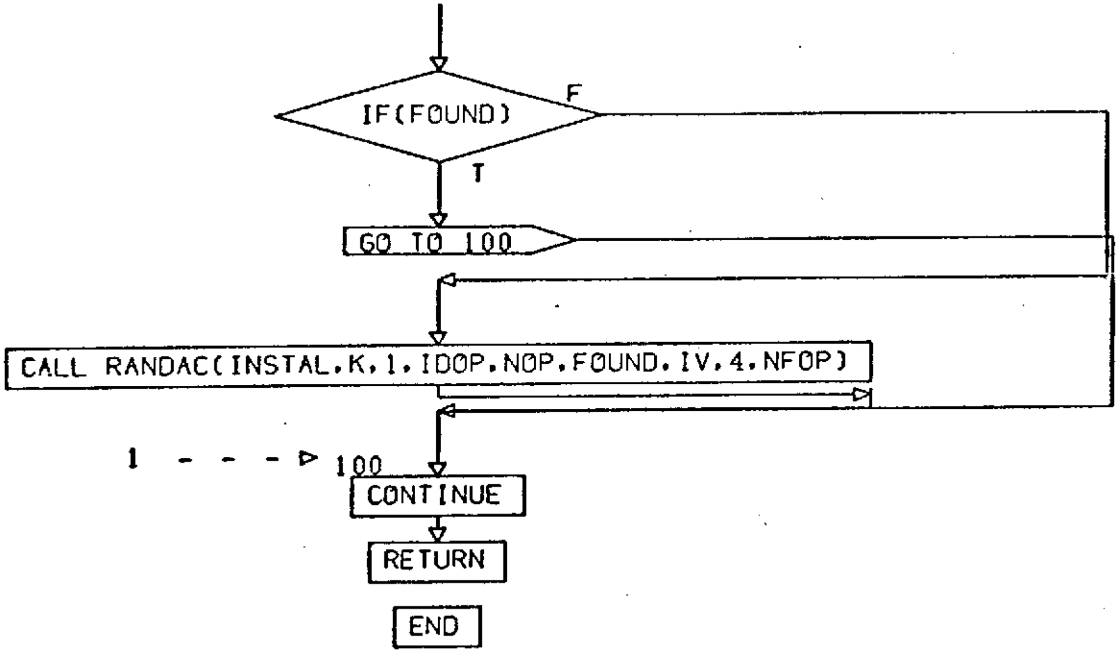
CONT. ON PG 2

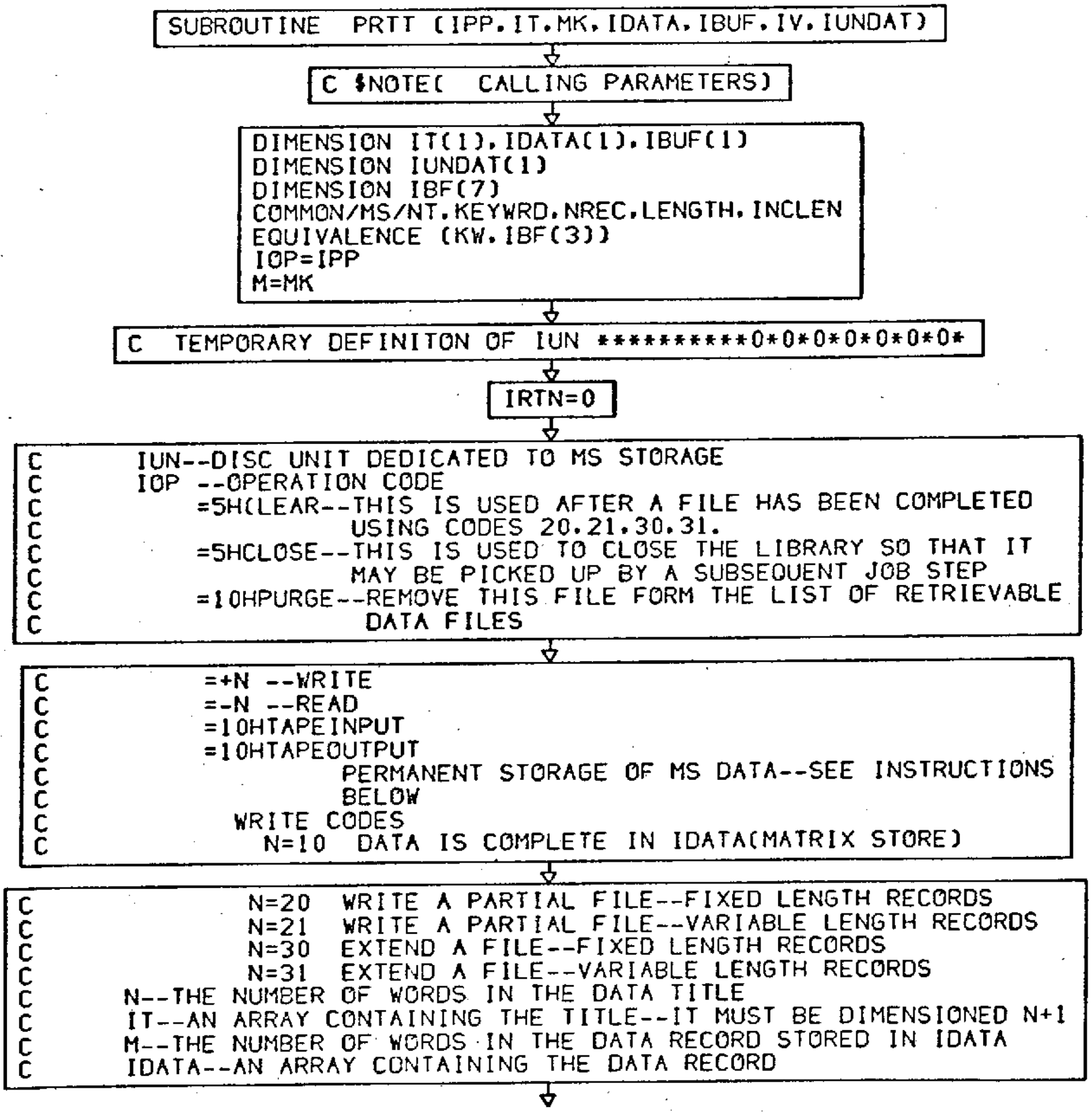

A0
GO TO 100
A1
IV(1)=I
IV(2)=NAMEOP(I)
CALL RANDAC(INSTAL ,CHAROP(I), 1,IOPR,NOPR,FOUND,IV,5,NFO)
1
100
CONTINUE
IV(2)=BLANK
DELIM=CHAROP(8)
RETURN
END



```
SUBROUTINE OPTION

C         THIS ROUTINE IS TO SET TO .TRUE. OPTIONS IN THE
C         DILOG COMMON BLOCK ACCORDING TO OTIN LETTERS ON
C         THE ENVOKING CARD.

COMMON/OPTDIR/NOP,NFOP,IDOP(52)
COMMON/DILOG/IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK )
EQUIVALENCE (IDILOG( 43), FIND  )
EQUIVALENCE (IDILOG(219), INITAL )
EQUIVALENCE (IDILOG(221), INSTAL)
LOGICAL FOUND,OPT
INTEGER LETTER(10),LOC(10)

INTEGER BLANK,FIND
DATA LETTER/1RA,1RB,1RC,1RE,1RM,1RL,1RO,1RD,1RS,1RT/
DATA LOC/340,292,36,250,311,300,301,304,305,307/
DATA NOP/13/
LOGICAL IDILOG
DEFINE OPT(I)=FLD(35-(1RZ-I),1,MASK).NE.0

CALL RANDAC(INITAL,BLANK,1,IDOP,NOP,FOUND,IV,4,NFOP)

CALL IOPT(MASK)

DO 100 I=1,10                - - - -   ⊳ 2

IV=LOC(I)
IDILOG(IV)=OPT(LETTER(I))

CALL GET(LETTER(I),6,K)

CALL RANDAC(FIND,K,1,IDOP,NOP,FOUND,IV,4,NFOP)
```

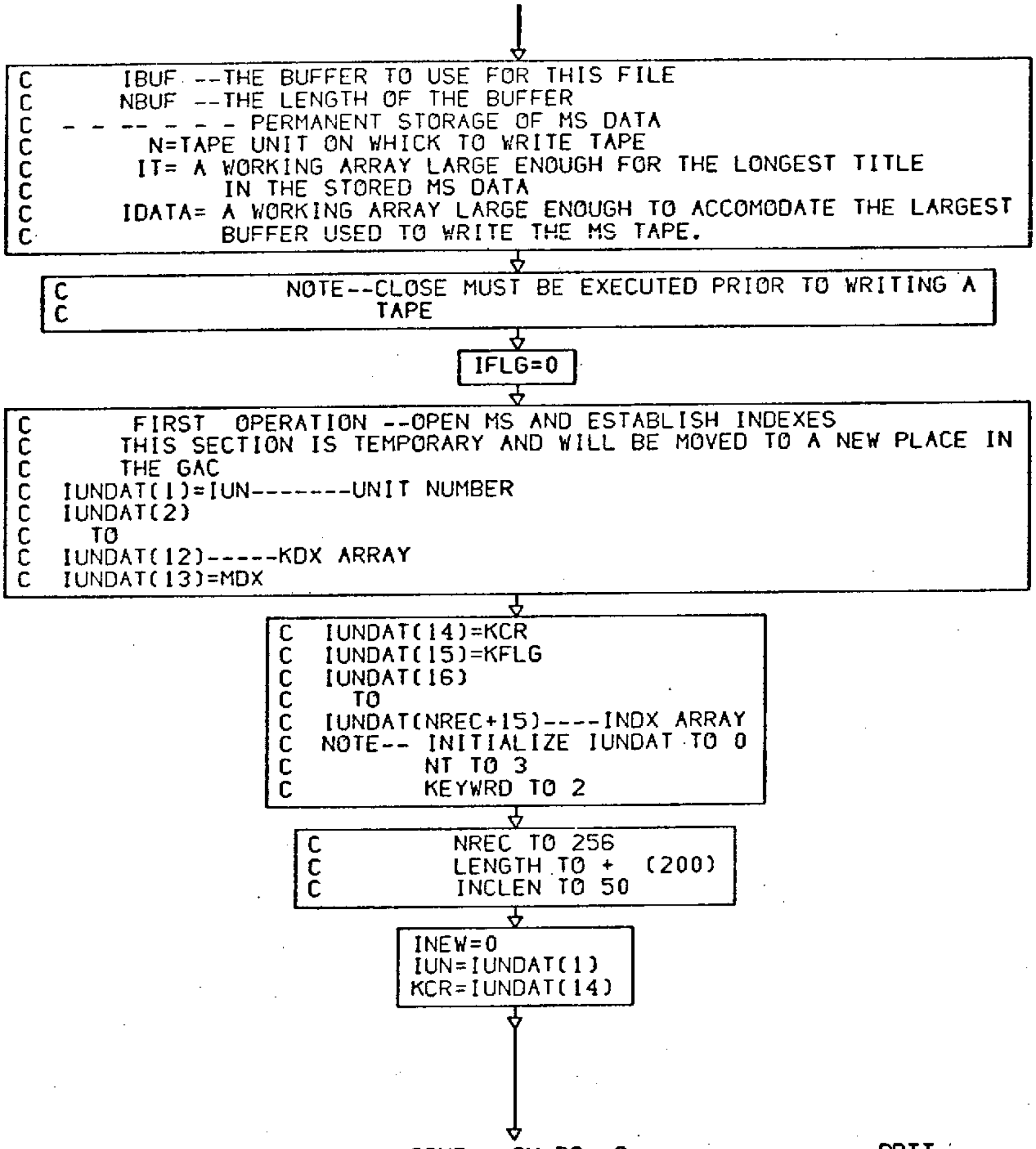
CONT. ON PG 2


IF(FOUND)
F
T
GO TO 100
CALL RANDAC(INSTAL,K,1,IDOP,NOP,FOUND,IV,4,NFOP)
1 - - - ▷ 100
CONTINUE
RETURN
END



```
SUBROUTINE  PRTT (IPP.IT.MK.IDATA.IBUF.IV.IUNDAT)

C $NOTE(  CALLING PARAMETERS)

DIMENSION IT(1).IDATA(1).IBUF(1)
DIMENSION IUNDAT(1)
DIMENSION IBF(7)
COMMON/MS/NT.KEYWRD.NREC.LENGTH.INCLEN
EQUIVALENCE (KW.IBF(3))
IOP=IPP
M=MK

C  TEMPORARY DEFINITON OF IUN **********0*0*0*0*0*0*0*

IRTN=0

C         IUN--DISC UNIT DEDICATED TO MS STORAGE
C         IOP --OPERATION CODE
C              =5HCLEAR--THIS IS USED AFTER A FILE HAS BEEN COMPLETED
C                        USING CODES 20.21.30.31.
C              =5HCLOSE--THIS IS USED TO CLOSE THE LIBRARY SO THAT IT
C                        MAY BE PICKED UP BY A SUBSEQUENT JOB STEP
C              =10HPURGE--REMOVE THIS FILE FORM THE LIST OF RETRIEVABLE
C                         DATA FILES

C              =+N --WRITE
C              =-N --READ
C              =10HTAPEINPUT
C              =10HTAPEOUTPUT
C                       PERMANENT STORAGE OF MS DATA--SEE INSTRUCTIONS
C                       BELOW
C                WRITE CODES
C                  N=10  DATA IS COMPLETE IN IDATA(MATRIX STORE)

C                 N=20  WRITE A PARTIAL FILE--FIXED LENGTH RECORDS
C                 N=21  WRITE A PARTIAL FILE--VARIABLE LENGTH RECORDS
C                 N=30  EXTEND A FILE--FIXED LENGTH RECORDS
C                 N=31  EXTEND A FILE--VARIABLE LENGTH RECORDS
C         N--THE NUMBER OF WORDS IN THE DATA TITLE
C         IT--AN ARRAY CONTAINING THE TITLE--IT MUST BE DIMENSIONED N+1
C         M--THE NUMBER OF WORDS IN THE DATA RECORD STORED IN IDATA
C         IDATA--AN ARRAY CONTAINING THE DATA RECORD
```

CONT. ON PG 2



```
C       IBUF --THE BUFFER TO USE FOR THIS FILE
C       NBUF --THE LENGTH OF THE BUFFER
C   - - -- - - - PERMANENT STORAGE OF MS DATA
C         N=TAPE UNIT ON WHICK TO WRITE TAPE
C          IT= A WORKING ARRAY LARGE ENOUGH FOR THE LONGEST TITLE
C               IN THE STORED MS DATA
C       IDATA= A WORKING ARRAY LARGE ENOUGH TO ACCOMODATE THE LARGEST
C               BUFFER USED TO WRITE THE MS TAPE.
```

```
C                 NOTE--CLOSE MUST BE EXECUTED PRIOR TO WRITING A
C                        TAPE
```

```
IFLG=0
```

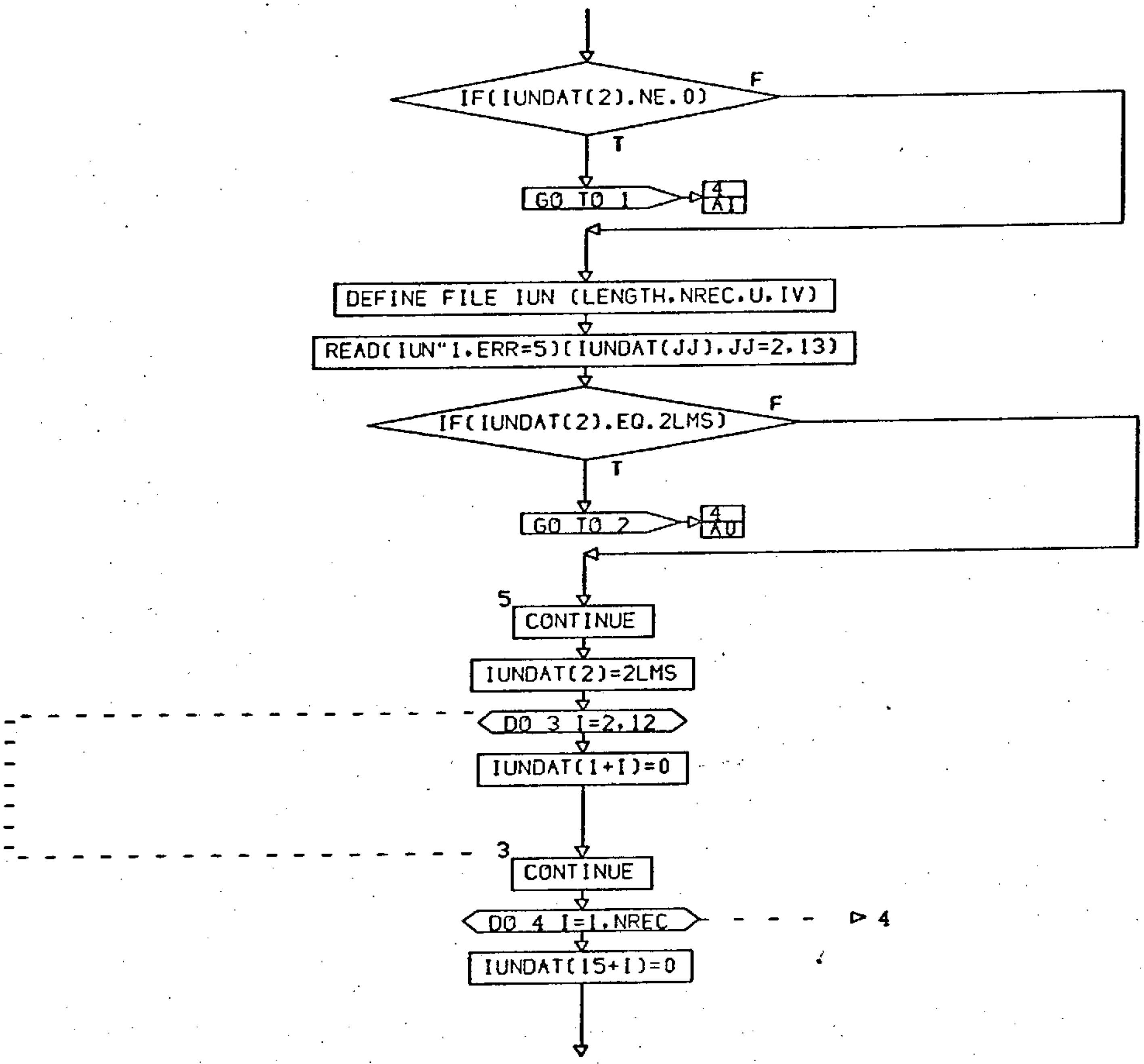
```
C        FIRST  OPERATION --OPEN MS AND ESTABLISH INDEXES
C       THIS SECTION IS TEMPORARY AND WILL BE MOVED TO A NEW PLACE IN
C       THE GAC
C   IUNDAT(1)=IUN-------UNIT NUMBER
C   IUNDAT(2)
C     TO
C   IUNDAT(12)-----KDX ARRAY
C   IUNDAT(13)=MDX
```

```
C   IUNDAT(14)=KCR
C   IUNDAT(15)=KFLG
C   IUNDAT(16)
C     TO
C   IUNDAT(NREC+15)----INDX ARRAY
C  NOTE-- INITIALIZE IUNDAT TO 0
C           NT TO 3
C           KEYWRD TO 2
```

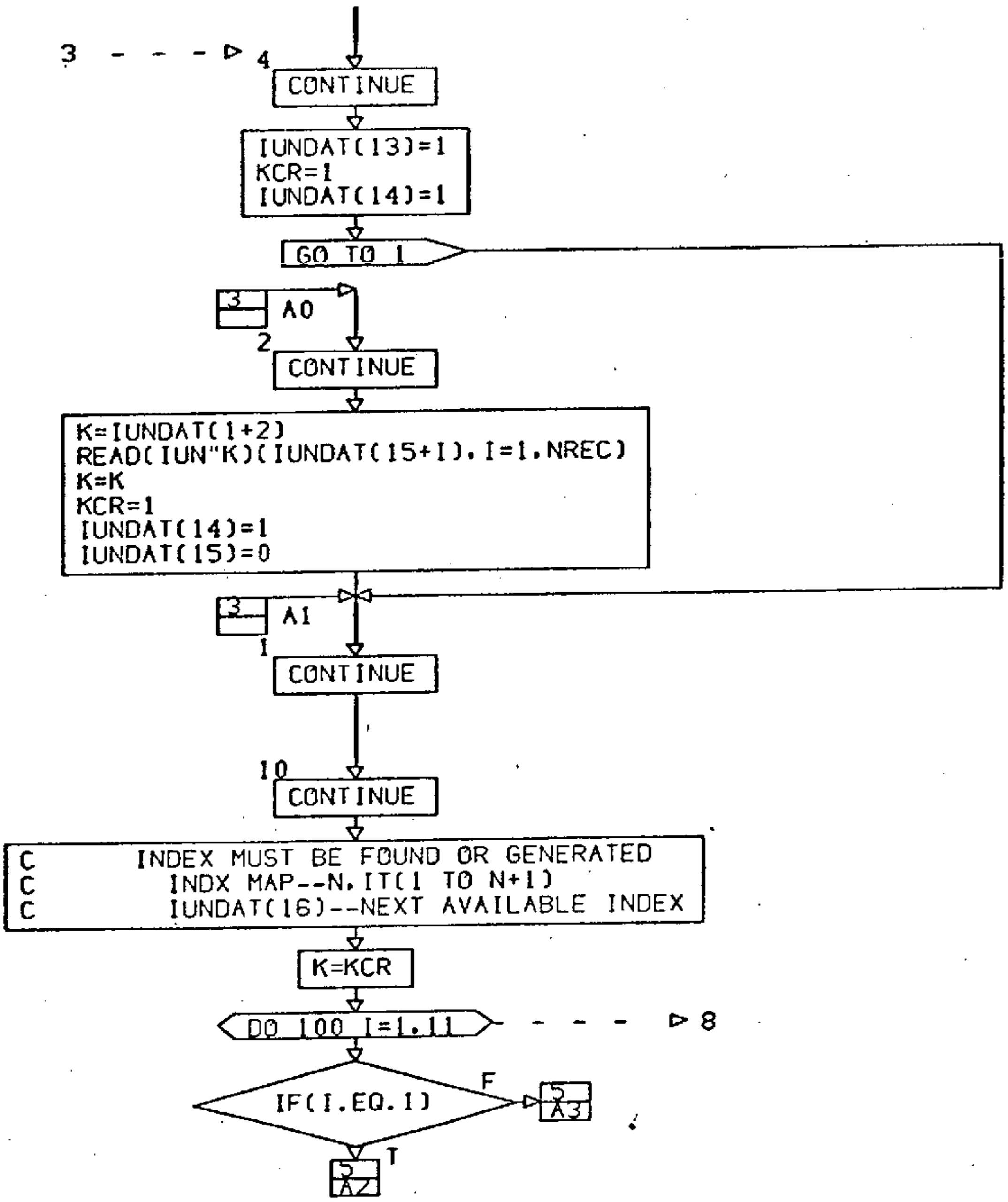
```
C          NREC TO 256
C          LENGTH TO +  (200)
C          INCLEN TO 50
```

```
INEW=0
IUN=IUNDAT(1)
KCR=IUNDAT(14)
```

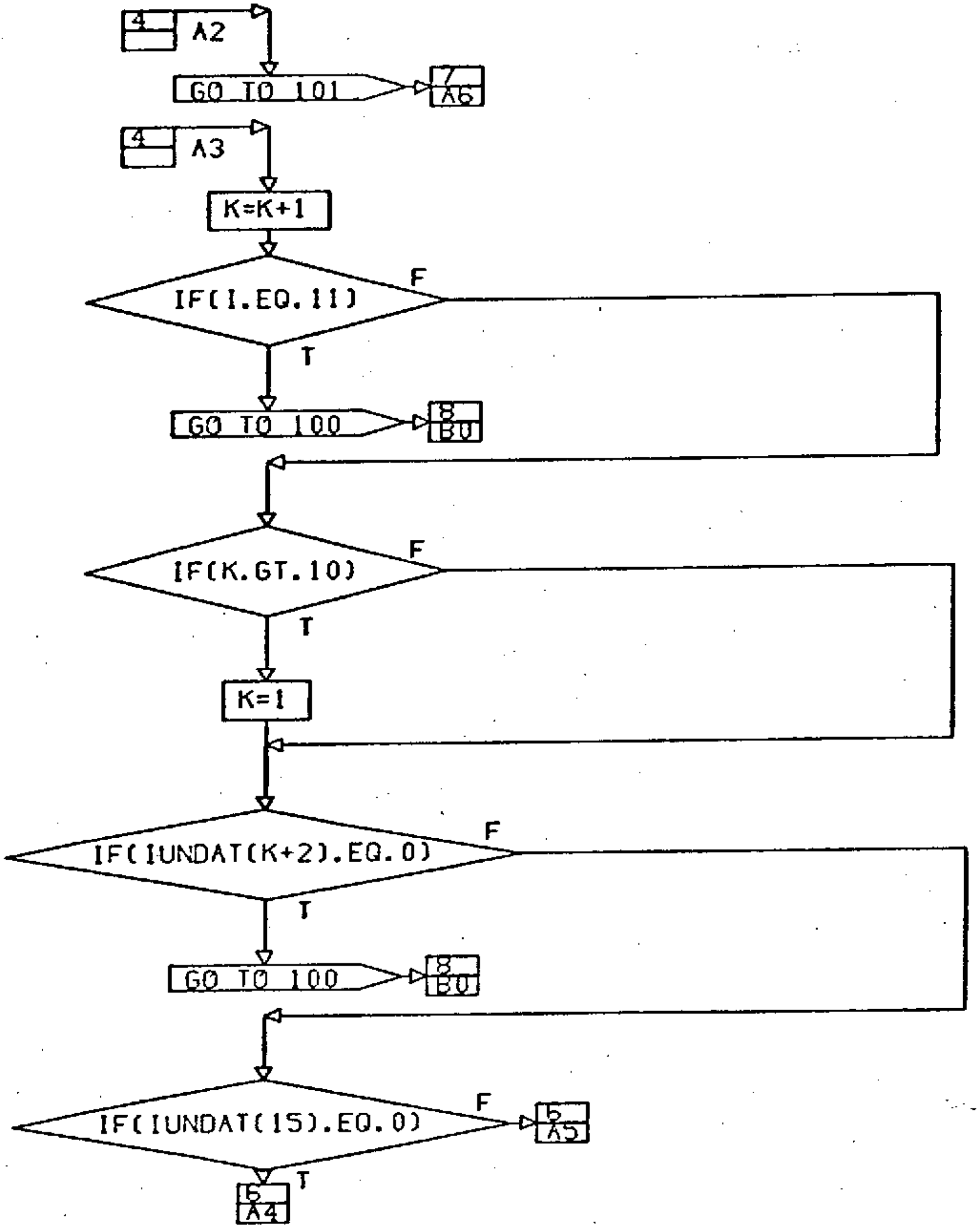
CONT. ON PG 3


IF(IUNDAT(2).NE.0)
F
T
GO TO 1
4
A1
DEFINE FILE IUN (LENGTH,NREC,U,IV)
READ(IUN"I,ERR=5)(IUNDAT(JJ),JJ=2,13)
IF(IUNDAT(2).EQ.2LMS)
F
T
GO TO 2
4
A0
5
CONTINUE
IUNDAT(2)=2LMS
DO 3 I=2,12
IUNDAT(1+I)=0
3
CONTINUE
DO 4 I=1,NREC
▷ 4
IUNDAT(15+I)=0

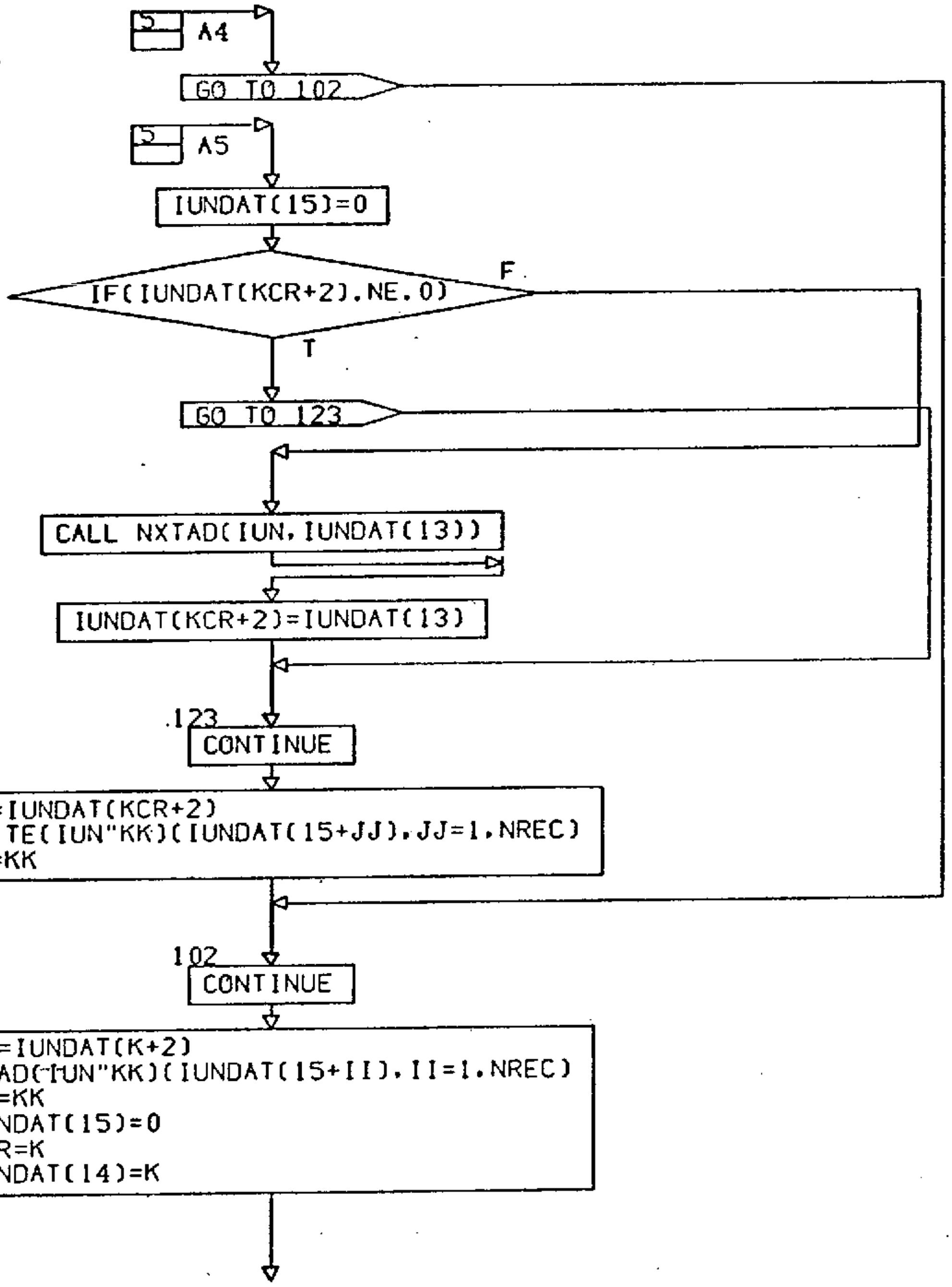
CONT. ON PG 4


3 - - - ▷ 4
CONTINUE
IUNDAT(13)=1
KCR=1
IUNDAT(14)=1
GO TO 1
3
A0
2
CONTINUE
K=IUNDAT(1+2)
READ(IUN"K)(IUNDAT(15+I),I=1,NREC)
K=K
KCR=1
IUNDAT(14)=1
IUNDAT(15)=0
3
A1
1
CONTINUE
10
CONTINUE
C INDEX MUST BE FOUND OR GENERATED
C INDX MAP--N,IT(1 TO N+1)
C IUNDAT(16)--NEXT AVAILABLE INDEX
K=KCR
DO 100 I=1,11
- - - - ▷ 8
IF(I.EQ.1)
F
5
A3
T
5
A2

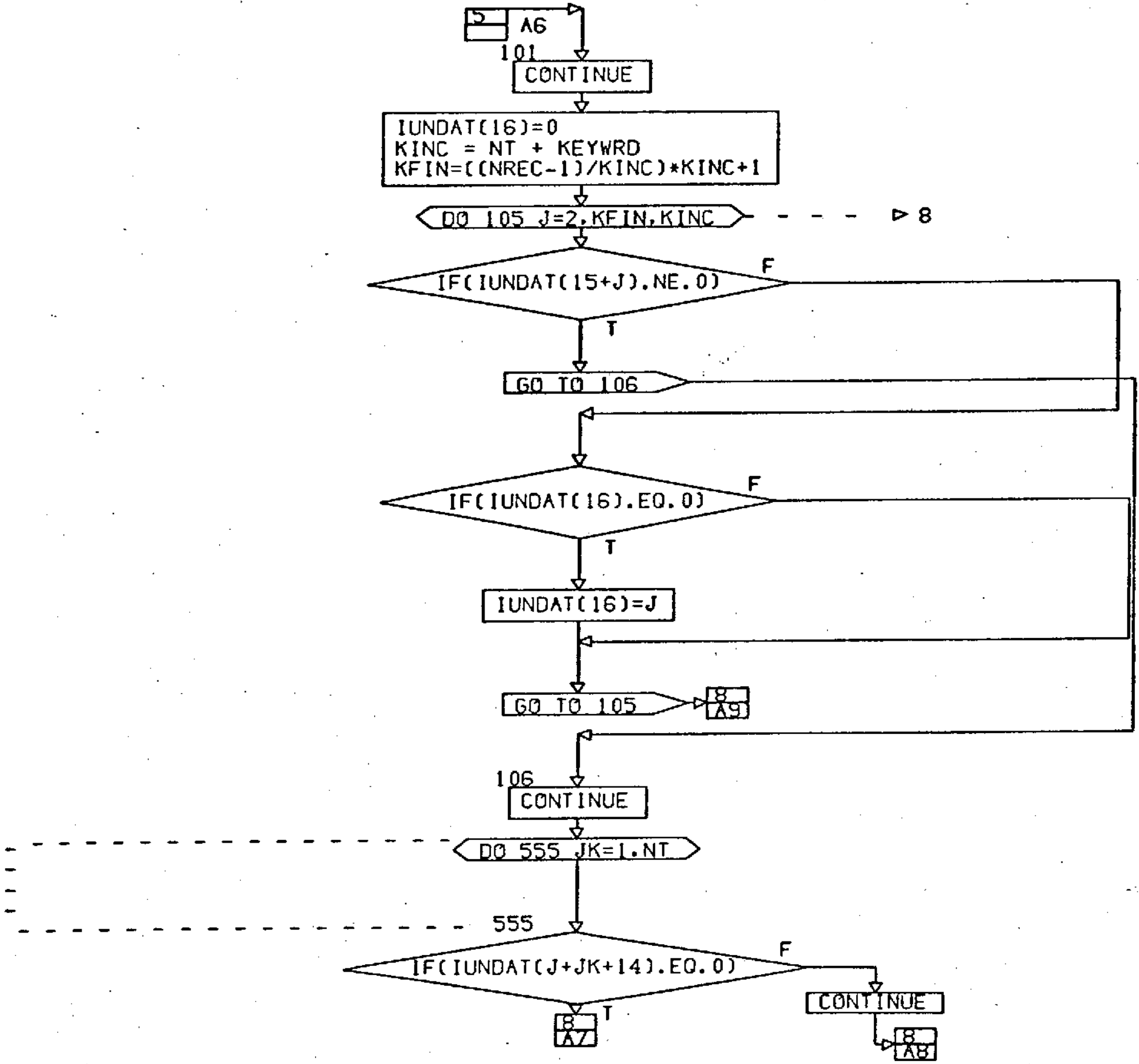
CONT. ON PG 5


4
A2
GO TO 101
7
A6
4
A3
K=K+1
IF(I.EQ.11)
F
T
GO TO 100
8
B0
IF(K.GT.10)
F
T
K=1
IF(IUNDAT(K+2).EQ.0)
F
T
GO TO 100
8
B0
IF(IUNDAT(15).EQ.0)
F
5
A5
T
6
A4

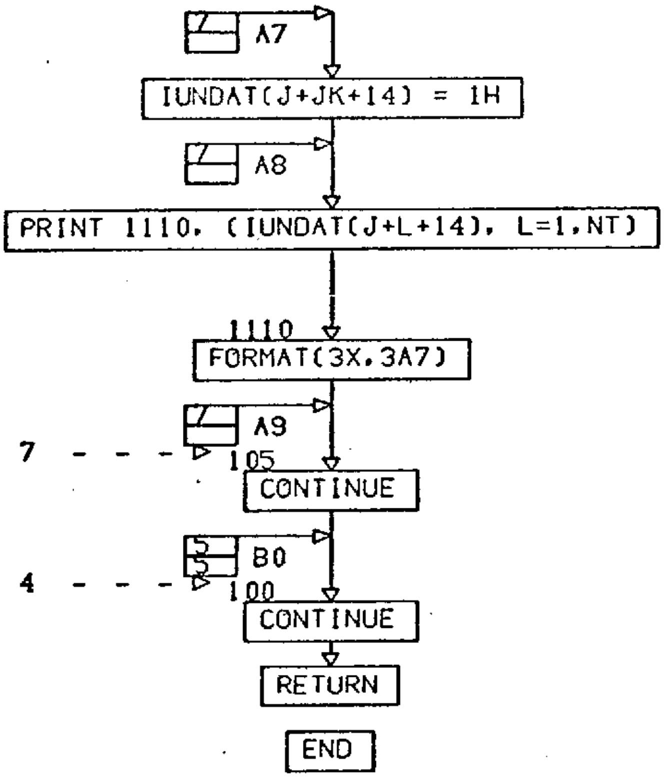
CONT. ON PG 6

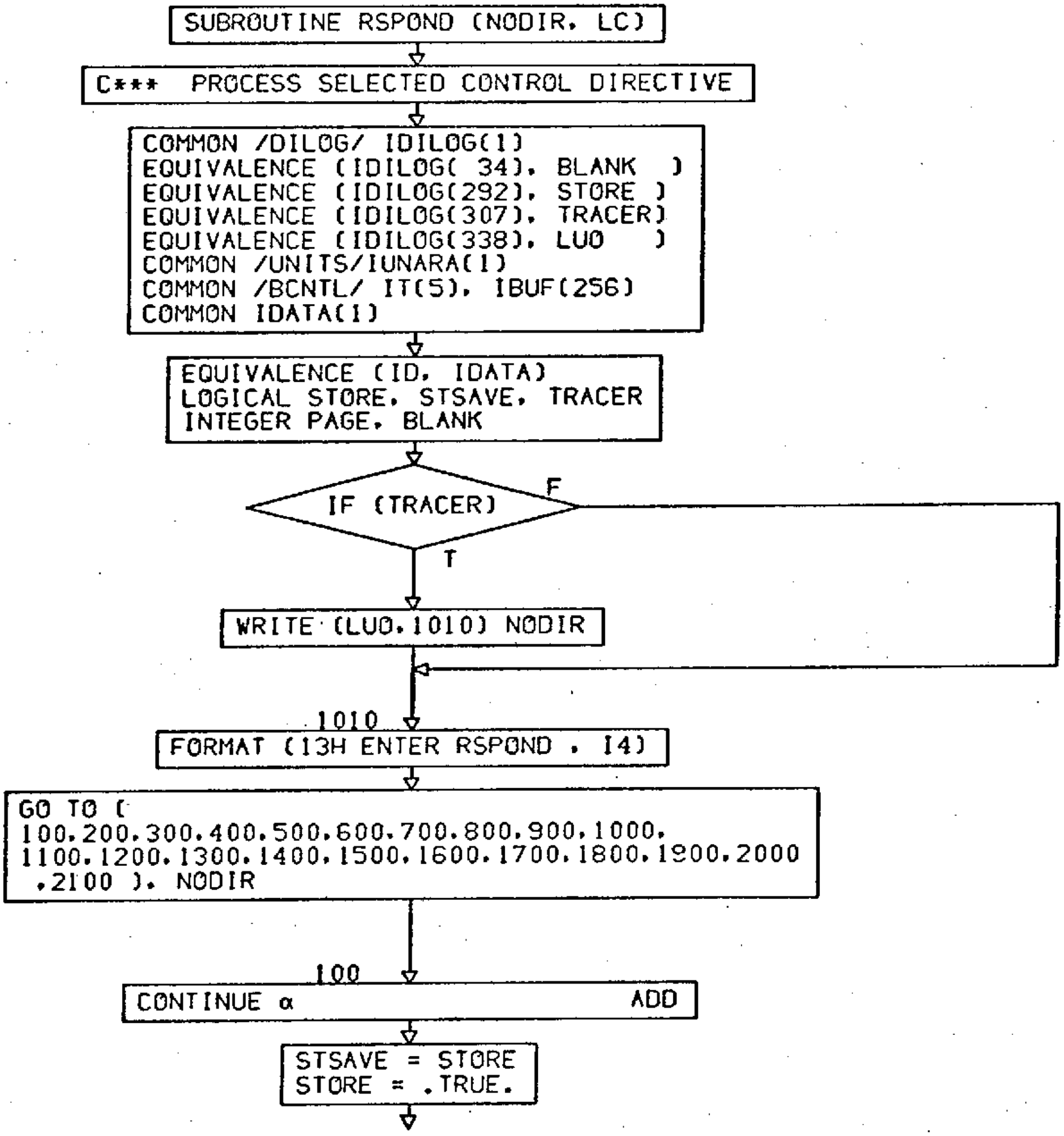

5
A4
GO TO 102
5
A5
IUNDAT(15)=0
IF(IUNDAT(KCR+2).NE.0)
F
T
GO TO 123
CALL NXTAD(IUN,IUNDAT(13))
IUNDAT(KCR+2)=IUNDAT(13)
123
CONTINUE
KK=IUNDAT(KCR+2)
WRITE(IUN"KK)(IUNDAT(15+JJ),JJ=1,NREC)
KK=KK
102
CONTINUE
KK=IUNDAT(K+2)
READ(IUN"KK)(IUNDAT(15+II),II=1,NREC)
KK=KK
IUNDAT(15)=0
KCR=K
IUNDAT(14)=K

CONT. ON PG 7


5
A6
101
CONTINUE
IUNDAT(16)=0
KINC = NT + KEYWRD
KFIN=((NREC-1)/KINC)*KINC+1
DO 105 J=2,KFIN,KINC
8
IF(IUNDAT(15+J).NE.0)
F
T
GO TO 106
IF(IUNDAT(16).EQ.0)
F
T
IUNDAT(16)=J
GO TO 105
8
A9
106
CONTINUE
DO 555 JK=1,NT
555
IF(IUNDAT(J+JK+14).EQ.0)
F
T
8
A7
CONTINUE
8
A8

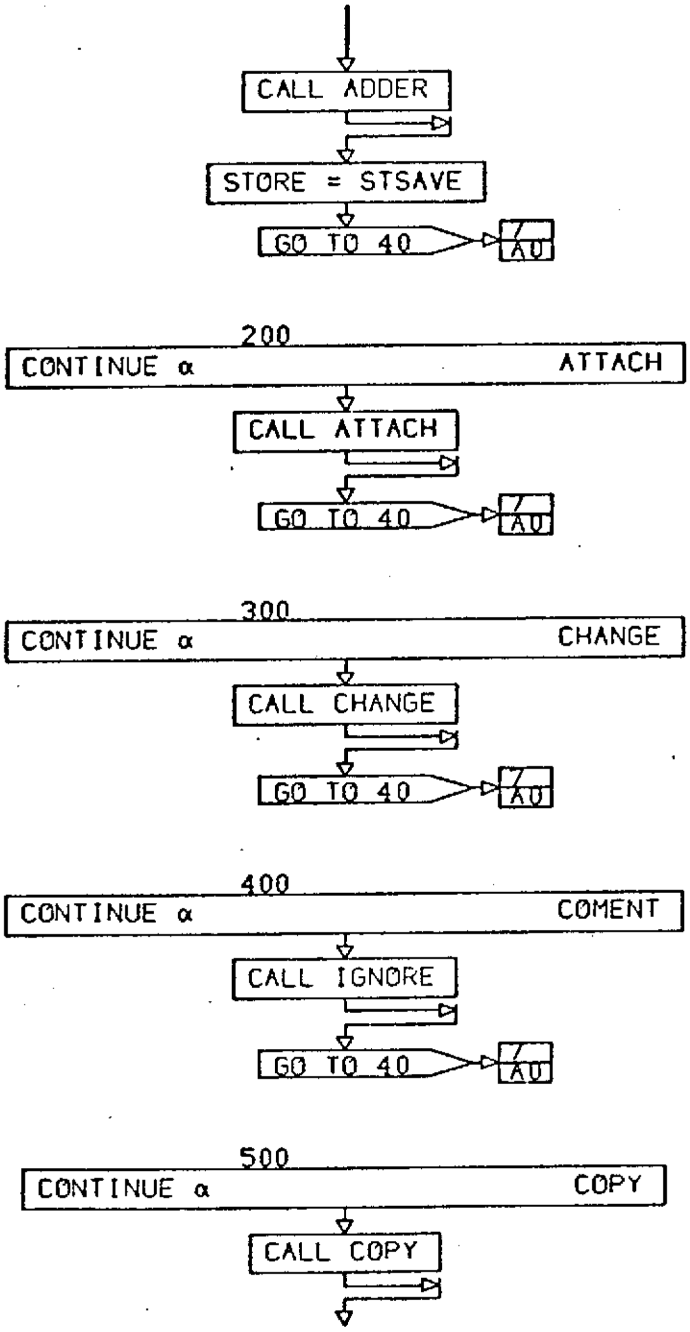
CONT. ON PG 8


A7
IUNDAT(J+JK+14) = 1H
A8
PRINT 1110, (IUNDAT(J+L+14), L=1,NT)
1110
FORMAT(3X,3A7)
A9
7
105
CONTINUE
B0
4
100
CONTINUE
RETURN
END

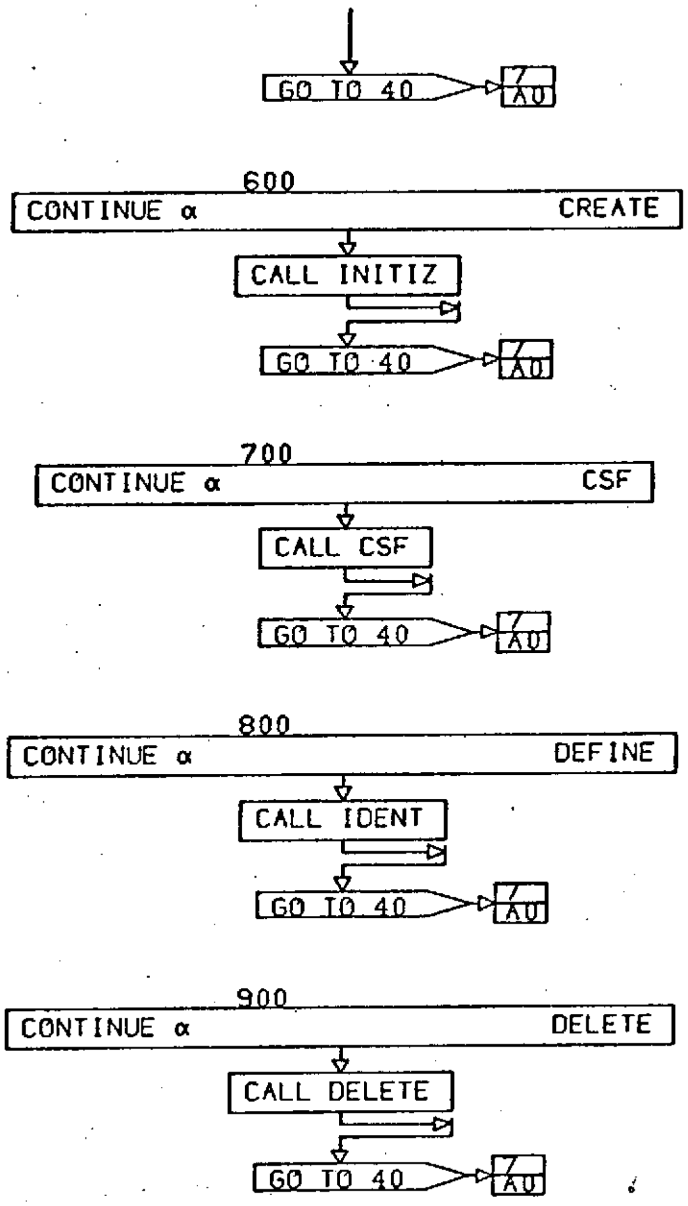


```
SUBROUTINE RSPOND (NODIR, LC)
C***  PROCESS SELECTED CONTROL DIRECTIVE
COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG( 34), BLANK  )
EQUIVALENCE (IDILOG(292), STORE )
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(338), LUO    )
COMMON /UNITS/IUNARA(1)
COMMON /BCNTL/ IT(5), IBUF(256)
COMMON IDATA(1)
EQUIVALENCE (ID, IDATA)
LOGICAL STORE, STSAVE, TRACER
INTEGER PAGE, BLANK
IF (TRACER)                                  F
    T
WRITE (LUO,1010) NODIR
1010
FORMAT (13H ENTER RSPOND , I4)
GO TO (
100,200,300,400,500,600,700,800,900,1000,
1100,1200,1300,1400,1500,1600,1700,1800,1900,2000
 ,2100 ), NODIR
100
CONTINUE α                                 ADD
STSAVE = STORE
STORE = .TRUE.
```

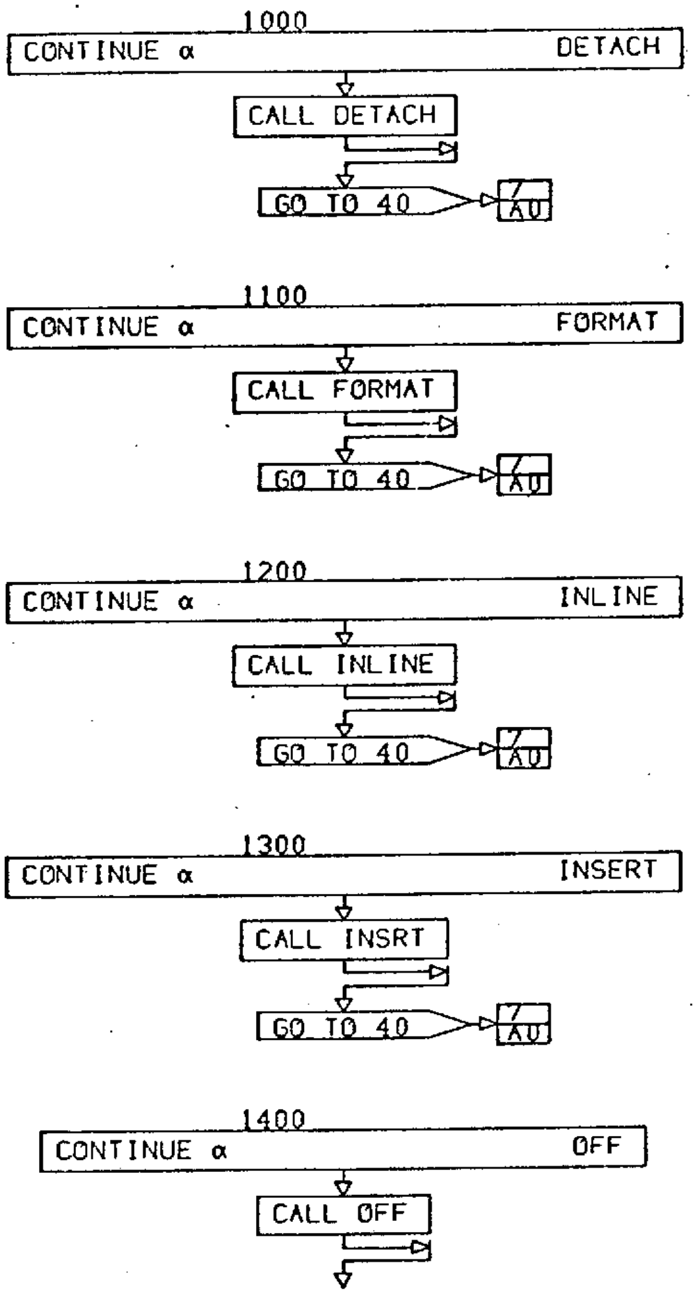
CONT. ON PG 2


CALL ADDER
STORE = STSAVE
GO TO 40
7
AU
200
CONTINUE α
ATTACH
CALL ATTACH
GO TO 40
7
AU
300
CONTINUE α
CHANGE
CALL CHANGE
GO TO 40
7
AU
400
CONTINUE α
COMENT
CALL IGNORE
GO TO 40
7
AU
500
CONTINUE α
COPY
CALL COPY

CONT. ON PG 3

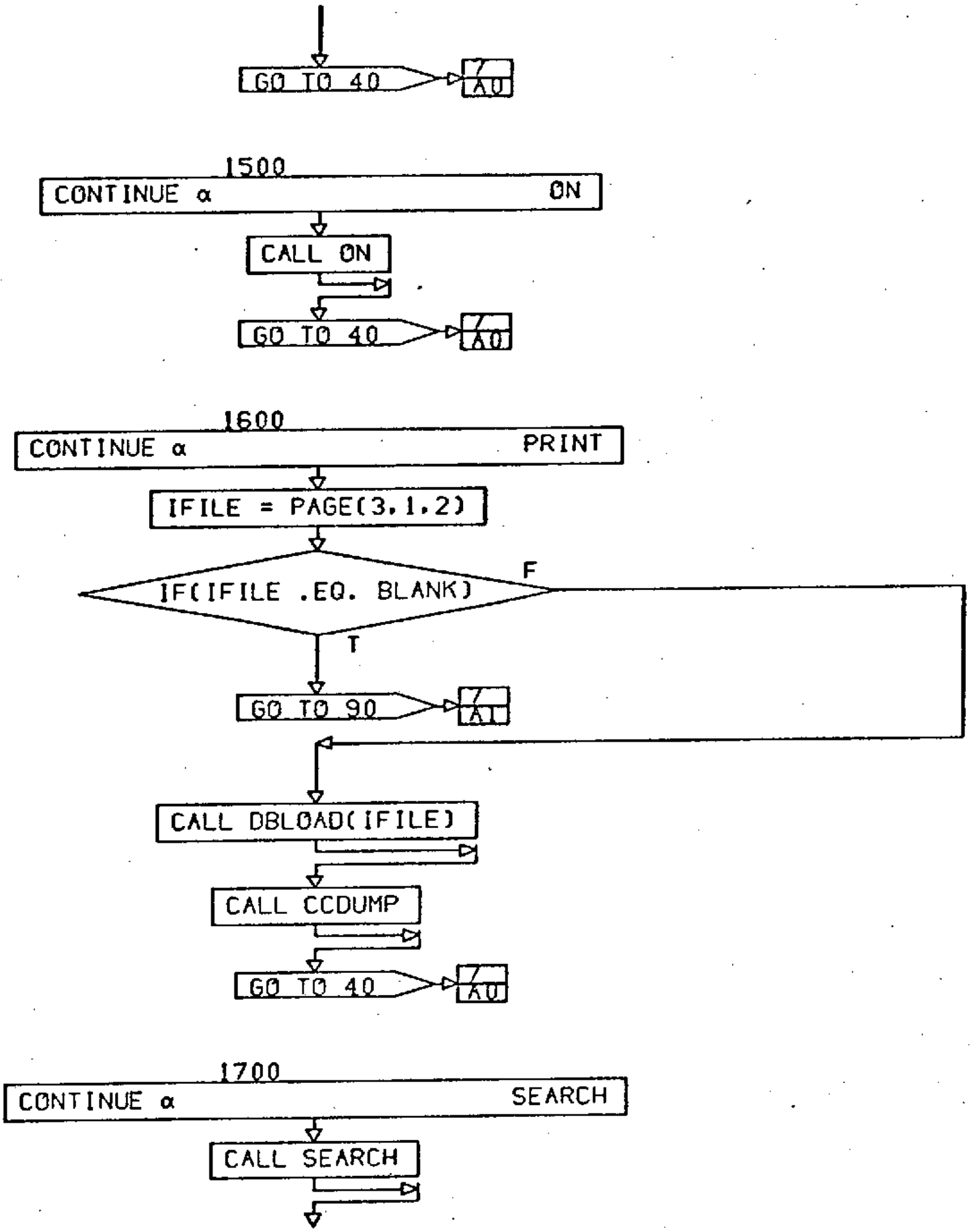

GO TO 40
7
AO
600
CONTINUE α
CREATE
CALL INITIZ
GO TO 40
7
AO
700
CONTINUE α
CSF
CALL CSF
GO TO 40
7
AO
800
CONTINUE α
DEFINE
CALL IDENT
GO TO 40
7
AO
900
CONTINUE α
DELETE
CALL DELETE
GO TO 40
7
AO

CONT. ON PG 4

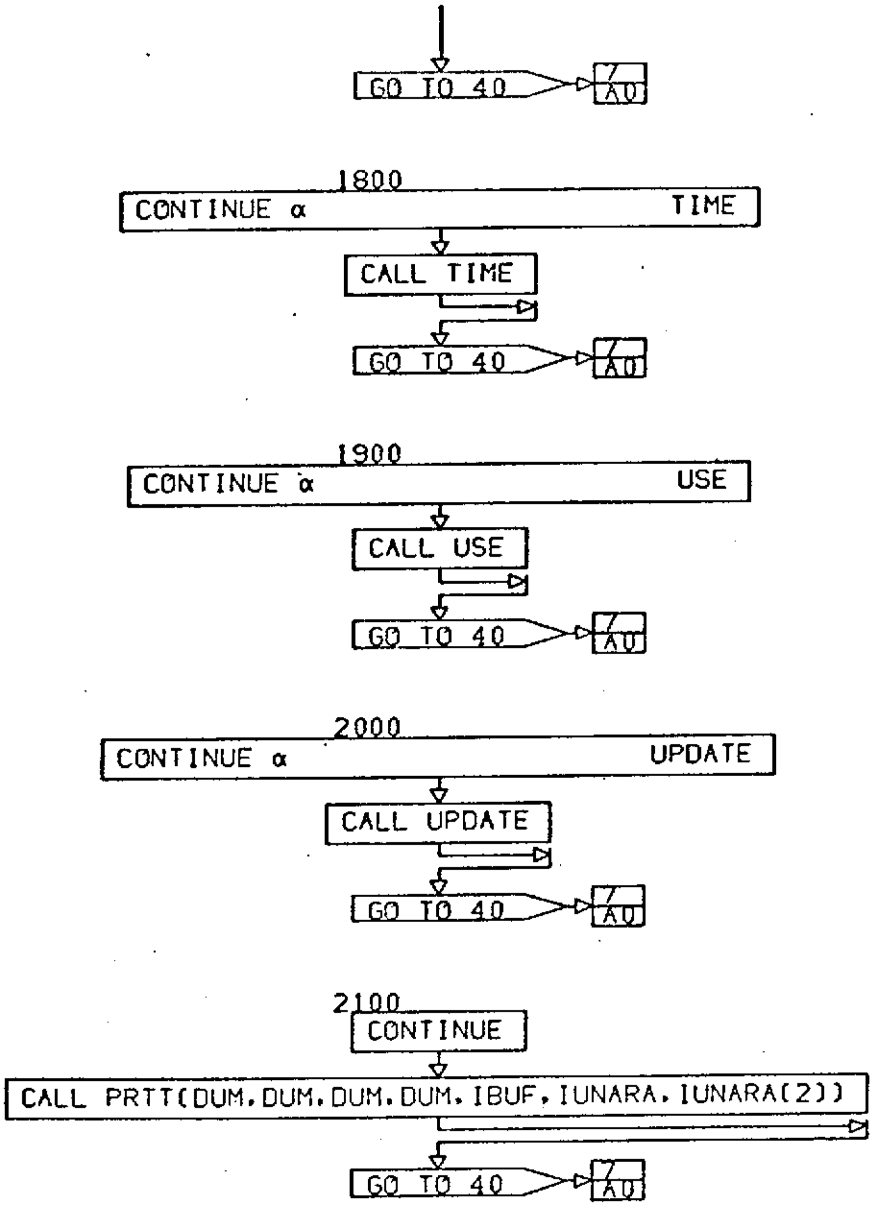


CONT. ON PG 5

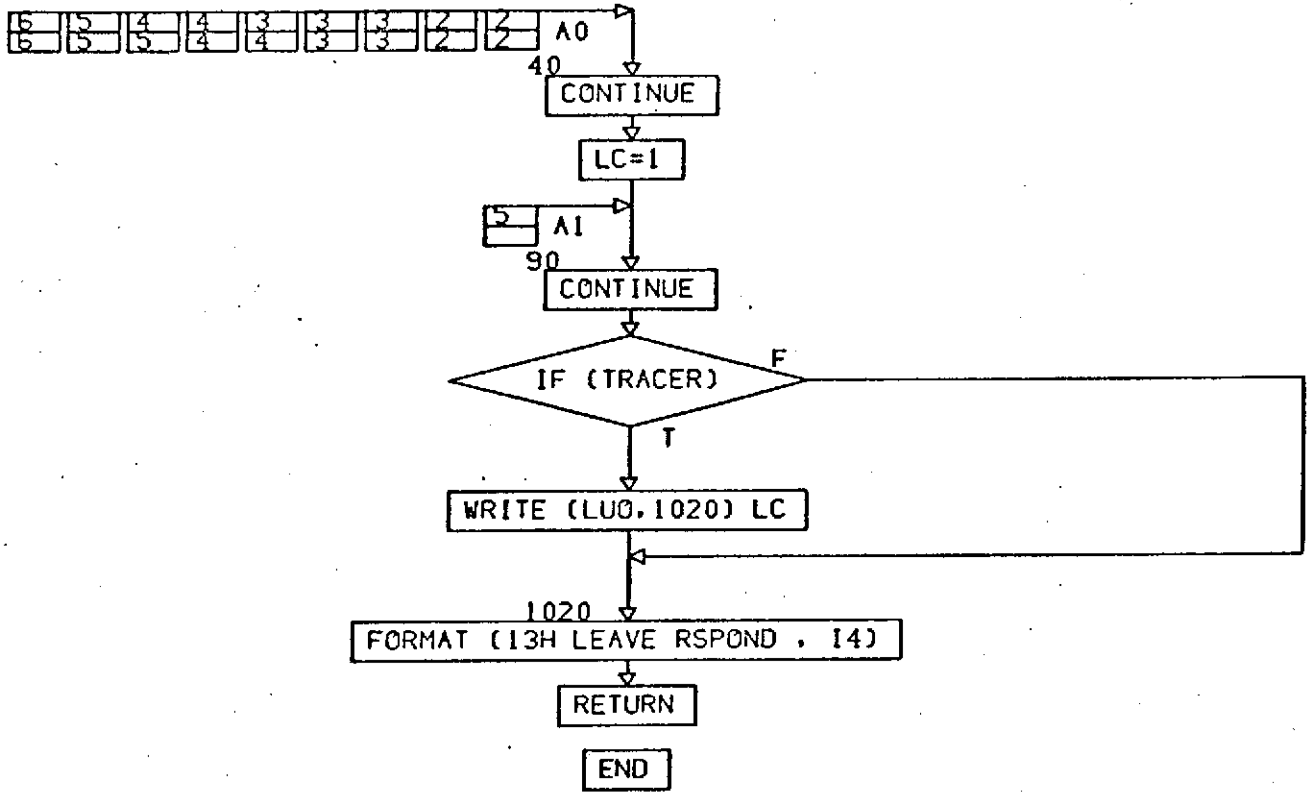
RSPOND
PG 4 OF 7

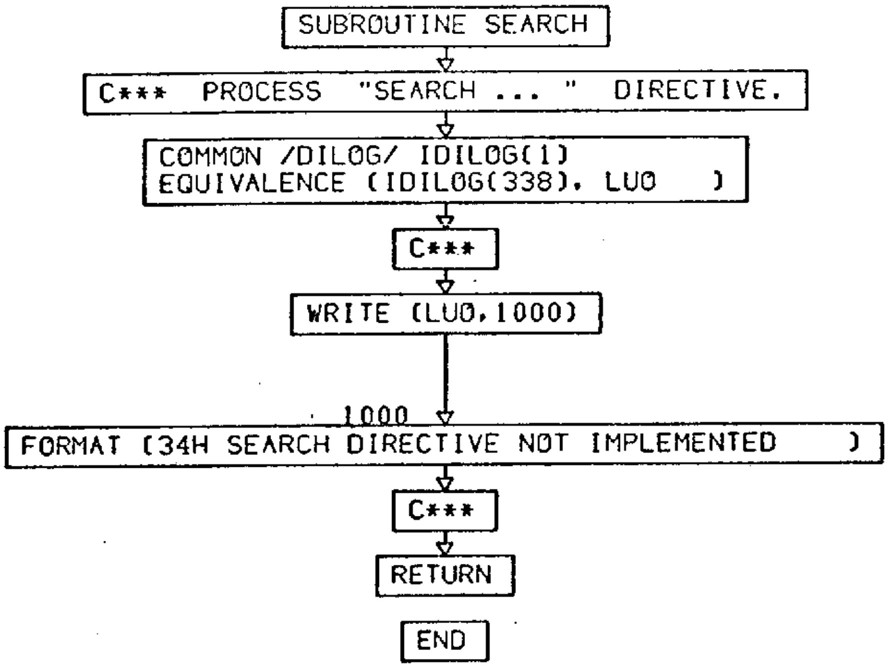

GO TO 40
7
A0
1500
CONTINUE α
ON
CALL ON
GO TO 40
7
A0
1600
CONTINUE α
PRINT
IFILE = PAGE(3,1,2)
IF(IFILE .EQ. BLANK)
F
T
GO TO 90
7
A1
CALL DBLOAD(IFILE)
CALL CCDUMP
GO TO 40
7
A0
1700
CONTINUE α
SEARCH
CALL SEARCH

CONT. ON PG 6

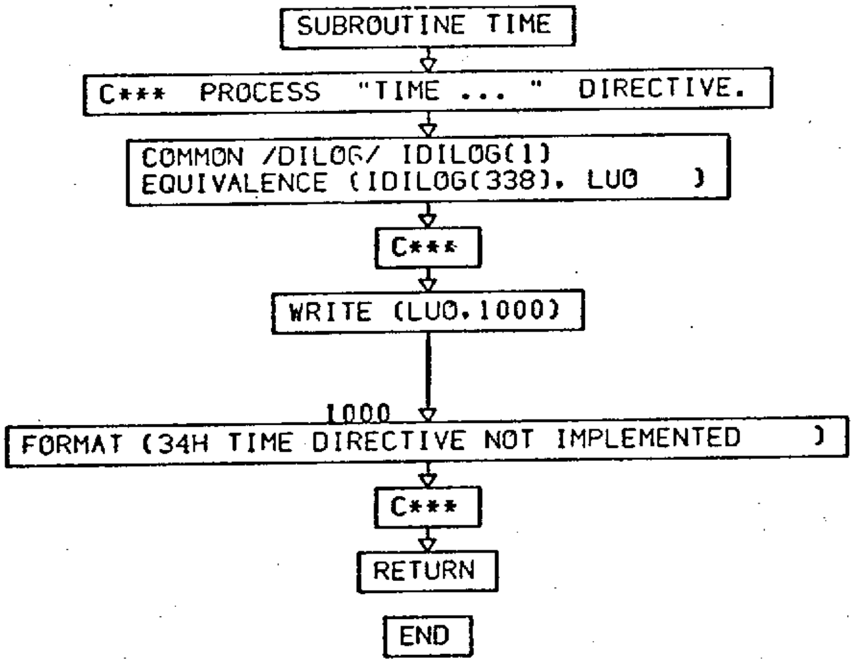

GO TO 40
7
AU
1800
CONTINUE α
TIME
CALL TIME
GO TO 40
7
AU
1900
CONTINUE α
USE
CALL USE
GO TO 40
7
AU
2000
CONTINUE α
UPDATE
CALL UPDATE
GO TO 40
7
AU
2100
CONTINUE
CALL PRTT(DUM.DUM.DUM.DUM.IBUF,IUNARA.IUNARA(2))
GO TO 40
7
AU

CONT. ON PG 7

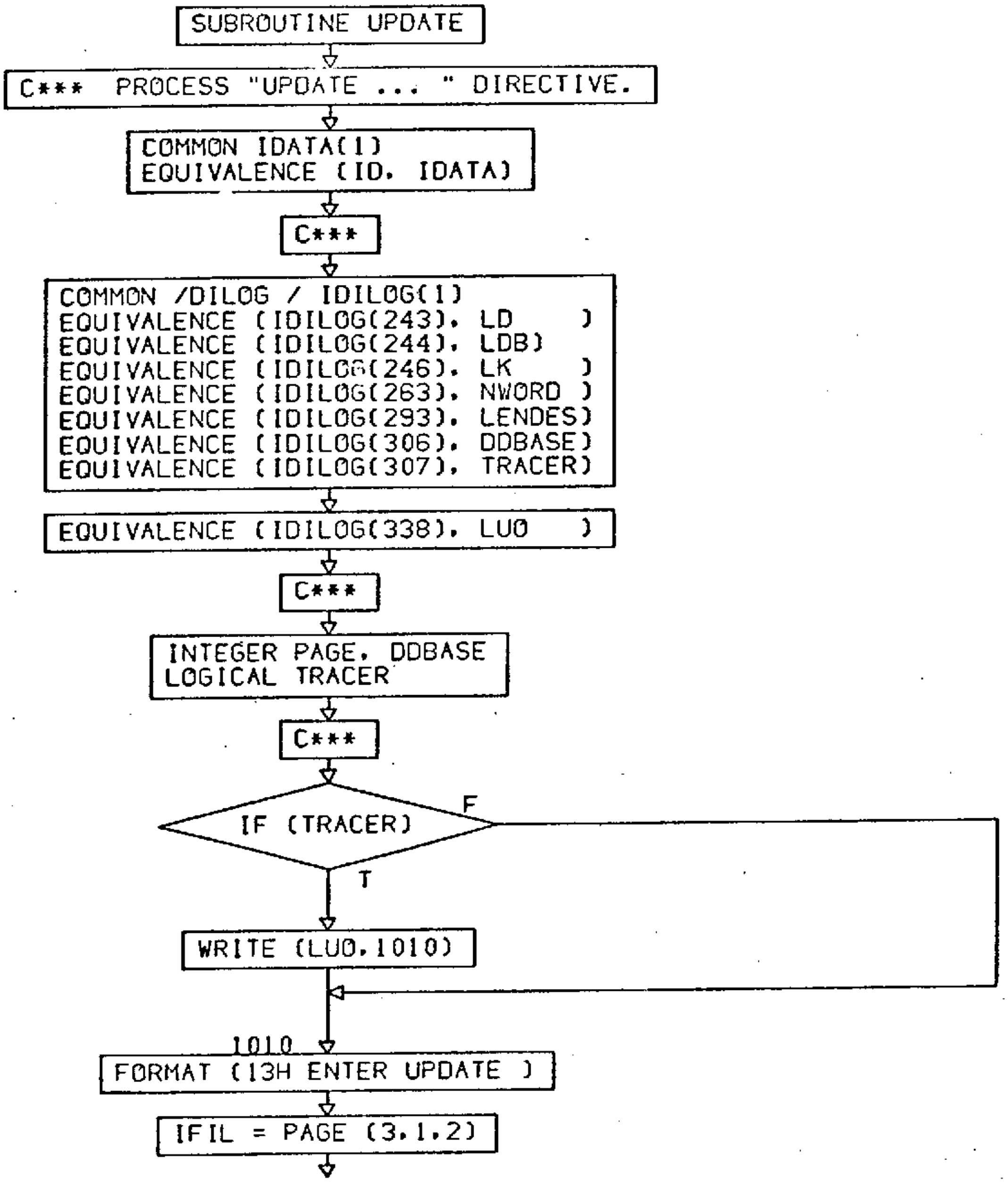

6
6
5
5
4
5
4
4
3
4
3
3
3
3
2
2
2
2
A0
40
CONTINUE
LC=1
5
A1
90
CONTINUE
IF (TRACER)
F
T
WRITE (LUO,1020) LC
1020
FORMAT (13H LEAVE RSPOND , I4)
RETURN
END



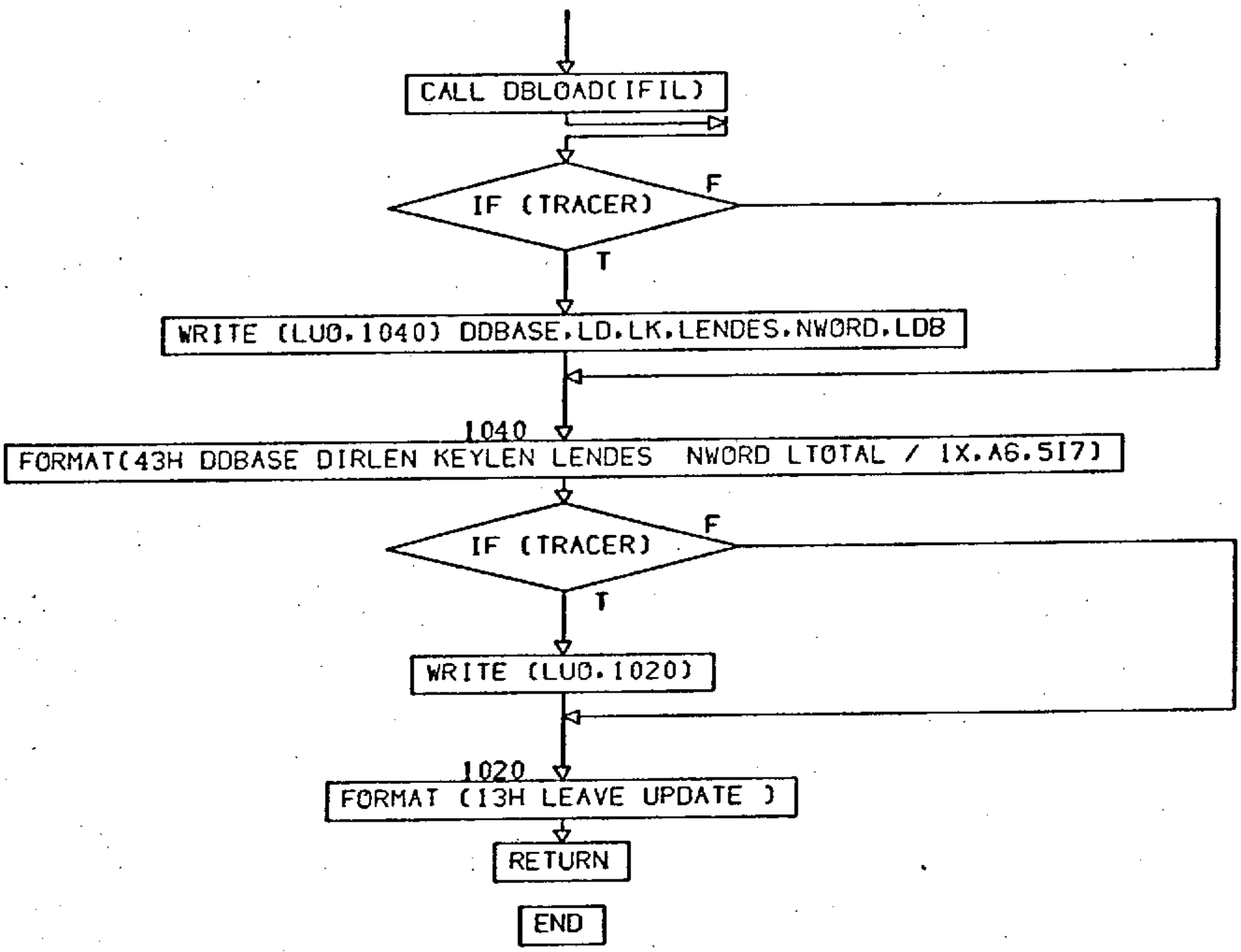
```
      SUBROUTINE SEARCH
C***  PROCESS  "SEARCH ... "  DIRECTIVE.
      COMMON /DILOG/ IDILOG(1)
      EQUIVALENCE (IDILOG(338), LUO    )
C***
      WRITE (LUO,1000)
 1000 FORMAT (34H SEARCH DIRECTIVE NOT IMPLEMENTED     )
C***
      RETURN
      END
```

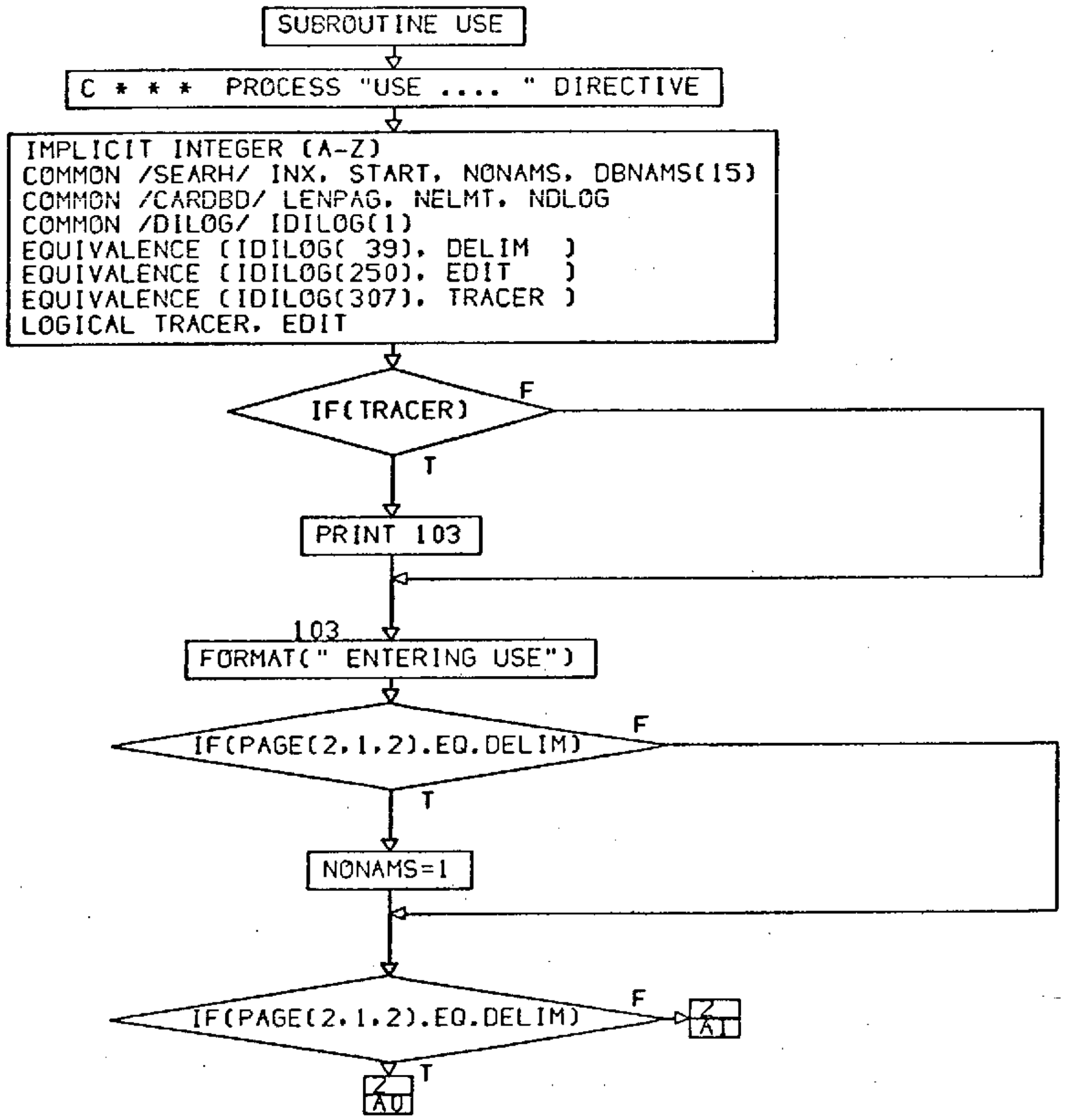


```
      SUBROUTINE TIME
C***  PROCESS  "TIME ... "  DIRECTIVE.
      COMMON /DILOG/ IDILOG(1)
      EQUIVALENCE (IDILOG(338), LUO    )
C***
      WRITE (LUO,1000)
 1000 FORMAT (34H TIME DIRECTIVE NOT IMPLEMENTED      )
C***
      RETURN
      END
```


SUBROUTINE UPDATE
C*** PROCESS "UPDATE ... " DIRECTIVE.
COMMON IDATA(1)
EQUIVALENCE (ID, IDATA)
C***
COMMON /DILOG / IDILOG(1)
EQUIVALENCE (IDILOG(243), LD   )
EQUIVALENCE (IDILOG(244), LDB)
EQUIVALENCE (IDILOG(246), LK   )
EQUIVALENCE (IDILOG(263), NWORD)
EQUIVALENCE (IDILOG(293), LENDES)
EQUIVALENCE (IDILOG(306), DDBASE)
EQUIVALENCE (IDILOG(307), TRACER)
EQUIVALENCE (IDILOG(338), LUO   )
C***
INTEGER PAGE, DDBASE
LOGICAL TRACER
C***
IF (TRACER)
F
T
WRITE (LUO,1010)
1010
FORMAT (13H ENTER UPDATE)
IFIL = PAGE (3,1,2)

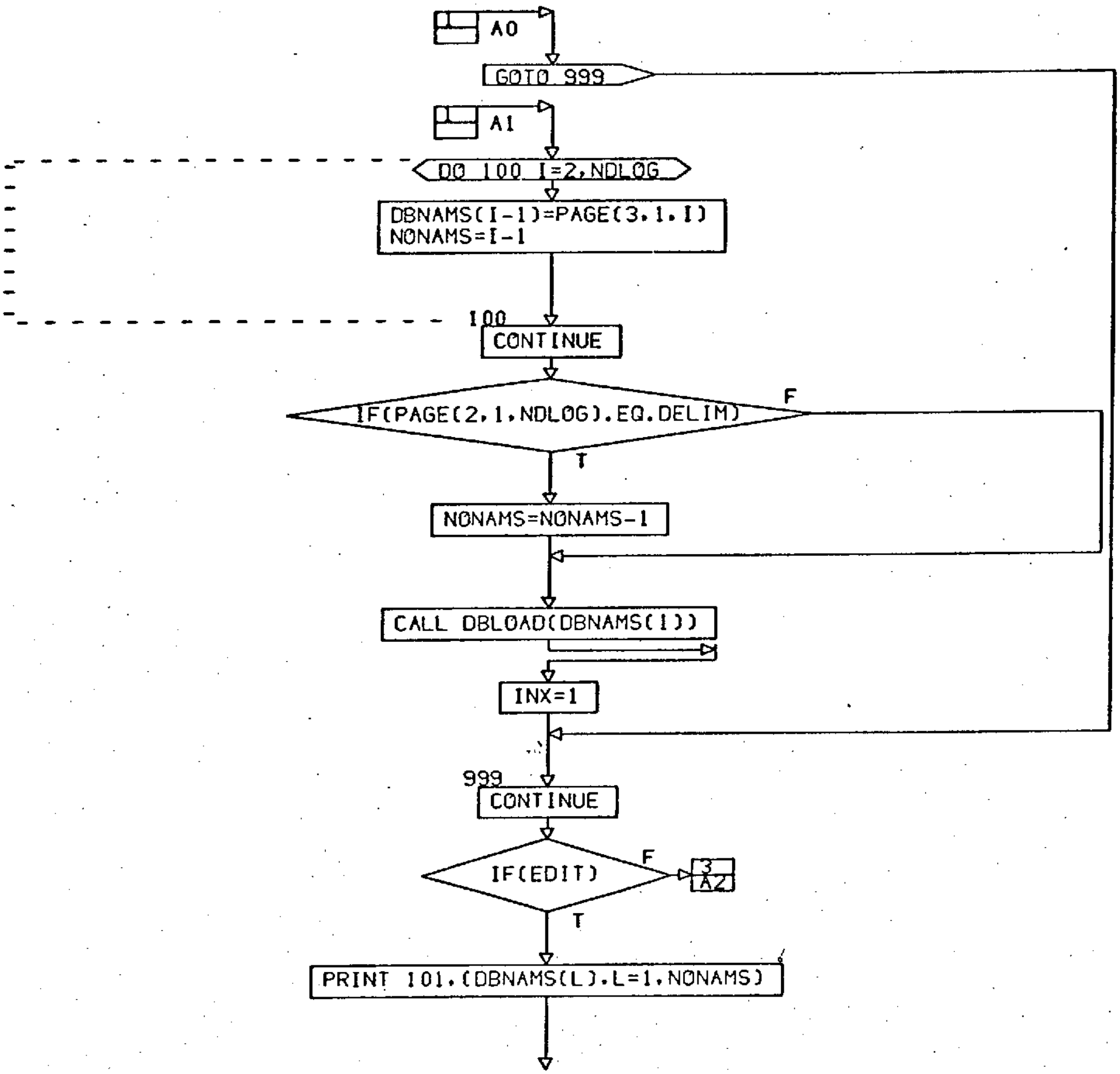
CONT. ON PG 2


CALL DBLOAD(IFIL)
IF (TRACER)
F
T
WRITE (LUO,1040) DDBASE,LD,LK,LENDES,NWORD,LDB
1040
FORMAT(43H DDBASE DIRLEN KEYLEN LENDES NWORD LTOTAL / 1X,A6,5I7)
IF (TRACER)
F
T
WRITE (LUO,1020)
1020
FORMAT (13H LEAVE UPDATE)
RETURN
END



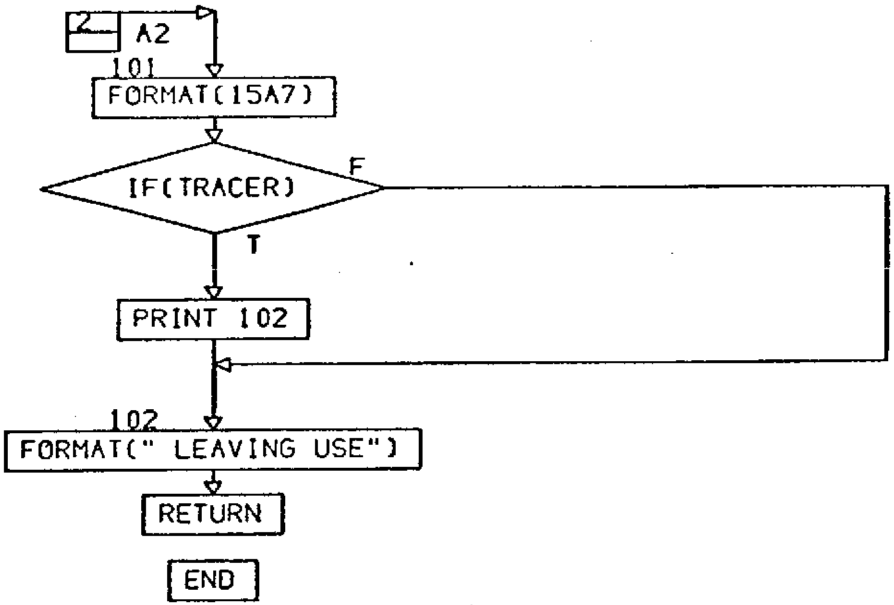
```
SUBROUTINE USE
C * * *  PROCESS "USE .... " DIRECTIVE
IMPLICIT INTEGER (A-Z)
COMMON /SEARH/ INX, START, NONAMS, DBNAMS(15)
COMMON /CARDBD/ LENPAG, NELMT, NDLOG
COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG( 39), DELIM  )
EQUIVALENCE (IDILOG(250), EDIT   )
EQUIVALENCE (IDILOG(307), TRACER )
LOGICAL TRACER, EDIT
IF(TRACER)
PRINT 103
103 FORMAT(" ENTERING USE")
IF(PAGE(2,1,2).EQ.DELIM)
NONAMS=1
IF(PAGE(2,1,2).EQ.DELIM)
```

CONT. ON PG 2



CONT. ON PG 3

USE
PG 2 OF 3



USE
PG 3 FINAL