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
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
**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.
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>
<thead>
<tr>
<th>Summary of Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>3</td>
</tr>
<tr>
<td>PROGRAM STRUCTURE</td>
<td>6</td>
</tr>
<tr>
<td>Concepts and Definitions</td>
<td>6</td>
</tr>
<tr>
<td>File Designations</td>
<td>8</td>
</tr>
<tr>
<td>Processor Specifications</td>
<td>8</td>
</tr>
<tr>
<td>Control Statement</td>
<td></td>
</tr>
<tr>
<td>Option Specifications</td>
<td></td>
</tr>
<tr>
<td>Syntax Definition</td>
<td>9</td>
</tr>
<tr>
<td>Summary of DLG Directives</td>
<td>10</td>
</tr>
<tr>
<td>Descriptions of Control Directives</td>
<td></td>
</tr>
<tr>
<td>Processor Interface</td>
<td>14</td>
</tr>
<tr>
<td>Usage</td>
<td>15</td>
</tr>
<tr>
<td>Restrictions</td>
<td></td>
</tr>
<tr>
<td>DMAN Software Package</td>
<td></td>
</tr>
<tr>
<td>DMAN Usage</td>
<td>16</td>
</tr>
<tr>
<td>A Discussion of IT</td>
<td>17</td>
</tr>
<tr>
<td>A Discussion of IOP</td>
<td></td>
</tr>
<tr>
<td>Technology Module Interface Package</td>
<td>19</td>
</tr>
<tr>
<td>Subroutine Descriptions</td>
<td>21</td>
</tr>
<tr>
<td>Subroutine ADDER</td>
<td></td>
</tr>
<tr>
<td>Subroutine ADDONE</td>
<td></td>
</tr>
<tr>
<td>Subroutine ANLSIS</td>
<td></td>
</tr>
<tr>
<td>Subroutine BCDDDB</td>
<td></td>
</tr>
<tr>
<td>Subroutine BCDDDEC</td>
<td></td>
</tr>
<tr>
<td>Subroutine BCDINT</td>
<td></td>
</tr>
<tr>
<td>Subroutine BCDVAL</td>
<td>22</td>
</tr>
<tr>
<td>Subroutine BILDOP</td>
<td></td>
</tr>
<tr>
<td>Subroutine CCDUMP</td>
<td></td>
</tr>
<tr>
<td>Subroutine CDINIT</td>
<td></td>
</tr>
<tr>
<td>Subroutine CHANGE</td>
<td></td>
</tr>
<tr>
<td>Subroutine CHARS</td>
<td></td>
</tr>
<tr>
<td>Subroutine CHRSNUM</td>
<td></td>
</tr>
<tr>
<td>Subroutine CREATEF</td>
<td></td>
</tr>
<tr>
<td>Subroutine CSF</td>
<td>23</td>
</tr>
<tr>
<td>Subroutine DBADD</td>
<td></td>
</tr>
<tr>
<td>Subroutine DBINIT</td>
<td></td>
</tr>
<tr>
<td>Subroutine DBLOAD</td>
<td></td>
</tr>
<tr>
<td>Subroutine DBWRT</td>
<td></td>
</tr>
<tr>
<td>Subroutine DECBCD</td>
<td></td>
</tr>
<tr>
<td>Subroutine DECIDE</td>
<td></td>
</tr>
<tr>
<td>Subroutine DELETE</td>
<td></td>
</tr>
<tr>
<td>Subroutine DIALEK</td>
<td></td>
</tr>
<tr>
<td>Subroutine DISECT</td>
<td>24</td>
</tr>
<tr>
<td>Main Program DLGDVR</td>
<td></td>
</tr>
</tbody>
</table>

PRECEDING PAGE BLANK NOT FILMED
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAN</td>
<td>24</td>
</tr>
<tr>
<td>DYNCOR</td>
<td></td>
</tr>
<tr>
<td>ENDFL</td>
<td></td>
</tr>
<tr>
<td>EOFTST</td>
<td></td>
</tr>
<tr>
<td>FLDATA</td>
<td></td>
</tr>
<tr>
<td>FORMAT</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>25</td>
</tr>
<tr>
<td>GETSUB</td>
<td></td>
</tr>
<tr>
<td>IDENT</td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td></td>
</tr>
<tr>
<td>IGNORE</td>
<td></td>
</tr>
<tr>
<td>INITDM</td>
<td></td>
</tr>
<tr>
<td>INITIZ</td>
<td></td>
</tr>
<tr>
<td>INITL</td>
<td></td>
</tr>
<tr>
<td>INMOD</td>
<td>26</td>
</tr>
<tr>
<td>INSRT</td>
<td></td>
</tr>
<tr>
<td>INTBCD</td>
<td></td>
</tr>
<tr>
<td>IOPT</td>
<td></td>
</tr>
<tr>
<td>IVCALC</td>
<td></td>
</tr>
<tr>
<td>IVDESC</td>
<td></td>
</tr>
<tr>
<td>LOCOP</td>
<td></td>
</tr>
<tr>
<td>MOVER</td>
<td></td>
</tr>
<tr>
<td>NLADD</td>
<td></td>
</tr>
<tr>
<td>NUMNIT</td>
<td>27</td>
</tr>
<tr>
<td>ONROFF</td>
<td></td>
</tr>
<tr>
<td>OPINIT</td>
<td></td>
</tr>
<tr>
<td>OPTION</td>
<td></td>
</tr>
<tr>
<td>PAGDMP</td>
<td>28</td>
</tr>
<tr>
<td>PAGE</td>
<td></td>
</tr>
<tr>
<td>PRINTF</td>
<td></td>
</tr>
<tr>
<td>PRTT</td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td></td>
</tr>
<tr>
<td>RPLACE</td>
<td></td>
</tr>
<tr>
<td>RSPOND</td>
<td>29</td>
</tr>
<tr>
<td>SCALE</td>
<td></td>
</tr>
<tr>
<td>SHELL</td>
<td></td>
</tr>
<tr>
<td>STRMOV</td>
<td></td>
</tr>
<tr>
<td>UPDATE</td>
<td></td>
</tr>
<tr>
<td>USE</td>
<td></td>
</tr>
<tr>
<td>VALING</td>
<td></td>
</tr>
<tr>
<td>WRITBR</td>
<td>30</td>
</tr>
<tr>
<td>WRITCR</td>
<td></td>
</tr>
<tr>
<td>COMMON VARIABLES</td>
<td>31</td>
</tr>
<tr>
<td>OVERLAY STRUCTURE</td>
<td>40</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

APPENDIX A - FLOW CHARTS OF SELECTED SUBROUTINES

<table>
<thead>
<tr>
<th>Routine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDER</td>
<td>A1</td>
</tr>
<tr>
<td>ANALYSIS</td>
<td>A16</td>
</tr>
<tr>
<td>ATTACH</td>
<td>A18</td>
</tr>
<tr>
<td>CCDUMP</td>
<td>A19</td>
</tr>
<tr>
<td>CDINIT</td>
<td>A23</td>
</tr>
<tr>
<td>CHANGE</td>
<td>A25</td>
</tr>
<tr>
<td>COPY</td>
<td>A30</td>
</tr>
<tr>
<td>CSF</td>
<td>A31</td>
</tr>
<tr>
<td>DELETE</td>
<td>A34</td>
</tr>
<tr>
<td>DETACH</td>
<td>A37</td>
</tr>
<tr>
<td>DIALEX</td>
<td>A38</td>
</tr>
<tr>
<td>DLGDRV</td>
<td>A44</td>
</tr>
<tr>
<td>FORMAT</td>
<td>A45</td>
</tr>
<tr>
<td>IDENT</td>
<td>A49</td>
</tr>
<tr>
<td>IGNORE</td>
<td>A56</td>
</tr>
<tr>
<td>INITDM</td>
<td>A62</td>
</tr>
<tr>
<td>INITZ</td>
<td>A63</td>
</tr>
<tr>
<td>INITL</td>
<td>A67</td>
</tr>
<tr>
<td>INLINE</td>
<td>A74</td>
</tr>
<tr>
<td>INMOD</td>
<td>A75</td>
</tr>
<tr>
<td>INRT</td>
<td>A81</td>
</tr>
<tr>
<td>NUMNIT</td>
<td>A91</td>
</tr>
<tr>
<td>ONROFF</td>
<td>A93</td>
</tr>
<tr>
<td>OPINIT</td>
<td>A95</td>
</tr>
<tr>
<td>OPTION</td>
<td>A97</td>
</tr>
<tr>
<td>PRTT</td>
<td>A99</td>
</tr>
<tr>
<td>RESPOND</td>
<td>A107</td>
</tr>
<tr>
<td>SEARCH</td>
<td>A114</td>
</tr>
<tr>
<td>TIME</td>
<td>A115</td>
</tr>
<tr>
<td>UPDATE</td>
<td>A116</td>
</tr>
<tr>
<td>USE</td>
<td>A118</td>
</tr>
</tbody>
</table>
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
FILE FORMATS

(elt1)

'CREATE...
'DEFINE...
@XQT A
123456
'ABC'
'ADD Q=..
'XYZ'
'ARA'

(elt2)

@XQT A
123456
25.327

NMLIST

$HAB X=33...,
ARA=2.,3.,4.,...
...
$END

$IN Q=14,
IAR=10,11,12,
...
$END

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.
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 (RESPOND) 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:

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>An absolute program element which is executed with a special Exec 8 processor control statement:</td>
</tr>
<tr>
<td></td>
<td>@name elt1,elt2</td>
</tr>
<tr>
<td></td>
<td>and which is interfaced with the elements named on the processor control statements.</td>
</tr>
<tr>
<td>Data Base</td>
<td>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.</td>
</tr>
<tr>
<td>Technology Module (Application Program)</td>
<td>An independent computer program which will receive or generate data base information.</td>
</tr>
<tr>
<td>Interrogation</td>
<td>The process of retrieving information from the data base. The disposition of the retrieved data is dependent upon the directive employed.</td>
</tr>
<tr>
<td>Directive (Also Command)</td>
<td>A language element used to specify a DLG Processor's action or function.</td>
</tr>
<tr>
<td>Data Management</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.elt1,1fn.elt2

1fn.elt1 Source Input (See I Option)
1fn.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' 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 @CKPT @RSPAR
@ASG @FREE @RSTRT
@BRKPT @LOG @START
@CAT @MODE @SYM
@CKPAR @QUAL @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:

```fortran
COMMON/UNITS/IAREA(273)
DATA IAREA/O,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:

```fortran
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:

\[ N=0 \text{ and } IDATA(1)=3LEOD. \]

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 n'\text{name}'.
- 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:

\[ \text{NAMGEN (NAME, K, J)} \]

- \( \text{NAME} \) = The desired root name.
- \( \text{I} \) = Concatenated number occupying the first one or two BCD character positions beyond the root name.
- \( \text{J} \) = Concatenated number occupying the second one or two BCD character positions beyond the root name.

An example would be:

\[ \text{NAM}=\text{NAMGEN (4HNAME,1,2)} \]

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

\[ \text{NAM}=6H\text{NAME12} \]

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

Usually the \text{NAMGEN} function is used in conjunction with the \text{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 BCDDDB. - BCDDDB 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 existence 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 NWORLD, 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 DIAMEK 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 described 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:

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

*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 replace-
ment 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 (alphabetically least 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 sub-
routine 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 start-
ing 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 minimum.

Subroutine VALIMG. - VALIMG is a Fortran subroutine for convert-
ing 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

<table>
<thead>
<tr>
<th>DILOG Locations</th>
<th>Value</th>
<th>Local Name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'E'</td>
<td>ALFE</td>
<td>Integer word containing the character (E), left justified and blank filled.</td>
</tr>
<tr>
<td>2</td>
<td>'F'</td>
<td>ALFF</td>
<td>Integer word containing the character (F), left justified and blank filled.</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>BCD(20)</td>
<td>Integer array of BCD characters used for scratch purposes.</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>BCDLEN</td>
<td>The number of characters in the BCD array.</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>BCDNUM(10)</td>
<td>Integer array containing powers of ten in sequential order from 0 to 9.</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>BLANK</td>
<td>Integer word containing blank characters.</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>COMMA</td>
<td>Integer word containing the character (,), left justified and blank filled.</td>
</tr>
<tr>
<td>36</td>
<td>.TRUE.</td>
<td>CONTIN</td>
<td>Logical variable set by option flag 'C'. If true, an end-of-record will signify the end of a command.</td>
</tr>
<tr>
<td>37</td>
<td>27</td>
<td>DBASE</td>
<td>Logical unit of the file containing the design data base.</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td>ICOPY</td>
<td>A counter for the cumulation of input operations on the logical unit SI element.</td>
</tr>
<tr>
<td>DILOG Locations</td>
<td>Value</td>
<td>Local Name</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>39</td>
<td>'('</td>
<td>DELIM</td>
<td>Integer word containing the DIALOG delimiter used in the simulation input data, usually has a value of ('').</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>ICONT</td>
<td>A counter for cumulating the number of output operations on the logical unit SO element.</td>
</tr>
<tr>
<td>41</td>
<td>'='</td>
<td>EQUAL</td>
<td>Integer word containing the character (=), left justified and blank filled.</td>
</tr>
<tr>
<td>42</td>
<td>8</td>
<td>MXCHAR</td>
<td>The maximum number of characters which can be used for interpreting a number in BCD format calculated as: [ \text{MXCHAR} = \text{NWORD} \times \text{NCAR} - (\text{LENEXP} - 1) ]</td>
</tr>
<tr>
<td>43</td>
<td>2</td>
<td>FIND</td>
<td>An integer word defining the FIND entry in RANDAC.</td>
</tr>
<tr>
<td>44</td>
<td>None</td>
<td>ICHAR(140)</td>
<td>Integer array containing the input image one character per word, left justified and blank filled.</td>
</tr>
<tr>
<td>184</td>
<td>None</td>
<td>IMAGE(36)</td>
<td>Integer array containing the input IMAGE.</td>
</tr>
<tr>
<td>219</td>
<td>1</td>
<td>INITIAL</td>
<td>An integer word which defines the initialization entry in RANDAC.</td>
</tr>
<tr>
<td>220</td>
<td>80</td>
<td>INRECL</td>
<td>Maximum number of characters in the input record (IMAGE).</td>
</tr>
<tr>
<td>DILOG Locations</td>
<td>Value</td>
<td>Local NAME</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>221</td>
<td>3</td>
<td>INSTAL</td>
<td>An integer word which defines the installation entry in RANDAC.</td>
</tr>
<tr>
<td>223</td>
<td>None</td>
<td>IV(20)</td>
<td>An integer array containing the data base location and descriptive information for the current directory entry.</td>
</tr>
<tr>
<td>243</td>
<td>47</td>
<td>LD</td>
<td>Length of the directory in terms of number of entries.</td>
</tr>
<tr>
<td>244</td>
<td>1016</td>
<td>LDB</td>
<td>Length of the data base in terms of number of entries.</td>
</tr>
<tr>
<td>245</td>
<td>1009</td>
<td>LFDB</td>
<td>Last free data base location.</td>
</tr>
<tr>
<td>246</td>
<td>1</td>
<td>LK</td>
<td>Length of the data base directory key in terms of number of words.</td>
</tr>
<tr>
<td>247</td>
<td>'('</td>
<td>LPAREN</td>
<td>Integer word containing the character '(', left justified and blank filled.</td>
</tr>
<tr>
<td>248</td>
<td>8</td>
<td>LT</td>
<td>Length of the array containing the data base location and descriptive information for the current data base entry in computer words.</td>
</tr>
<tr>
<td>249</td>
<td>47</td>
<td>NCD</td>
<td>Maximum number of control directives.</td>
</tr>
<tr>
<td>250</td>
<td>.FALSE.</td>
<td>EDIT</td>
<td>Logical variable set by option character 'E'. If true, DLG requests and responses will be printed.</td>
</tr>
<tr>
<td>DILOG Locations</td>
<td>Value</td>
<td>Local Name</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>251</td>
<td>11</td>
<td>MAXINT</td>
<td>The number of characters representing the maximum size integer for the computer on which this program is installed.</td>
</tr>
<tr>
<td>252</td>
<td>0</td>
<td>ICNML</td>
<td>Integer word containing the record count from the NMLIST file.</td>
</tr>
<tr>
<td>253</td>
<td>0</td>
<td>ICNDB</td>
<td>Integer word containing the number of database read and write requests.</td>
</tr>
<tr>
<td>254</td>
<td>None</td>
<td>MYEOF</td>
<td>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.</td>
</tr>
<tr>
<td>255</td>
<td>NAME</td>
<td></td>
<td>Integer word containing the current database name.</td>
</tr>
<tr>
<td>256</td>
<td>6</td>
<td>NCAR</td>
<td>The number of characters per computer word.</td>
</tr>
<tr>
<td>257</td>
<td>12</td>
<td>NCDBV</td>
<td>The number of characters per database variable.</td>
</tr>
<tr>
<td>258</td>
<td>'-'</td>
<td>NEG</td>
<td>Integer word containing the character ('-'), left justified or blank filled.</td>
</tr>
<tr>
<td>260</td>
<td>14</td>
<td>NMLIST</td>
<td>Logical unit number for potential database information. Also used for reading inserted files. See insert command.</td>
</tr>
<tr>
<td>DILOG Locations</td>
<td>Value</td>
<td>Local Name</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>261</td>
<td>15</td>
<td>NNUM</td>
<td>Maximum number of entries in the number directory.</td>
</tr>
<tr>
<td>262</td>
<td>∞</td>
<td>NOPER</td>
<td>Number of operations per dialog in the page array.</td>
</tr>
<tr>
<td>263</td>
<td>2</td>
<td>NWORD</td>
<td>Number of words per data base entry.</td>
</tr>
<tr>
<td>264</td>
<td>14</td>
<td>NWREC</td>
<td>Number of words per input record.</td>
</tr>
<tr>
<td>265</td>
<td>'.'</td>
<td>POINT</td>
<td>Integer word containing the character (.), left justified and blank filled.</td>
</tr>
<tr>
<td>266</td>
<td>'+'</td>
<td>POS</td>
<td>Integer word containing the character (+), left justified and blank filled.</td>
</tr>
<tr>
<td>267</td>
<td>')'</td>
<td>RPAREN</td>
<td>Integer word containing the character ()), left justified and blank filled.</td>
</tr>
<tr>
<td>268</td>
<td>None</td>
<td>VALUE</td>
<td>A real word containing the value of the current data base variable or result of an arithmetic operation.</td>
</tr>
<tr>
<td>269</td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>270</td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>271</td>
<td>4</td>
<td>DELET</td>
<td>Integer variable used for delete entry in RANDAC.</td>
</tr>
<tr>
<td>272</td>
<td>None</td>
<td>IDESC(20)</td>
<td>An integer array used for temporary storage of the current data base variable descriptive information.</td>
</tr>
<tr>
<td>DILOG Locations</td>
<td>Value</td>
<td>Local Name</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>292</td>
<td>.FALSE.</td>
<td>STORE</td>
<td>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 database. Otherwise, only previously defined data will be stored.</td>
</tr>
<tr>
<td>293</td>
<td>~</td>
<td>LENDES</td>
<td>Length of the descriptive information in IV and IDESC.</td>
</tr>
<tr>
<td>294</td>
<td></td>
<td>COMAND</td>
<td>Integer word containing the name of the current control directive (i.e., ADD, PRINT,...etc.)</td>
</tr>
<tr>
<td>295</td>
<td>0</td>
<td>ICNSRT</td>
<td>Integer word containing the record count of inserted records.</td>
</tr>
<tr>
<td>296</td>
<td>0</td>
<td>IRANDC</td>
<td>Integer word containing the number of collisions from RANDAC.</td>
</tr>
<tr>
<td>297</td>
<td>0</td>
<td>IRANDF</td>
<td>Integer word containing the number of RANDAC FIND requests.</td>
</tr>
<tr>
<td>298</td>
<td>0</td>
<td>IRANDE</td>
<td>Integer word containing the number of RANDAC entries.</td>
</tr>
<tr>
<td>299</td>
<td></td>
<td>NFCD</td>
<td>Next free control directive directory location.</td>
</tr>
<tr>
<td>300</td>
<td>FALSE</td>
<td>LISTI</td>
<td>Logical variable set to true by option character 'L'. If true, source input data will be listed.</td>
</tr>
<tr>
<td>301</td>
<td>FALSE</td>
<td>LISTO</td>
<td>Logical variable set to true by the option character 'O'. If true, the source output file will be listed.</td>
</tr>
<tr>
<td>DILOG Locations</td>
<td>Value</td>
<td>Local Name</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>302</td>
<td>None</td>
<td>COMSAV</td>
<td>Integer word used for saving the value in COMMA when COMMA is being used for storing alternate delimiter.</td>
</tr>
<tr>
<td>303</td>
<td>~</td>
<td>CONDIR</td>
<td>Integer word containing the current control directive (i.e., CREATE, PRINT...).</td>
</tr>
<tr>
<td>304</td>
<td>FALSE</td>
<td>PGDUMP</td>
<td>Logical variable set to true by the option character 'D'. If true, card cracking information (IPAG array) will be printed.</td>
</tr>
<tr>
<td>305</td>
<td>FALSE</td>
<td>SPLITR</td>
<td>Logical variable set to true by the option character 'S'. If true, the list interrupt mode will be invoked.</td>
</tr>
<tr>
<td>306</td>
<td>'DBASE'</td>
<td>DDBASE</td>
<td>Integer word containing the name of the requested data base (DBASE), left justified and blank filled.</td>
</tr>
<tr>
<td>307</td>
<td>FALSE</td>
<td>TRACER</td>
<td>Logical variable set to true by the option character 'T'. If true, trace printout option will be invoked.</td>
</tr>
<tr>
<td>308</td>
<td>~</td>
<td>DIRIN</td>
<td>Name of the directory (data base) which is currently in core.</td>
</tr>
<tr>
<td>309</td>
<td>0</td>
<td>FERROR</td>
<td>A counter for cumulating the number of fatal errors which have occurred since the start of execution.</td>
</tr>
<tr>
<td>Locations</td>
<td>Value</td>
<td>Local Name</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td>310</td>
<td></td>
<td>IFILE</td>
<td>Integer word containing the name of the data base which is to be loaded, left justified and blank filled.</td>
</tr>
<tr>
<td>311</td>
<td>FALSE</td>
<td>INIT</td>
<td>Logical variable set by option character 'G' to specify a new data base file is being created.</td>
</tr>
<tr>
<td>312-331</td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>332</td>
<td>0</td>
<td>ICDLG</td>
<td>Integer variable for counting DLG input data.</td>
</tr>
<tr>
<td>333</td>
<td></td>
<td></td>
<td>Unknown usage.</td>
</tr>
<tr>
<td>334</td>
<td></td>
<td></td>
<td>Unknown usage.</td>
</tr>
<tr>
<td>335</td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>336</td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>337</td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>338</td>
<td>6</td>
<td>LUO</td>
<td>Logical unit number for the output file.</td>
</tr>
<tr>
<td>339</td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>340</td>
<td>FALSE</td>
<td>ANALY</td>
<td>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'</td>
</tr>
<tr>
<td>341</td>
<td>1</td>
<td>STCOLM</td>
<td>Start column for processing input records. Generally has a value</td>
</tr>
<tr>
<td>DILOG Locations Value</td>
<td>Local Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>342</td>
<td></td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>344 20</td>
<td>MAXFER</td>
<td>The maximum number of fatal errors which can occur before execution is terminated.</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td></td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>346</td>
<td>CRDFMT</td>
<td>An integer array containing the BCD definition of a card format.</td>
<td></td>
</tr>
<tr>
<td>347-350</td>
<td></td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>351-352</td>
<td></td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>353</td>
<td>NWPAGE</td>
<td>The number of words which define the width of a namelist record.</td>
<td></td>
</tr>
<tr>
<td>354</td>
<td></td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>355 *EOF</td>
<td>ENDATA</td>
<td>Integer word containing the character string '*EOF', left justified and blank filled, used to identify a user end-of-file.</td>
<td></td>
</tr>
<tr>
<td>356</td>
<td></td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>IOCONT</td>
<td>A counter used to accumulate the total number of input/output requests such as READ, WRITE, etc.</td>
<td></td>
</tr>
<tr>
<td>358-400</td>
<td></td>
<td>Not used.</td>
<td></td>
</tr>
</tbody>
</table>
OVERLAY STRUCTURE

1. TYPE CLRAFCM
2. LIB WORK
3. LIB LECUR,,MSG=LOCALIB.
4. 
5. SEG MAIN
6.   IN DLGDRVR
7.   IN DIALEK;IF, DYNCOR
8. SEG A=* (MAIN)
9.   IN INMOD,RESPOND
10. SEG B*;A
11.   IN INITL,INITM,OPTION,OPINIT,HNUMIT,CDINIT
12. SEG D*; (A)
13.   IN ADDER
14. SEG E*;D
15.   IN ATTACH
16. SEG F*;D
17.   IN CHANGE
18. SEG G*;D
19.   IN IGNORE
20. SEG H*;D
21.   IN COPY
22. SEG I*;D
23.   IN INITIZ
24. SEG J*;D
25.   IN CSF
26. SEG K*;D
27.   IN IDENT
28. SEG L*;D
29.   IN DELETE
30. SEG M*;D
31.   IN DETACH
32. SEG N*;D
33.   IN FORMAT
34. SEG O*;D
35.   IN INLINE
36. SEG P*;D
37.   IN INSERT
38. SEG Q*;D
39.   IN OERRFF
40. SEG R*;D
41.   IN CCDUMP
42. SEG S*;D
43.   IN SEARCH
44. SEG T*;D
45.   IN TIME
46. SEG U*;D
47.   IN USE
48. SEG V*;D
49.   IN UPDATE
50. SEG W*;D
51.   IN PRRTT
52. END

ORIGINAL PAGE IS OF POOR QUALITY
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

RCNTL(COMMONBLOCK)
040401 041005

EXTERNAL REFERENCES:
NTAF$, FNCTR$, IOCDD$, WRELLK$

SYS$RLIB$.HSUTC$/FDR69
$(1) 001000 001024
$(2) 041006 041017
EXTERNAL REFERENCES: NTAB$, NS11$, NHPPA$, IOCDD$, NFCHK$, WAIT$, NIDERS$, R$, DRAINS$, NREBS$, REW$, ID$, STREG$, PRINT$, NWALK$

SYS$RLIB$.NRELLK$/FDR68
$(1) 001107 001131
EXTERNAL REFERENCES: NTAB$, UNITS$, WAIT$, NIDERS$, R$, UPDDA$, ID$

SYS$RLIB$.NMEF$/JSC69
$(1) 001132 001337
$(2) 041020 041037
EXTERNAL REFERENCES: NTAB$, NS11$, NHPPA$, IOCDD$, NFCHK$, NRECM$, PACK$, RDLLK$, UNITS$, UPDDA$, WAIT$, NREBS$, DRAINS$, NFRETS$, NIDERS$, NFERLS$, NSUTC$, NHPPA$, FUNCH$, FREFCI$, STREG$, PRINT$, NWALK$, CLOSE$, WEF$, ID$

SYS$RLIB$.NDCW$/FDR64
$(1) 001340 001465
$(2) 041040 041102
EXTERNAL REFERENCES: NC1UL0$, NFDP$, NC1UL1

SYS$RLIB$.NFITY$/FDR
$(1) 001466 001510

SYS$RLIB$.NCNTI$/FDR68
$(1) 001511 001732
$(2) 041103 041177
EXTERNAL REFERENCES: STREG$, NSTSYS$, NSSTAT$, NCOMB$, NERCP$, NFTRIL$, NCIODP$, NERCT$

SYS$RLIB$.NCLOD$/FDR68
$(1) 001733 002123
$(2) 041200 041239
EXTERNAL REFERENCES: NTAB$, NS11$, UNITS$, CSS$, IDM$, MB$, NMEF$, WAIT$, NREWS$, NREBS$, STREG$, NCEP$, PRINT$, NWALK$, NTESS$, NIDERS$, W$, ID$

41

ORIGINAL PAGE IS
OF POOR QUALITY
EXTERNAL REFERENCES: CENDS, FIELDS, PRINTS, NEE$, OPT$, INTRUP, IALL$.

EXTERNAL REFERENCES: COMS, EXIT$, NR3FS, REST$, CONDs, ERR$, IALL$, ERR$, PRINTS.

EXTERNAL REFERENCES: NWDUS$, NID2$, NERR3$.

EXTERNAL REFERENCES: NMCOD$, NERTRN$, NID2$, NWDUS$, NERR3$.

EXTERNAL REFERENCES: PAGE, CREATE, DISECT, PAGMP, RANDAC, ADDER, RSPOND, NWDUS$, NID2$, NERR2$, NERR3$.

EXTERNAL REFERENCES: LOC$, NWDUS$, NID3$, NID1$, NID2$, NERR3$.

EXTERNAL REFERENCES: LOCP, NWDUS$, NID3$, NID1$, NID2$, NERR3$.

EXTERNAL REFERENCES: NERR3$.
EXTERNAL REFERENCES: PAGE, INITL, INITDM, OPTION, OPINIT, NUMNIT, 
CDINIT, DISCER, PAGEIMP, PANDAC, INMOD, MLADI, RSPOND, ANLSIS, 
UNLOAD, DREM, DONE, EXIT, NRTST, NIO2$, NMDUS$, NERR2$, NERR3$. 

EXTERNAL REFERENCES: READ$, SDF1, SDFD, PRINT$, SDFIC, SDFDC, 
PRF$, POSTPR, PRFRM, SDFIO, ERR$, PWUL$, SIFDO 

EXTERNAL REFERENCES: LAST$, MCDRS$, LDCRS$, PRINT$, EABT$ 

SEGMENT A+; 024317 0311102 054125 055164 
FOLLOWS SEGMENT MAIN 

EXTERNAL REFERENCES: NERR1$, NERRC$ 

EXTERNAL REFERENCES: NERRA$, NERR1$, NERRC$ 

EXTERNAL REFERENCES: NMDUS$, XPRI$, NIO1$, NIO2$, NERR3$. 

EXTERNAL REFERENCES: PRT, MOVER, DECBCD, STRMOV, INTECD, NERR2$, 
NMDUS$, NIO2$, NERR3$ 

EXTERNAL REFERENCES: DECIDE, BCDDEC, BCDINT, NERR2$, NERR3$
EXTERNAL REFERENCES: CHRNUN, NERR3$

EXTERNAL REFERENCES: CHRNUN, XPRI, XRPR, NERR3$

EXTERNAL REFERENCES: CHRNUN, XPRI, XRPR, NERR3$

EXTERNAL REFERENCES: PAGE, RANDAC, DBLOAD, RPLACE, PGMOUT,
WRITOR, NMODUS, NI02$, NI03$, NERR3$

EXTERNAL REFERENCES: PAGE, RMPR, ATTACH, CHANGE, IGNORE, COPY,
INITIZ, CSF, INCHT, DELETE, DETAHC, FORMAT, INLINE, insert, OFF,
ON, DBLOAD, PGDMPP, SEARCH, TIME, USE, UPDATE, PRTT, NMODUS, NI02$, 
NERR3$, NERR3$

SEGMENT B+ 024317 025846 054105 054556
HAS THE SAME STARTING ADDRESS AS SEGMENT A

EXTERNAL REFERENCES: OPT$

EXTERNAL REFERENCES: NERR3$

EXTERNAL REFERENCES: NERR3$
EXTERNAL REFERENCES: RANDAC, IDPT, GET, NERR3$

EXTERNAL REFERENCES: RANDAC, NERR3$

EXTERNAL REFERENCES: RANDAC, NERR3$

EXTERNAL REFERENCES: RANDAC, NERR3$

EXTERNAL REFERENCES: RANDAC, NERR3$

EXTERNAL REFERENCES: PAGE, LOOP, DECIDE, RANDAC, GETSUB, DCINT, SCALE, INTBCD, MOVER, DECBCD, ADDONE, NWIDU$, NIO2$, NERR2$, NERR3$
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%, NID2%, HERR2%, HERR3%

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 CARDED
EXTERNAL REFERENCES: PAGE, MOVER, STRMOV, PGMOUT, WRITCP, NWDU%, NID2%, HERR3%

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%, NID2%, HERR3%

SEGMENT I 031103 031552 055165 055321
HAS THE SAME STARTING ADDRESS AS SEGMENT D

EX42-00002 WORK.DBINIT/DLG
%1 031103 031424 %0 055165 055267
%3 CARDED %2 BLANK%COMMON
%4 DILOG
EXTERNAL REFERENCES: PAGE, DECIDE, BCDDEC, BCDINT, DISECT, NWDU%, NID2%, HERR2%, HERR3%

EX42-00002 WORK.INITIZ/DLG
%1 031425 031553 %0 055370 055321
%3 DILOG %2 BLANK%COMMON
EXTERNAL REFERENCES: PAGE, DBINIT, DYNCP, DYNARC, NWDU%, NID2%, HERR3%

SEGMENT J 031103 031232 055165 055213
HAS THE SAME STARTING ADDRESS AS SEGMENT D
EXTERNAL REFERENCES: PAGE, GET, PUT, NERTRN, NPERT$, NID2$, NERR3$

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

EXTERNAL REFERENCES: PAGE, DECIDE, RDDEC, RDINT, RANDAC, IVDESC, MOVER, STRMDV, IVCALC, NUDUS$, NID2$, NERR3$

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

EXTERNAL REFERENCES: PAGE, RANDAC, IVDESC, MOVER, NUDUS$, NID2$, NERR3$

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

EXTERNAL REFERENCES: NUDUS$, NID2$, NERR3$

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

EXTERNAL REFERENCES: MOVER, PAGE, STRIP6, SDINT, STRMDV, DMAN, NNCOD$, PGMOUT, NPERT$, NID2$, NID1$, NID2$, NERR3$

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

53
EXTERNAL REFERENCES: HWDUS$, NI02$, NERR3$

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

EXTERNAL REFERENCES: EDFTST, NDI1$, NI01$, NI02$, NRIUS$, HERR3$
BUFFER(COMMONBLOCK) 055266 055310

EXTERNAL REFERENCES: PAGE, DECIDE, BCDDEC, BCDDINT, CREATE, READER, PGMDUT, WRITER, HWDUS$, NI02$, NERR3$

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

EXTERNAL REFERENCES: PAGE, RANDAC, HWDUS$, NI01$, NI02$, NERR3$
SEGMENT R$  031103 031724  055165 055742
HAS THE SAME STARTING ADDRESS AS SEGMENT D

EXTERNAL REFERENCES: NERR3$

EXTERNAL REFERENCES: NERR3$
EX42-00002*WORK.DMWRT
$(1) 031244 031516 $(0) 055216 055313
$(3) DILDS $(2) BLANK$COMMON
EXTERNAL REFERENCES: INTECD, SHELL, RANDAC, IVDESC, NWDUS$, NI2$, NI2$, N101$, NERR3$

EX42-00002*WORK.CDUMP/DLG
$(1) 031517 031724 $(0) 055314 055742
$(3) DILDS $(2) BLANK$COMMON
EXTERNAL REFERENCES: DMWRT, NWDUS$, NI2$, 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) DILDS $(2) BLANK$COMMON
EXTERNAL REFERENCES: NWDUS$, NI2$, 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) DILDS $(2) BLANK$COMMON
EXTERNAL REFERENCES: NWDUS$, NI2$, NERR3$

SEGMENT U+ 031103 031120 055165 055212
HAS THE SAME STARTING ADDRESS AS SEGMENT D

EX42-00002*WORK.USE/DLG
$(1) 031103 031234 $(0) 055165 055212
$(3) SEARCH $(2) BLANK$COMMON
$(5) DILDS $(4) CARDED
EXTERNAL REFERENCES: PAGE, DELLOAD, NPT$, NI2$, NI2$, N101$, NERR3$

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) DILDS $(2) BLANK$COMMON
EXTERNAL REFERENCES: PAGE, DELLOAD, NWDUS$, NI2$, NERR3$

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

ORIGINAL PAGE IS
OF POOR QUALITY
EX42-00002*WORK.PRTT

$(1)$ 031103 031477  $(2)$ 055165 055251
$(3)$ MS  $(4)$ BLANK\COMMON

EXTERNAL REFERENCES: NXTAD, NDEFS, NPBA\$s, NI01\%, NI02\%, NWD\$, NPRTS, NERR3\$

IBANK DRAWN TO SCALE:  200 WORDS DECIMAL PER DASH

MAIN (9935)
INDIRECT LOAD TABLE

CALLS ON THE FOLLOWING 16-BIT ENTRY POINTS IN INDIRECT LOAD SEGMENTS ARE ROUTED VIA THESE INDIRECT LOAD ADDRESSES, TO INSURE SEGMENTS ARE LOADED

<table>
<thead>
<tr>
<th>ADDER</th>
<th>ADDONE</th>
<th>ATTACH</th>
</tr>
</thead>
<tbody>
<tr>
<td>040134</td>
<td>040137</td>
<td>040142</td>
</tr>
<tr>
<td>ECIDEC</td>
<td>ECINIT</td>
<td>ECRVAL</td>
</tr>
<tr>
<td>040145</td>
<td>040150</td>
<td>040153</td>
</tr>
<tr>
<td>CCDUMP</td>
<td>CDINIT</td>
<td>CHANGE</td>
</tr>
<tr>
<td>040156</td>
<td>040161</td>
<td>040164</td>
</tr>
<tr>
<td>CHRMN</td>
<td>COPY</td>
<td>CSF</td>
</tr>
<tr>
<td>040167</td>
<td>040172</td>
<td>040175</td>
</tr>
<tr>
<td>DBINIT</td>
<td>DBURD</td>
<td>DECPD</td>
</tr>
<tr>
<td>040200</td>
<td>040203</td>
<td>040206</td>
</tr>
<tr>
<td>DECIDE</td>
<td>DELETE</td>
<td>DETACH</td>
</tr>
<tr>
<td>040211</td>
<td>040214</td>
<td>040217</td>
</tr>
<tr>
<td>FORMAT</td>
<td>GETSUR</td>
<td>IDENT</td>
</tr>
<tr>
<td>040222</td>
<td>040225</td>
<td>040230</td>
</tr>
<tr>
<td>IGNORE</td>
<td>INITDM</td>
<td>INITIZ</td>
</tr>
<tr>
<td>040233</td>
<td>040236</td>
<td>040241</td>
</tr>
<tr>
<td>INITL</td>
<td>INLINE</td>
<td>IIMOD</td>
</tr>
<tr>
<td>040244</td>
<td>040247</td>
<td>040252</td>
</tr>
<tr>
<td>INSERT</td>
<td>INTED</td>
<td>INTLD</td>
</tr>
<tr>
<td>040255</td>
<td>040260</td>
<td>040263</td>
</tr>
<tr>
<td>IOFT</td>
<td>IVCALC</td>
<td>IVDESC</td>
</tr>
<tr>
<td>040266</td>
<td>040271</td>
<td>040274</td>
</tr>
<tr>
<td>NRBUS</td>
<td>NRMNIT</td>
<td>OFF</td>
</tr>
<tr>
<td>040277</td>
<td>040302</td>
<td>040305</td>
</tr>
<tr>
<td>ON</td>
<td>ONINIT</td>
<td>OPTION</td>
</tr>
<tr>
<td>040310</td>
<td>040313</td>
<td>040316</td>
</tr>
<tr>
<td>PRTT</td>
<td>PDBRP</td>
<td>PPLACE</td>
</tr>
<tr>
<td>040321</td>
<td>040324</td>
<td>040327</td>
</tr>
<tr>
<td>RESPND</td>
<td>SCALD</td>
<td>SEARCH</td>
</tr>
<tr>
<td>040332</td>
<td>040335</td>
<td>040340</td>
</tr>
<tr>
<td>SHELL</td>
<td>STRHDY</td>
<td>TIME</td>
</tr>
<tr>
<td>040343</td>
<td>040346</td>
<td>040351</td>
</tr>
<tr>
<td>UPDATE</td>
<td>USE</td>
<td>VALMS</td>
</tr>
<tr>
<td>040354</td>
<td>040357</td>
<td>040362</td>
</tr>
<tr>
<td>WRITER</td>
<td>XPIT</td>
<td>XPRI</td>
</tr>
<tr>
<td>040365</td>
<td>040370</td>
<td>040373</td>
</tr>
</tbody>
</table>

ORIGINAL PAGE IS OF POOR QUALITY.
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(I)
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), IY )

EQUIVALENCE (IDILOG(223), IV )
EQUIVALENCE (IDILOG(243), LD )
EQUIVALENCE (IDILOG(248), LT )
EQUIVALENCE (IDILOG(251), MAXINT)
EQUIVALENCE (IDILOG(257). NCDBV )
EQUIVALENCE (IDILOG(259), NF )
EQUIVALENCE (IDILOG(263), NIORD )

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(I), BCD(I), BLANK, DELIM, EQUAL
INTEGER SAVTYP, SAVOP, END, IDESC(I)

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

CONT. ON PG 2
WRITE (LUO, 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

IF (INAM.EQ.BLANK.OR.INAM.EQ.END)
F
T
GO TO 700
CALL DECIDE (INAM, BCD, ITYPE, NCHAR)

IF (TRACER)

WRITE (LU0,1030) INAM, ITYPE

FORMAT (9H TYPE OF A9, 4H IS II)

GO TO (100,200,300,530), ITYPE

CONT. ON PG 5
CALL RANDAC (FIND, INAM, LK, IDATA(ID+I), LD, FOUND, IV, LT, NF)

IF (FOUND)
  T
  GO TO 110
  A4

DO NOT STORE NEW VARIABLES IN DIRECTORY IF STORE IS FALSE AND OPERATOR IS BLANK.

IF (STORE)
  T
  GO TO 102

IF (IOPR.EQ.BLANK)
  F
  GO TO 600

102 CONTINUE

CONT. ON PG 6

ADDER
PG 5 OF 15
IF (J-IS) 500, 104, 105

0

104

IF (K.EQ.IS)

F

T

GO TO 500

105

CONTINUE

IF (I0PR.NE.EQUAL)

F

T

GO TO 107

WRITE (LUO.6002) INAM

FORMAT( 14H0*** WARNING. A10. 38H NOT IN DATA BASE. EXPRESSION ST ORED. )

GO TO 530

107

CONTINUE

CONT. ON PG 7

A6
IF (IOPR.NE.BLANK)
  GO TO 109

IF (IOPP.EQ.EQUAL)
  WRITE (LU0.6002) INAM
  GO TO 500

109 CONTINUE
WRITE (LU0.6001) INAM

FORMAT(12H0*** ERROR, A10, 39H NOT IN DATA BASE, EXPRESSION IGNORE D. )

GO TO 600

C CHECK FOR INPUT SUBSCRIPT.

A4
CONTINUE

CONT. ON PG 8

A7
IF (J.EQ.1)  
   T  
   ISUBS=1  

IF (ISUB.NE.BLANK)  
   F  
   T  
   GO TO 120  

KOUNT=1  
GO TO 140  

SUBSCRIPT WAS INPUT. LOAD ITS VALUE IN KOUNT.

120  
CONTINUE  

CALL GETSUB (ISUB, KOUNT)  

IF (IOPR.EQ.BLANK.AND.IOPP.EQ.EQUAL)  
   F  
   T  

ISUBS = KOUNT  

CONT. ON PG. 9  

ADDER  
PG 8 OF 15
FLOATING POINT VALUE WAS INPUT. SCALE IF REQUIRED.
C INTEGER VALUE WAS INPUT. SAVE VALUE IN IVAL.

IF (IOPR.EQ.EQUAL)

T

GO TO 530

F

IF (SAVTYPE.EQ.3 .AND. SAVOP.EQ.EQUAL)

T

VALUE = 1.0

F

GO TO 330

C *** NOTE *** INTEGER ARRAYS MAY NOT BE INITIALIZED BY N*VALUE

CONT. ON PG 11

A10
ONLY METHOD ALLOWED IS IA=IVALU,IVALU,IVALU.....

THIS IS DONE TO ALLOW INTEGER ARITHMETIC SUCH AS SUBSCRIPT CALCS. IE B("JJJ*2+1")= VALUE

CALL BCD INT (BCD.MAXINT,IVAL)

SCALE DATA UNLESS CURRENT DATA IS A CONTINUATION

CONT.

CONT. ON PG 12

ADDEND

PG 11 OF 15

ORIGINAL PAGE IS OF POOR QUALITY
B2
GO TO 500

B3
ARRAY=TRUE.
SAVTYP=0
GO TO 500

B4
CONTINUE

C
SAVE DATA FROM CURRENT OPERATION.
IVLSAV=IVAL
SAVOP=IOPR
SAVTYP=ITYPE

500
CONTINUE

C
END OF EXPRESSION. CALL ADDONE TO INSERT DATA BASE NAMES AND/OR VALUES INTO THE DATA BASE.

IF (ITYPE.EQ.1) 

CONT. ON PG 13
ADDER
PG 12 OF 15

A12
IF (ITYPE.NE.3) THEN
  CALL INTBCD (VALUE, BCD, NC)
ENDIF

CONTINUE

CALL MOVER (BLANK, 0, BCD, 1, BCDLEN)

CONT. ON PG 14
CALL DECBCD (VALUE, NCDBV, BCD)

CONTINUE

CALL ADDONE (IP31K, BCD, ISUBS, FIRST, LAST)

IF (.NOT.ARRAY)

FIRST=.FALSE.

DO 580 L=2, IYLSAV

CALL ADD ONE (BLANK, BCD, KOUNT, FIRST, LAST)

580 CONTINUE

GO TO 600

C

STORE AN ELEMENT OF AN ARRAY

CONT. ON PG 15
CALL ADDONE (BLANK, IPAG(I+2), KOUNT, FIRST, LAST)

GO TO 600

IF (.NOT.STORE)

GO TO 700

LAST = .TRUE.

CALL ADDONE (BLANK, BLANK, KOUNT, FIRST, LAST)

IF (TRACER)

WRITE (LUO, 1020)

FORMAT (13H LEAVE ADDER )

RETURN

END

ADDER
PG 15 FINAL

A15
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

WRITE (LUO,1000)

1000 FORMAT (34H INPUT/OUTPUT PROCESSING BREAKDOWN )

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

1010 FORMAT (50H TOTAL SIREAD SOWRIT NMLIST DBASE INSRT DIALOG / 917 )

WRITE (LUO,1020) IRANDE, IRANDF, IRANDC

1020 FORMAT (33H0 RANDOM ACCESS DIRECTORY ANALYSIS /

CONT. ON PG 2

ANLSIS
PG. 1 OF 2

A16
2IH ENTRY FIND COLL /317 )

C***

RETURN

END
SUBROUTINE ATTACH

*** PROCESS "ATTACH ... " DIRECTIVE.

COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG(338), LUO )

WRITE (LUO,1000)

FORMAT (34H ATTACH DIRECTIVE NOT IMPLEMENTED )

RETURN

END
SUBROUTINE CC DUMP

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(309), 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

ORIGINAL PAGE IS OF POOR QUALITY

CCDUMP
PG 1 OF 4
IF (IDATA(ID+J).EQ.BLANK)
    K = K+1
    IF (K.EQ.1)
    IB = 1
    DBNAME(K) = IDATA(ID+J)
    IF (K.EQ.100)
    CONTINUE
1 100  CONTINUE
110 CONTINUE
NUSED = MIN0(K, 100)
IC = IC + 1
CONT. ON PG 3
IF (IC.GT.1)
    F

    L = LD - IB + 1

    LENDB = (LDB-LT*LD)/NWORD
    NDBVAR = LENDB - (LDB-LFDB)/NWORD
    WRITE (LUO,1009)

1009
    FORMAT (IH, 36(2H* ))
    WRITE (LUO,1000) DIRIN, LDB, LD, LT, LENDB, NDBVAR, NWORD

1000
    FORMAT (IH0, 10H PRINT OF A10,
    16H TOTAL LENGTH = ,16,6H WORDS/
    21H DIRECTOY LENGTH = 15, 10H ITEMS, 15.
    13H ARE IN USE ( 15, 13H WORDS/ITEM) /
    21H DATA BASE LENGTH = 15, 10H ENTRIES, 15.
    13H ARE IN USE ( 15, 15H WORDS/ENTRY) /
    11X, 46M APhEBETIZED LIST IN GROUPS OF 100 FOLLOW // )

200
    CONTINUE

    WRITE (LUO,1001) IC

1001
    FORMAT (13H *** GROUP 12, 6H *** / )

    CONT. ON PG 4

A21
CALL DB WRIT (DBNAME, KEY, NUSED)

IF (IE .LT. LD)
  T
    GO TO 10
  F

WRITE (LUO, 1019)

1019
  FORMAT (IH, 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), INITIAL)
EQUIVALENCE (IDILOG(221), INSTAL)

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

DATA (NAMENC(I), I=1,40)/
"ADD", "ATT", "ATTACH", "CHA", "CHANGE", ".", "COM", "COMEN", "COP",
"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)/
1,2,3,4,4,4,5,5,6,6,7,7,8,8,8,9,9,9,9,9,10,11,11,11,12,12,12,13,13,
15,14,16,16,17,17,18,19,20,20,-1,-1,-21,21

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

CALL RANDAC (INITIAL, BLANK, I, ICD, NCD, FOUND, IVAL, 4, NFCD)

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

DO 100 I=1,NN
CALL RANDAC (FIND, NAMENC(I), I, ICD, NCD, FOUND, IVAL, 4, NFCD)

CONT. ON PG 2

CDINIT

PG 1 OF 2

A23

ORIGINAL PAGE IS OF POOR QUALITY
CALL RANDAC(INSTAL,NAMEN(I),1,ICD,NCD,FOUND,IVAL,4,NFCD)

IF (FOUND)

T

GO TO 100

IVAL=IVALN(I)

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

1 ----→ 100

CONTINUE

RETURN

END
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), NCOBV)
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

CHANGE

PG 1 OF 5

A25
WRITE(LU0,1005)

FORMAT( 'ARRAY INDEX NOT AN INTEGER, NO CHANGE DONE' )

IF(ITYP.NE.3)

GO TO 800

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

CONTINUE

CALL BCDDEC(BCD,NCDBV,VAL)

GO TO 800

CONT. ON PG 3
CALL BCDINT(BCD, NCDBV, VAL)

GO TO 800

CONTINUE

ONELES = NCDBV - 1

DO 450 I = 1, ONELES

IF(BCD(I) .EQ. POINT)

GO TO 475

CONTINUE

GO TO 450

CONTINUE

WRITE(LUO, 1006)

FORMAT("VALUE IS NOT ".TRUE." OR ".FALSE.")

GO TO 900

CONT. ON PG 4
WRITE(LUO,1002) INDX, VAL, VAL, VAL
IDILOG(INDX)=VAL

A5
900
CONTINUE

IF(Tracer)
T

WRITE(LUO,1004)

1004
FORMAT(" LEAVE CHANGE")
RETURN

END
SUBROUTINE COPY

C*** PROCESS "COPY ... " DIRECTIVE.

COMMON /DILOG/ IDILOG(1)
EQUIVALENCE (IDILOG(338), LUO )

WRITE (LUO,1000)

FORMAT (34H COPY DIRECTIVE NOT IMPLEMENTED )

RETURN

END
SUBROUTINE CSF

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/'a'/
LIM = PAGE(1.1.2)
CHAR = 0
ASSIGN 10 TO LEAP

DO 30 I=LIM, INRECL    
   CALL GET(IMAGE, I, K)
   GO TO LEAP
30    CONT. ON PG 2

IF(K.NE.MASTSP)
   ASSIGN 20 TO LEAP
   GO TO 30

ASSIGN 20 TO LEAP
CALL PUT(BCD.CHAR.K)
CALL PUT(BCD.CHAR+.BLANK)
CALL PUT(BCD.CHAR+2.POINT)
CALL PUT(BCD.CHAR+3.BLANK)
K=RETURN(6.BCD)
IF(K.NE.0)

CONT. ON PG 3
FORMAT(" ERTRN STATUS IN CSF WAS ="013")
RETURN
END
SUBROUTINE DELETE

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

COMMON IDATA(i)
EQUIVALENCE (ID, IDATA)

COMMON /DILOG /
IDILOG(I)
EQUIVALENCE (IDILOG(34). BLANK)
EQUIVALENCE (IDILOG(39). DELIM)
EQUIVALENCE (IDILOG(43). FIND)
EQUIVALENCE (IDILOG(43). 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(I), BLANK, DELIM, FIND
INTEGER PAGE, DELET

LOGICAL FOUND
IS=1

IF (PAGE(2,1,1).NE.DELIM)
- F (PAGE(3,1,1).EO.BLANK)

CONT. ON PG 2 DELETE P6 1 OF 3 A34
A0
RETURN

A1
IS=2

A2
CONTINUE

DO 60 K=15,NDLOG
   INAM = PAGE(3,1,K)
   IF (INAM.EQ.BLANK)
      T
      GO TO 900
      F
      CALL RANDAC(FIND,INAM,LK,IDATA(I0+1),LD,FOUND,IV,LT,NF)
      IF (FOUND)
         T
         GO TO 20
         F
         WRITE (LUD,7001) INAM

CONT. ON PG 3
DELETE
PG 2 OF 3
7001

FORMAT(20HNAME 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 MOVE (BLANK, 0, IDATA(ID+1), I, LAST-LOC+1)

2 - - - 60
CONTINUE

4 A4
900
CONTINUE
RETURN

END
SUBROUTINE DETACH
C*** PROCESS "DETACH ... " DIRECTIVE.
COMMON /DILOG/ IDILOG(I)
EQUIVALENCE (IDILOG(338), LUO )
C***
WRITE (LUO,1000)
1000
FORMAT (34H DETACH DIRECTIVE NOT IMPLEMENTED )
C***
RETURN
END
COMMON /CARDBD/ IPAG(685)

INTEGER FIND, DIRIN
INTEGER CONDIR, PAGE
LOGICAL FOUND, MYEOF, EDIT, TRACER, ANALY, ATTN
DATA IC/O/
PRINT 8887

FORMAT(" BEGIN DIALEK")

DATA ID /5/

FORMAT(" BEGIN DLG")

CONT. ON PG 3
CONTINUE
ENTRY INTRUP
CALL DISECT (O)
CALL PAGDMP

IF (.NOT.MYE0F)
T
GO TO 50
GO TO 9900
F

CONTINUE
IC=IC+1

IF (IC.EQ.1)
T
CONDIR=PAGE(3,1,1)

CONT. ON PG 4

A40
IC=0

*** LOCATE CONTROL DIRECTIVE IN DIRECTORY

CALL RANDAC(FIND, CONDIR, I, ICD, NCD, FOUND, NODIR, 4, NFCD)

IF (TRACER)

WRITE(LUO, 6001) CONDIR, NODIR

6001 FORMAT(I8, A6, I6)

IF (FOUND)

GO TO 80

CALL INMOD

CONT. ON PG 5
C*** PROCESS CONTROL DIRECTIVE.

1 A3

CONTINUE

IF (NODIR.GT.0)

T

GOTO 90

CALL NLADD

GOTO 40

F

CONTINUE

CALL RESPOND (NODIR, LC)

GOTO (40.50), LC

CONT. ON PG 6

CONT. ON PG 6

DIALEK

PG 5 OF 6

A42
CALL ANALYSIS

CALL UNLOAD(DIRIN)

CALL DBEND (IFIL)

CALL DONE($9999)

PRINT 8888

FORMAT("O DIALEK PROCESSING COMPLETE")

FORMAT("O DLG DONE")

CALL EXIT

RETURN 0

END
SUBROUTINE FORMAT

PROCESS "FORMAT ..." DIRECTIVE.

IMPLICIT INTEGER (A-Z)
DIMENSION TEMP(14)
COMMON /UNITS/ IUNARA(1)
COMMON /SCNTL/ 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), ICND )
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)
    F
    PRINT 1100
T
1100
FORMAT(13H ENTER FORMAT )
CALL MOVER(BLANK, 0, TEMP, 1, 14)

CONT. ON PG 2

FORMAT
PG 1 OF 4

A45
CALL MOVER(0,0,IT,1,5)

\[ M = \text{PAGE}(3,3,2) \alpha \text{NUM OF ITEMS PER LINE} \]

CALL STRIPG(M,TEMP,DUMMY)

CALL BCDINT(TEMP,G,M)

\[ N = \text{PAGE}(3,2,2) \alpha \text{NUM OF LINES=TOTAL ITEMS/NUM PER LINE} \]

CALL MOVER(BLANK,0,TEMP,1,14)

CALL STRIPG(N,TEMP,DUMMY)

CALL BCDINT(TEMP,G,6N)

\[ N = N/M \]

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)

\[ \text{IT}(1) = \text{PAGE}(3,1,2) \]

CONT. ON PG 3
DO 200 I=1,N
ICNDB=ICNDB+1
CALL DMAN(-21,IT,M,ICHAR,IBUF,IUNARA,IUNARA(2))

IF(EDIT)
   PRINT 1120,(ICHAR(K),K=1,M)
FORMAT(9H DATA IS (/5013))
   IF(ICHAR(1).EQ.3LE0D)
      CALL DMANC_PRED(IT,M,ICHAR,IBUF,IUNARA,IUNARA(2))
   IF(M.LE.0)
      CALL MOVER(BLANK,0,TEMP,1,14)
GO TO 300

CONT. ON PG 4

A47
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

FORMAT(" ENCODED DATA IS:/(IX,14A6)"))

3 200 CONTINUE

3 300 CONTINUE
RETURN
END
WRITE (LUO,1010)

1010

FORMAT (13H ENTER IDENT )

IF (LENDES .LE. 2)

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

IS=1

COMAND = CONDIR

IF (NDLOG.LT.2)

CONT. ON PG 3

IDENT

PG 2 OF 7

A50
NUMBER OF ELEMENTS ARE INPUT

NUMBER = PAGE(3,2,K)

CALL DECIDE (NUMBER, BCD, ITYPE, NC)

IF (ITYPE .NE. 2)

CALL BCD DEC (BCD, NC, VALUE)

IVAL = VALUE

GO TO 70

CONTINUE

IF (ITYPE .NE. 3)

CALL BCD INT (BCD, NC, IVAL)

CONT. ON PG 5

IDENT

PG 4 OF 7
CONTINUE
WRITE (LUD, 1000) NAME, NUMBER

FORMAT (1H 30X, 15H ERROR IN IDENT., 22H THE DEFINED VARIABLE A10/
1H .30X, 10H CANNOT BE A10, 23H SET ARRAY LENGTH TO 1.)

IVAL = 1

CALL RANDAC (FIND, NAME, LK, IDATA(ID+1), LD, FOUND, IV, LT, NF)

IF (FOUND) THEN
CONTINUE
GO TO 30

CONT. ON PG 6
IDENT
PG 5 OF 7

A53
C DEFINE NEW VARIABLES

DEFINE NEW VARIABLES

A7

IDESC(2) = COMAND

CREATE KOUNT LOCATIONS IN THE DATABASE FOR THE NEW VARIABLE

LOC = LFDB + NWORD
NUM = KOUNT
LFDB = LFDB + KOUNT * NWORD

GO TO 40

C DELETE EXISTING DESCRIPTION

A8

CONTINUE

CALL IV DESC (NUM, LOC)

CALL MOVER (BLANK, 0, IV, 1, LENDES)

CONTINUE

40

CONTINUE

NBCD = LENDES - 2

CALL MOVER (BLANK, 0, IDESC(3), 1, NBCD)

C ADD NEW DESCRIPTION

IS1 = PAGE(1, 1, K+1)

CONT. ON PG 7

IDENT
PG 6 OF 7

A54
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)

CONTINUE

IF (TRACER)

WRITE (LUO,1020)

FORMAT (13H LEAVE IDENT )

RETURN

END
SUBROUTINE IGNORE

C THIS ROUTINE BLANKS OUT THE CHARACTERS ASSOCIATED WITH COMMENT COM

COMMON /DIOLOG / IDILOG(I)
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(I), BLANK, IMAGE(I)
LOGICAL SKIP, ATTN, TRACER, EDIT
DATA ATTN / .FALSE. /

C***

IF (TRACER) F

WRITE (LUO.1010)

1010 FORMAT (13H ENTER IGNORE )

SKIP= .FALSE.
NS= 1

CONT. ON PG 2

IGNORE
PG 1 OF 6

A56
C
SEARCH THRU PAGE ARRAY LOOKING FOR BEGINNING AND
CLOSING COMMENT DELIMITERS

DO 200 K=1,NDLOG

NOPER = IPAG(LENPAG-K+1)

DO 100 J=1, NOPER

IOPR = PAGE(2,J,K)

NF = PAGE(1,J,K)

IF (IOPR.NE.DELIM)

T

GO TO 90

IF (.NOT.ATTN)

F

GO TO 60

C
CLOSING DELIMITER FOUND. RESET FIRST DELIMITER SWITCH

ATTN = .FALSE.

C
REPLACE COMMENT WITH BLANKS

CONT. ON PG 3

IGNORE
PG 2 OF 6

A57
BEGINNING COMMENT DELIMITER FOUND, SET FIRST DELIMITER SWITCH(ATTN) TO .TRUE. AND SAVE STARTING COLUMN OF COMMENT

DELETION NOT FOUND, CHECK FOR END OF CURRENT DIALOG, AND CHECK FOR END OF CURRENT CARD IMAGE

CONT. ON PG 4

A58
IF (K.EQ.NDLOG)
T
GO TO 40
CONTINUE

IF (N.EQ.INRECL)
T
GO TO 100

IF (.NOT.ATTN)
N=INRECL
CONT. ON PG 5

CONTINUE

GO TO 200

CONTINUE

GO TO 220

N=INRECL

CONT. ON PG 5

IGNORE
PG 4 OF 6
CALL MOVER(BLANK,0,ICHR(NS),1,N)

CALL STRMOV( ICHR(NS),1,N, IMAGE,NS)

CONTINUE

C TEST FOR ALL BLANK CARD

DO 300 I=1,NWREC

IF (IMAGE(I) .NE. BLANK)

GO TO 900

300 CONTINUE

IF (SKIP) .TRUE.

900 CONTINUE

IF (SKIP) .FALSE.

GO TO 999

CONT. ON PG 6

A60

IGNORE PG 5 OF 6
CALL PGMOUT ($999, $999, IMAGE, BLANK)

IF (EDIT) T

CALL WRITCR (LUO, IMAGE, NWREC)

IOCONT = IOCONT + 1

IF (TRACER) T

WRITE (LUO, 1020)

FORMAT (13H LEAVE IGNORE)

RETURN

END
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 /DIALOG/ IDILOG(I)
EQUIVALENCE (IDILOG(37), DBASE)
INTEGER DBASE

C*** BEGIN EXECUTION

INCLLEN = 10
KEYWRD = 2
LENGTH = 800
LUD = DBASE
NREC = 256
NT = 3

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), INITIAL)
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
WRITE (LUO, 1010)

FORMAT (13H ENTER INITIZ )

IFIL = PAGE(3,1,2)

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)

WRITE (LUO, 1040) DDBASE, LD, LK, LENDES, NWORD, LDB

FORMAT(43H DDBASE DIRLEN KEYLEN LENDES NWORD LTOTAL / IX.A6.517)

CALL DYNCOR (IDATA, LDB)

CONT. ON PG 3

INITIZ

PG 2 OF 3
CALL RANDAC (INITIAL, BLANK, LK, DATA(ID+1), LD, FOUND, IV, LT, NF)

900 CONTINUE

IF (TRACER)

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 / DILG/ 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), EONAM)
EQUIVALENCE (IDILOG( 42), MXCHAR)
EQUIVALENCE (IDILOG( 43), FIXED)
EQUIVALENCE (IDILOG(220), INRECL)
EQUIVALENCE (IDILOG( 44), ICHR)
EQUIVALENCE (IDILOG(184), IMAGE)
EQUIVALENCE (IDILOG(219), INITIAL)
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

INITL
PG 2 OF 8
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, 6HUUPDATE, 6HDESIGN, 6HEXECUT, 
  6HLOOP, 6HENDD, 6HRESTAR, 6HPRINT, 
  6HCREATE, 6HEDIT /

DATA NULL/1H/, NCHAR/6/, DELIMM/1H"", EQU/1H=", COM/1H,/
DATA PLUS/1H+, MINUS/1H-, IE/IHE", IF/1HIF", IF/1H,/
DATA RP/1H/, LP/IH("/, 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 ENDDATA /4H*EOF/
DATA INSERT /6HINSERT/
DATA IFIELD/28000/

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

DO 20 I=1,NCDBV

BCD(I)=NULL

CONT. ON PG 5
IF(KSTAT.LT.0)

KSTAT=NERTRN(6,"&ASG.T 25.,F/2/POS/2 . ")

IF(KSTAT.NE.0)

PRINT 101,KSTAT

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 )

WRITE (LUO,1000)

FORMAT (34H INLINE DIRECTIVE NOT IMPLEMENTED )

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 / IDLOG(1)
EQUIVALENCE (IDLOG( 34), BLANK )
EQUIVALENCE (IDLOG( 39), DELIM )
EQUIVALENCE (IDLOG( 40), ICONT )

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

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

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

IF(TRACER)
A0
WRITE(LU0,1010)
A1
1010
FORMAT(13H ENTER INMOD )
IF (PAGE(1,1,1) .EQ. INRECL) F
  GO TO 800 A0
IF (PAGE(2,1,1) .NE. DELIM) F
  GO TO 200 A4
INAM = PAGE (3,1,1)
START=INX
   DO 125 I=1,NONAMS D 3
   CALL RANDAC (FIND,INAM,LK,IDATA(ID+1),LD,FOUND,IV,LT,NF)
IF (FOUND) F
  A3
  T
CONT. ON PG 3
INMOD
A2
GO TO 150

A3
IF (NONAMS.EQ.1)
T
GO TO 125
F

INX = MOD (START + I - 1, NONAMS) + 1
CALL DBLOAD (DBNAMS (INX))

CONTINUE

IF (INAM.EQ.BLANK)
T
GO TO 200
F

INX = INX + 1

CONT. ON PG 4

INMOD
PG 3 OF 6
WRITE (LUO,1005) INAM, IMAGE

1005

FORMAT(1HO,50HILLEGAL COMMAND OR DATA BASE NAME NOT IN DIRECTORY/ 1X,A10,10HON CARD *14A6)

GO TO 900

IF(EDIT)

CALL PGMOUT($890,$890,IMAGE,BLANK)

CALL WRITCR(LUO,IMAGE,NWREC)

CONTINUE

CONTINUE

MYEOF = .TRUE.

CONT. ON PG 6
IF(Tracer)

WRITE(LUO, 1020)

FORMAT(13H LEAVE INMOD )

RETURN

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

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

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

DATA NRMAX /10000/
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  LU

IS  CONTINUE

IF (K.GT.NDLOG) F
  T

CONT. ON PG 3

INSRT
PG 2 OF 10
DECIDE NUMBER, ITYPE.

IF (PAGE(2,2,K).NE.EQUAL)

CONT. ON PG 4
CALL BCD DEC (BCD, NC, VALUE)
IVAL = VALUE
GO TO 70

IF (ITYPE NE 3)

GO TO 60

CALL BCD INT (BCD, NC, IVAL)
GO TO 70

CONTINUE

WRITE (LUO, 1000) NAME, NUMBER

FORMAT (IH 30X, 16HERROR IN INSERT,, 21H THE FIRST RECORD ON A10/ IH .30X, 10HCANNOT BE A10.23H SET RECORD NUMBER TO 1)
IVAL = 1

CONT. ON PG 5

INSRT
PG 4 OF 10
CONTINUE

ISTART = IVAL

GO TO 40

CONTINUE

IEND = NRMAX

GO TO 200

CONTINUE

40

IOP = PAGE (2.3.K)

IF (IOP.NE.BLANK.AND.IOP.NE.DELIM) F

GO TO 42

IEND = ISTART

GO TO 200

CONTINUE

NUMBER = PAGE(3.3.K)

CONT. ON PG 6

INSRT

PG 5 OF 10

A85
CALL DECIDE (NUMBER, BCD, ITYPE, NC)

IF (ITYPE .NE. 2)

   T

   GO TO 150

   CALL BCD DEC (BCD, NC, VALUE)

   IVAL = VALUE

   GO TO 170

   CONTINUE

   IF (ITYPE .NE. 3)

   T

   GO TO 160

   CALL BCD INT (BCD, NC, IVAL)

   GO TO 170

   CONTINUE

CONT. ON PG 7
IVAL = NRMAX

CONTINUE

IEND = IVAL

CONTINUE

CALL CREATF (IADD, DBUFFR, LDBUF, DBFET, NAME)

IF (ISTART.LE.0) THEN
  ISTART = 1
END IF

IF (IEND.LT.ISTART) THEN
  IEND = ISTART
END IF

IC = 0

DO 280 I = 1, NRMAX
  IC = IC + 1
CONT. ON PG 8

DO 280 I = 1, NRMAX
  IC = IC + 1
CONT. ON PG 8

CONT. ON PG 8

INSRT

PG 7 OF 10
CALL READBR (IADD, IMAGE, NWREC)

IF (MYEOF)
  F
  T
    GO TO 290

IF (I.GT.IEND)
  F
  T
    GO TO 290

IF (I.LT.ISTART)
  F
  T
    GO TO 280

CALL PGMOUT($290,$290,IMAGE,BLANK)

IOCONT=IOCONT+1

IF (EDIT)
  F
  T

CONT. ON PG 9

INSRT
PG 8 OF 10

A88

ORIGINAL PAGE IS
OF POOR QUALITY
CALL WRITCR(LUO,IMAGE,NWREC)

CONTINUE

IF (IC.EQ.1)

GO TO 300

GO TO 600

WRITE (LUO,3000) NAME

FORMAT (IH,30X,SHFILE A10,35HDOES NOT EXIST. EXECUTION CONTINUES)

CONTINUE

IF (K.GE.NDLOG)

CONT. ON PG 10

INSRT

PG 9 OF 10
CONTINUE
MYEOF=.FALSE.
IF( TRACER )
WRITE(LU0,1020)
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), INITIAL)
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)
/1HO,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 (INITIAL, BLANK, INUM, NNUM, FOUND, IVAL, 4, NFN)
C............ LOAD THE DIRECTORY ..............
DO 100 I=1,NN
CALL RANDAC (FIND, NAMEN(I), INUM, NNUM, FOUND, IVAL, 4, NFN)
IF (FOUND) THEN
  T
ELSE
  F
END IF
GO TO 100
CONT. ON PG 2
NUMNIT
PG 1 OF 2

A91
CALL RANDAC(INSTALL, NAMEN(I), 1, INUM, NNUM, FOUND, IVAL, 4, NFN)

1 - - - -
A0
IVAL=IVALN(I)

A1
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
ONGING=.TRUE.
OFFING=.FALSE.

GO TO 100

ENTRY OFF
OFFING=.TRUE.
ONGING=.FALSE.

CONTINUE

LETTER=PAGE(3,1,2)

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

IF(FOUND.AND.ONGING)

F

T

IDILOG(IV)=.TRUE.

CONT. ON PG 2

ONROFF
PG 1 OF 2

A93
IDILOG(IV) = .FALSE.

IF (TRACER)
WRITE (LUO, 1000) LETTER, IDILOG(IV)

1000
FORMAT (IX, 14HLEAVING ON/OFF/I4, 6H = .L2)
RETURN

END
SUBROUTINE OP INIT
COMMON /DIRECT/ IDIREC(I)
EQUIVALENCE (IDIREC(1), IOPR)
COMMON /DILOG/ IDILOG(I)
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(I), IV(I)
INTEGER BLANK, DELIM, FIND
INTEGER NAMEOP(10), CHAROP(10)
DATA NOPR/10/

DATA (NAMEOP(I), I=1.8)
/SHEQUAL, 4HPLUS, 5HMINUS, 6HMLTPLY, 6HDIVIDE, 5HEXPN
, 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, S, NFO)

DO 100 I=1,NNAME
CALL RANDAC(FIND, CHAROP(I), 1, IOPR, NOPR, FOUND, IV, S, NFO)

IF (FOUND)

CONT. ON PG 2  OPINIT
PG 1 OF 2
IV(1) = I
IV(2) = NAMEOP(I)
CALL RANDAC(INSTAL, CHAROP(I), 1, IOPR, NOPR, FOUND, IV, S, NFO)

CONTINUE

IV(2) = BLANK
DELIM = CHAROP(B)
RETURN

END
SUBROUTINE OPTION

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

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

INTEGER BLANK, FIND
DATA LETTER/IRA,IRB,IRC,IRE,IRM,IRL,IRO,IRD, IRS,IRT/
DATA LOC/340,292,36,250,311,300,301,304,305,307/
DATA NOP/13/
LOGICAL IDILIG
DEFINE OPT(I)=FLD(35-(IRZ-I),1,MASK),NE.0

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

CALL IOPT(MASK)

DO 100 I=1,10

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

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

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

CONT. ON PG 2
IF(FOUND)
  CALL RANDAC(INSTAL.KI1.,IDOP,NOP,FOUND,IV,4,NFOP)
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 IBFC(7)
COMMON/MS/NT. KEYWRD, NREC, LENGTH, INCLEN
EQUIVALENCE (KW, IBF(3))
IOP=IPP
M=MK

C TEMPORARY DEFINITION OF IUN ******0*0*0*0*0*0*

IRTN=0

C IUN--DISC UNIT DEDICATED TO MS STORAGE
C IOP --OPERATION CODE
  =5H(LEAR--THIS IS USED AFTER A FILE HAS BEEN COMPLETED
  USING CODES 20, 21, 30, 31.
  =5HCLOSE--THIS IS USED TO CLOSE THE LIBRARY SO THAT IT
  MAY BE PICKED UP BY A SUBSEQUENT JOB STEP
  =10HPURGE--REMOVE THIS FILE FORM THE LIST OF RETRIEVABLE
  DATA FILES
C =+N WRITE
C =N READ
C =10HTAPEINPUT
C =10HTAPEOUTPUT
PERMANENT STORAGE OF MS DATA--SEE INSTRUCTIONS
BELOW
WRITE CODES
N=10 DATA IS COMPLETE IN IDATA(MATRIX STORE)

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

CONT. ON PG 2
IBM. -- THE BUFFER TO USE FOR THIS FILE

NBUF -- THE LENGTH OF THE BUFFER

--- --- --- PERMANENT STORAGE OF MS DATA

N=TAPE UNIT ON WHICH TO WRITE TAPE

IT= A WORKING ARRAY LARGE ENOUGH FOR THE LONGEST TITLE
IN THE STORED MS DATA

IDATA= A WORKING ARRAY LARGE ENOUGH TO ACCOMODATE THE LARGEST
BUFFER USED TO WRITE THE MS TAPE.

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

IFLG=0

FIRST OPERATION -- OPEN MS AND ESTABLISH INDEXES

This section is temporary and will be moved to a new place in
the GAC

IUNDAT(1)=IUN------UNIT NUMBER

IUNDAT(2)

to

IUNDAT(12)------KDX ARRAY

IUNDAT(13)=MDX

IUNDAT(14)=KCR

IUNDAT(15)=KFLG

IUNDAT(16)

to

IUNDAT(NREC+15)----INDEX ARRAY

NOTE-- INITIALIZE IUNDAT TO 0

NT TO 3

KEYWRD TO 2

NREC TO 256

LENGTH TO + (200)

INCLEN TO 50

INNEW=0

IUN=IUNDAT(1)

KCR=IUNDAT(14)

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

PRIT
PG 3 OF 8
CONTINUE

IUNDAT(13)=1
KCR=1
IUNDAT(14)=1

GO TO 1

CONTINUE

K=IUNDAT(1+2)
READ(IUN'K')(IUNDAT(15+I),I=I,NREC)
K=K
KCR=1
IUNDAT(14)=1
IUNDAT(15)=0

CONTINUE

INDEX MUST BE FOUND OR GENERATED
INDEX MAP--N,IT(1 TO N+1)
IUNDAT(16)--NEXT AVAILABLE INDEX

K=KCR

DO 100 I=1,11

IF(I.EQ.1)

CONT. ON PG 5

CONTINUE

CONT. ON PG 5

PRTT

PG 4 OF 8
A3

K = K + 1

IF (I.EQ.11)

F

T

GO TO 100

IF (K.GT.10)

F

T

K = 1

IF (IUNDAT(K+2).EQ.0)

F

T

GO TO 100

IF (IUNDAT(KS).EQ.0)
IF (IUNDAT(KCR+2).NE.0)
CALL NXTAD(IUN,IUNDAT(13))
IUNDAT(KCR+2)=IUNDAT(13)
CONTINUE

KK=IUNDAT(KCR+2)
WRITE(IUN"KK)(IUNDAT(15+JJ),JJ=1,NREC)
KK=KK
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
A6
101
CONTINUE

IUNDAT(16)=0
KINC = NT + KEYWRD
KFIN=((NREC-1)/KINC)*KINC+1

DO 105 J=2,KFIN,KINC

IF(IUNDAT(15+J),NE.0)

GO TO 106

IF(IUNDAT(16).EQ.0)

IUNDAT(16)=J

GO TO 105

CONTINUE

DO 555 JK=1,NT

IF(IUNDAT(J+JK+14).EQ.0)

CONT. ON PG 8

PG 7 OF 8

A105
IUNDAT(J+JK+14) = IH

PRINT 1110, (IUNDAT(J+L+14), L=1,NT)

1110 FORMAT(3X, 3A7)

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 /BCNTRL/ IT(5), IBUF(256)
COMMON IDATA(1)

EQUIVALENCE (ID, IDATA)
LOGICAL STORE, STSAVE, TRACER
INTEGER PAGE, BLANK

IF (TRACER)
WRITE (LUO, 1010) NODIR

1010
FORMAT (13H ENTER RSPOND, 14)

GO TO (100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100).

CONTINUE
ADD
STSAVE = STORE
STORE = .TRUE.

CONT. ON PG 2

A107
CALL ADDER
STORE = STSAVE
GO TO 40

CONTINUE α
ATTACH

CALL ATTACH
GO TO 40

CONTINUE α
CHANGE

CALL CHANGE
GO TO 40

CONTINUE α
COMENT

CALL IGNORE
GO TO 40

CONTINUE α
COPY

CALL COPY
CONT. ON PG 3

RESPOND
PG 2 OF 7

A108
1000
CONTINUE α
DETACH

CALL DETACH

GO TO 40

1100
CONTINUE α
FORMAT

CALL FORMAT

GO TO 40

1200
CONTINUE α
INLINE

CALL INLINE

GO TO 40

1300
CONTINUE α
INSERT

CALL INSRT

GO TO 40

1400
CONTINUE α
OFF

CALL OFF

CONT. ON PG 5

A110
CONTINUE α

CALL TIME

GO TO 40

CONTINUE α

CALL USE

GO TO 40

CONTINUE α

CALL UPDATE

GO TO 40

2100

CONTINUE

CALL PRT(DUM,DUM,DUM,DUM,IBUF,IUNA,IUNA(2))

GO TO 40
CONTINUE

LC=1

CONTINUE

IF (TRACER)

WRITE (LUO, 1020) LC

RETURN

END
SUBROUTINE SEARCH

C*** PROCESS "SEARCH ... " DIRECTIVE.

COMMON /DILOG/ IDILOG(I)
EQUIVALENCE (IDILOG(338), LUO)

WRITE (LUO, 1000)

1000 FORMAT (34H SEARCH DIRECTIVE NOT IMPLEMENTED )

RETURN

END
SUBROUTINE TIME

C*** PROCESS "TIME ..." DIRECTIVE.

COMMON /DLOG/ 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)

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 )

INTEGER PAGE, DDBASE
LOGICAL TRACER

C***

IF (TRACER)

WRITE (LU0.1010)

FORMAT (13H ENTER UPDATE )

IFIL = PAGE (3,1,2)

CONT. ON PG 2

ORIGINAL PAGE IS OF POOR QUALITY
CALL DBLOAD(IFIL)

IF (TRACER)
  T
  WRITE (LUO.1040) ODBASE, LD, LK, LENDES, NWORD, LDB
  1040
  FORMAT(43H ODBASE DIRLEN KEYLEN LENDES NWORD LTOTAL / IX, A6, 517)
  IF (TRACER)
    T
    WRITE (LUO.1020)
    1020
    FORMAT (13H LEAVE UPDATE )
    RETURN
    END
  F

UPDATE
PG 2 FINAL
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)
  T
  PRINT 103
  103
  FORMAT( " ENTERING USE"")
  IF(PAGE(2,1,2).EO.DELIM)
    F
    T
    NONAMS=1
  F
  A
  T

CONT. ON PG 2

USE
PG 1 OF 3

A118
CONTINUE

IF(PAGE(2,1,NDLOG).EQ.DELEM)

T

NONAMS=NONAMS-1

CALL DBLOAD(DBNAMS(I))

INX=1

CONTINUE

IF(EDIT)

F

T

PRINT 101.(DBNAMS(L).L=1,NONAMS)

CONT. ON PG 3

USE

PG 2 OF 3

A119
2 A2
101 FORMAT(15A7)
IF( TRACER )
T
PRINT 102
F
102 FORMAT( " LEAVING USE" )
RETURN
END