

UNCLASSIFIED

(U) CONSTRUCTION ENGINEERING RESEARCH LAB (AR
CHAMPAIGN IL E S NEELY OCT 79 DOD/DF-83/002J

. F/G 9/2

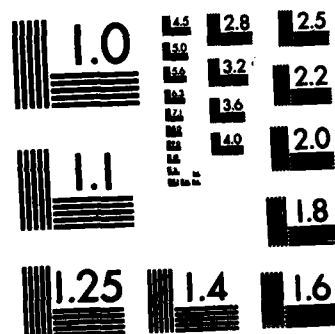
NL

END

2.4.3. MEQ

9

ONE



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A 124212

EDITSPEC SYSTEM MANUAL
VOLUME 8: PROGRAMMERS MANUAL

by
E. S. Neely

October 1979

DTIC
ELECTE
S FEB 8 1983 D
A

Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
P.O. Box 4005
Champaign, IL 61820

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22101

This document has been approved
for public release and sale; its
distribution is unlimited.

DTIC FILE COPY

83 02 08 144

77



UNITED STATES DEPARTMENT OF COMMERCE
National Technical Information Service
5205 Port Royal Road
Springfield, Virginia 22161

Date 14 Feb 1983

NTIS Control # 035042

TO: Defense Technical Information
Center - DTIC
Cameron Station
Alexandria, Virginia 22314

FROM: NTIS, Input Branch
5285 Port Royal Road
Springfield, Virginia 22161

Report # DOD-DF-83-0429 ADA # _____

Title: EDITS PEC - Sig manual v 8

Subject report is ☒ Standard Process ☐ STG report. STG - Special
Technology Group. ☐ Computer Product ☐ Follow up date _____

☐ The report will be accessioned by DDC. The form noting the ADA number
is returned.

☐ The report has been assigned the ADA number noted above and is return
to NTIS for processing.

☐ DDC will not process the report. It is returned to NTIS

Mag Tape Price _____
PC & MF Price _____

Source DOD A

Stock Quantity _____

Source Share _____

Comments:

Dottie Adams
Signature (for billing)

Copy when completed to
Finance Branch

DOD Report Action Request
(Replaces NTIS-164 5-72)

Please return to NTIS - Information Services Branch, Room 301, Yorktown Building.
Attention: Dottie Adams

REPORT DOCUMENTATION PAGE		1. REPORT NO. DOD/DF-83/002j	2.	3. Recipient's Accession No. AD-A 124 212
4. Title and Subtitle EDITSPEC: System Manual, Volume VIII: Programmers Manual				5. Report Date October 1979
7. Author(s) Edgar S. Neely, Jr.				6.
9. Performing Organization Name and Address Department of the Army Construction Engineering Research Laboratory P.O. Box 4005 Champaign, IL 61820				8. Performing Organization Rept. No.
10. Project/Task/Work Unit No. 4A762731AT41/T1/009				11. Contract(C) or Grant(G) No. (C) (G)
12. Sponsoring Organization Name and Address (same)				13. Type of Report & Period Covered
15. Supplementary Notes For magnetic tapes, see				14.
16. Abstract (Limit: 200 words) The EDITSPEC System is an automated system designed to produce construction specifications from Corps of Engineers Guide Specifications. The System uses one central computer and a communications network to provide remote terminal access by Corps offices, nationwide to a central data base. This volume provides programmers with the procedures to master the EDITSPEC system and to add new features to the system. EDITSPEC's internal character representation and command processing are presented first. The standard application commons and subroutines are described next. The required access methods to tables are then given. The last chapter provides the suggested method for adding new functional commands to the system.				
17. Document Analysis a. Descriptors Construction Specifications Guide Specifications Military Construction b. Identifiers/Open-Ended Terms c. COSATI Field/Group				
18. Availability Statement:		19. Security Class (This Report) UNCLASSIFIED		21. No. of Pages
		20. Security Class (This Page) UNCLASSIFIED		22. Price

ABSTRACT

This volume provides programmers with the procedures to master the EDITSPEC system and to add new features to the EDITSPEC system. EDITSPEC's internal character representation and command processing is presented first. The standard application commons and subroutines are described next. The required access methods to tables are then given. The last chapter provides the suggested method for adding new functional commands to the system.



Accession For	
NTIS ORAL	☒
NTIS TAP	☐
Unpublished	☐
Classification	
Distribution	
Availability Codes	
Availability	
Special	
A	

CONTENTS

	<u>Page</u>
ABSTRACT	
FOREWORD	
1. MASTERING THE EDITSPEC SYSTEM	
2. EDITSPEC CHARACTER REPRESENTATION	
3. EDITSPEC COMMAND PROCESSING	
4. STANDARD APPLICATION COMMONS	
5. STANDARD APPLICATION SUBROUTINES	
6. STANDARD ACCESS TO TABLES	
7. ADDING NEW FUNCTIONAL COMMANDS	

APPENDICES

	<u>Page</u>
A - USER MANUAL TYPING NOTES	
B - SIT TEST DECK EXAMPLE	
C - DOCUMENTATION EXAMPLE	
D - PROGRAMMING NOTES AND PROCEDURES	
E - PROGRAM COMMON AREAS	

1. MASTERING THE EDITSPEC SYSTEM

This chapter attempts to guide the programmer who knows nothing about the EDITSPEC system into a position of understanding the logic of the system design and being ready to begin program design and implementation.

The best way to obtain a working knowledge of the system is to understand the functions that the system is currently performing. The programmer should read the first few chapters of the users manual in detail and then review the purpose of each command in the remaining chapters.

The basic programmer's tools are FORTRAN, the data handler, and the table handler. The programmer should already have several years of experience in coding production programs in FORTRAN. FORTRAN manuals should be obtained for reference. All codes should follow the standards of ANSI FORTRAN.

A solid working knowledge of the data handler and table handler should be obtained next by reading Volumes IV and then III of the EDITSPEC System Manual. Both systems are basic tools needed to perform coding for EDITSPEC.

The programmer should then start to learn the system design concepts by reading the "Construction Specification Preparation Within the EDITSPEC System" report. Volume 1, Systems Overview, should be read next followed by Volume 2, System Design Concepts.

Detailed programming instructions are given in the remaining sections of this manual.

2. EDITSPEC CHARACTER REPRESENTATION

The actual representation of a character in machine code and the sequential order of the machine character representation are entirely machine dependent. The EDITSPEC system uses one consistent internal code to represent all allowable characters. This internal code is shown in Table 2.1. *where*

The first 26 internal characters identify all special characters recognized by the system. Special characters are identified as punctuation or nonpunctuation characters as shown in the table. A word within the text of a document may have punctuation characters before and/or after the characters in the word. For example, the word "house" may appear as "(house)" in the text. The punctuation characters must be overlooked when trying to locate the word "house."

The next 26 characters with internal codes 27 through 52 represent the alphabetic character set. EDITSPEC internal representation represents lower-case alphabetic characters as the characters themselves. Upper-case alphabetic characters are represented by two internal characters. The first is an internal code of 63 to indicate that the next character is to be printed as an upper-case character. The second character is the lower-case representation of the upper-case character.

The two-character notation was adopted to minimize computer searching time when trying to locate all occurrences of the word "house." The text may contain house in several forms:

1. house
2. House
3. HOUSE
4. hoUSE
5. H0use

Text searches are normally performed in all lower case with the internal code 63 ignored during comparison.

The internal codes 53 through 62 represent the characters "0" through "9."

When the user enters a character that is not contained in the internal code, the characters will be translated to a question mark (?).

3. EDITSPEC COMMAND PROCESSING

The FORTRAN subroutine CMMND controls the processing of all commands. CMMND calls another subroutine GTCMD to get the next command. GTCMD calls subroutine RDLIN to read the command from an external device. RDLIN calls one of four subroutines to perform the actual reading of the command.

The /PARSC/ common is used to store all related information about the command. The machine character representation of the command is stored in array MCLIN in "A1" format for machine-independent writing. Array STRNG contains the command in EDITSPEC internal codes in a packed format.

The commands are decoded by the subroutine DECOD. Array PMPTR contains the total number of parameters in the command in PMPTR (1, 3). The actual information about each parameter is stored in the rows of PMPTR. The parameter types and PMPTR information is given in Table 2.2, Decode Variables.

The DECODE subroutine calls subroutine MATCH to identify the type of command that has been entered. MATCH compares the command name string entered with the allowable command name acronyms and sets variable ISUB to point to the correct subroutines for CMMND to transfer control to for actual processing of the command itself.

When the command processing routine returns control to CMMND, CMMND will start the process all over again.

SUBROUTINE DECOD

PROGRAM: SUBROUTINE DECOD

FUNCTION: INTERPRET A COMMAND AND CALL THE PROPER SUBROUTINE

AUTHOR: PETER KARP

MODIFICATIONS:

LANGUAGE: FORTRAN

CALLING SEQUENCE: CALL DECOD

TASKS OR MODULES:

VARIABLES:

CHARY - ARRAY WHICH CONTAINS UNPACKED COMMAND STRING
ICHAR - INDEX INTO CHARY (POINTS TO CURRENT CHARACTER)
ISUB - VALUE USED IN COMPUTED GO TO DETERMINE WHICH
COMMAND SUBROUTINE TO CALL
KOUNT - USED TO CALCULATE EACH PARAMETER SIZE
NCHAR - MAXIMUM NO. OF CHARACTERS IN THIS COMMAND STRING
MAXST - MAXIMUM ALLOWABLE STRING LENGTH
NCHST - COUNTER FOR CURRENT LENGTH OF CURRENT STRING
PMPTR - A 2-DIMENSIONAL ARRAY WHICH CONTAINS THE DESCRIPTION
OF ALL PARAMETER FIELDS FOUND FOR THE PRESENT COMMAND
PMPTR (1,1) = NOT USED
PMPTR (1,2) = NOT USED
PMPTR (1,3) = THE NO. OF PARAMETER FIELDS FOUND FOR THIS
COMMAND
PMPTR (N,1) = THE CHARACTER LOCATION WITHIN THE COMMAND
STRING OF THE FIRST CHARACTER OF THIS
PARAMETER
PMPTR (N,2) = THE LENGTH OF THIS PARAMETER (NO. OF CHAR)
PMPTR (N,3) = PARAMETER TYPE
PTYPE - TYPE OF PARAMETER

- 1 - NUMERIC
- 2 - ALPHANUMERIC STRING
(NOT ENCLOSED WITHIN DELIMITERS)
- 3 - ALPHANUMERIC STRING
(ENCLOSED WITH DELIMITERS)
- 4 - SEMICOLON
- 5 - HYPHEN
- 6 - COMMA
- 7 - DOUBLE ASTERISK
- 8 - PLUS
- 9 - COLON

PRMNO - THE CURRENT PARAMETER NO.
PRVCH - CONTAINS THE PREVIOUS CHARACTER
STEND - LAST USED STRING DELIMITER
STRNG - ORIGINAL COMMAND STRING

Table 2.2. DECODE VARIABLES.

4. STANDARD APPLICATION COMMONS

Several labeled common areas have been defined to contain information which insures that the code is as machine independent as possible. Programmers should use the variables in common instead of recalculating the values. The programmer should review all common areas to understand their application and contents.

The basic commons that the programmer should review are as follows:

- 1) Blank Common (unlabeled) - contains all general system information.
- 2) DEBUGC Common - Contains the debug switch.
- 3) EDITC Common - Contains all general variables related to editing a document.
- 4) FDITC Common - Contains all general variables related to accessing a second document while editing a document.
- 5) IOC Common - Contains all input and output logical device numbers.
- 6) LOCKC Common - Contains all resource allocation information.
- 7) MC Common - Contains several machine-dependent constants derived from NCU.
- 8) PARSC Common - Contains all information about the command being processed.
- 9) SCRTC Common - Contains one array 243 words long that can be used for table-handler access rather than dimensioning a new local array.
- 10) SIZEC Common - Contains all information about the number of standard units required to represent specific length character strings.
- 11) SYSTM Common - Contains all general information about the EDITSPEC system and system table.
- 12) TABLC Common - Contains information used by the table-handler system.
- 13) TPXXXX Commons - XXXX is the name of a system or document table. Contains offset information to access the tables and data records.

14) VFMT Common - Contains variable formats for printing different types of character strings.

A listing of all commons used in the EDITSPEC system is given in Appendix E.

5. STANDARD APPLICATION SUBROUTINES

Several basic application programs are available for the programmer's use. To insure uniformity in programming and easy modifications to basic functions, all programmers must use the standard application subroutines provided.

Two lists are provided for reference. The first (Table 2.3) contains the functions to be performed, followed by the name of the subprogram that performs the function. The second (Table 2.4) contains the calling sequence and the definitions of the parameters to be used.

<u>FUNCTION</u>	<u>SUBROUTINE</u>
<u>Character Conversions</u>	
<u>EDITSPEC to Machine</u>	
Packed Strings	ED2MC RCNVR IN2EX
Single Character	
<u>Machine to EDITSPEC</u>	
A1 Format String	MC2ED
Single Character	EX2IN
<u>EDITSPEC to Integer Number</u>	
Within Packed Strings	INTER
Beginning of Packed Strings	INTGR
<u>Integer Number to EDITSPEC</u>	
Packed	NO2IC
<u>Copy Characters to Another String</u>	
From left to right	ICOPY
From right to left	ICOPB
<u>Data Set Number</u>	
Primary	FDSMN
Backup	FDSBK
<u>EDIT A Second Document While EDITING a</u>	
Primary Document	FDIT FSTOR
<u>Move Characters Within a String</u>	CHIFT
<u>Packing Characters</u>	
EDITSPEC to EDITSPEC	PACK
<u>Unpacking Characters</u>	
Machine to Machine	
packed to A1 format	AN2A1
EDITSPEC to EDITSPEC	
packed to unpacked	UNPAK

Table 2-3. Programming Functions Supported by Standard Subroutines

SUBROUTINE AND VARIABLES		I/O	TYPE	CHANGED
SUBROUTINE: AN2A1 (SOURC,LENGTH,DESTN)				
FUNCTION: TO CONVERT FROM (AN) FORMAT; (N=NCU).				
VARIABLES:	SOURC () IS SOURCE STRING PACKED IN A4 FORMAT (FOR IBM360).	INPUT	A(t)	NO
	LENGTH IS NUMBER OF CHARACTERS IN SOURC.	INPUT	WORD	NO
	DESTN() IS DESTINATION ARRAY, WHERE STRING IS PUT IN A1 FORMAT.	OUTPUT	A(t)	YES
SUBROUTINE: CHIFT (LA,NSHFT,NSHS, NSHE,LP)				
FUNCTION: SHIFT CHARACTERS TO EITHER THE LEFT OR THE RIGHT OR TO PAD WITH A SPECIFIED CHARAC- TER.				
VARIABLES:	CALLING SEQUENCE: CALL CHIFT (LA,NSHFT,NSHS,NSHE, LP)			
	WHERE: LA - THE ARRAY IN WHICH SHIFTING IS TO OCCUR. (MAX DIMENSIONS = 126 WORDS)	INPUT/ OUTPUT	A(126)	YES
	NSHFT - THE AMOUNT OF SHIFT IN NO. OF CHARS. (-LT, +RT) VALUE OF 0 WILL BLANK AREA WITH EXT. BLANKS.	INPUT	WORD	NO
	NSHS - LEFTMOST CHAR. POSITION OF AREA TO BE SHIFTED.	INPUT	WORD	NO
	NSHE - RIGHT-MOST CHAR. POSITION OF AREA TO BE SHIFTED.	INPUT	WORD	NO
	LP - THE PADDING OR BLANK- ING CHARACTER.	INPUT	WORD	NO
SUBROUTINE: ED2MC(EDARR,SC,NC, UCSET,MCARR,NN)				
FUNCTION: TO CONVERT A STRING FROM EDITSPEC FORMAT TO MACHINE (A1) FORMAT.				
VARIABLES:	EDARR IS THE SOURCE STRING IN EDITSPEC FORMAT.	INPUT	A(t)	NO

Table 2.4. Standard Application Subroutines

SUBROUTINE AND VARIABLES		I/O	TYPE	CHANGED
SC IS THE STARTING CHARACTER NUMBER OF EDARR TO START CONVRTING.		INPUT	WORD	NO
NC IS NUMBER OF CHARACTERS OF EDARR TO CONVERT.		INPUT	WORD	NO
UCSET IS OUTPUT CHARACTER SET CODE.		INPUT	WORD	NO
MCARR IS OUTPUT STRING IN IN MACHINE (A1) FORMAT.		OUTPUT	A(t)	YES
NW IS NUMBER OF WORDS IN MCARR THAT HAVE BEEN CONVERTED.		OUTPUT	WORD	YES
SUBROUTINE: FUNCTION EX2IN(IXCAR)				
FUNCTION: CONVERTS ONE CHARACTER AT A TIME FROM MACHINE INTERNAL CODE TO EDITSPEC INTERNAL CODE.				
VARIABLES: EXTRN - AN ARRAY WHICH IS DEFINED N ELEMENTS LONG WHERE N EQUALS THE NO. OF POSSIBLE INTERNAL CHARACTER REPRESENTATION CODES ON A PARTICULAR MACHINE.				
IXCAR - IS THE MACHINE CODE VALUE FOR A PARTICULAR CHARACTER.		INPUT	WORD	NOT CHANGED
SUBROUTINE: FDIT(JDOCN,JREAD)				
FUNCTION: TO SET UP A SECONDARY EDIT DOCUMENT.				
VARIABLES: JDOCN(3) IS SECONDARY DOCUMENT NAME.		INPUT	A(3)	NO
IDAC IS DOCUMENT ACCESS CODE RETURNED BY ROUTINE FDAC.		OUTPUT	WORD	YES
SUBROUTINE: FDSBK(DSNM,DSNO)				
FUNCTION: TO GET DATASET NUMBER FROM DATASET NAME (BACKUP - NOT MAIN)				
VARIABLES: DSNM(2) IS 6 CHARACTER DATASET NAME.		INPUT	A(2)	NO
DSNO IS DATASET NUMBER (FORTRAN LOGICAL I/O UNIT NUMBER).		OUTPUT	WORD	YES

SUBROUTINE AND VARIABLES		I/O	TYPE	CHANGED
TEMPO(8) IS TEMPORARY ARRAY TO STORE UNPACKED NAME. DSERR IS USED TO INDICATE AN ERROR CODE.				
<u>SUBROUTINE: FDSMN(DSNM,DSNO)</u>				
FUNCTION: TO GET DATA SET NUMBER FROM DATASET NAME (MAIN - NOT BACKUP)				
VARIABLES:	DSNM(2) IS 6 CHARACTER DATASET NAME.	INPUT	A(2)	NO
	DSNO IS DATSET NUMBER (FORTRAN LOGICAL I/O UNIT NUMBER).	OUTPUT	WORD	YES
<u>SUBROUTINE: FSTOR</u>				
FUNCTION: TO TERMINATE ACCESS TO A SECONDARY DOCUMENT.				
<u>SUBROUTINE: ICOPB(SOURC,SFST,NUM, DEST,DFST)</u>				
C - COPIES NUM CHARACTERS FROM SOURC (STARTING AT SFST) TO				
C - DEST (STARTING WITH DFST) BACKWARDS				
C - ALL OTHER CHARACTERS OF DEST ARE UNCHANGED				
VARIABLES:	SOURCE - PACKED INPUT STRING	INPUT	A(+)	NO
	SFST - RIGHT-MOST CHARACTER TO MOVE	INPUT	WORD	NO
	NUM - TOTAL NUMBER OF CHARAC- TERS TO MOVE	INPUT	WORD	NO
	DEST - PACKED OUTPUT STRING	INPUT/ OUTPUT	A(+)	YES
	DFST - RIGHT-MOST CHARACTER TO START	INPUT	WORD	NO
<u>SUBROUTINE: ICOPY(SOURC,SFST,NUM, DEST,DFST)</u>				
C - COPIES NUM CHARACTERS FROM SOURC (STARTING AT SFST) TO				
C - DEST (STARTING WITH DFST)				
C - ALL OTHER CHARACTERS OF DEST ARE UNCHANGED				

SUBROUTINE AND VARIABLES		I/O	TYPE	CHANGED
VARIABLES:	SOURCE - PACKED INPUT STRING	INPUT	A(+)	NO
	SFST - RIGHT-MOST CHARACTER TO MOVE	INPUT	WORD	NO
	NUM - TOTAL NUMBER OF CHARACTER TO MOVE	INPUT	WORD	NO
	DEST - PACKED OUTPUT STRING	INPUT/OUTPUT	A(+)	YES
	DFST - RIGHT-MOST CHARACTER TO START	INPUT	WORD	NO
<u>SUBROUTINE: INTEGER FUNCTION INTER(STRNG,IPTR,LENG)</u>				
FUNCTION: THIS ROUTINE COPIES A NUMERIC STRING FROM (STRNG) AND PLACES IT INTO THE (DIGIT) ARRAY. THE (DIGIT) ARRAY IS THEN CONVERTED INTO A NUMBER.				
VARIABLES:	CALLING SEQUENCE:			
	STRNG = THE STRING OF CHARACTERS REPRESENTING THE USERS INPUT COMMAND.	INPUT	A(+)	NO
	IPTR = FIRST CHARACTER IN THE STRING FOR THE FIRST NUMBER.	INPUT	WORD	NO
	LENG = TOTAL NUMBER OF CHARACTERS IN THE NUMBER.	INPUT	WORD	NO
<u>SUBROUTINE: INTEGER FUNCTION INTGR (A,NDIG)</u>				
FUNCTION: THIS ROUTINE FORMS THE ACTUAL NUMBER FOR THE STRING INTO THE MACHINE INTERNAL REPRESENTATION OF THE ACTUAL NUMERICAL VALUE BEING REPRESENTED.				
VARIABLES:	CALLING SEQUENCE:			
	A = NUMERIC STRING	INPUT	A(+)	NO
	NDIG = NUMBER OF DIGITS TO CONVERT	INPUT	WORD	NO
<u>SUBROUTINE: INTEGER FUNCTION INZEX(ICAR)</u>				
FUNCTION: CONVERTS THE EDITSPEC REPRESENTATION OF CHARACTERS INTO THE STANDARD MACHINE CODE REPRESENTATION.				

SUBROUTINE AND VARIABLES

I/O

TYPE

CHANGED

ICAR IS A WORD WHICH CONTAINS
ONLY ONE CHARACTER RIGHT-
JUSTIFIED
EXTRN IS THE MACHINE'S
REPRESENTATION OF CHARACTERS
THE EDITSPEC REPRESENTATION
IS A CODE FROM 1 TO 62 WHICH
CORRESPONDS EXACTLY TO THE
ORDERING OF EXTRN

SUBROUTINE NO2IC(NUMBR,CHARS,NCHAR)

FUNCTION: TO CONVERT AN INTEGER
NUMBER TO FOI INTERNAL
CHARACTERS. RESTRICTED
TO 10 SIGNIFICANT DIGITS.

VARIABLES: CALLING SEQUENCE:

CALL NO2IC(NUMBR,CHARS,NCHAR)

NUMBR IS THE INTEGER NUMBER
PASSED TO NO2IC

INPUT

WORD

NO

CHARS(3) IS AN ARRAY CONTAIN-
ING THE CHARACTERS AND IS
FILLED AND RETURNED BY NO2IC.

OUTPUT

A(3)

YES

NCHAR IS RETURNED BY NO2IC
AS THE TOTAL NUMBER OF
CHARACTERS STORED IN CHARS
BY NO2IC.

OUTPUT

WORD

YES

SUBROUTINE: PACK(SOURC,L1,NUM,
DEST,L2)

FUNCTION: PACK NUM CHARACTERS INTO
THE STRING DEST, STARTING
WITH CHARACTER L2 OF DEST.
THE CHARACTERS ARE TAKEN
FROM THE LEAST SIGNIFICANT
CHARACTER OF THE ELEMENTS OF
THE ARRAY SOURC, STARTING
WITH STANDARD UNIT L1.

VARIABLES: SOURC - PACKED INPUT
STRING

INPUT

A(+)

NO

L1 - RIGHT-MOST CHAR TO MOVE
NUM - TOTAL NUMBER OF CHARAC-
TERS TO MOVE

INPUT

WORD

NO

INPUT

WORD

NO

DEST - PACKED OUTPUT STRING

INPUT/
OUTPUT

A(+)

YES

L2 - FIRST CHARACTER IN
OUTPUT

INPUT

WORD

NO

SUBROUTINE AND VARIABLES	I/O	TYPE	CHANGED
--------------------------	-----	------	---------

SUBROUTINE: RCHVR (INREP, EXREP,
NCHAR)

FUNCTION: C - CONVERTS THE EDITSPEC
INTERNAL REPRESENTATION OF
CHARACTERS IN 'INREP' INTO THE
MACHINE INTERNAL REPRESENTATION
IN 'EXREP'.

'NCHAR' CHARACTERS ARE CON-
VERTED.

VARIABLES:	INREP - EDITSPEC PACKED ARRAY	INPUT	A(+)	NO
	EXTREP - MACHINE PACKED ARRAY	OUTPUT	A(+)	YES
	NCHAR - NO. OF MACHINE CHARAC- TERS	INPUT/ OUTPUT	WORD	YES

SUBROUTINE: UNPAK (SOURC,L1,NUM,
DEST)

FUNCTION: UNPACK NUM CHARACTERS
FROM THE STRING SOURC,
STARTING WITH CHARACTER L1
OF SOURC. THE CHARACTERS
ARE PLACED INTO THE LEAST
SIGNIFICANT CHARACTER OF
THE ELEMENTS OF THE ARRAY
DEST. THE REMAINING
CHARACTERS OF EACH ELEMENT
OF DEST ARE FILLED WITH THE
CONTENTS OF THE VARIABLE
'FILL' (ZEROES IN THIS
IMPLEMENTATION).

VARIABLES:	SOURC - PACKED INPUT STRING	INPUT	A(+)	NO
	L1 - STARTING CHARACTER TO UNPACK	INPUT	WORD	NO
	NUM - TOTAL NUMBER OF CHAR- ACTERS TO UNPACK	INPUT	WORD	NO
	DEST - OUTPUT ARRAY	OUTPUT	A(+)	YES

6. STANDARD ACCESS TO TABLES

During the actual execution of EDITSPEC in an interactive or multi-programming environment, several users may be executing different copies of the EDITSPEC code at the same time. Some of the users may wish to access (to read and/or write) the same resources (document or system table) at the same time. The "multi-user" feature will permit concurrently executing EDITSPEC programs to share the use of system tables and documents. Shared use of these resources must be strictly controlled in order to ensure that one program does not interfere with the correct execution of others. This control takes the form of synchronizing use of these resources on the part of the programs involved.

When a program requires use of a resource, it must request control of that resource from the operating system. Control of a resource can be either exclusive or shared. Exclusive control of a resource guarantees that no other program will be granted access to that resource (write access). Shared control guarantees that no other program will be granted exclusive control of that resource, but other programs will be granted shared control of that resource (read only access).

A request for either kind of control of a resource can be either conditional or unconditional. For conditional requests, control is granted only if the resource is immediately available. The requesting program is informed as to whether or not control was granted. For unconditional requests, control is granted as soon as the resource becomes available. The requesting program may have to wait for an indefinite amount of time. This eventuality is entirely transparent to the program itself.

When a program has finished using a resource, it must relinquish control of that resource so that it becomes available for use by other programs.

REQUESTING AND RELINQUISHING CONTROL OF RESOURCES IS THE PROGRAMMER'S RESPONSIBILITY. Two subroutines, LOCK and UNLOCK, have been made available for this purpose.

Required Programming Before Calling LOCK or UNLOCK

INTEGER RSRCS
COMMON /LOCKC/ RSRCS (7,40), NRSRCS

Before calling LOCK or UNLOCK, you must describe the resources to be requested or relinquished via the array RSRCS. Each column corresponds to one resource. The number of resources described in RSRCS must be defined in the variable NRSRCS.

Before a call to LOCK, RSRCS(2,I), RSRCS(3,I), RSRCS(4,I), and RSRCS(5,I) must contain the name of the I'th resource requested. RSRCS(1,I) must contain the number of the data set on which this resource resides. RSRCS(7,I) should = 1 if exclusive control of this resource is requested, 0 if shared control is requested. NRSRCS should contain the number of resources. RSRCS(6,I) should not be used within a processing program.

LOCK, as its name implies, is to be used to request control of resources.

LOCK has a single argument, the integer variable IND, which should = 1 if the request is conditional, 0 if the request is unconditional. If IND = 1, LOCK will return the status of the request via the argument IND. IND = 1 if control of all requested resources was granted, 0 if control over none of the resources was granted due to the non-availability of one or more of the resources. If the request was unconditional, then control of all requested resources was granted (IND was not modified and still = 0). Note that LOCK (or UNLOCK) never modifies LOCKC.

UNLOCK is to be used to relinquish control of resources. UNLOCK has no arguments, since control of resources is always relinquished unconditionally.

Application Rules

The following rules must be strictly observed:

- (1) Control of a resource should not be requested until it is needed;
- (2) A resource must not be used until after its control has been requested and granted;
- (3)
 - a. The use of DKOPN, DKPUT, DKINS, DKLOS, DKSET, DKRNM, DKXIT, or DKCLR in connection with a resource requires EXCLUSIVE CONTROL of that resource;
 - b. The use of DKFIL, DKLEN, or DKGET in connection with a resource requires shared or exclusive control of that resource;
 - c. The use of DKNDS, DKNIT, or DKDMP is unrestricted;
- (4) Control of a resource must not be relinquished until it is consistent with respect to both itself and other resources;

- (5) Control of a resource should be relinquished as soon as it is no longer needed;
- (6) A resource must not be used after its control has been relinquished;
- (7) The logic of execution should be such that: UNLOCK is not called without a previous call to LOCK; two calls to LOCK/UNLOCK are never made without an intervening call to UNLOCK/LOCK; LOCK is not called without a subsequent call to UNLOCK;
- (8) a. The same resource must not be referenced by two different columns in RSRCS;
 b. The maximum number of resources to be locked is 40;
- (9) No assumptions may be made regarding the contents of a resource at the time its control is granted;

Failure to observe one or more of the above rules will not necessarily result in the occurrence of a perceptible error condition (such as an ABEND), but resource integrity may nevertheless be seriously impaired. It is most important that great care in coding be exercised.

Obtaining Access From System Commands

When CMMND transfers control to a system command, no resources have been locked and the programmer should start from the first column to request resources. The programmer should obtain access to all resources needed at the beginning and give up all resources at the end of the system subroutine. NRSRCS should always be set to zero before returning from a system command.

Obtaining Access From EDIT and Internal Commands

When CMMND transfers control to an edit or internal command, access to the resources for the document tables have already been obtained by the EDIT or FDIT subprograms which were previously called.

Access to system tables must be performed by unlocking all resources, adding the system table to the end of the current list, and locking all resources. Removing access to system tables must be done by unlocking all resources, removing the last system tables that were locked, and locking the remaining resources if there are any in the array.

7. ADDING NEW FUNCTIONAL COMMANDS

Addition of the new functional commands should be performed in the following manner:

1. Write Users Manual Entries - Review the format for user commands as given in the Users Manual. The command description to be entered into the Users Manual should be prepared from the functional requirements provided by the proponent agency. Typing instructions are in Appendix A.

2. Review Users Manual Entries - The new text should be reviewed and approved by the proponent agency before any detailed design or coding is started.

3. Prepare SIT Test Deck - The complete test deck for each new command should be written. See Appendix B for an example.

4. Review SIT Test Deck - The test deck should be reviewed and approved by the proponent agency.

5. Design Subroutines - Prepare a description of the design in correct English prose. Prepare this description on coding sheets as comments to be used for documentation in the actual code.

6. Code Each Subprogram - The programmer should set up the standard subprogram documentation forms prior to coding the first line of code. A detailed description of the design in correct English prose should be prepared on coding sheet before actual coding is started. Code and document all coding as the coding is performed.

Subroutines CMMND and MATCH must be modified to transfer control to the correct processing program. See Appendix C for a Documentation Example.

See Appendix D for Programming Notes and Procedures.

7. Set up standard debug options for each subprogram.

The command .DEBUG X. will work in both system and edit mode to turn on the debug trace variable DEBUG located in /DEBUGC/ common. Every program should be written as follows with an IF statement to check the value of DEBUG:

X = 0 no debug print of any type

X = 1 The message "IN <SUBROUTINE NAME>" printed with the values of all variables passed to the routine in the subroutine call and in common. This is the first executable statement in the subroutine. If a common variable may be changed, it should be printed here first.

The message "OUT <SUBROUTINE NAME>" printed with the same variables as the "IN" write statement. This is the last executable statement in the subroutine.

X = 2 Detailed debug output

All debug statements should be labeled as "GUBED-C" as the last seven characters in card column 74 through 80. This automatically identifies all debug statements.

Debug statements without the "IF" check are permissible for debugging purposes. These cards should never be removed from the program. Instead, the cards should be reversed to show the C-DEBUG as the first characters. This will allow another programmer to reverse the cards again to obtain the real detailed debug print.

If this procedure is followed, no debug work will ever be lost and have to be redone by a future programmer.

The list routines .LIST. and .LT. are provided to allow the programmer to adequately test his programs. If new tables are added to the system, new list routines should be rewritten for each new table.

8. Hand Debug the SIT Deck - The programmers should play computer and write out a complete trace of each and every command in the SIT deck. This should insure that the complete logic of the program has been tested and the program is correct.

9. Computer Debug the SIT Deck - The programmer should run the SIT deck with the DEBUG switch set to 2 and compare the results with the hand debug output.

10. Complete Cross-Reference Indices - There are several indices that need updating:

1. Commons used
2. Tables used
3. Subprogram cross-reference (called & called by) index cards
4. Conversion forms

APPENDIX A
USER MANUAL TYPING NOTES

The text to be typed is a portion of the "EDITSPEC USER'S MANUAL."
The manual is composed of several hundred "COMMANDS" and "CHAPTERS."

Each command and chapter should be typed and stored as a separate document (or file).

Each command and chapter document (or file) should be named
"EDITSPEC USER'S MANUAL - command name" or chapter number.

Example - Command Name = INPUT
EDITSPEC USER'S MANUAL = INPUT

All documents will be stored until FY84.

PLEASE TYPE IN FINAL FORM ACCORDING TO ATTACHED INSTRUCTIONS.

USER MANUAL TYPING INSTRUCTIONS

1. A one-inch margin should be placed on all four sides of an 8 by 10 1/2" sheet of paper.
2. Each command will begin on a new page.
3. The complete phrase applied to define the command will be typed in capital letters and right justified on the first text line of every page related to the command.
4. The shortest acceptable character string will be typed in capital letters on the second text line of every page related to the command. The first character will be typed immediately below the first character of the complete phrase applied to define the command.
5. The titles of the subject areas:

<u>TITLE</u>	<u>FIRST CHARACTER INDENTION</u>
PURPOSE	45
GENERAL FORM	42
FIELD OPTIONS	42
SPECIAL NOTES	42
EXECUTION PROCEDURES	39
COMMAND VARIATIONS	39
MESSAGES	43

will be typed in capital letters, centered, and underlined on a new line. Two lines are to be skipped before and after the subject area title. If the subject is not required, the word "None" will be printed after the title.

Example: FIELD OPTIONS - None

6. The "GENERAL FORM" subject area is composed of two different typing formats. The first typing format will be boxed in by an illustrator before printing. An additional three lines will be skipped after the GENERAL FORM title to allow for the box. This is a total of 5 lines to be skipped after the general format title. Two lines after the title and three lines for the box before the first text line. All lines should be indented 5 spaces from each margin. The first few characters, the shortest acceptable character string, will be typed in capital letters. The rest of the text will be typed in italics. Stop codes should be placed to allow type ball changes. Italics should not be used in drafts of the commands. This will speed up the draft preparation. Three lines will be left after the completion of this format for the illustrator's box.

The second typing format contains a three-column table. The first column, 12 characters in width, contains the word "where," left justified in the first row. The first column is blank for all other rows. The text in the second column, 24 characters in width, will be typed left justified and in italics. Stop codes should be placed to allow type ball changes. The text in the third column, 36 characters in width, will be typed left justified in regular type. The first column should have 12 spaces, the second column 24 spaces, and 36 for the third column. The first word should not be capitalized.

7. FIELD OPTIONS. The field options section is composed of a three-column, 24 spaces each, table. The table rows are separated by one skipped line, one straight line, and one skipped line. The column titles "FIELD, OPTIONS, DEFAULT" should appear on each page that contains this section. Text in the first column is in italics, and the rows are numbered by integers with a period and 2 spaces following the integer. The second and third columns are in normal type. The first, second, and third columns should be 24 spaces wide. The first word should not be capitalized. Leave at least 2 spaces between columns.

8. SPECIAL NOTES. Use italics where noted.

9. EXECUTION PROCEDURES. Use italics where noted.

10. COMMAND VARIATIONS. Each variation may contain the following formats: The first is the variation number, left justified, marked "Var. #n". This is followed by the command in upper case regular, then the italics type. This line should be underlined. The second is the Example number indented 5 spaces and underlined. The general form of the third format indented 5 spaces and underlined is:

Text before:

Command:

Text after:

11. MESSAGES. If no messages are to be typed the phrase "All messages self-explanatory" should be placed after "MESSAGES."

↓ 1"

ENTER
EN

PURPOSE

↕ 2 Lines

↕ 2 Lines

This edit command allows the user with write access to insert a new line of text into an existing document.

GENERAL FORM

↕ 2 Lines

5 Lines

.ENTER line number; text; text segment id

↕ 3 Lines

where

line number

is the 1- to 8-digit line number of the new text to be inserted.

text

is line of not more than 384 characters, counting capital letters twice. The string must be enclosed in parentheses or a special character.

1" 12 spaces

2" text segment 34 spaces

3" 36 spaces

is a 1- to 4-character alphanumeric id.

FIELD OPTIONS

↕ 2 Lines

↕ 2 Lines

FIELD	OPTIONS	DEFAULT
1. line number	valid line number	add text line to end of document.
2. text	1-384 alphanumeric characters, counting capital letters twice	no default. field required.

2" 34 spaces

2"

2"

ENTER
EN

FIELD	OPTIONS	DEFAULT
3. <i>text segment</i>	1- to 4-character alpha-numeric id.	no text segment assigned.

SPECIAL NOTES - None

EXECUTION PROCEDURE

If the line number field is blank, then the document's current increment is added to the last number in the document to calculate the entered text line number.

COMMAND VARIATIONS

Var. #1. .EN_line number; (text) text segment

Example #1.

Text before:

Text Table

LN
14000
14200

Text
Additional ----- shingles
*p3*open ----- accordance

Command: .EN_14100; (on each side of the valley);A.

Text after:

Text Table

LN
14000
14100
14200

Text
Additional ----- shingles
on each side of the valley.
*p3*open ----- accordance

The text is entered on line 14100 with a text segment id of A.

ENTER
EN

Var. #2. .EN_line number; (text)

Example #1.

Text before:

Text Table

<u>LN</u>	<u>Text</u>
15200	two ----- each
15400	*p3* ----- surfaced

Command: EN_15300; (shingle tab along open metal valleys.)

The text will be added as line 15300 with no text segment.

Text after:

Text Table

<u>LN</u>	<u>Text</u>
15200	two ----- each
15300	shingle tab along open metal valleys.
15400	*pe* ----- surfaced.

Var #3. .EN_ (text); text segment

Text before:

<u>LN</u>	<u>Text</u>
22500	construction ---- wind
end	

Command: .EN_ (areas, *sl*).

The new line will be added after line 22500 with the documents increment (100).

Text after:

Text Table

<u>LN</u>	<u>Text Table</u>
22500	construction ---- wind
22600	areas, *SL*
end	

MESSAGES - All messages self-explanatory.

↓ 2 Lines

SIT TEST DECK EXAMPLE

SIT TEST DECK EXAMPLE

27

CERL TEST DECK III

NEW PRINTTEST:TESTDATA .
~~EDIT PRINTTEST~~ .
 INPUT .

THIS DOCUMENT TESTS THE PRINT COMMAND . ALL FIELDS WILL BE TESTED FOR EACH SURFIELD DELIMETER AND EACH PARAMETER TYPE .
 THE PRINT COMMAND MUST INTERACT WITH THE BEGIN PARAGRAPH AND PAGE NUMBER COMMANDS, AND ALL FORMAT TABLES.

THERE ARE NO INTERRELATIONSHIPS BETWEEN FIELDS.
 INITIALIZATION REQUIRED TO TEST PRINT COMMAND
 DEFINE DOCUMENT FORMAT UNDER ID OF 1 & 22

FIELD 1 - DOCUMENT FORMAT ID

TEST NUMBER	DESCRIPTION
-------------	-------------

- | | |
|----|--|
| 1. | BLANK FIELD -(EX10 |
| 2. | NOT IN THE SYSTEM TABLE-(EX 30 |
| 3. | MORE THAN EIGHT INTEGER CHARACTERS (EX23 |
| 4. | IN THE SYSTEM TABLE (EX1,8,9,11 THR |

FIELD 2 - COLUMNS TO BE PRINTED

- | | |
|-----|--|
| 5. | BLANK FIELD (EX8,9,23 |
| 6. | UNALLOABLE LETTER (EX1,10 |
| 7. | LENGTH GREATER THAN ONE CHARACTER (EX2 |
| 8. | F 16, 14.-F (EX12,31 |
| 9. | L 17 15.-L (EX13,32 |
| 10. | T 18 16.-T (EX11,33 |
| 11. | A 21 17.-A (EX12,31 |
| 12. | X 20, 18.-X (EX11 |
| 13. | P 21, 19.-P (EX22 |
| 20. | REPETITION OF THE SAME CHARACTER (EX14 |
| 21. | BOTH THE LETTER AND ITS NEGATIVE (EX14 |

FIELD 3 - ARFA

STARTING LINE NO.	ENDING LINE NO.	TEXT SEGMENTS		
		ONE	TWO	THREE
22.	BLANK	BLANK	BLANK	BLANK (EX8,9
23.	0	0	BLANK	BLANK (EX10,20
24.	0	GTR SLN	BLANK	BLANK (EX11
25.	0	LES SLN	BLANK	BLANK (EX12
26.	GTR ELN	0	BLANK	BLANK (EX13
27.	LES ELN	0	BLANK	BLANK (EX14
28.	0	0	VALUE1	VALUE2 VALUE3 (EX34
29.	BLANK	BLANK	VALUE1	VALUE2 (EX16,17
30.	NUMMER	NUMBER	VALUE1	(EX18
31.	9 CTR NO	9 CTR NO	VALUE1	VALUE1 (EX23(9),12
32.			5 CTR ALP	5CTR ALPHA (EX13
33.	ONE OR THREE NOS		FOUR TEXT SEGMENTS (EX24,25,26,27	
34.A	TEXT SEGMENT	LINE NUMBER LIST	(EX35	
34.B	NUMMER1	NUMBER1	(EX1	
34.C		NUMMER	(EX1	
34.D	ZERO	BLANK	(EX15,19	

FIELD 4 - LINE SPACING

- | | |
|-----|------------------|
| 35. | BLANK (EX8,9 |
| 36. | 0 (EX14 |
| 37. | SINGLE-1 (EX1,18 |

APPENDIX C
DOCUMENTATION EXAMPLE

DOCUMENTATION EXAMPLE

```

C SUBROUTINE PRINT      (STRNG,PMPTR)
C PROGRAM NAME: PRINT  (STRNG,PMPTR)
C FUNCTION: THIS ROUTINE CHECKS ALL INPUT DATA AND PRINTS ALL ERRORS. IT
C           THEN TRANSFERS CONTROL TO (PRMON) TO DO PRINTING.
C AUTHOR: UNKNOWN AUGUST 1976
C MODIFICATIONS: NEELY SEPT 76
C LANGUAGE: FORTRAN
C CALLING SEQUENCE : STRNG=THE STRING OF CHARACTERS REPRESENTING THE
C                   USERS INPUT COMMAND.
C                   PMPTR=A Z-DIMENSIONAL ARRAY CONTAINING THE
C                   DESCRIPTION AND LOCATION OF ALL PARAMETER
C                   FIELDS WITHIN THE USERS COMMAND-STRNG.
C
C ROUTINES CALLED:
C   INTER-FUNCTION TO FORM AN INTEGER.
C   UNPAK
C   LNLIM =          OBTAINS FIRST AND LAST LINE NUMBERS
C   PRMON          PRINTS TEXT
C   TBLCS-RECORD LOCATE IN TABLE
C   ICOPY-
C
C TASKS OR MODULES:
C
C VARIABLES:
C   STRNG(1)= USERS COMMAND
C   PMPTR(100,3)= 99 PARAMETER FIELDS,3 COLUMNS-1ST CTR.LENGTH,TYPE
C   NPRI=NO OF THE DEVICE TO PLACE THE PRINTED DOCUMENT ON
C   FLGOP(6)= DATA COLUMNS TO BE PRINTED: 1- X-50      .2- 38-L
C           3-32-F          4-46-T          5-27-A 6-42-P
C   IMERR= 1-GENERAL ERROR MESSAGE PRINTED. 0-NOT PRINTED FOR ONE FIELD.
C   IERR = 0-NO ERRORS. 1- ERRORS. STOP AFTER PARAMETER ANALYSIS
C   MODE = THE FIELD NO. BEING PROCESSED. NUMBERED IN ORDER 1 TO 8
C   IMYPH= 0=HYPHEN NOT PREVIOUS DELIM. 1= HYPHEN IS PREV DELIM.
C   IELN= ENDING LINE NUMBER TO PRINT.USER INPUT
C   ISLN= STARTING LINE NO   TO PRINT.USER INPUT
C   ISEG=      TEXT SEGMENT 1 THRU3
C   EDITF= TABLE BEING EDITED
C   ISPC= LINE SPACING01=SINGLE 2= DOUBLE
C   ICNT = NO. OF PARMS PLUS ONE -ROW LOCATION OF LAST PARAMETER
C   ITYPE= PARM. TYPE
C   IPTR = 1ST CTR OF PARM. IN STRNG ARRAY
C   LENG = PARM. CTR LENGTH
C   DEBUG= 2 OR GTR FOR DEBUG TESTING
C   FLGNM= 1-50=X .2-38=L .3-32=F .4-46=T .5-27= A  ALL AVAILABLE FIELDS
C           6-31=E
C   LETTER = THE COLUMN PRINT OPTION USER INPUT VALUE
C   JSLN= STARTING LINE NO. IN TEXT TABLE.
C   JELN= ENDING LINE NO. IN TEXT TABLE.
C   ISET= 1=PRINT .-1= NO PRINT
C   I = THE ROW IN PMPTR BEING ANALIZED.
C   J      COUNTER
C   NPPIN= NO P/P INITIALIZATION
C   NOD=NUMBER OF OUTPUT DEVICE 0 OR 1
C   NPRF= ACTUAL PRINT DEVICE

```



```

EQUIVALENCE (ISPRN(1),ISQTY(1,1)),(ISTTF(1),ISQTY(1,12))
EQUIVALENCE (ISBAK(1),ISQTY(1,13))
EQUIVALENCE (EDTBL,IJBLNK),(MCHBL,IJBLNK)

```

```

DOCUMENT EDIT COMMON
INTEGER EDITF,EDS,EBDS
LOGICAL EREAD
COMMON /EDITC/ EDITF(4),EDS,EBDS,ERead,IEDIT,MGO,ICYC,IDAC,INC,

```

```

1 ITQTY(5,15),IERR
DIMENSION ITAUG(5),ITAU0(5),ITAUT(5),ITDRT(5),ITERT(5),
1 ITERC(5),ITERL(5),ITERS(5),ITFLG(5),ITIX (5),
2 ITLC (5),ITPUL(5),ITABC(5),ITEXT(5),ITBAK(5)
EQUIVALENCE (ITAUG(1),ITQTY(1, 1)),(ITAU0(1),ITQTY(1, 2))
EQUIVALENCE (ITAUT(1),ITQTY(1, 3)),(ITDRT(1),ITQTY(1, 4))
EQUIVALENCE (ITERT(1),ITQTY(1, 5)),(ITERC(1),ITQTY(1, 6))
EQUIVALENCE (ITERL(1),ITQTY(1, 7)),(ITERS(1),ITQTY(1, 8))
EQUIVALENCE (ITFLG(1),ITQTY(1, 9)),(ITIX (1),ITQTY(1,10))
EQUIVALENCE (ITLC (1),ITQTY(1,11)),(ITPUL(1),ITQTY(1,12))
EQUIVALENCE (ITABC(1),ITQTY(1,13)),(ITEXT(1),ITQTY(1,14))
EQUIVALENCE (ITBAK(1),ITQTY(1,15))

```

```

I/O UNIT NUMBERS
COMMON /IOC/ INP1,INP2,NLST,NPRO,NECO,NPRI,MESG ,NPRT,NPRP

```

```

READ AND PARSE COMMAND ROUTINES COMMON AREA.
INTEGER EBUFF,PMPTR,STRNG
LOGICAL EOFIL,NXTRD,CMDOK
COMMON /PARSC/EBUFF(400),STRNG(400),PMPTR(100,3),NXTLN(80),
1 NCHAR,ISUB,EOFIL,NXTRD,CMDOK

```

```

AUXILIARY MACHINE SPECIFIC CONSTANTS
COMMON /MC/ MAXCH,IWD1,NCUM,IWD2,NCUM2,NCU2,IWD3

```

```

TABLE HANDLING ROUTINES COMMON AREA
COMMON /TABLC/ NLOC,ISHDR(11),IRHDR(243)

```

```

PRINT ROUTINES COMMON AREA
COMMON /PRINT/ NITC,LCO,NOD,NPPNI
1 ,DFDRC(48),IRSLN,IRELN,ISPAC,ISEG(3)

```

```

INTEGER EDITF .DFDRC
DIMENSION IHDR(243)
INTEGER FLGNM(6),FLGOP(6)
1,STRNG(1),PMPTR(100,3)
DATA FLGNM/50,38,32,46,27,42/

```

C-----PUT PRINTED DOCUMENT OUT ON FT08F001 OR FT07 HIGH SPEED PTR
 C *** THE FOLLOWING 10 CARDS ARE FOR DEBUG ONLY

IF (DEBUG .NE. 2) GO TO 9994
 I2= PMPTR(1,3) +1
 I1= (PMPTR(12,1)+ PMPTR(12,2) / NCU + 1
 WRITE (MSG.9996) (STRNG(I),I=1,I1)

9996 FORMAT(1H .12Z10)

WRITE(MSG.9995)

1 ((PMPTR(I,J),J=1,3),I=1,I2)

9995 FORMAT (1H .30I4)

9994 CONTINUE

C- INITIAL ALL VARIABLES TO THE DEFAULT OPTIONS.

LNP=0

IS=0

NITC = 0

LCO = 0

NOD = 0

NPRI=NPRT

NPPIN= 0

DO 4 I= 1,3

4 ISEG(I)= EDTBL

C- MARK COLUMNS TO BE PRINTED AS ZERO FOR NO DECISION MADE.

DO 3 I=1,6

FLGOP(I)=0

3 CONTINUE

C SET TO -1 TO INDICATE NO USER ENTRY

IELN= -1

ISLN= -1

IMERR=0

IERR=0

MODE=1

IMYPH=0

IFORM=0

ISPAC=1

C- CHECK FOR NO PARAMETERS IN COMMAND. GTR THAN 0, LESS THAN 35

ICNT = PMPTR(1,3)+1

IF (ICNT .LE. 1) GO TO 991

IF (ICNT .GT. 37) GO TO 992

C

C

C

C

C-

C

C

C

C

PROCESS EACH PARAMETER IN SEQUENCE. CHECK FOR ALL ERRORS.

5 DO 500 I=2,ICNT

ITYPE=PMPTR(1,3)

IPTR =PMPTR(1,1)

LENG =PMPTR(1,2)

IF (DEBUG .EQ. 2) WRITE(6,9000) I,ITYPE,IPTR,LENG

9000 FORMAT(20H I,ITYPE,IPTR,LENG =, 4I5)

C

C

*Note
 use of
 Blank Comment
 Cards*

```

C      GO TO (10,20,900,40,50,60,900,900),ITYPE ← Transfer on
C                                     Field Type 1 thru 8
C
C-     PROCESS NUMBERS.
C      GO TO (150,900,100,180,130,140,160,170),MODE ← Transfer on
C                                     Field number in
C                                     actual command/
150    IF (LENG .LE. 8 ) GO TO 959
        WRITE(MESG,9959)
9959   FORMAT(5X,45H*** DOCUMENT FORMAT GREATER THAN 8 CHARACTERS )
        IERR=1
        GO TO 500
959    CONTINUE
        IFORM = INTER(STRNG,IPTR,LENG)
C LOAD THE DOCUMENT FORMAT TABLE INTO THE DATA HANDLER FILE LOCATION
600    DO 601 J=1,4
601    FILNM(J) = ISDOC(J)
        ISHID  = ISDOC(5)
        CALL TBLCS( ISHID,IFORM,IHDR,ITEM,LOC)
        IF(ITEM .GT. 0 ) GO TO 602
940    WRITE(MESG,9940)
9940   FORMAT(5X,35H*** DOCUMENT FORMAT DOES NOT EXIST.)
        IERR=1
        GO TO 500
602    IRDFI=IHDR(LOC)
C READ THE DATA RECORD INTO PRINT COMMON FOR USE IN PRMON ROUTINE
        I1=NCU* 48
        CALL DKGET(IRDFI,DFDRC,1,I1,NC )
        GO TO 500
C
C
C-     LINE NUMBERS
100    IF(ILNP .EQ. 1)GO TO 9111
        J=1+1
        IF ((PMPTR(J,3) .EQ. 5) .OR. (IMYPH .EQ. 1))GO TO 111
        GO TO 911
111    IF (IMYPH) 110,111,120
C- HAVE A STARTING LINE NO.,PROCESS THE NO.
110    IF ( LENG .LE. 8 ) GO TO 957
112    WRITE( MESSG,9955)
9955   FORMAT( 5X,40H*** LINE NUMBER MUST BE 1 TO 8 INTEGERS )
        IERR=1
        GO TO 500
957    ISLN = INTER(STRNG,IPTR,LENG)
        GO TO 500
C- IF A HYPHEN ENCOUNTERED PREVIOUSLY WE HAVE ENDING LINE NO. (AREA FIELD).
120    IF(LENG .LE. 8)GO TO 959
        WRITE(MESG, 9955)
        IERR=1
        GO TO 500
959    IELN = INTER(STRNG,IPTR,LENG)

```

0.1701.MODE ← Transfer on
Field number in
actual comments/

```

      IF(I .LT. ICNT)GO TO 500
C
C
C
C
C- CHECK TO SEE IF THE USER HAS GIVEN BOTH A START AND END LINE NUMBER.
C- EITHER START OR END LINE HAS BEEN SPECIFIED. DETERMINE WHICH AND
C- ISET DEFAULT FOR THE OTHER.
510 JSLN=1
    JELN=99999999
      IF( ISEG(1) .EQ. ISEG(2) ) WRITE(MESG,5001 )
      IF( ISEG(1) .EQ. ISEG(3) ) WRITE(MESG,5001 )
      IF( ISEG(2) .EQ. ISEG(3) ) WRITE(MESG,5001 )
5001 FORMAT ( 5X,32H*** DUPLICATE TEXT SEGMENT 10 )
      IF (ISLN .LE. 0) ISLN = JSLN
      IF (IELN .LE. 0) IELN = JELN
      IF( ISLN .LT. JSLN ) ISLN= JSLN
      IF( IELN .GT. JELN ) IELN= JELN
C- CHECK TO BE SURE THAT BOTH STARTING AND ENDING LINES EXIST.
550 DO 557 J=1,4
      FILNM(J)= ITEXT(J)
557 CONTINUE
      ISHID = ITEXT(5)
      CALL TBLCS (ISHID, ISLN, IHDR, ITEM, LOC)
      IF(ITEM .GT. 0) GO TO 556
930 WRITE(MESG,9930)
9930 FORMAT(5X,40H*** STARTING LINE NUMBER DOES NOT EXIST.)
      WRITE(MESG,552) ISLN
552 FORMAT(5X,16H*** LINE NUMBER , 18 )
      ISLN = IHDR(-ITEM)
      WRITE(MESG,553) ISLN
553 FORMAT(5X,46H*** DOES NOT EXIST AND HAS BEEN REPLACED WITH ,18 )
556 IRSLN=IHDR(LOC)
551 CALL TBLCS (ISHID, IELN , IHDR, ITEM, LOC)
      IF(ITEM .GT. 0 ) GO TO 554
931 WRITE (MESG,9931)
9931 FORMAT(5X,39H*** ENDING LINE NUMBER DOES NOT EXIST.)
      WRITE(MESG,552) IELN
      IELN = IHDR(-ITEM)
      WRITE(MESG,553) IELN
554 CONTINUE
      IRELN=IHDR(LOC)
C CHECK FOR SLN GTR THAN ELN
      IF(ISLN .LE. IELN ) GO TO 410
      WRITE(MESG,555)
555 FORMAT( 5X,49H*** STARTING LINE NO. GREATER THAN ENDING NUMBER. )
      IERR=1
      GO TO 410
9111 J=I+1
      IF(PMPTR(J,3) .EQ. 5) GO TO 9112
      IF(IHYPH .EQ. 1) GO TO 9112
      GO TO 911

```

C- SPACING FIELD IS PRESENT.

180 ISPAC = INTER(STRNG,IPTR,LENG)

800 IF ((ISPAC .GE. 1) .AND. (ISPAC .LE. 2)) GO TO 500

WRITE(MESG,9950)

ISPAC=2

9950 FORMAT(5X,50H*** (LINE SPACING INCORRECT. DOUBLE SPACING ASSUMED)

GO TO 500

9112 WRITE(MESG,9113)

9113 FORMAT(5X,38H*** ONLY ONE LINE NUMBER PAIR ALLOWED)

IERR=1

GO TO 500

C
C
C
C

130 NITC = INTER(STRNG,IPTR,LENG)

IF ((NITC .GE. 0) .AND. (NITC .LE. 1)) GO TO 500

WRITE(MESG,9951)

9951 FORMAT(5X,42H*** INDEX CREATION INCORRECT. NONE CREATED)

NITC=1

GO TO 500

C
C
C
C

140 LCO = INTER(STRNG,IPTR,LENG)

IF ((LCO .GE. 0) .AND. (LCO .LE. 1)) GO TO 500

WRITE(MESG,9952)

9952 FORMAT(5X,46H*** LOGIC OVERRIDE INCORRECT. OVERRIDE ASSUMED)

LCO=1

GO TO 500

C
C
C
C

160 NOD = INTER(STRNG,IPTR,LENG)

IF(NOD .EQ. 0) GO TO 500

IF(NOD .EQ. 1) GO TO 161

WRITE(MESG,9953)

9953 FORMAT(5X,46H*** OUTPUT DEVICE INCORRECT. PRINTER ASSUMED)

161 NPRI= NPRP

GO TO 500

C
C
C

170 NPPIN= INTER(STRNG,IPTR,LENG)

IF ((NPPIN .GE. 0) .AND. (NPPIN .LE. 1)) GO TO 500

WRITE(MESG,9954)

9954 FORMAT(5X,50H*** P/P INITIALIZATION INCORRECT. NONE PERFORMED)

NPPIN=1

GO TO 500

C
C
C
C

```

C
C- ALPHA
20 GO TO (900,250,210,900,900,900,900,900),MODE

C
C
C
C
C- OPTIONS FIELD ENCOUNTERED.
250 CALL UNPAK(STRNG,IPTR,1,LETR)
      IF (LENG .LE. 1) GO TO 961
      WRITE(MESG,995)
9961  FORMAT (5X,43H*** COLUMN CODE GREATER THAN ONE CHARACTER. /
1      5X,34H*** ONLY THE FIRST CHARACTER USED. )
961  CONTINUE
      DO 260 J=1,6
      IF (LETR-FLGNM(J)) 260,270,260
260  CONTINUE
      GO TO 900
C- RECORD OPTION ENTERED.
270 FLGOP(J)=1
      IF (IHYPH) 500,500,275
275 FLGOP(J)=-1
      IHYPH=0
      GO TO 500

C
C

C- AREA FIELD - TEXT SEGMENT IS PRESENT.
210 J=1+1
      IF (PMPTR(J,3) .EQ. 5) GO TO 112
      IF (IHYPH .EQ. 1) GO TO 112
      IF (LENG-4 ) 220,220,911
911  WRITE (MESG,9911)
9911  FORMAT ( 5X,47H*** TEXT SEGMENT ID MUST BE 1-4 A/N CHARACTERS. )
      IERR=1
      GO TO 500
220  IS= IS+1
      IF (IS .LE. 3 ) GO TO 221
      WRITE(MESG,9912)
9912  FORMAT(5X,41H*** ONLY THREE TEXT SEGMENT IDS PERMITTED. )
      IERR= 1
      GO TO 500
221  CALL ICOPY (STRNG,IPTR,LENG,ISEG(IS),1)
      GO TO 500

C
C
C
C
C- SEMICOLON ENCOUNTERED.
40  GO TO (410,410,510,410,410,410,920),MODE
C- PROCESSING OF THIS PARAMETER COMPLETE. MOVE TO NEXT PARM. RESET KEYS
410 MODE=MODE+1
      IMERR=0
      IHYPH=0
      GO TO 500
C

```


NOTE USE OF STANDARD FORMAT ERRORS FOR

1. FIELD

Then

1/1 TYPE error.

C
C
C
C
C

```

PRINT FORMAT ERROR ONCE PER FIELD
900 IF (IMERR) 901,901,960
901 IMERR=1
    IERR=1
    WRITE (MSG,9901)
9901 FORMAT(5X,20H*** FORMAT ERROR IN )
    GO TO ( 903,904, 902, 905, 906, 907, 908,909),MODE
902 WRITE(MSG,9902)
9902 FORMAT(25X,80H AREA FIELD. MUST BE SLN-ELN(INTEGERS) AND/OR 3 TEX
1T SEGMENTS (ALPHANUMERIC).
    GO TO 960
903 WRITE(MSG,9903)
9903 FORMAT(25X,60H DOCUMENTH FORMAT FIELD. MUST BE A 1 TO 8 CHARACTER
1 INTEGER.
    GO TO 960
904 WRITE(MSG,9904)
9904 FORMAT(25X, 52H COLUMNS TO BE PRINTED. MUST BE F,L,T,A,X,P,OR - .
1
    GO TO 960
905 WRITE(MSG,9905)
9905 FORMAT(25X,40H LINE SPACING FIELD. MUST BE 1 OR 2.
    GO TO 960
906 WRITE(MSG,9906)
9906 FORMAT(25X,55H NO INDEX AND TABLE CREATION FIELD MUST BE 0 OR 1.
1
    GO TO 960
907 WRITE(MSG,9907)
9907 FORMAT(25X,52H LOGIC CONDITION OVERRIDE FIELD. MUST BE 0 OR 1.
1
    GO TO 960
908 WRITE(MSG,9908)
9908 FORMAT(25X,48H OUTPUT DEVICE NUMBER FIELD. MUST BE 0 OR 1.
    GO TO 960
909 WRITE(MSG,9909)
9909 FORMAT(25X,67H NO PAGE/PARAGRAPH NUMBER INITIALIZATION FIELD. MUS
1T BE 0 OR 1.
    GO TO 960

```

FIELDS

C
C

C
C
C
C
960 GO TO (9001,9002,9003,9004,9005,9006,9007,9008),ITYPE

9001 WRITE(MESG,99001)
99001 FORMAT (5X,22H*** NUMERIC NOT VALID.)
GO TO 500
9002 WRITE(MESG,99002)
99002 FORMAT (5X,27H*** ALPHANUMERIC NOT VALID.)
GO TO 500
9003 WRITE(MESG,99003)
99003 FORMAT (5X,37H*** ALPHANUMERIC DELIMITED NOT VALID.)
GO TO 500
9004 WRITE(MESG,99004)
99004 FORMAT (5X,25H*** SEMI-COLON NOT VALID.)
GO TO 500
9005 WRITE(MESG,99005)
99005 FORMAT (5X,20H*** HYPHEN NOT VALID)
GO TO 500
9006 WRITE(MESG,99006)
99006 FORMAT (5X,20H*** COMMA NOT VALID)
GO TO 500
9007 WRITE(MESG,99007)
99007 FORMAT (5X,28H*** DOUBLE ASTERIX NOT VALID)
GO TO 500
9008 WRITE(MESG,99008)
99008 FORMAT (5X,28H*** PLUSS SIGN NOT VALID.
GO TO 500

C
C
C
C
C
C
*Note transfer to end of Do Loop
for para meta processing.*

920 WRITE(MESG,9920)
9920 FORMAT(5X,26H*** TOO MANY FIELDS GIVEN.)
IF(IFORM .EQ. 0) GO TO 991
IERR=1
GO TO 990
991 WRITE (MESG,9991)
9991 FORMAT(5X, 50H*** NO FIELDS FOUND. DOCUMENT FORMAT ID REQUIRED.)
GO TO 990
992 WRITE (MESG,9992)
9992 FORMAT(5X, 50H*** TOO MANY FIELDS FOUND. PROCESSING CONTINUES.)
GO TO 5
990 WRITE(MESG,9990)
9990 FORMAT(5X,46H*** COMMAND IGNORED. BUT PROCESSING CONTINUES.)
GOTO 1000
END

APPENDIX D
PROGRAMMING NOTES & PROCEDURES

D.1 WORK PROGRESSION

1. Users Manual Description

- a. Review Functional Criteria.**
- b. Discuss Functional Criteria with Supervisor.**
- c. Check all other commands for possible interactions.**
- d. Review table structure and command with supervisor.**
- e. Review code for additional knowledge.**
- f. Design and write the final description**
- g. Supervisor review of description.**
- h. Typing.**
- i. TIB review.**
- g. Drafting.**

2. Complete Coding

- a. Review existing code.**
- b. Document code to standards.**
- c. Design coding changes to complete the command. Update table descriptions.**
- d. Code the changes.**
- e. Design a 100% complete test deck that will operate as a stand-alone run.**
- f. Hand debug the test run thoroughly.**
- g. Review the test run and hand calculations with your supervisor.**
- h. Run on the computer.**

3. Reporting Standards

- a. Each team member will keep a daily log book. This log will detail the time spent on the project. The log book will be daily dated and entries made recording start and completion times. Computer run costs will be entered directly into the log. Log books will be checked by the supervisor no later than COB each Friday.
- b. All computer trailer sheets must be saved and numbered by the programmer. Supervisor's permission should be granted before submission of a computer run.
- c. All decisions must be recorded in writing within the log books of all parties involved in the system.
- d. All considerations reviewed must be entered into the log book for future reference. This log will become a detailed daily diary of your work.

D.2 DOCUMENTATION STANDARDS

These standards apply to all programming performed.

All program documentation should be contained in comments in the programs. The comments which will be included in the documentation should be in the format described below.

The resulting listed documentation will be in the following format:

Program Name: as in the first line of the function or subroutine, with arguments.

Function: brief description of the function of the program.

Author: author's name, date of first writing.

Modifications: Each mod should be numbered. Author, date and brief description of the reason for the change(s). All cards associated with each mod should be so numbered in c.c. 79-80.

Language: programming language.

Calling Sequence: description of arguments and function result.

Routines Called: list of subroutines and functions called by the program.

Tasks or Modules: tasks or modules containing this routine.

Variables: a description of all variables used in the program. **COMMON** variables will generally be described in a "main" program, referenced in this section. Check to insure that the command description is correct. The structure of arrays and meaning of variables should be fully described.

Any figures or drawings which would aid in the description of an algorithm should be named with the program name and a figure number, such as ABC.1 for program ABC. These are referenced from the comments in the subroutine and included in the documentation book.

D.3 TECHNICAL DOCUMENTATION GUIDELINES

A. Contents

1. All textual documentation will be contained in comments in the programs. It will be possible to obtain a listing of the appropriate comments separately.

B. FORM

1. Placement of comments

All comments start in cc7.

2. Separating sections

Blank comment lines and lines of asterisks may be used to separate sections of code.

- a. DO loops - A blank comment line before the DO statement and a blank comment line after the CONTINUE statement is used to 'bracket' major DO loops. Comments pertaining to the loop are included within the 'bracket'.
- b. Major headings are 'bracketed' by a blank comment line before and after the headings.
- c. A line of asterisks may be used to signify the end of a major program section.
- d. Headings may be caused to stand out further by inserting a space after each character and by using a line of asterisks right, left, above, or below the heading.

Note: All the above devices are used to enhance clarity for anyone reading the program. Uniform use of format within each program system will amplify the effectiveness of same.

3. Statement numbers.

- a. A FORMAT statement follows the first WRITE or READ statement using it.
- b. Error messages start at statements numbered 99, or 999 or 9999.
- c. RETURN is statement number 100, 1000, or 10000.
- d. Statement numbers can be sequenced in two ways: 1) Increasing value sequence, 2) Collating sequence. If increasing sequence is used, the statement numbers should be right justified. Some people reserve 50's for ending DO loops, and derive the number of a FORMAT statement from the associated READ or WRITE statement by adding 'one'. Read or Write statements are frequently numbered in hundreds:

```
1800 WRITE(3,1801)x,y,z  
1801 FORMAT(3F8.4)
```

In a long program with many READs and WRITEs 1802,3, etc. would be preferable to using 1821, 1822, etc., (If one is in danger of using up all available hundreds, that is.) If Collating sequence is used, the statement numbers should be left justified. 50's might be reserved for ending DO loops. FORMAT numbers may be derived from associated READ or WRITE statement numbers of multiplying with 10.

4. Debugging statements.

For automatic removal of tracers, traps, etc., an asterisk in cc2 followed by the number of lines to be removed from the final program might be used.

C. FORTRAN Standards

The American National Standard FORTRAN (ANSI x3.9-1966) standards are followed throughout.

D. Additional Guidelines

1. FUNCTIONS may not define or redefine any arguments or variables in COMMON.
2. DIMENSION and TYPE statements come before COMMON, which come before EQUIVALENCE statements.
3. No recursive procedures will be written.
4. All DO loops end on unique CONTINUE statements.

5. I/O device unit members should be variables in COMMON.
6. All specifications statements appear before all executable statements.
7. Each subroutine or function subprogram has a single entry point, at the beginning, and a single RETURN statement as the last executable statement.
8. Avoid word-size-, character-size-, or absolute-address-dependent operations. Similarly, do not assume the format of negative integers (1's or 2's complement) or the internal representation of character or numeric data.
9. Profuse comments should be used.

E. Programming "Tricks"

1. Keep all statement numbers in order. This can be done by considering all statement numbers having fewer than 5 digits as being filled with trailing zeroes. This will guarantee that statement numbers are not duplicated and that a programmer does not have to search throughout a program to find if a particular statement number has been used.
2. Filler variables in COMMON should be named according to their word location in COMMON. This allows for easy computation of the size of filler arrays and insertion of new variables in COMMON.
3. Subroutines should be kept to less than 100 executable statements if possible. Individual subroutines should each serve only a single function.
4. Error and warning messages should state in the user's terms what the problem is and what action is taken by the program.
5. Intermediate output for additional data should be available by changing an input variable. This variable will set dump switches which are either stored or are accessible through COMMON.

F. Program Conventions

1. If a transfer within an IF statement is impossible under normal data transfer to an error message that states

"The computer program has transferred to an impossible location in routine number _____"

2. An error message, written in English phrases, should give the user the same information that the computer has. The object is to provide maximum service to the user and to eliminate all confusion.
3. During debugging list the subroutine parameters (1) before the call, (2) immediately inside the subroutine, (3) before returning, (4) immediately after the return from the call in calling program. Place IF statements around the debugs and mark in Col 73-80 as GUBED-C.
4. Do not remove existing coding cards from the deck. Mark a C in column one and retain.
5. Mark your card entries with the number of the modification in card columns 77-80.
6. Standard error message prefix will be (5x,--Hxxx.
7. All common blocks will be set up as follows:
 - a. Date Modified
 - b. Common Statements
 - c. Dimension Statements
 - d. Equivalence Statements
 - e. Integer Statements
 - f. Comment cards defining each variable in order
8. Array ITNM (3,8) has been added to /SYSTM/ common for writing error messages. The columns correspond to the parameter field type values in PMPRTR (,3).
9. All system routines should set FILNM to SYSFL before returning. All edit routines should set FILNM to EDITF before returning.

APPENDIX E

PROGRAM COMMON AREAS

JULY 22, 1977
 INTEGER FILNM(4)
 INTEGER RVRSE,TRU,FALS,ZEROS,ONES,FOLID,PREID,FSTAT,P
 COMMON P(200)
 EQUIVALENCE (P(5),NBW),(P(6),NCW),(P(7),NAW),(P(8),NBC),(P(9),NCU)
 EQUIVALENCE (P(10),NBU),(P(11),NAU),(P(12),NBU2),(P(13),NC2U)
 EQUIVALENCE (P(14),RVRSE),(P(15),NWPSU),(P(16),LFIW),(P(17),LFIWU)
 EQUIVALENCE (P(24),ICRD),(P(25),IPRT),(P(26),IPCH),(P(27),ITAP)
 EQUIVALENCE (P(38),TRU),(P(39),FALS),(P(40),ZERCS),(P(41),ONES)
 EQUIVALENCE (P(187),MAXGP),(P(188),NDS),(P(189),FILNM(1))
 EQUIVALENCE (P(193),FOLID),(P(194),PREID),(P(195),LRLEN)
 EQUIVALENCE (P(196),FSTAT),(P(197),NDA),(P(198),IDHTR)
 EQUIVALENCE (P(199),IDHER),(P(200),IDKER)

NBW IS NUMBER OF BITS IN A MACHINE WORD (A NON-FORTRAN CONCEPT)
 A MACHINE WORD IS USUALLY EQUAL TO A STANDARD-UNIT.
 A STANDARD UNIT IS THE AMOUNT OF STORAGE NEEDED FOR AN INTEGER
 NCW IS NUMBER OF CHARACTERS IN A MACHINE WORD
 NAW IS NUMBER OF MACHINE ADDRESSES THAT WILL FIT IN A MACHINE WORD
 THIS IS PROBABLY NEVER USED
 NBC IS NUMBER OF BITS PER CHARACTER.
 NCU IS NUMBER OF CHARACTERS PER STANDARD-UNIT (ATLEAST 4 FOR EDITSPEC)
 NBU IS NUMBER OF BITS PER SU (STANDARD-UNIT)
 NAU IS NUMBER OF ADDRESSES PER SU. PROBABLY UNUSED.
 NBU2 IS NBU/2
 NC2U IS NCU*2
 RVRSE DENOTES ORDER OF CHARACTERS IN AN INTEGER VARIABLE.
 0 = LEFT TO RIGHT (A1 REFERS TO HIGH-ORDER OR LEFTMOST BITS)
 1 = RIGHT TO LEFT (A1 REFERS TO LOW-ORDER OR RIGHTMOST BITS)
 NWPSU IS NUMBER OF WORDS PER SU (USUALLY 1)
 LFIW IS NUMBER OF BITS IN A FORTRAN INTEGER (SAME AS NBU)
 LFIWU IS NUMBER OF USABLE BITS IN A FORTRAN INTEGER.
 MOST OF THE ABOVE VARIABLES EXIST FOR HISTORICAL REASONS
 ONLY. PROGRAMMERS SHOULD NOT NEED ANYTHING BUT NCU OUT OF THESE.
 ICRD IS CARD-READER LUN (PROBABLY NOT USED)
 IPRT IS PRINTER LUN (PROBABLY NOT USED)
 IPCH IS PUNCH LUN (PROBABLY NOT USED)
 ITAP IS MAGNETIC TAPE LUN (PROBABLY NOT USED)
 THE ABOVE LOGICAL-INIT-NUMBERS EXIST FOR HISTORICAL REASONS
 FOR THE LUN VARIABLES USED IN EDITSPEC. SEE COMMON /IOC/
 TRU IS AN INTEGER REPRESENTATION OF LOGICAL .TRUE.
 FALS IS AN INTEGER REPRESENTATION OF LOGICAL .FALSE.
 ZEROS IS AN INTEGER REPRESENTATION OF BINARY ZERO (ALL BITS OFF)
 ONES IS AN INTEGER REPRESENTATION OF BINARY ONES (ALL BITS ON).
 THE ABOVE WERE USED IN THE OLD SYSTEM TO PERFORM SOME
 TRICKY TYPE CONVERSION. THEY CONTINUE TO EXIST SOLELY
 FOR HISTORICAL REASONS.
 MAXGP STANDS FOR MAXIMUM NUMBER OF CHARACTERS PER GET OR PUT. IT
 WAS USED IN THE EARLIER DATA-HANDLER FOR I/O. IT IS
 CURRENTLY SUPERFLUOUS, AND HAS A VALUE OF 32768 (2**15)
 THE FOLLOWING ARE USED TO COMMUNICATE BETWEEN EDITSPEC
 AND THE DATA-HANDLER.

NDS - DATASET NUMBER (INDEX INTO DATASET TABLE)
 FILNM - 4 INTEGER ARRAY TO STORE FILE-NAME
 FOLID - RECORD ID OF SUBSEQUENT OR FOLLOWING RECORD
 PREID - RECORD ID OF PREVIOUS RECORD
 LRLEN - LOGICAL RECORD LENGTH
 FSTAT - FILE STATUS
 NDA - MAXIMUM NUMBER OF DIRECT-ACCESS DATASETS

NDA - MAXIMUM NUMBER OF DIRECT-ACCESS DATASETS
 IDHTR - DATA-HANDLER TRACE FLAG
 IDHER - DATA-HANDLER ERROR FLAG
 IDKER - DATA-HANDLER ERROR CODE

DECEMBER 1, 1977

ACCESS SYNTAX COMMON.

INTEGER DNAMS,ACODE,ANUMS,UIDS,UALLS,ALLSW,RCIDID,RCATID,RCUID
 COMMON /ACSNC/ DNAMS(3,20),ACODE(20),ANUMS(3,20),UIDS(3,20),
 UALLS(3,20),NNAMS,NNUMS,NIDS,NALLS,ALLSW,RCIDID(20),RCATID(20),
 2RCUID(20)

DNAMS(3,20) IS USED TO STORE UPTO 20 DOCUMENT NAMES.
 ACODE(20) IS ACCESS CODE FOR DOCUMENT NAME LIST.
 ANUMS(3,20) IS LIST OF ACCOUNT NUMBERS.
 UIDS(3,20) IS LIST OF USER-IDS.
 UALLS(3,20) IS LIST OF ALL-MASKS.
 NNAMS IS NUMBER OF DOCUMENT NAMES IN LIST (<=20)
 NNUMS IS NUMBER OF ACCOUNT NUMBERS.
 NIDS IS NUMBER OF USER-IDS.
 NALLS IS NUMBER OF ALL MASKS.
 ALLSW IS 1 IF 'ALL' IS SPECIFIED.
 RCIDID(20) HOLDS RECORD-ID OF FIRST DATA RECORD ASSOCIATED
 WITH EACH DOCUMENT NAME IN DNAM(3,20) (IN DIR-TABLE).
 RCATID(20) IS DATA RECORD ASSOCIATED WITH ANUMS(3,20) IN ACS-TBL
 RCUID(20) IS DATA RECORD ASSOCIATED WITH UIDS(3,20) IN USR-TBL.

C AUGUST 30, 1979

COMMON /ACTAC/ IACTA

C IACTA =0 OFF,=1 WHEN A *CT* FOUND INSIDE A *TB*

C MARCH 8, 1979

C BACKUP COMMON BLOCK

C INTEGER STAPE

COMMON /BACKC/ ISBUP(5), ISBUC(5), ISBUT(5), MULFIL,
 1 NUNITS, STAPE.

1 IRSTR, IDOCDS

C ISBUP IS TABLE OF ENTITIES EDITED SINCE LAST BACKUP

C ISBUC IS BACKUP DIRECTORY BY DOCUMENT/DATASET/SYSTEM-TABLE NAME

C ISBUT IS BACKUP DIRECTORY BY TAPE NAME

C MULFIL IS TYPE OF BACKUP TAPE USED

C =0 IF THERE IS ONLY ONE FILE ON THE TAPE

C =1 IF THERE MAY BE MORE THAN ONE FILE ON THE TAPE

C NUNITS IS TOTAL NUMBER OF THINGS THAT CAN BE BACKED UP

C =MAXIMUM NUMBER OF FILES ON THE TAPE IF MULFIL=1

C =MAXIMUM NUMBER OF ENTITIES THAT MAY BE BACKED UP IF MULFIL=0.

C STAPE IS UNIT NUMBER OF THE "SPECIAL TAPE" TO HOLD ISBUP, ISBUC, AND ISBUT.

C IRSTR - RESTORE SWITCH

C APRIL 12, 1979

C COMMON HOLDS NDS AND FILNM FOR RESTORE READ ROUTINE

C INTEGER RENOS, REFIL

C 1. RRECIO

C 2. INISAV

C RRECIO RECORD ID TO READ NEXT

C INISAV SAVE AREA FOR INP1

C COMMON /BCKRSC/ RENOS, REFIL(4)

C 1. RRECIO

CONVERSION-SYSTEM COMMON. DOCUMENT PARTITION

FEBRUARY 6, 1979

COMMON /CNVDOC/ DOCNM, DSNAM, NEWSN, OPTION, CMDOPT,
PROCES, CNVERR, DSCAT, DSKEY, DSRID,
DSBUF, SEQNO

INTEGER	DOCNM(3)	(PACKED) EDIT SPEC NAME OF DOC TO BE CONVERTED
INTEGER	DSNAM(2)	DATASET-NAME ON WHICH THE DOCUMENT RESIDES.
INTEGER	NEWSN(2)	TARGET DATASET-NAME FOR THIS DOCUMENT. IF USER CHOOSES TO MODIFY THE DATASET THAT THE DOCUMENT RESIDES ON AT TAPE GENERATION TIME
INTEGER	OPTION	SWITCH THAT CONTAINS THE OPTION VALUE THE USER SELECTED WHEN THE ".DSCN" COMMAND WAS ISSUED.
INTEGER	CMDOPT	OPTION SELECTED ON THE SPECIAL COMMAND RECORDS THAT FOLLOW THE ".DSCN" COMMAND WITH OPT SET TO 2.
LOGICAL	PROCES	LOGICAL SWITCH TO INDICATE WHETHER THE "LOOKUP" PHASE HAS SELECTED ANOTHER DOCUMENT TO BE MOVED TO THE CONVERSION MEDIUM. (IF FALSE, INDICATES TO TERMINATE CONVERSION PROCESSING).
LOGICAL	CNVERR	SWITCH TO INDICATE THAT THE CONV PROCESS HAS FAILED
INTEGER	DSCAT(5)	TABLE IDENTIFIER TO ACCESS THE SUB-TABLE THAT INDICATES THE DOCUMENT CATALOG FOR A PARTICULAR DATASET.
INTEGER	DSKEY	SUBSCRIPT IN DSBUF() WHERE THE FIRST WORD OF THE SELECTED KEY (DOCUMENT NAME) IS TO BE FOUND
INTEGER	DSRID	SUBSCRIPT IF DSBUF() WHERE THE RECORD-ID OF THE RECORD ASSOCIATED WITH

NDA - MAXIMUM NUMBER OF DIRECT-ACCESS DATASETS
OF THE RECORD ASSOCIATED WITH
THE KEY # DSKEY IF FOUND

INTEGER DSBUF(243)

BUFFER AREA FOR HEADER RECORDS IF
THE ISDDT TABLE ACCESSES WHILE
DOCUMENT TRANSFERS ARE BEING
PERFORMED

INTEGER SEONO

SEQUENCE NUMBER OF THE NEXT CONV
RECORD TO BE GENERATED

CONVERSION SYSTEM GENERAL COMMON

AUGUST 3, 1979

COMMON /CONVRC/ ITONR, CVLUN, ISDDT, CVDLIM, CVBUF, CVCNT

INTEGER ITONR(5)

TABLE IDENTIFIER FOR RECORD-ID TRANSATION
TABLE.

INTEGER CVLUN

LOGICAL UNIT NUMBER FOR CONVERSION DATA
OUTPUT.

INTEGER ISDDT(5)

TABLE IDENTIFIER FOR THE DATASET-NAME
TABLE.

INTEGER CVDLIM

MACHINE (A1) REPRESENTATION OF THE
CHARACTER TO BE USED AS THE "SPECIAL
CHARACTER DELIMITER" FOR ALL EDITSPEC
COMMANDS ISSUED BY THE CONVERSION SYSTEM.

INTEGER CVBUF(400)

GENERAL BUFFER AREA FOR USE BY THE CONVERSION
SYSTEM.

INTEGER CVCNT

COUNT OF ALL CONVERSION RECORDS
GENERATED BY THE CONV SYSTEM

COMMON /CONVR2/

1ZLOGON(6),ZDOCU (54),ZFOOT (102),IDUMMY(245),
1JDUMMY(245),ISDATA(40),IUDATA(30),ISPEC(625),
1IKEY (3)

INTEGER ZDOCU,ZFOOT

AUGUST 31, 1977

INTEGER INPCS,OUTCS

COMMON /CSETC/ INPCS,OUTCS

INPCS IS INPUT CHARACTER SET.
OUTCS IS OUTPUT CHARACTER SET.

OCTOBER 27, 1978

CONVERSION-SYSTEM LOGIN COMMON

COMMON /CVLGNC/ LOGIDE

INTEGER LOGIDE(3)

CONTAINS THE USER-ID (IN EDITSPEC FORM)
THAT WAS USED IN THE LAST CALL TO
"CVLGIN". SHOULD BE DATAED IN BLOCK DATA
TO / 3*0 /

COMMON BLOCK: CVUSRC 06-SEP-78

COMMON /CVUSRC/ SEQNM, RECSIZ, DATALN

COMMON VARIABLE DESCRIPTIONS:

TYPE	NAME	DESCRIPTION
------	------	-------------

INTEGER	SEQNM	
---------	-------	--

CONTAINS THE SEQUENCE NUMBER OF
THE NEXT CONVERSION RECORD TO
BE EMITTED.

INTEGER	RECSIZ	
---------	--------	--

CONTAINS THE RECORD SIZE (IN
CHARACTERS) THAT WAS READ
DURING THE LAST DKGET. USED
PRIMARILY BY CVUSR2.

INTEGER	DATALN(68)	
---------	------------	--

BUFFER TO HOLD THE ALPHAMERIC
PORTIONS OF THE CONVERSION
RECORDS.

JULY 22, 1977

INTEGER DEBUG

COMMON /DEBUGC/DEBUG

DEBUG IS TRACE OUTPUT SWITCH.

DOCUMENT KEYWORD INDEX TABLE COMMON

MAY 25, 1978

INTEGER DKYWDI

COMMON /DKYWDC/ DKYWDI(5,6),KWUSE

DKYWDI: DOCUMENT KEYWORD TABLE

DKYWDI(1,N)= THE PARTICULAR PART OF THE FILE-NAME

DKYWDI(2,N)= THE ID OF THE FIRST RECORD

DKYWDI(3,N)= THE SIZE OF THE PRIMARY KEY

DKYWDI(4,N)= THE SIZE OF THE SECONDARY KEY

NDA = MAXIMUM NUMBER OF DIRECT-ACCESS DATASETS
 WHERE N IS THE KEYWORD INDEX (1 TO 6)
 KWUSE=NUMBER OF KEYWORD INDEX TABLES USED BY THE DOCUMENT

JULY 22, 1977

DOCUMENT EDIT COMMON

INTEGER EDITF,EDS,EBDS,EACC,EREL

LOGICAL EREAD,EDBAK

COMMON /EDITC/ EDITF(4),EDS,EBDS,EREL,IEDIT,MGO,ICYC,IDAC,INC,

1 ITQTY(5,15),IERR,IAUD,EDBAK,EACC,EREL

DIMENSION ITAUG(5),ITAUD(5),ITAUT(5),ITDRT(5),ITER(5),

1 ITERC(5),ITERL(5),ITERS(5),ITFLG(5),ITIX(5),

2 ITLC(5),ITPUL(5),ITABC(5),ITEXT(5),ITBAK(5)

EQUIVALENCE (ITAUG(1),ITQTY(1,1)),(ITAUD(1),ITQTY(1,2))

EQUIVALENCE (ITAUT(1),ITQTY(1,3)),(ITDRT(1),ITQTY(1,4))

EQUIVALENCE (ITER(1),ITQTY(1,5)),(ITERC(1),ITQTY(1,6))

EQUIVALENCE (ITERL(1),ITQTY(1,7)),(ITERS(1),ITQTY(1,8))

EQUIVALENCE (ITFLG(1),ITQTY(1,9)),(ITIX(1),ITQTY(1,10))

EQUIVALENCE (ITLC(1),ITQTY(1,11)),(ITPUL(1),ITQTY(1,12))

EQUIVALENCE (ITABC(1),ITQTY(1,13)),(ITEXT(1),ITQTY(1,14))

EQUIVALENCE (ITBAK(1),ITQTY(1,15))

EDITF IS AN ARRAY CONTAINING THE DOCUMENT-NAME OF THE DOCUMENT
 BEING EDITED, IN THE FIRST 3 WORDS, AND ALL BLANKS IN
 THE 4TH WORD

EDS IS THE DATASET NUMBER OF THE DATASET CONTAINING THE DOCUMENT
 BEING EDITED

EBDS IS THE DATASET NUMBER OF THE DOCUMENT COMMAND BACKUP DATASET

EREL IS READ-ONLY-FLAG.

.FALSE. MEANS READ/WRITE ACCESS TO DOCUMENT

.TRUE. MEANS READ-ONLY ACCESS.

IEDIT IS IN-EDIT-MODE SWITCH.

0 MEANS NOT-IN-EDIT-MODE

1 MEANS IN-EDIT-MODE

MGO IS XTRNL-REFERENCE-OVERRIDE SWITCH

0 MEANS DO NOT OVERRIDE

-1 MEANS OVERRIDE

ICYC IS CURRENT CYCLE NUMBER

IDAC IS USER DOCUMENT ACCESS CODE

INC IS LINE INCREMENT

ITQTY IS 15 DIFFERENT ARRAYS, EACH 5 WORDS LONG - ONE FOR EACH OF
 THE 14 DIFFERENT DOCUMENT TABLES, AND ONE FOR THE BACKUP
 TABLE. FOR EACH ARRAY, THE 5 WORDS HAVE THE FOLLOWING
 INFORMATION-

1. THE PARTICULAR PART OF THE FILE-NAME (FILNM(4))

2. THE ID OF THE FIRST RECORD

3. SIZE OF PRIMARY KEY

4. SIZE OF SECONDARY KEY

5. STORAGE MODE (PACK OR/AND POINT)

THE 15 DIFFERENT ARRAYS ARE AS FOLLOWS. FOR EACH TABLE, THE FORTRAN
 VARIABLE NAME, THE PARTICULAR PART OF THE FILE-NAME OF
 THE TABLE, AND A DESCRIPTION OF THE TABLE.

IT---	VALUE OF IT---(1) (FILNM(4))	DESCRIPTION
-------	---------------------------------	-------------

ITAUG	'AUG'	AUDIT TABLE (GENERAL)
-------	-------	-----------------------

ITAUD	'AUD'	AUDIT TABLE (OTHER)
-------	-------	---------------------

ITAUT	'AUT'	AUDIT TABLE (TEXT)
-------	-------	--------------------

ITDRT	'DRT'	DOCUMENT DIFFERENCED TABLE
-------	-------	----------------------------

NDA - MAXIMUM NUMBER OF DIRECT-ACCESS TABLES ACCUMULATED WITH
 2. THE ID OF THE FIRST RECORD
 3. SIZE OF PRIMARY KEY
 4. SIZE OF SECONDARY KEY
 5. STORAGE MODE (PACK OR/AND POINT)
 IFDIT IS 1 IF WE ARE IN FDIT-MODE; 0 OTHERWISE.

DOCUMENT KEYWORD INDEX TABLE COMMON

MAY 25, 1978

INTEGER DKYWDT

COMMON /FKYWD/ DKYWDT(5,6),KWUSE

DKYWDT: DOCUMENT KEYWORD TABLE

DKYWDT(1,N)= THE PARTICULAR PART OF THE FILE-NAME

DKYWDT(2,N)= THE ID OF THE FIRST RECORD

DKYWDT(3,N)= THE SIZE OF THE PRIMARY KEY

DKYWDT(4,N)= THE SIZE OF THE SECONDARY KEY

DKYWDT(5,N)= THE STORAGE MODE (PACK/POINT)

WHERE N IS THE KEYWORD INDEX (1 TO 6)

KWUSE=NUMBER OF KEYWORD INDEX TABLES USED BY THE DOCUMENT

DECEMBER 15, 1977

COMMON /FLSLC/ IFLSW,IFLNO,IFLCH

IFLSW = 0 NORMAL CALL OF FLGCH

IFLSW = 1 FLGCH CALLED BY TEXTGC AND HENCE REQUIRES THE VALUES
 OF IFLNO & IFLCH TO BE SPECIFIED VIA COMMON AND NOT VIA STRING

IFLNO THE FLAG ID

IFLCH THE FLAG CHOICE NO OF THE IFLNO FLAG ID

JULY 22, 1977

COMMON /FORMAT/

1 PFMTT(15), PFMTB(15),PFMS(29),PFMNS(12)
 1,PFMHF(12) ,PFMTC(15)

INTEGER PFMTT,PFMTB,PFMS,PFMNS,PFMHF ,PFMTC

PFMTT(15)- PRINT FORMAT FOR THE MARGIN AT THE TOP OF PAGE
 PFMTB(15)- PRINT FORMAT FOR THE MARGIN AT THE BOTTOM OF PAGE
 PFMS(29)- PRINT FORMAT FOR SKIPPING LINES - ALL COLUMNS
 PFMNS(12)- PRINT FORMAT FOR NONSKIP OF LINES- TEXT ONLY
 PFMHF(12)- PRINT FORMAT FOR HEADER,FOOTER,PAGENUMBER
 PFMTC(15) - PRINT FORMAT FOR PAGE EJECT ON TERMINALS

CCCCCCCC

cccccc

CCCC

CCCCCCCCCCCC

[illegible]

CC
CC

cc

5

2. THE ID OF THE FIRST RECORD
OF CONVERSION RECORDS

INTEGER	INDEX	KEYWORD INDEX # OF TABLE BEING CURRENTLY RESTORED
INTEGER	TYPE	TYPE CODE OF CONVERSION RECORD JUST READ
INTEGER	PRVTYP	TYPE CODE OF CONVERSION RECORD PREVIOUS TO CURRENT ONE
INTEGER	DOCCNT	COUNT OF DOCUMENTS CURRENTLY RECORDED ON ACCESS LIST WE ARE NOW RESTORING
INTEGER	EXDOC	NUMBER OF DOCUMENT NAMES WE EXPECT TO RESTORE TO THIS ACCESS LIST
INTEGER	EXSEQ	SEQUENCE NUMBER WE EXPECT THE NEXT RESTORE RECORD TO HAVE.
INTEGER	SEQNM	SEQUENCE NUMBER OF RESTORE RECORD JUST READ.
LOGICAL	KFERR	"FATAL ERROR" SWITCH DURING KEYWORD RESTORE PHASE
INTEGER	DOCBUF(14)	BUFFER TO HOLD DOCUMENT NAMES AND FLAGS BEFORE APPENDING THEM TO THE DOCUMENT-ACCESS LISTS
INTEGER	WRKBUF(20)	WORK BUFFER
INTEGER	SUBTBL(5)	TABLE-IDENTIFIER FOR INDIVIDUAL KEYWORD SUB-TABLES
INTEGER	KNOTBL(5)	TABLE-IDENTIFIER FOR KEYWORD NUMBER TABLE.
INTEGER	LVORID	RECORD-ID OF LEVEL 0 RECORD (WHICH CONTAINS CRE-ID, KW-TBL POINTER, ETC.)
INTEGER	KACRID	RECORD-ID OF LAST KEYWORD ACCESS RECORD GENERATED
INTEGER	KYSIZE	SIZE OF CURRENT KEYWORD(S) IN CHARACTERS
INTEGER	KYWNO	

RESTORED

MARCH 6, 1978

INTEGER RSRCS

COMMON /LOCKC/ RSRCS(7,20),NRSRCS

RSRCS(1,ALL) DATA SET NUMBER

RSRCS(2,ALL) SYSTMTABLE WORD OR FIRST WORD OF DOCUMENT NAME

RSRCS(3,ALL) SYSTMTABLE WORD OR 2 ND WORD OF DOCUMENT NAME

RSRCS(4,ALL) SYSTMTABLE WORD OR 3 RD WORD OF DOCUMENT NAME

RSRCS(5,ALL) SYSTMTABLE WORD OR 4 TH WORD OF DOCUMENT NAME

RSRCS(6,ALL) DATA BASE NUMBER (STARTING FROM 0).

RSRCS(7,ALL) SHARED OR EXCLUSIVE USE REQUEST

NRSRCS NUMBER OF RESOURCES REQUESTED

JULY 22, 1977

LOG-ON AND LOG-OFF COMMON

INTEGER ACTNO,TLOG,CLTIM

COMMON /LOGC/ ACTNO(3),TLOG(5),CLTIM

ACTNO IS ACCOUNT NUMBER CURRENTLY LOGGED-ON UNDER.

TLOG CONTAINS THE TIME AND DATE OF LOG-ON.

LIST/PRINT COMMON USED TO PRODUCE A HARD COPY TEXT FOR EDITOR/ENGR

COMMON /LTPTC/ LTPT

LTPT SWITCH 0= OFF,1= ON

JULY 22, 1977

AUXILIARY MACHINE SPECIFIC CONSTANTS

COMMON /MC/ MAXCH,IND1,NCUM,IND2,NCUM2,NCU2,IND3,NCU243

THIS COMMON BLOCK CONSISTS OF VALUES THAT COULD HAVE BEEN
COMPUTED FROM NCU (NUMBER OF CHARACTERS PER STANDARD-UNIT),
AND THAT ARE PRE-COMPUTED AND STORED IN COMMON TO SAVE TIME

MAXCH = NCU*80+1

IND1 = 1

NCUM = NCU-1

IND2 = NCU+1

NCUM2 = NCU*2-1

NCU2 = NCU*2

IND3 = NCU*2+1

NCU243 = NCU*243

JULY 22, 1977

2. THE ID OF THE FIRST RECORD
OF CONVENTION RECORDS
1. INPNFC(72.6), INPAGE(72)
1. INPNFCC

INTEGER PAGE, PAGENO

DIMENSION PAGENO(60), ISFL(6), ISTN(6), ISTA(6), ISLI(6), ISLP(6)

EQUIVALENCE (PAGE(1), ISFI), (PAGE(8), LOP), (PAGE(9), ISOP)

1. (PAGE(10), IPJ), (PAGE(35), ITLP)
2. (PAGE(2), ISFL(1)), (PAGE(11), ISTN(1))
3. (PAGE(17), ISTA(1)), (PAGE(23), ISLI(1))
4. (PAGE(29), ISLP(1)), (PAGE(37), PAGENO(1))
5. (PAGE(36), NSD)

NUMBR COMMON DESCRIPTION

IPAGNO(15) -THE PAGE NUMBER IN MACHINE CHARACTERS ,
PACKED FORMAT.

NCPGNO -THE NUMBER OF WORDS IN (IPAGNO(15))

PAGE(96) -INFORMATION ABOUT THE PAGE NUMBER.

EQUIVALENCED AS FOLLOWS:

ISFI= SUBFIELD TO INCREMENT

ISFL= SUBFIELD LENGTH 1-6 SUBFIELDS

THE FOLLOWING READ FROM PAGE NUMBER FORMAT RECORD

LOP = LOCATION ON PAGE

ISOP= START ON PAGE NUMBER

IPJ = PAGE JUSTIFICATION

ISTN= SUBFIELD TYPE NUMERIC

ISTA= SUBFIELD TYPE ALPHABETIC

ISLI= SUBFIELD LAST CHARACTER OF INCREMENT

ISLP= SUBFIELD LAST CHARACTER TO PRINT

ITLP= TOTAL LENGTH OF PAGE NUMBER

NSD NUMBER OF SUBFIELDS DEFINED

PAGENO= ACTUAL PAGE NUMBER IN MACHINE CHARACTER

UNPACKED FORM(RIGHT JUSTIFIED)

NPNFC = NO OF PARAGRAPH NUMBER FORMATS IN COMMON

IPNFC(96.6) - SIX DIFFERENT PARAGRAPH NUMBERS

- INFORMATION IN SAME ORDER AS DEFINED

- FOR (PAGE)

INPNFC = INITIALIZATION FOR PARAGRAPH NOS

INPAGE = INITIALIZATION FOR PAGE NOS

IPNFC PAGE NUMBER FORMAT IN COMMON

JULY 22, 1977

COMMON /PAGEBF, FLAG(64), LNNUM(64), PGBUF(40,64)

1. AUDDOC(3,64), AUDTCN(64), TEXTSG(64)

3. IHEADO(40,12), IHEADE(40,12), HDLNO, HDLNE

4. IFOOTO(40,12), IFOOTE(40,12), FTLNO, FTLNE

5. NPFC, IPFT(4,12)

6. MLBODY, KLN

4. IHEADS(40), IHEAO, IHEAE, INC, IFOOTS(40), IFOOO, IFOOE, IFC

1. IPGCNT, ILRJ, NLCFP

1. INTEGER FLAG, PGBUF, AUDDOC, AUDTCN, TEXTSG
MDLNO, MDLNE, FTLNO, FTLNE

PAGEBF COMMON DESCRIPTION

FLAG(64) - THE FIRST FLAG FOUND ON THIS ROW .NEGATIVE IF
- MORE FLAGS ARE ON THIS ROW.
LNNUM(64)- THE LINE NUMBER IN THIS DOCUMENT OR THE NEGATIVE
- OF THE LINE NUMBER COPIED FROM ANOTHER DOCUMENT.
PGBUF(40,64)- THE TEXT COLUMN FOR ONE PAGE
AUDDOC(3,64)- NAME OF THE DOCUMENT THAT THIS LINE HAS
- BEEN COPIED FROM.
AUDTCN(64) - CYCLE NUMBERS FOR DOCUMENT COPIED AND FOR
- TEXT ENTRANCE INTO CURRENT DOCUMENT
TEXTSG(64) - THE TEXT SEGMENT FOR THIS LINE

IHEAD0 = ODD PAGE HEADER STORAGE

IHEAE = EVEN PAGE HEADER STORAGE

MDLNO = NO LINES IN HEADER -ODD

MDLNE = NO LINES IN HEADER -EVEN

IFOOT0 = ODD PAGE FOOTER STORAGE

IFOTE = EVEN PAGE FOOTER STORAGE

FTLNO- NO. LINES IN FOOTER -ODD

FTLNE- NO. LINES IN FOOTER -EVEN

NPFC - NO. OF PARAGRAPH FORMATS LOADED INTO COMMON

IPFT(4,12) = PARAGRAPH FORMAT TABLE

IPFT(1, ALL) = LEFT INDENTION OF FIRST LINE

IPFT(2, ALL) = LEFT INDENTION ALL OTHER LINES

IPFT(3, ALL) = RIGHT INDENTION OF ALL LINES

IPFT(4, ALL) = PARAGRAPH ID

MLBODY = MAXIMUM NO. OF PRINTABLE LINES IN PGBUF FOR BODY OF TEXT

KLN = LINE NUMBER IN PGBUF BEING WRITTEN. NEG. NOT FULL YET.

IHEADS(40) = HEADER STORAGE

IHEAE = LINE LOCATION OF HEADER STORAGE -EVEN

IHEAO = LINE LOCATION OF HEADER STORAGE -ODD

IHC = HEADER STORAGE CHANGED 1= YES 0= NO

IFOOTS(40) = FOOTER STORAGE

IFOE = LINE LOCATION OF FOOTER STORAGE - EVEN

IFOOO = LINE LOCATION OF FOOTER STORAGE - ODD

IFC = FOOTER STORAGE CHANGED 2= YES 0= NO

IOJPN = OUTSIDE JUSTIFICATION OF PAGE INFORMATION

• INDICATES AN EVEN PAGE

- INDICATES AN ODD PAGE

LFL = LAST FOOTNOTE LINE

LMAX = TOTAL NUMBER OF LINES ALLOWED ON THIS PAGE (MAX-FN)

IPFSN = PRINT FORMAT STATEMENT NUMBER IDENTIFIER

IPGCNT NUMBER OF PAGES PRINTED IN THIS PRINT COMMAND

ILRJ 1= LEFT/RIGHT JUSTIFICATION OF EACH LINE BEFORE PRINT

NLCFP NO. LINES TO FOLD/TERMINAL TYPEWRITER ONLY

AUGUST 29, 1977

READ AND PARSE COMMAND ROUTINES COMMON AREA.

INTEGER PMPTR, STRNG, EDLIN, EDSIZ

LOGICAL EOFIL, NXTD, CMDOK

COMMON /PARSC/MCLIN(400), STRNG(400), PMPTR(100,3), NXTLN(80),

1. NCHAR, ISUB, EOFIL, NXTD, CMDOK, EDLIN(400), MCSIZ, EDSIZ

NO IN IS USED IN STORE LINE READ IN MACHINE FORMAT (ALL FOR WRITE)

NDA = MAXIMUM NUMBER OF DIRECT ACCESS DATASETS

PRINTOUT, NIS THE SIZE OF THE SECONDARY KEY

2. THE ID OF THE FIRST RECORD
 STRNG CONTAINS THE CONVERTED COMMAND (IN EDITSPEC FORMAT)
 PMPTR IS USED TO BREAK UP A COMMAND STRING INTO FIELDS OR PARAMETERS
 PMPTR(,1) IS THE CHARACTER INDEX INTO THE ARRAY STRNG
 INDICATING START OF THE FIELD
 PMPTR(,2) IS LENGTH OF FIELD IN CHARACTERS
 PMPTR(,3) IS FIELD TYPE CODE
 NXTLN IS THE CONTENTS OF THE NEXT LINE (NEXT COMMAND) IF IT HAS
 ALREADY BEEN INADVERTANTLY READ
 NCHAR IS NUMBER OF CHARACTERS OF COMMAND IN ARRAY STRNG.
 ISUB IS COMMAND NUMBER CODE RETURNED BY COMMAND DECODER
 EOFIL IS END-OF-FILE INDICATOR FOR READING
 NXTRD IS .FALSE. IF NXTLN HAS NOT BEEN READ
 .TRUE. IF NEXT LINE HAS BEEN READ INTO ARRAY NXTLN
 CMDOK IS .FALSE. IF COMMAND COULD NOT BE EXECUTED DUE TO ERROR
 .TRUE. IF COMMAND WAS EXECUTED AND MAY BE BACKED-UP
 EDLIN STORES LINE READ IN PACKED EDITSPEC FORMAT.
 MCSIZ IS NUMBER OF CHARACTERS READ INTO MCLIN.
 EDSIZ IS NUMBER OF CHARACTERS PLACED INTO EDLIN.

COMMON /PRERRC / IPRERR
 FOUND ONLY IN GTCMD AND DEBOG FOR SUPPRESSION OF NOT A COMMAND PRINTING

AUGUST 1, 1977
 MODIFIED JULY 10, 1979: CMBUF 126 CHANGED TO 400.

COMMON /PRINTC/ DFDR(58)
 1. FLGOP(6), JRSLN, IRELN, ISEG(3), NITC, LCO, NOD, NPPNI
 1. ICNTD, ICNOD, IODNM(3)
 1. INSTR(126), CMBUF(400), LPBUF(126)
 1. MORLN, MORIL, NCHCO, LSTRT, KSTRT, LINE, CTBP, LENG, ITAB1, ITAB2
 1. RID1, RID2, ITEXT(743), IFLAGT, NOLIN, RTAB
 1. NPSW, K2PSW, KSW, LWID, SLEN, INDIL, INDLT, INDRT
 1. NEWST(126), MINSTR, MNEWST, LINSTR, LNEWST, NINSTR, NNEWST
 1. XBLNK, LWIDSU, IOFSET
 1. IPERID
 1. ILNCLN
 1. IUNSC
 1. LFLAG, INLINE, LAUD(3), ICYCLE, ITXTSG
 1. NSLVL(15), LSTCT(15), LENCT(15), ITOPS, LEVHST, INDST, LSNEW(15)
 1. LENEW(15)

DIMENSION IPP(4,10)
 INTEGER CMBUF, CTBP, RTAB, XBLNK, SLEN, FLGOP
 INTEGER DFDR
 INTEGER RID2, RID1, TEXTDR(126)

EQUIVALENCE (DFDR(1), IFOOT), (DFDR(2), IHEAD)
 1. (DFDR(3), ILNG), (DFDR(4), IWID), (DFDR(5), ILEVH)
 2. (DFDR(6), IPPNF), (DFDR(7), IPPBP), (DFDR(8), IPPBP1)
 3. (DFDR(9), IPPBP2), (DFDR(10), IPAF), (DFDR(11), INS)
 4. (DFDR(12), IRLJ), (DFDR(13), MT), (DFDR(14), MB)
 5. (DFDR(15), ISAF), (DFDR(16), ISALN), (DFDR(17), ISAT)
 6. (DFDR(18), ISAA), (DFDR(19), IPP(1,1))
 1. (LPBUF(110), LINES), (LPBUF(111), LNSTA), (LPBUF(112), LNEED)
 2. (LPBUF(113), LSFID), (LPBUF(114), LNSPC), (LPBUF(115), NLINE)
 3. (LPBUF(116), X1), (LPBUF(117), JUST), (LPBUF(118), X2)
 4. (LPBUF(119), X3), (LPBUF(120), LENGX), (LPBUF(121), X5)

6. (LPBUF(125).LENGE) . (LPBUF(126).NCHCE)
EQUIVALENCE (TEXTOR(1),ICNTD)

DFDRC(48) - ALL INFORMATION RELATED TO THE DOCUMENT FORMAT
EQUIVALENCED AS FOLLOWS:
THE FOLLOWING ARE VARIABLES IN THE DOCUMENT FORMAT RECORD

IFOOT = FOOTER FORMAT ID
IHEAD = HEADER FORMAT ID
ILNG = PAGE LENGTH
IWID = PAGE WIDTH
ILEVM = MAXIMUM NUMBER OF LEVELS
IPPNF = PARAGRAPH NUMBER FORMAT
IPBP = PRINT PARG NO AT BOTTOM OF PAGE
IPBP1 = PRINT PARG NO AT BOTTOM OF PAGE - 1 ST SUBFIELD
IPBP2 = PRINT PARG NO AT BOTTOM OF PAGE - LAST SUBFIELD
IPNF = PAGE NUMBER FORMAT ID
INS = NO SKIP BETWEEN HEADER/FOOTER AND PAGE NUMBER
IRLJ = RIGHT / LEFT JUSTIFICATION
MT = TOP MARGIN IN LINES
MB = BOTTOM MARGIN IN LINES
ISAF = SPACE BETWEEN FLAGS AND LINE NUMBER COLUMNS
ISALN = SPACE BETWEEN LINE NO AND TEXT COLUMNS
ISAT = SPACE BETWEEN TEXT AND AUDIT TRAIL
ISAA = SPACE BETWEEN AUDIT NAME AND CYCLE NOS
IPP(1,A) = PARAG SUBFIELD TO INCREMENT
IPP(2,A) = PARAG SUBFIELD TO PRINT FIRST
IPP(3,A) = PARAG SUBFIELD TO PRINT LAST
IPP(4,A) = PARAG FORMAT IDENT
FLGOP(6) = DATA COLUMNS TO BE PRINTED: 1- X-50 2- 38-L
3-32-F 4-46-T 5-27-A 6-42-P
IRSLN- SLN REC ID
IRELN- ELN REC ID
ISLN= STARTING LINE NO TO PRINT.USER INPUT
IELN= ENDING LINE NUMBER TO PRINT.USER INPUT
ISEG= TEXT SEGMENT 1 THRU
ISPAC= LINE SPACING01=SINGLE 2= DOUBLE
NITC/ NO INDEX/TABLE CONTENTS
LCO/ LOGIC CHECK OVERRIDE
NOD=NUMBER OF OUTPUT DEVICE 0 OR 1
NPPNI- NO P/P INITIALIZATION
IFORM= DOCUMENT FORMAT ID
ICNTD = CYCLE NUMBER FOR THIS LINE IN THIS DOCUMENT
ICNOD = CYCLE NUMBER FOR THIS LINE IN OTHER DOCUMENT
IODNM(3)= OTHER DOCUMENT NAME
INSTR = ONE LINE OF INPUT FROM A DOCUMENT TEXT TABLE
CMBUF = THE ARRAY THAT CONTAINS THE COMMAND NEST
LPBUF = 1-109 WORDS HOLD THE CURRENT LINE TO BE PRINTED IN
MACHINE CHARACTERS
LINES = TOTAL NUMBER OF LINES TO PRINT IN THE TEXT COLUMN
LNSTA = STARTING LINE NUMBER TO PRINT
LNEND = ENDING LINE NUMBER TO PRINT
LSFID = DOCUMENT FORMAT ID
LNSPC = LINE SPACING FOR PRINTING
NLINE = CURRENT LINE NUMBER
JUST = TEXT JUSTIFICATION
LD = NUMBER OF CAPITAL LETTERS FOUND IN ARRAY INSTR
LENGE = NUMBER OF MACHINE CHARACTERS IN ARRAY INSTR BEFORE
THE COMMAND STRING
NCHCE = NUMBER OF CHARACTERS IN COMMAND STRING (MACHINE CTRS)
MORLN = NUMBER OF CHARACTERS (MACHINE) IN LPBUF
MORIL = THERE ARE MORE CHARACTERS IN THIS LINE TO BE CHECKED
IF EQUIL TO ONE . NO MORE IN LINE IF EQUIL TO ZERO
NCHCO = NUMBER OF CHARACTERS IN COMMAND STRING (EDITSPED CTRS)
LSTRT = FIRST WORD IN ARRAY INSTR THAT HAS NOT BEEN SEARCHED
FOR AN INTERNAL COMMAND USED IN LOGIC

NMA = MAXIMUM NUMBER OF DIRECT ACCESS DATA SETS
INVTOT(12) IS THE SIZE OF THE COMMANDARY FILE

KSTRT = FIRST CHARACTER IN FIRST WORD IN ARRAY INSTR THAT HAS NOT BEEN SEARCHED FOR AN INTERNAL COMMAND. USED IN LOCAC
 LINE = INSTRUCTIONS FOR PROCESSING A LINE :NEG- COPY COMMAND ZERO- THIS DOCUMENT ,POSITIVE- CONTINUE TO SCAN PRESENT LINE FOR COMMAND.
 CTBP = COMMAND TO BE PROCESSED IF EQUIL TO ONE,NONE IF ZERO.
 LENG = NUMBER OF INTERNAL CHARACTERS IN ARRAY INSTR BEFORE THE COMMAND STRING
 ITAB1 = LEFT MARGIN OF TEXT COLUMN
 ITAB2 = RIGHT MARGIN OF THE TEXT COLUMN
 RID1 = RECORD ID OF THE LINE TO BE READ NEXT
 RID2 = THE RECORD WHERE THE START OF THE TABLE IS FOUND
 INTEXT(243) = HEADER RECORD IN TEXT TABLE -CURRENT RECORD
 IFLAGT = POINTER TO THE LAST/NEXT TEXT TABLE KEY.
 NOLIN = 1=NO MORE LINES TO BE PRINTED.=0,MORE LINES TO BE PRINTED
 RTAB= +10 = TABLR ROUTINE SAID THE TABLE IS COMPLETE
 RTAB= -2 = PROCESSING A COPY TABLE COMMAND
 RTAB= +2 = PROCESSING A TB COMMAND
 RTAB= 0 = NORMAL LINE PROCESSING
 NPSW =
 -1 = *P* NOT NEXT CTR TO BE PROCESSED
 -3 = *LJ,RJ,CJ* NOT NEXT
 0 =
 1 = *P* COMMAND READY
 3 = *JUST* COMMAND REAY
 -1 = *SL* NO PREVIOUS *P*
 K2PSW = 0 =
 1 = TWO *P* COMMANDS ON SAME LINE
 1 = *SL* FOUND AFTER A *P*
 KSW =
 LWID = NUMBER OF CTRS IN PRINT LINE
 SLEN = SPACE AT LINE END IN THE (KSW) CURRENT PRINT LINE
 NEWST(126) = INPUT (INSTR) IN EXTERNAC-MACHINE CTRS
 MINSTR = MAXIMUM WORDS ACTUALLY READ INTO INSTR
 MNEWST = MAXIMUM WORDS ACTUALLY PLACED INTO NEWST
 LINSTR = LAST CHARACTER PROCESSED IN INSTR
 LNEWST = LAST CHARACTER PROCESSED IN NEWST
 NINSTR = TOTAL CTRS ACTUALLY IN INSTR
 NNEWST = TOTAL CTRS ACTUALLY IN NEWST
 XBLNK = ONE MACHINE BLANK RIGHT JUST
 LWIDSU = NO. SUS IN LINE WIDTH
 IOFSET = NO. COMMAND CTRS TO SKIP TO REACH NEXT CTR TO PROCESS
 INDRT = RIGHT INDENTATION
 INDLT = LEFT INDENT
 INDIL = FIRST PARAGRAPH INDENT
 IPERID = ONE MACHINE PERIOD IN RIGHT LOCATION
 ILNCLN = CLEAN UP BLANKS ON LINE:1=YES,0=NO
 IUNSC = MACHINE UNDERSCORE CHARACTER
 IFLAG = CURRENT FLAG ID
 INLINE = CURRENT LINE NUMBER TO PRINT
 IAUD(3) = CURRENT DOC NAME TO PRINT
 ICYCLE = CURRENT CYCLE TO PRINT
 ITXTSG = CURRENT TEXT SEGMENT TO PRINT
 NSLVL = STACK HOLDING NESTED LEVEL NUMBER FOR INTERNAL COMMANDS.
 LSTCT = STACK HOLDING LOCATIONS OF THE STARTING CHARACTER OF NESTED INTERNAL COMMANDS.
 LENCT = STACK HOLDING LOCATIONS OF THE ENDING CHARACTER (*) OF NESTED INTERNAL COMMANDS.
 ITOPS = POINTER POINTS TO THE TOP OF STACK. (NOTE THAT THE FIRST ELEMENT OF ARRAY IS CONSIDERED AS THE BOTTOM OF THE STACK. E.G. NSLVL(1) IS THE BOTTOM OF THE STACK IF ITOPS .GT. 1)
 LEVMST = HIGHEST NESTING LEVEL.

LSNEW = STACK HOLDING STARTING LOCATION OF COMMANDS IN NEWST.
 LENEW = STACK HOLDING ENDING LOCATION (*) OF COMMANDS IN NEWST.

SEPT 1979
 COMMON /PRTBLC / ACTALN
 ACTALN *CT* COMMAND LINE NUMBER

.. RESTORE COMMON - DOCUMENT PARTITION

.. MARCH 25, 1979

COMMON /RESDOC/ DOCNM, DSNAME, DSNO, TYPE, OLDTP, SEQNM,
 EXSEQN, CVREC, CVIN, RESERR, CREUID, RESBUF

INTEGER	DOCNM(3)	NAME (IN EDITSPEC CHARS) OF DOC CURRENTLY BEING TRANSFERRED TO THE CONVERSION MEDIUM
INTEGER	DSNAME(2)	DATASET NAME TO WHERE THE CURRENT DOCUMENT IS TO BE STORED
INTEGER	DSNO	DATASET NUMBER OF THE DATASET IDENTIFIED IN DSNAME()
INTEGER	TYPE	TYPE CODE OF RESTORE RECORD JUST OBTAINED
INTEGER	OLDTP	TYPE CODE OF THE LAST RESTORE RECORD (USED TO ENFORCE SEQUENC OF RESTORE OPERATIONS)
INTEGER	SEQNM	SEQUENCE NUMBER OF THE CURRENT RESTORE RECORD
INTEGER	EXSEQN	(EXPECTED) SEQUENCE NUMBER OF THE NEXT RESTORE RECORD TO BE READ
INTEGER	CVIN	LOGICAL UNIT NUMBER OF THE CONV INPUT DATA SOURCE
LOGICAL	RESERR	SWITCH TO SHOW IF AN ERROR HAS OCCURED DURING THE RESTORE PROCESS
INTEGER	CREUID(3)	CREATOR USER ID OF THE OWNER OF THE CURRENT DOCUMENT

INTEG. CUDFC(79)
 NDB - MAYTMIN NUMBER OF DIRECT-ACCESS DATASETS
 PRYUNT/4.NIS THE SIZE OF THE SECONDARY KEY

RAW FORM OF THE CONVERSION RECORD
DATA. TO BE SUPPLIED TO THE
INDIVIDUAL RESTORE ROUTINES

INTEGER RESBUF(100)

GENERAL PURPOSE BUFFER AREA FOR
THE RESTORE ROUTINES TO SHUTTLE
INFORMATION BETWEEN THEMSELVES.
THE ACTUAL FORMAT OF DATA IN THIS
BUFFER IS DEPENDENT ON THE
CLUSTER OF RESTORE ROUTINES
CURRENTLY ACTIVE (IE, RESTORING
TEXT TABLE, AUDIT TABLES, ETC)

AUG 17, 1977

SCRATCH COMMON, FOR QUANTITIES USED AS DUMMY ARRAYS IN DIFFERENT ROUTINES.

COMMON /SCRTC/ IHED(243)

IHED(243) IS USED AS ARGUMENT TO TABLE-HANDLER ROUTINES.

SEARC ROUTINE COMMON

THIS COMMON AREA WAS ADDED TO REDUCE THE NUMBER OF PARAMETERS
PASSED TO SEARC TO A TOLERABLE LEVEL.

THE ROUTINES CALLING SEARC ARE:LOCAT,CHANG,ERASE,LDHAR

INTEGER RITO,TSID,SW50

COMMON /SEARCH/ LBLNK,LSBLN,LITO(50,4),LIPA(50),RITO(50,2),

*ILITO,ILIPA,IMODE,IBYT,NRMTCH,IASEN,NTEFO,TSID(10),SW50,

*ICNT,ITST

LBLNK: LEADING BLANKS IN A SOUGHT STRING.

LSBLN: TRAILING BLANKS IN THE SOUGHT STRING.

LITO: ARRAY TO HOLD LINE NUMBERS AND POSITIONS OF STRING.

LITO(N,1): LINE NUMBER OF LINE WHERE STRING STARTS.

LITO(N,2): POSITION IN LINE OF THE FIRST CHARACTER
OF THE SOUGHT STRING..

LITO(N,3): LINE NUMBER OF LINE WHERE STRING ENDS.

LITO(N,4): POSITION IN LINE WHERE STRING ENDS.

LIPA: SAME AS LITO USED FOR NEAR MATCH FINDINGS.

RITO(50,2): RECORD ID'S OF STARTING AND ENDING LINE NUMBERS
OF EACH OCCURANCE OF THE SOUGHT STRING.

ILITO: NUMBER OF EXACT OCCURANCES.

ILIPA: NUMBER OF NEAR MATCH OCCURANCES.

IMODE: SWITCH THAT INDICATES TO SEARC ROUTINE WHETHER TO
SEARCH FOR ALL OR ONLY ONE OCCURANCE OF THE SOUGHT
STRING IN EACH AREA SPECIFIED.

NRMTCH: A SWITCH THAT IS SET TO ONE IF THE NEAR MATCH
OCCURANCES ARE TO BE LISTED.

IASEN: WHEN THIS SWITCH IS ZERO WE DECAPITALIZE THE STRING
LEAVE THE STRING EXACTLY AS IT IS.

(REMOVE ALL CENT SIGNS) BEFORE MATCHING. ELSE WE

NTEFO: NUMBER OF TEXT SEGMENT ID'S GIVEN BY USER.

TSID(10): ARRAY CONTAINING THE TEXT SEGMENT ID'S. SEARCHING
WILL OCCUR ONLY IN SPECIFIED LINES OR PAIRS OF LINES
AND THEN ONLY AMONG THOSE LINES THAT HAVE ONE OF THE
TEXT SEGMENT ID'S IN THE TSID ARRAY.

SW50: SINCE THE NUMBER OF OCCURANCES OF A STRING IN AN AREA
IS LIMITED BY THE SIZES OF ARRAY LITO(TO 50) THIS
SWITCH WILL BE SET TO ONE IF THERE ARE MORE OCCURANCES

JULY 22, 1977

SIZE COMMON

COMMON /SIZEC/ NUSID,NACTN,NDSNM,NDOCN,NDATE,NTIME

NUSID IS SIZE OF USER-ID (12 CHARACTERS) IN WORDS = (NCU+1)/NCU

NACTN IS SIZE OF ACCOUNT-NUMBER (12 CHARACTERS) IN WORDS

NDSNM IS SIZE OF DATASET NAME (6 CHARACTERS) IN WORDS.

NDOCN IS SIZE OF DOCUMENT NAME (12 CHARACTERS) IN WORDS.

NDATE IS SIZE OF DATE FIELD (DD-MMM-YY 9 CHARACTERS) IN WORDS.

NTIME IS SIZE OF TIME-OF-DAY (8 CHARACTERS) IN WORDS.

C AUGUST 29, 1979

COMMON /SKPGC/ ISP

C ISP SKIP PAGE SWITCH =0 NOSKIP, =1 SKIP PAGE

C COMMON USED IN PRNTB AND STRME SUBROUTINES

SYSTEM KEYWORD INDEX TABLE COMMON

APRIL 13, 1978.

COMMON /SKYWDC/KYWD(5,6)

KYWD : SYSTEM KEYWORD TABLE

KYWD(1,N)= THE PARTICULAR PART OF THE FILE-NAME

KYWD(2,N)= THE ID OF THE FIRST RECORD

KYWD(3,N)= THE SIZE OF THE PRIMARY KEY

KYWD(4,N)= THE SIZE OF THE SECONDARY KEY

KYWD(5,N)= THE STORAGE MODE (PACK/POINT)

WHERE N IS THE KEYWORD INDEX (1 TO 6)

AUG 17, 1977

SYSTEM COMMON

INTEGER SDS,SBDS,SYSFL,BATCH,USID,EDTBL

LOGICAL MULTI,BAKUP

COMMON /SYSTEM/ IBLNK,JBLNK,SDS,SBDS,SYSFL(4),USID(3),LOGGD,MULTI,

1BAKUP,BATCH,ISQTY(5,13),ITNM(3,8),IGEN,ISUP,ISBDD(5),IDOC

2,ICLEAR

DIMENSION ISACS(5),ISCHA(5),ISDIR(5),ISSPC(5),ISUSR(5),

1ISDOC(5),ISFOF(5),ISHEF(5),ISPGN(5),ISPRF(5),ISPRN(5),ISTTF(5),

2ISBAK(5)

EQUIVALENCE (ISACS(1),ISQTY(1,1)),(ISCHA(1),ISQTY(1,2))

EQUIVALENCE (ISDIR(1),ISQTY(1,3)),(ISSPC(1),ISQTY(1,4))

EQUIVALENCE (ISUSR(1),ISQTY(1,5)),(ISDOC(1),ISQTY(1,6))

EQUIVALENCE (ISFOF(1),ISQTY(1,7)),(ISHEF(1),ISQTY(1,8))

EQUIVALENCE (ISPGN(1),ISQTY(1,9)),(ISPRF(1),ISQTY(1,10))

EQUIVALENCE (ISPRN(1),ISQTY(1,11)),(ISTTF(1),ISQTY(1,12))

EQUIVALENCE (ISBAK(1),ISQTY(1,13))

EQUIVALENCE (EDTBL,IBLNK),(MCHBL,JBLNK)

IBLNK IS A WORD FULL OF EDITSPEC BLANKS (ALSO KNOWN AS EDTBL)

JBLNK IS A WORD FULL OF MACHINE BLANKS (ALSO KNOWN AS MCHBL)

SDS IS THE DATASET NUMBER OF THE SYSTEM DATASET.

SBDS IS THE DATASET NUMBER OF THE SYSTEM BACKUP DATASET

SYSFL(4) CONTAINS THE SYSTEM TABLE GENERIC FILE NAME IN ITS

LAST 3 WORDS. THE FIRST WORD IS BLANK.

(SYSFL(1),I=2,4) =/124 SYSTEM TABLE/

USID(3) IS THE CURRENT USER ID

LOGGD IS ZERO IF NOBODY HAS LOGGED-ON. IF A VALID USER LOGS ON,

LOGGD CONTAINS THE RECORD ID OF THE DATA RECORD ASSOCIATED

WITH THE USER IN THE USER TABLE.

MULTI IS .TRUE. IF THIS IS A MULTIPLE USER EDITSPEC SYSTEM.

IF MULTI IS .FALSE. THEN THE FIRST ACCESS DATASET IS THE SECONDARY KEY

KEYWORD(1,N) IS THE SIZE OF THE SECONDARY KEY

USER EDITSPEC SYSTEM IS CREATED.
 BAKUP IS .TRUE. IF BACK-UPS ARE TO BE KEPT, SO THAT THE SYSTEM
 MAY BE RECREATED AFTER FAILURE.
 BATCH IS ZERO IF THE EDITSPEC SYSTEM IS FOR USE IN INTERACTIVE MODE,
 ONE IF IN BATCH MODE.
 ISQTY IS ACTUALLY 13 DIFFERENT ARRAYS. FOR EACH OF THE 13 ARRAYS,
 THE FOLLOWING 5 QUANTITIES ARE DEFINED, IN ORDER
 1. THE PARTICULAR PART OF THE FILE-NAME (FILNM(1))
 2. THE ID OF THE FIRST RECORD
 3. THE SIZE OF THE PRIMARY KEY
 4. THE SIZE OF THE SECONDARY KEY
 5. THE STORAGE MODE (PACK/POINT)
 FOR EACH FIVE WORD ARRAY, THE FORTRAN VARIABLE NAME, THE PARTICULAR
 PART OF THE FILE-NAME (FILNM(1)), AND A DESCRIPTION OF THE
 TABLE (FILE) FOLLOWS.

IS---	VALUE OF IS---(1) (FILNM(1))	DESCRIPTION
ISACS	'ACS'	ACCOUNTS TABLE
ISCHA	'CHA'	CHARGES TABLE
ISDIR	'DIR'	DOCUMENT DIRECTORY
ISSPC	'SPC'	SPECS TABLE
ISUSR	'USR'	USER TABLE
ISDOC	'DOC'	DOCUMENT FORMAT
ISFOF	'FOF'	FOOTR FORMAT
ISHEF	'HEF'	HEADER FORMAT
ISPGN	'PGN'	PAGE NUMBER FORMAT
ISPRF	'PRF'	PARAGRAPH FORMAT
ISPRN	'PRN'	PARAGRAPH NUMBER FORMAT
ISTTF	'TTF'	TEXT TABLE FORMAT
ISBAK	'BAK'	BACK-UP TABLE

ITNM IS USED TO PRINT ERROR MESSAGES BASED ON PMPTR CODE.
 IGEN IS USUALLY ZERO; EXCEPT DURING GENERATE/UPDATE, WHEN IT IS ONE.
 ISUP IS SUPERVISOR SWITCH.
 ISBDD(5) IS TABLE DEFINITION ARRAY FOR ANOTHER SYSTEM TABLE.
 IDOC IS FORMAT-ID OF LEVEL 1 DOCUMENT FORMAT COMMAND LAST PROCESSED.
 ICLAR IS 1 IF BUFFERS ARE TO BE CLEARED AFTER EVERY COMMAND.

AUG 5, 77
 TABLE HANDLING ROUTINES COMMON AREA
 INTEGER MORID
 COMMON /TABLC/ NLOC,ISHDR(11),IRHDR(243),MORID
 ISHDR,IRHDR ARE SCRATCH SUPER- AND REGULAR- HEADERS USED BY TBL-HNDLR.
 MORID IS ID OF IHDR RETURNED BY TBLCS

SEPT 27, 1977
 TABL AND COPY TABLE SWITCHES COMMON.
 INTEGER TBINIT,TBCTSW,TBERR
 COMMON /TBCTSC/ TBINIT,TBCTSW,TBERR

OCTOBER 22, 1977

/TABLE-NOTE/ COMMON.

INTEGER TNARR, TNLEN, TNNUM

COMMON /TBNOTC/ TNARR(500), TNLEN, TNNUM

TNARR(500) IS USED TO STORE TABLE-NOTES, IN MACHINE A1.

TNLEN IS INDEX OF LAST WORD OF TNARR THAT HAS BEEN FILLED UP.

TNNUM IS LAST TABLE-NOTE IDENTIFICATION NUMBER USED SO FAR.

OCTOBER 22, 1977

INTEGER TBNAM, TBBRKC(6), TBCLL(40,6), TBNCL(6), TBCLM(40,5)

INTEGER TBRWM(2,100,5), TBNRW(5), TBLVL, TBDOC(3,5), TBLNO(5)

INTEGER TBHDR(243), TBFLG, TBLOC

COMMON /TBPARM/ TBNAM, TBBRKC, TBCLL, TBNCL, TBRWM, TBNRW, TBLVL, TBDOC,
TBLNO, TBHDR, TBFLG, TBLOC

TBNAM IS TABLE-FORMAT-ID.

TBBRKC(6) IS BREAK CHARACTERS, AT VARIOUS LEVELS OF *CT.

TBCLL IS COLUMN-ID-LIST IN VARIOUS *TB COMMANDS.

TBNCL IS NUMBER OF ENTRIES IN TBCLL

TBRWM IS ROW MAP (LIST OF SOURCE-ID, DESTN-ID) AT VARIOUS LEVELS.

TBNRW IS NUMBER OF ENTRIES IN TBRWM

TBLVL IS CURRENT *CT LEVEL.

TBDOC SAVES DOCUMENT NAMES FOR *CT.

TBLNO SAVES THE LINE NUMBER OF THE *CT COMMANDS, FOR GOING BACK.

TBHDR, TBFLG, TBLOC ARE FOR TABLE-HANDLING ROUTINES.

16 MAY, 1978

TABLE-PARAMETERS ACS-TABLE COMMON.

INTEGER DR1ACS, DR2ACS, DR3ACS, DR4ACS

COMMON /TPACSC/ DR1ACS(4), DR2ACS(9), DR3ACS(1), DR4ACS(30), ID1ACS,
ID2ACS, ID3ACS, ID4ACS, I2ACS3, I2ACS4, L1ACS, L2ACS, L3ACS, L4ACS

DRNACS() IS FOR STORING DATA RECORD N OF ACS-TABLE.

IDNACS IS RECORD-ID OF DATA RECORD N OF ACS-TABLE.

I2ACS3 IS INDEX OF ELEMENT 3 OF DATA RECORD 2 OF ACS-TABLE.

I2ACS4 IS INDEX OF ELEMENT 4 OF DATA RECORD 2 OF ACS-TABLE.

LNACS IS LENGTH OF DATA RECORD N OF ACS-TABLE (IN CHARACTERS).

JULY 22, 1977

DOCUMENT AUDIT TRAIL (GENERAL) TABLE PARAMETERS COMMON.

INTEGER DR1AUG, DR2AUG

COMMON /TPAUG/ DR1AUG(1), DR2AUG(240), ID1AUG, ID2AUG, I1AUG5,
I1AUG6, L1AUG, L2AUG

DR1AUG HOLDS DATA RECORD 1 FROM AUG-TABLE.

DR2AUG HOLDS DATA RECORD 2 FROM AUG-TABLE.

ID1AUG IS RECORD ID OF DR1AUG.

ID2AUG IS RECORD ID OF DR2AUG.

I1AUG5 IS INDEX OF ELEMENT 5 OF DR1AUG.

I1AUG6 IS INDEX OF ELEMENT 6 OF DR1AUG.
 L1AUG IS LENGTH OF DR1AUG (IN CHARACTERS).
 L2AUG IS LENGTH OF DR2AUG (IN CHARACTERS).

JULY 22, 1977

TABLE PARAMETERS COMMON. FOR DIR-TABLE.

INTEGER DR1DIR, DR2DIR, DR3DIR

COMMON /TPDIRC/ DR1DIR(15), DR2DIR(12), DR3DIR(28), ID1DIR, ID2DIR,

ID3DIR, I1DIR9, I1DIRA, L1DIR, L2DIR, L3DIR

DR1DIR HOLDS DATA RECORD 1 FROM DIR-TABLE.

DR2DIR HOLDS DATA RECORD 2 FROM DIR-TABLE.

DR3DIR HOLDS DATA RECORD 3 FROM DIR-TABLE.

ID1DIR IS ID OF DATA RECORD 1 FROM DIR-TABLE.

ID2DIR IS ID OF DATA RECORD 2 FROM DIR-TABLE.

ID3DIR IS ID OF DATA RECORD 3 FROM DIR-TABLE.

I1DIR9 IS INDEX OF ELEMENT 9 IN DATA RECORD 1 OF DIR-TABLE.

I1DIRA IS INDEX OF ELEMENT 10 IN DATA RECORD 1 OF DIR-TABLE (A=10).

L1DIR IS LENGTH, IN CHARACTERS, OF DATA RECORD 1 OF DIR-TABLE.

L2DIR IS LENGTH, IN CHARACTERS, OF DATA RECORD 2 OF DIR-TABLE.

L3DIR IS LENGTH, IN CHARACTERS, OF DATA RECORD 3 OF DIR-TABLE.

JULY 22, 1977

TABLE PARAMETERS COMMON. FOR DRT-TABLE.

INTEGER DR1DRT

COMMON /TPDRTC/ DR1DRT(32), ID1DRT, L1DRT

DR1DRT HOLDS DATA RECORD 1 FROM DRT-TABLE.

ID1DRT IS ID OF DATA RECORD 1 FROM DRT-TABLE.

L1DRT IS LENGTH, IN CHARACTERS, OF DATA RECORD 1 OF DRT-TABLE.

TABLE PARAMETERS COMMON. FOR LC-TABLE.

INTEGER DR1LC

COMMON /TPLCC/ DR1LC(22), ID1LC, L1LC

DR1LC HOLDS DATA RECORD 1 FROM LC-TABLE.

ID1LC IS ID OF DATA RECORD 1 FROM LC-TABLE.

L1LC IS LENGTH, IN CHARACTERS, OF DATA RECORD 1 OF LC-TABLE.

DECEMBER 8, 1977

PUL-TABLE PARAMETERS COMMON.

INTEGER DR1PUL

COMMON /TPPULC/ DR1PUL(64), ID1PUL, I1PULD, L1PUL

DR1PUL(64) IS DATA RECORD 1 OF PUL-TABLE.

ID1PUL IS ID OF DATA RECORD 1 OF PUL-TABLE.

I1PULD IS INDEX OF ENTRY 13 IN DR1PUL

L1PUL IS LENGTH OF DR 11 = (I1PULD-1+NOOCN+5SPECS)*NCU

SEP 26, 1977

TABLE PARAMETER COMMON FOR TTF-TABLE.

INTEGER NDITTF, I1TTF

NDITTF = MAXIMUM NUMBER OF DIRECT ACCESS DATA RECORDS

I1TTF = FIRST OF THE SECONDARY RECORDS

```
COMMON /TPTTFC/ ID1TTF,DR1TTF(243),L1TTF
INTEGER SKPLN,RWCHR,COLMX,FORMT(6,40)
EQUIVALENCE (SKPLN,DR1TTF(1)), (RWCHR,DR1TTF(2))
EQUIVALENCE (COLMX,DR1TTF(3)), (FORMT(1,1),DR1TTF(4))
```

ID1TTF IS RECORD-ID OF DATA RECORD 1 OF TTF-TABLE.
 DR1TTF(243) IS USED TO HOLD CONTENTS OF DATA RECORD 1 OF TTF-TABLE.
 L1TTF IS LENGTH (IN CHARACTERS) OF DATA RECORD 1 OF TTF-TABLE.
 SKPLN IS NUMBER OF LINES TO BE SKIPPED BETWEEN ROWS OF TABLE.
 RWCHR IS CHARACTER TO BE USED IN PRINTING A SOLID ROW BETWEEN ROWS.
 COLMX IS NUMBER OF COLUMNS (MAXIMUM COLUMN-ID) FOR TABLE.
 FORMT(1,1) = PREFIX TYPE. (A=1,I=2,D=3,B=4,C=5,SP=NEGATIVE).
 FORMT(2,1) = PREFIX LENGTH.
 FORMT(3,1) AND FORMT(4,1) = DATA TYPE AND LENGTH.
 FORMT(5,1) AND FORMT(6,1) = SUFFIX TYPE AND LENGTH.
 FORMT SPECIFIES THE COLUMN FORMATS.

JULY 1979

```
COMMON /USAGEC/ MONITR , NDSUSG
```

```
MONITR      0= NO COMMAND MONITORING
             1=  COMMAND MONITORING
NDSUSG      DATA HANDLER DATASET NUMBER FOR  A1MONITR DATASET
```

JULY 22, 1977

```
INTEGER R80CF,W80CF,W12CF,RTMCF,WLNCF
COMMON /VFMT/
```

```
1 R80CF(3),W80CF(4),ISUB0C,LNCU,LDN1,LDN2,MCH(35),W12CF(6),ISU12C
```

```
2, ISUTMC,RTMCF(3),ISULNC,WLNCF(3)
```

R80CF= FORMAT STATEMENT TO READ 80 CTRS INTO POSITIONS 1-80

W80CF= FORMAT STATEMENT TO WRITE 80 CTRS ON ONE LINE

ISUB0C= NUMBER OF SUS TO CONTAIN 80 CTRS

LNCU=LITERAL FOR NCU

LDN1,LDN2 = LITERAL FOR FORMAT TO PRINT 12 CTRS/3A4, /

MCH IS NUMBERS IN CHARACTER FORM. MCH(1)=1H1; MCH(35)=2H35.

W12CF= FORMAT ST. TO WRT. 12CTRS STARTING AT LOCATION 1-120

USER MUST PLACE THE NUMBER OF COLUMNS TO SKIP

IN WORD 3 OF W12CF BY AN ASSIGNMENT FROM A VARIABLE

DEFINED IN A DATA STATEMENT. (OR USE THE ARRAY MC)

ISU12C= NUMBER OF SUS TO CONTAIN 12 CTRS

ISUTMC IS NUMBER OF WORDS TO STORE A TERMINAL LINE.

RTMCF IS FORMAT TO READ LINE FROM TERMINAL.

ISULNC IS 80/NCU (WHOLE WORDS NEEDED TO STORE AN OUTPUT LINE

FOR PRINTING ON A MAX. 80 CHAR DEVICE)

WLNCF IS FORMAT FOR ISULNC WORDS

JULY 24, 1979

```
COMMON /WIDOWC/ PFLAG
INTEGER PFLAG
```

```
PFLAG      SWITCH INDICATING
```

-1 = SET BEFORE CALL TO WIDOW PROGRAM. INIT. IN BLOCK DATA

0 = SET BY WIDOW PROGRAM TO MEAN NORMAL PROCESSING
 LINE

3-8

DT