NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.
Flight Design System-1
System Design Document

Executive Logic Flow - Program Design Language

Mission Planning and Analysis Division
December 1979

NASA
National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas
77-FM-18
Vol. IX

SHUTTLE PROGRAM

FLIGHT DESIGN SYSTEM-1
SYSTEM DESIGN DOCUMENT

EXECUTIVE LOGIC FLOW - PROGRAM DESIGN LANGUAGE

By Mission Analysis and Engineering
Federal Systems Division - Houston
IBM Corporation

JSC Task Monitor: Software Development Branch

Approved:
Elric N. McHenry, Chief
Software Development Branch

Approved:
Ronald L. Berry, Chief
Mission Planning and Analysis Division

Mission Planning and Analysis Division
National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas
December 1979
The Flight Design System-1 (FDS-1) is a pilot project to evaluate current concepts and to determine the hardware/software capability that will be required for the operational era to support Shuttle flight planning. This software system is being implemented on a Hewlett-Packard 21MX computer with a Daconics documentation system and will provide terminal-based interactive flight planning capability.

The System Design Document (SDD) for FDS-1 is the specification for and description of this hardware/software facility. The SDD is logically organized into 10 published volumes. This organization is presented in the accompanying table. The material in the early volumes is primarily presented from the user's point of view, whereas the latter material is software-developer oriented. The SDD will be published by volumes over a period of time, and various volumes will be updated and republished during the development of FDS-1.
FDS-1 SYSTEM DESIGN DOCUMENT

Volume I  Introduction, Overview, and User Interface
Volume II  Utility Processor Library
Volume III Processor Library
Volume IV  System Architecture and Executive
Volume V  Data Management and Data Base Documentation Support System
Volume VI Standards
Volume VII Utility Support Software
Volume VIII Build and Delivery Procedures, Software Development, Debug, and System Build Aids
Volume IX  Executive Logic Flow - Program Design Language
Volume X  Document Change Request Procedure and Submittal Form

*Combined as one volume with title: Volume III FDS-1 Processor Library
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1-1</td>
</tr>
<tr>
<td>2.0 COMMON FOR FDS EXECUTIVE</td>
<td>2-1</td>
</tr>
<tr>
<td>3.0 FDS EXECUTIVE MESSAGES</td>
<td>3-1</td>
</tr>
<tr>
<td>4.0 PDL LISTING PROGRAM</td>
<td>4-1</td>
</tr>
<tr>
<td>5.0 FDS EXECUTIVE DETAILED LOGIC FLOW</td>
<td>5-1</td>
</tr>
<tr>
<td>6.0 DETAILED LOGIC FLOW LISTING - PROGRAM EXECUTION</td>
<td>6-1</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The flight design system can be divided functionally into two major areas: the FDS Executive and the application processor. The requirements for and the architecture of the FDS Executive is presented in volume I and volume IV, respectively, of this document. Volume IX presents the detailed logic flow for the FDS Executive.
2.0 COMMON FOR FDS EXECUTIVE

Three commons (XE, XB, and XS) presented in this section are used throughout the FDS Executive.
COMMON XE(400), XB(1400), XS(2000)

XE - EXECUTIVE FIXED COMMON (GLOBAL)
XB - EXECUTIVE DYNAMIC BLOCK (SUBSA LEVEL DEPENDENT)
XS - EXECUTIVE SCRATCH SPACE (VOLATILE ACROSS ALL CALLS

TO FOS ROUTINES EXCEPT XR,...

********

XE CONTENTS

INTEGER
# CARPTG
# CLASNO
# COMPR
# FLGDS
# FRCNAM(3)
# RBCBUF(64)
# SEREND
# SUBSTA
# TKMLNG
# TOKEMS(32)
# XE
DIMENSION
# INTNAM(3)

EQUIVALENCE
# XE(0), AU
# XE(3), SAL
# XE(5), MASSTA
# XE(7), SEGNUM(1)
# XE(11), SEREND
# XE(13), INTNAM(1)
# XE(19), RBCPTR
# XE(86), JMLNG
# XE(159), EXEND
# XE(142), CARTRG
# XE(144), COMPR
# XE(145), COMBUF(1)

CARTRG - NUMBER OF THE DISK CARTRIDGE CONTAINING EXECUTIVE MASTER FILES
CLASNO - EXECUTIVE/MANAGER REQUEST BLOCK CLASS I/O NUMBER
COMBUF - TERMINAL COMMUNICATIONS OUTPUT BUFFER
(1) - NUMBER OF TOKENS IN BUFFER
(2) - NUMBER OF USED WORDS IN BUFFER
(3-256) - TOKENS REPRESENTING USER'S RESPONSE
(5) - PROTOTYPE OF TOKENS SHOWN ON PRINTOUT
EXTEND - SEQUENCE # WHERE EXECUTION IS TO END (RETURN TO$)
FLGS - EXECUTIVE FLAG WORD (U-OFF, S-ON)

$0-10 UNSURE
11 PROCESSOR ON-LINE DEBUG
12 MANAGER ON-LINE DEBUG
13 EXECUTIVE ON-LINE DEBUG
14 PRODUCE A DUMP ON ALL TERMINATIONS
15 - MANAGER REQUEST TRANSACTION TRACE FLAG

INTNAM - NAME OF INTERFACE TABLE INPUT TO INTERFACE TABLE EDITOR
147 1 CD  XB CONTENTS (DIRECTIVE LEVEL)
148 1 CD  INTEGER
150 1 CD  = DIRECT(50)
152 1 CD  EQUIVALENCE
153 1 CD  = (XB(1),NUMDIR),(XB(2),DIRECT(1))
154 1 CD  = (XB(151),BEGINNING OF DIRECTIVE DEPENDENT ALLOCATION)
155 1 CD  DIRECT - FIXED ORDER LIST OF FDS DIRECTIVES (FOUR CHARACTERS PER
156 1 CD  DIRECTIVE)
158 1 CD  NUMDIR - NUMBER OF DIRECTIVES ACTUALLY IN DIRECT
160 1 CD  *********
XB CONTENTS (EXECUTION CONTROL LEVEL)

**INTEGER**

* ASCEZ
* RESIND
* SEINO

**DIMENSION**

* LIBB(150)

**EQUIVALENCES**

* (XB(1), MOPRC2)  = (XB(2), LIBB(1))
* (XB(235), RESIND) = (XB(236), ASCEZ(1))
* (XB(249), SEINO) = (XB(251), SENTAB(1))

**ASCEZ** - SEQUENCE TABLE ENTRY IN ASCII TO PROMPT USER IN SEMI MODE ONLY

**CURIND** - CURRENT INDEX TO EXECUTING SEQUENCE ENTRY

**LIBB** - LIBRARY DIRECTORY PROCESSOR NAME TABLE

**MOPRC2** - NUMBER OF PROCESORS IN LIBB (SAME AS XE(143))

**RESIND** - INDEX OF RESET ENTRY WHEN RESET SEQUENCE 0 IS REQUESTED

**SEINO** - NUMBER OF ENTRIES IN SEQUENCE TABLE

**SENTAB** - SEQUENCE TABLE CURRENTLY BEING EXECUTED
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>251</td>
<td>XB CONTENTS (INTERFACE TABLE EDIT LEVEL)</td>
</tr>
<tr>
<td>252</td>
<td>INTEGER BITNO, DFLAG, INSTRING, PLINE, PROMPT(30), SFLAG</td>
</tr>
<tr>
<td>253</td>
<td>RENNAME, VERSION, XKRBLNG, WKBUFF(1300)</td>
</tr>
<tr>
<td>254</td>
<td>DIRECTION ISIZES(9), NEWTAG(3)</td>
</tr>
<tr>
<td>255</td>
<td>EQUIVALENCE</td>
</tr>
</tbody>
</table>
JSUB F - CURRENT SECOND SUBSCRIPT (OR 0) FOR THIS ARGUMENT
LENFF - NO. OF WORDS/ELEMENT FOR THIS ARGUMENT
LISTLU - LU TO WHICH PRINT SHOULD GO (USED WHEN
LIST DIRECTIVE CALLS XILSD OR XICHR)
LITDSP - DISPL. TO LITERAL DATA FOR THIS ARGUMENT
LITDW - INDEX IN WKBUF TO END OF LITERAL DATA
LITLEN - LENGTH OF LITERAL DATA AREA OF WKBUF
LITPR - INDEX IN WKBUF TO START OF LITERAL DATA
LITSZ - NUMBER OF ARRAY ELEMENTS FOR THIS ARGUMENT (LOGICAL SIZE)
LISTLG - FLAG USED TO DETERMINE ORIGIN OF A
CALL TO XILSD OR XICHR
= 0, CALLED FROM LIST DIRECTIVE (PER CENT PROMPT) TO LIST
AN INTERFACE TABLE
= 3, CALLED FROM INTERFACE TABLE EDITOR TO LIST ARGUMENT
DATA, PARAMETER OR INCOMPLETE INDICATORS
= 4, CALLED FROM LIST DIRECTIVE (PER CENT PROMPT) TO LIST
A DATA ELEMENT'S VALUE(S)
MODAV - PREVIOUS VALUE OF PRMTDM WHILE PRMTDM = 4 (CONTINUE)
MARG - INDEX IN WKBUF TO START OF SHORT PROMPTS
MDSTM - INDEX IN WKBUF TO BIT MASK WORD(S) FOR THIS ARGUMENT
WENSTAB - ASCII NAME OF TABLE BEING GENERATED
NOCTM - NO. OF BIT MASKS ASSOCIATED WITH THIS ARGUMENT
NUMARG - NO. OF ARGUMENTS IN THIS INT. TABLE
NUMDIR - NUMBER OF DIRECTIVES ACTUALLY IN DIRECT
PRMLEN - LENGTH (IN WORDS) OF PROMPT BUILT
PRMTDM - CURRENT PROMPTING MODE
= 1, PROMPT FOR INCOMPLETE ARGUMENTS (MISSING)
= 3, PROMPT WITH A LIST OF CURRENT VALUES
= 4, CONTINUATION OF SAME ARGUMENT
= 5, PROMPT WITH "=" (CREATE MODE)
PROMPT - ASCII ARRAY FOR PROMPT BUILT
SFLAG - FLAG SET TO VALUE OF SUBSCRIPT BIT FOR THIS ARGUMENT
VERSON - VERSION NO. OF THIS INTERFACE TABLE
WKBLNG - LENGTH OF WKBUF
WKBUF - WORKING BUFFER FOR INTERFACE TABLE BEING EDITED
ORGANIZED AS :

HEADER (7 WORDS)
ARGUMENT CHARACTERISTICS (NUMARG+7 WORDS)
ARGUMENT PROMPTS (NUMARG+3 WORDS)
LITERAL DATA AND PARAMETER SUBSCRIPTS (LITLEN WORDS)
3.0 FDS EXECUTIVE MESSAGES

The list of messages generated by the Executive are presented in this section.
FD$ ERROR MESSAGES

* ATTENTION FUNCTION

** XA01 MANAGER H/S TERMINATED - REPLY TO CONTINUE TERMINATION
** XA02 USER INITIATED INTERRUPT ENTER REQUEST - KILL(X),STATUS(S),RETURN( )
** XA03 ERROR, LU IS NOT SIGNED ON TO FD$
** XA04 FD$ ATTENTION FUNCTION TERMINATING
** XA05 FD$ MANAGER SIGNALED TO TERMINATE BOTTOM ASSOC TASK
** XA06 FD$ EXECUTIVE ACTIVE - NO ACTION TAKEN
** XA07 FD$ PROCESSOR 'NAME' SCHEDULED TO ABORT
** XA08 MANAGER WAITING FOR SYSTEM RESOURCES ... NO ACTION TAKEN

* XB BATCH JOB CREATION

* XC CONFIGURATION PROGRAMS
** XC01 LU 'NN' SIGNED ON TO FD$
** XC03 LU IS CURRENTLY USING ID 'ID' - SIGN ON REJECTED
** XC04 -LU 'LU' IS AN INVALID LU
** XC05 FD$ CURRENTLY AT MAX USER'S. CANNOT SIGN ON
** XC06 'LU' IS ALREADY SIGNED ON TO FD$
** XC07 CANNOT FIND 'NAME' ID - SIGN ON TERMINATED
** XC08 ENTER VALID USER ID (A-Z)

* XM MANAGER

** XM01 INVALID REQUEST II FROM 'NAME'
** XM02 FD$ SIGN OFF FOR LU 'LU'
** XM03 INVALID BACK CHAIN FOR CURRENT PROCESSOR
** XM04 'N' TRACKS NOT AVAILABLE FOR DWA
** XM05 NO DWA SPACE FOR 'NAME'
** XM06 NO AWA FOR DWA DIRECTORY
** XM07 I/O ERROR FOR DWA, STATUS = 'NNNNNN'
** XM08 TOO SEARCH ERROR, PHASE 3 COMPRESS

* XP PROCESSOR SERVICES

** XP01 TYPE OF RESPONSE DOES NOT MATCH TYPE REQUESTED
** XP02 DATA AREA OVERFLOW
** XP03 SUBSCRIPT OUT OF RANGE
** XP04 INVALID ENTRY AFTER SUBSCRIPT
** XP05 INVALID REPEAT ENTRY
** XP06 TOO MANY NESTED REPEATS
** XP07 INVALID SUBSCRIPT
** XP08 RESPONSE IS TOO LONG FOR BUFFER
** XP09 INVALID RESPONSE ON OR AFTER COLUMN NN
** XP10 PROCESSOR INITIALIZATION ERROR
** XP11 AWA ACCESS FAILURE FOR ......
** XP12 PARAMETER I/O INCONSISTENCY
** XP13 INVALID PARAMETER REQUEST
** XP14 INVALID ORDE FILE ATTRIBUTE(S)
** XP15 RETRIEVAL OF TOO MUCH DATA REQUESTED

* XV SYSTEM SERVICES

** XV01 'ID-NAME' NAMED PROGRAM MADE A PM REQUEST RESERVED FOR MANAGER
** XV02 'ID-NAME' NAMED PROGRAM NOT IN WAIT LIST FOR MANAGER REPY
** XV03 'ID-NAME' NAMLO PROGRAM HAS INVALID BACK CHAIN TO MANAGER
4.0 PDL LISTING PROGRAM

The detailed logic flow of the program that generates PDL listing is presented as follows.
PDL STRUCTURED LISTING PROGRAM

INPUT
80 COLUMN PDL IMAGES SUBJECT TO THE FOLLOWING CONVENTIONS:
* IN COLUMN 1 INDICATES PAGE EJECT AND THE FIRST TOKEN (6 OR LESS
CHARACTERS) IS REPRODUCED IN COLUMNS 127-132 OF OUTPUT LISTING
UNTIL NEXT . RECORD IS DETECTED. IF RECORD CONTAINS ONLY THE . THE
RECORD CONTINUES TO APPEAR IN THE
IDENTIFICATION COLUMNS OF THE OUTPUT
* IN COLUMN 1 INDICATES A COMMENT TO BE COPIED TO THE OUTPUT LISTING.
OTHER IN COLUMN 1 INDICATES PDL RECORD TO BE STRUCTURED BASED ON KEY
WORDS APPEARING AS FIRST NON-BLANK CHARACTERS (SEE LOGIC).

OUTPUT
132 COLUMN LISTING AS FOLLOWS
2-6 SEQUENCE NUMBER
8-11 STRUCTURE LEVEL NUMBER
13-125 STRUCTURED LISTING OF 80 COLUMN INPUT RECORDS
127-132 IDENTIFICATION
FOLLOWING THE LISTING IS A SYMBOL DEFINITION TABLE INDICATING THE
SEQUENCE NUMBER OF THE LINE CONTAINING EACH 'BEGIN NAME' AND
"LABEL:".

NOTES
USES FSTND & SORT1
1 BEGIN PDLIST
2 INITIALIZE SEQUENCE NUMBER, LEVEL AND DEFINITION TABLE COUNTER
3 DO UNTIL END-OF-FILE INPUT
4 READ RECORD
5 INCREMENT SEQUENCE NUMBER
6 IF COLUMN 1 = -
7 THEN
8 SET PAGE EJECT IN IMAGE
9 IF REMAINDER OF IMAGE IS NOT BLANK
10 THEN
11 SET ID TO CONTENTS OF FIRST NON-BLANK FIELD
12 ENDIF
13 ELSE
14 CLEAR LEVEL INCREMENT
15 IF COLUMN 1 NOT = -
16 THEN
17 IF FIRST CHARACTER = : - INDICATES :LABEL:
18 THEN
19 SET LINE SKIP IN IMAGE
20 GENERATE DEFINITION TABLE ENTRY FOR LABEL
21 ELSE
22 CALL FSTDND TO GET FIRST WORD OF PDL
23 LOOKUP FIRST WORD IN KEY WORD TABLE
24 KEY WORD TABLE CONTAINS
25 1 BEGIN - BEGIN SECTION INDICATOR
26 2 IF - SECTION INDICATOR
27 3 DO - SECTION INDICATOR
28 4 DFOR - SECTION INDICATOR
29 5 DOUNTI - SECTION INDICATOR
30 6 DOWNIL - SECTION INDICATOR
31 7 CASE - SECTION INDICATOR
32 8 START - SECTION INDICATOR
33 5 STARTS - SECTION INDICATOR
34 10 ELSE - SECTION SEPARATOR
35 11 THEN - SECTION SEPARATOR
36 12 EXIT - SECTION SEPARATOR
37 13 EXITIF - SECTION SEPARATOR
38 14 OR - SECTION SEPARATOR
39 15 ORELSE - SECTION SEPARATOR
40 16 ENDLODO - SECTION SEPARATOR
41 17 END - END OR END LOOP?
42 18 ENDF - SECTION TERMINATOR
43 19 ENDO - SECTION TERMINATOR
44 20 ENDCAS - SECTION TERMINATOR
45 21 ENDOSE - SECTION TERMINATOR
46 IF KEY WORD LOCATED
47 THEN
50 (:END:, :TERM:, :TERM:, :TERM:, :TERM:)
51 :BEGIN: GENERATE DEFINITION TABLE ENTRY FOR SECTION NAME
52 SET LEVEL INCREMENT = 1
53 :SECOND: SET LEVEL INCREMENT = 1
REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR
5.0 FDS EXECUTIVE DETAILED LOGIC FLOW

A directory listing the major programs and subroutines in alphabetical order is presented initially. The detailed logic flow of each then follows in alphabetical order.
* NAME DEFINITION

1  EXECUTION SEGMENT INTERFACE ROUTINE
2  EXECUTION SEGMENT INTERFACE ROUTINE
3  EXECUTION SEGMENT INTERFACE ROUTINE
4  EXECUTION SEGMENT INTERFACE ROUTINE
5  EXECUTION SEGMENT INTERFACE ROUTINE
6  EXECUTION SEGMENT INTERFACE ROUTINE
7  EXECUTION SEGMENT INTERFACE ROUTINE
8  EXECUTION SEGMENT INTERFACE ROUTINE
9  EXECUTION SEGMENT INTERFACE ROUTINE
10 EXECUTION SEGMENT INTERFACE ROUTINE
11 EXECUTION SEGMENT INTERFACE ROUTINE
12 EXECUTION SEGMENT INTERFACE ROUTINE
13 EXECUTION SEGMENT INTERFACE ROUTINE
14 EXECUTION SEGMENT INTERFACE ROUTINE
15 EXECUTION SEGMENT INTERFACE ROUTINE
16 EXECUTION SEGMENT INTERFACE ROUTINE
17 EXECUTION SEGMENT INTERFACE ROUTINE
18 EXECUTION SEGMENT INTERFACE ROUTINE
19 EXECUTION SEGMENT INTERFACE ROUTINE
20 EXECUTION SEGMENT INTERFACE ROUTINE
21 EXECUTION SEGMENT INTERFACE ROUTINE
22 EXECUTION SEGMENT INTERFACE ROUTINE
23 EXECUTION SEGMENT INTERFACE ROUTINE
24 EXECUTION SEGMENT INTERFACE ROUTINE
25 EXECUTION SEGMENT INTERFACE ROUTINE
26 EXECUTION SEGMENT INTERFACE ROUTINE
27 EXECUTION SEGMENT INTERFACE ROUTINE
28 EXECUTION SEGMENT INTERFACE ROUTINE
29 EXECUTION SEGMENT INTERFACE ROUTINE
30 EXECUTION SEGMENT INTERFACE ROUTINE
31 EXECUTION SEGMENT INTERFACE ROUTINE
32 EXECUTION SEGMENT INTERFACE ROUTINE
33 EXECUTION SEGMENT INTERFACE ROUTINE
34 EXECUTION SEGMENT INTERFACE ROUTINE
35 EXECUTION SEGMENT INTERFACE ROUTINE
36 EXECUTION SEGMENT INTERFACE ROUTINE
37 EXECUTION SEGMENT INTERFACE ROUTINE
38 EXECUTION SEGMENT INTERFACE ROUTINE
39 EXECUTION SEGMENT INTERFACE ROUTINE
40 EXECUTION SEGMENT INTERFACE ROUTINE
41 EXECUTION SEGMENT INTERFACE ROUTINE
42 EXECUTION SEGMENT INTERFACE ROUTINE
43 EXECUTION SEGMENT INTERFACE ROUTINE
44 EXECUTION SEGMENT INTERFACE ROUTINE
45 EXECUTION SEGMENT INTERFACE ROUTINE
46 EXECUTION SEGMENT INTERFACE ROUTINE
47 EXECUTION SEGMENT INTERFACE ROUTINE
48 EXECUTION SEGMENT INTERFACE ROUTINE
49 EXECUTION SEGMENT INTERFACE ROUTINE
50 EXECUTION SEGMENT INTERFACE ROUTINE
51 EXECUTION SEGMENT INTERFACE ROUTINE
52 EXECUTION SEGMENT INTERFACE ROUTINE
53 EXECUTION SEGMENT INTERFACE ROUTINE
54 EXECUTION SEGMENT INTERFACE ROUTINE
55 EXECUTION SEGMENT INTERFACE ROUTINE
56 EXECUTION SEGMENT INTERFACE ROUTINE
57 EXECUTION SEGMENT INTERFACE ROUTINE
58 EXECUTION SEGMENT INTERFACE ROUTINE
59 EXECUTION SEGMENT INTERFACE ROUTINE
60 EXECUTION SEGMENT INTERFACE ROUTINE

**ATTENTION FUNCTION**

**ATTENTION TASK**

**BATCH JOB CREATION**

**CONFIGURATION PROGRAMS**

**FILE**

**DIRECTIVES**

1. EXEC
2. USR
3. EXEC
4. EXEC
5. EXEC
6. EXEC
7. EXEC
8. EXEC
9. EXEC
10 EXEC
11 EXEC
12 EXEC
13 EXEC
14 EXEC
15 EXEC
16 EXEC
17 EXEC
18 EXEC
19 EXEC
20 EXEC
21 EXEC
22 EXEC
23 EXEC
24 EXEC
25 EXEC
26 EXEC
27 EXEC
28 EXEC
29 EXEC
30 EXEC
31 EXEC
32 EXEC
33 EXEC
34 EXEC
35 EXEC
36 EXEC
37 EXEC
38 EXEC
39 EXEC
40 EXEC
41 EXEC
42 EXEC
43 EXEC
44 EXEC
45 EXEC
46 EXEC
47 EXEC
48 EXEC
49 EXEC
50 EXEC
51 EXEC
52 EXEC
53 EXEC
54 EXEC
55 EXEC
56 EXEC
57 EXEC
58 EXEC
59 EXEC
60 EXEC
179 1 * XXSEM  SEMI - AUTOMATIC NODE  DIRECT
180 1 * XXSTO  STORE SEQUENCE TABLE IN ASERTAB  DIRECT
181 1 * XXTRP  TEMPORARY EXECUTION OF ONE ENTRY WITH BINTAB  DIRECT
182 1 *  UTILITY PROCESSORS  DIRECT
183 1 * ASSGN  ASSIGN PROCESSOR  DIRECT
184 1 * DBDSP  DATA BOX DISPLAY PROCESSOR  DIRECT
185 1 * DEFIN  DEFINE PROCESSOR  DIRECT
186 1 * DO  CONDITIONAL LOOP IN SEQUENCE TABLE  DIRECT
187 1 * ELSE  EXECUTION POINT FOR FALSE IF CONDITION  DIRECT
188 1 * ENDEIF  TERMINATES AN IF STRUCTURE  DIRECT
189 1 * ENDDO  TERMINATES A DO LOOP STRUCTURE  DIRECT
190 1 * ENDSCE  END SCAN PROCESSOR  DIRECT
191 1 * IF  CONDITIONAL EXECUTION OF SEQUENCE TABLE ENTRIES  DIRECT
192 1 * SCAN  SCAN PROCESSOR  DIRECT
193 1 * XICHR  CHARACTER OBJECT STORE FOR ASSGN  DIRECT
194 1 * XIDFT  FIND ANY TOKEN IN A SYMBOLIC STRING  DIRECT
195 1 * XIDIN  DATA BOX DISPLAY INPUT PROCESSOR  DIRECT
196 1 * XIDMK  DATA BOX DISPLAY CONSTRAINT MASKER  DIRECT
197 1 * XIDOT  DATA BOX DISPLAY OUTPUT ROUTINE  DIRECT
198 1 * XIDP1  DATA BOX DISPLAY PASS 1 PROCESSOR  DIRECT
199 1 * XIDP2  DATA BOX DISPLAY PASS 2 PROCESSOR  DIRECT
200 1 * XIDPL  PERFORMS EVALUATION BETWEEN TWO REAL NUMBERS  DIRECT
201 1 * XIFCL  FIND PROCESSOR CLASS NUMBER  DIRECT
202 1 * XIFMC  FUNCTIONAL OPERATIONS FOR ASSGN  DIRECT
203 1 * XIFMLE  FUNCTIONAL OPERATIONS FOR ASSGN  DIRECT
204 1 * XIFRE  FREE OBJECT STORE FOR ASSGN  DIRECT
205 1 * XIFRD  FIXED OBJECT STORE FOR ASSGN  DIRECT
206 1 * XIZSP  REMOVE DUPL. BLANKS & BLANK FILL  DIRECT
207 1 * XILS  SYMBOLIC STRING SYNTAX ERROR LISTER  DIRECT
208 1 * XIMSG  FRS PROCESSOR MESSAGE ROUTINE  DIRECT
209 1 * XIOR  MATH OPERATIONS FOR ASSGN  DIRECT
210 1 * XIPS  DATA CONVERSION AND STORAGE FOR ASSGN  DIRECT
211 1 * XIPS1  PASS 1 SUBROUTINE FOR ASSGN PROCESSOR  DIRECT
212 1 * XIPS2  PASS 2 SUBROUTINE FOR ASSGN PROCESSOR  DIRECT
213 1 * XNUM  DATA RETRIEVAL FOR ASSGN  DIRECT
214 1 * XISCH  SEARCHES SEQUENCE TABLE FOR IF STRUCTURES  DIRECT
215 1 * XSYM  SYMBOL TABLE INTERFACE FOR ASSGN  DIRECT
216 1 * XSTTY  SYMBOL TABLE MAINTENANCE  DIRECT
SAVE ESR ADDRESS(IN BREG ON ENTRY)
CALL ENLU(BREG) GET LU IN ASCII & BINARY
STARTSEARCH UNTIL LAST STATUS TABLE ENTRY
EXITIF STBLU EN LU
SET STB ENTRY ADDRESS
ENDLOOP
SET STB ENTRY TO ZERO
ENDSEARCH
IF STB ENTRY FOUND, THEN
GET MANAGER'S ID ADDRESS(STMG)
IF MANAGER IS DORMANT, THEN
WRITE "**XAOO- MANAGER HAS TERMINATED;
REPLY TO CONTINUE TERMINATION:
READ(LU) ** WAIT FOR REPLY **
LOCK ON THE FDS TABLE RESOURCE
CALL SLIBM DISABLE
IF STB(EXECUTIVE ADDRESS NE 0, THEN
IF STAT(CURRENT) NE STAT, THEN
IF CURRENT AT IS NOT DORMANT AND BACK CHAIN POINTS TO OLD XMCX, THEN
FIND BOTTOM AT
DO UNTIL NEXT-AT .EQ. STMG(MANAGER)
CALCULATE NEXT-AT FROM BOTTOM'S FATHER ID NUMBER
CLEAR BOTTOM'S WAIT BIT & FATHER ID NUMBER
CLEAR NEXT-AT'S PARH ONE(PI)
CALL SLIBM ENABLE
CALL MESS 'OFFBUTTON'
CALL SLIBM DISABLE
SET BOTTOM TO NEXT-AT
ENDO
ENDIF
ENDIF
CALL SLIBM MAKE EXEC DORMANT
CLEAR EXEC'S ID & STBEX
ENDIF
DECREMENT NUMBER ACTIVE(STBAC)
GET ESR ADDRESS
RESTORE INTERRUPT HANDLER(FROM STBER)
CLEAR STBER
CLEAR MANAGER'S ID, STMG, & STBEX
ENABLE....(VIA A JMP TO EXIT(DISPATCHER))
RELEASE EXEC'S AND PROCESSOR'S CLASS NUMBERS
CLEAR LOCK ON FDS TABLE
ELSE ** MANAGER IS STILL ALIVE **
WRITE "**XAOO- USER INITIATED INTERRUPT?
WRITE 'ENTER REQUEST- KILL(S), STATUS(S), OR RETURN(BLANK)'
READ (LU) REQUEST
IF REQUEST IS KILL OR S, THEN
PERFORM XAKILL
ELSE
IF REQUEST IS STATUS OR S, THEN
PERFORM XASTAT
ENDIF
ENDIF
ELSE
WRITE "**XAOO ERROR LU IS NOT SIGNED-ON TO FDS"
ENDIF
WRITE '***XAOO FDS ATTENTION FUNCTION TERMINATING'
END
77 1:GO TO XSTAT
78 2:PRODUCE A DFS STATUS REPORT
79 3:CALL SLIBR - DISABLE
80 4:GET CURRENT-TIME FROM STIME
81 5:GET MANAGER'S ADDRESS FROM STMBG
82 6:MOVE NAME, STATUS, PARTITION, & PRIORITY
83 7:GET EXECUTIVE'S ADDRESS FROM STMBG
84 8:MOVE NAME, STATUS, PARTITION, & PRIORITY
85 9:GET CURRENT AT FROM STBAT
86 10:MOVE NAME, STATUS, PARTITION, & PRIORITY
87 11:PERFORM XABTM(CURRENT) FIND BOTTOM AT
88 12:SET BOTTOM TO CURRENT
89 13:GO WHILE FATHER-ID NE ZERO
90 14:SET FATHER-ID FROM CURRENT
91 15:CALCULATE NEXT
92 16:IF MAX ENTRIES HAVE NOT BEEN PROCESSED, THEN USE NEXT TO
93 17:MOVE NAME, STATUS, PARTITION, & PRIORITY
94 18:ENDIF
95 19:IF NEXT IS THE MANAGER, THEN
96 20:SET CURRENT AS TOP
97 21:ENDIF
98 22:SET CURRENT TO NEXT
99 23:ENDDO
100 24:IF TOP EQ ZERO, THEN
101 25:USE TOP TO MOVE NAME, PARTITION, & PRIORITY
102 26:SET STATUS TO 'IN USE' OCTAL 17
103 27:ENDIF
104 28:CALL SLIBR - ENABLE
105 29:WRITE FIRST SET OF HEADERS
106 30:SET TOP AS REPORT DATA
107 31:WRITE REPORT LINE
108 32:SET MANAGER AS REPORT DATA
109 33:WRITE REPORT LINE
110 34:SET EXECUTIVE AS REPORT DATA
111 35:WRITE REPORT LINE
112 36:SET CURRENT AS REPORT DATA
113 37:WRITE REPORT LINE
114 38:WRITE INTERMEDIATE HEADERS
115 39:DO UNTIL MAX ENTRIES OR NO MORE DATA
116 40:WRITE REPORT LINE
117 41:SET NEXT REPORT DATA
118 42:ENDDO
119 43:END XSTAT
120 44:SAMPLE REPORT ****************************
121 45:FD STATISM FOR LU 10 HH:MM:SS 360
122 46:NAME PRIOR PART# STATUS
123 47:TOP AT-  PROCB 922 3 GENERAL WAIT
124 48:MANAGER- XNPKN 40 3 GENERAL WAIT
125 49:EXECUTIVE- XEEKH 80 3 GENERAL WAIT
126 50:CURRENT AT- PROCB 11311 4 GENERAL WAIT
127 51:BACK CHAIN (UP TO 8) FROM BOTTOM VIA FATHER-ID
128 52:PROCB 32767 6 DISC ALLOCATE SUSPEND
129 53:PROCB 11311 4 GENERAL WAIT
130 54:PROCB 2060 3 GENERAL WAIT
131 55:PROCB 845 6 GENERAL WAIT
140 1 BEGIN XKILL
141 2 * TERMINATE CURRENT FDS FUNCTION
142 3 * IF MANAGER IS ACTIVE-SET FLAG FOR SEQUENCE TERMINATION
143 4 * ON NEXT RETURN VIA A PM.
144 5 * IF THE EXEC IS ACTIVE-DO NOTHING
145 6 * IF A PROCESSOR IS ACTIVE-USE RTE MESS TO OFF THE PROCESSOR
146 7 * SET MANAGER'S ID ADDRESS (STBMG)
147 8 * IF STATUS OF MANAGER IS NOT WAIT, THEN
148 9 * SET TERMINATE FLAG IN STB-ENTRY
149 10 WRITE "***XAO6 FDS MANAGER SIGNALED TO TERMINATE SEQUENCE"
150 11 ELSE
151 12 IF CURRENT(STBAT) EQ EXEC(STBEX), THEN
152 13 WRITE "***XAO6 FDS EXECUTIVE ACTIVE; NO ACTION TAKEN"
153 14 ELSE
154 15 IF CURRENT(STBAT) NE EXEC(STBEX), THEN
155 16 PERFORM XABTM(CURRENT) FIND BOTTOM AT
156 17 IF BOTTOM AT IS D.RTR OR SNMP THEN
157 18 WRITE "***XAO6 MANAGER IS WAITING FOR SYSTEM RESOURCES...NO ACTION TAKEN."
158 19 EXIT XAKIL
159 20 ELSE
160 21 IF RETURNED BOTTOM IS MANAGER THEN
161 22 IF MANAGER IS NOT WAITING ON A PROGRAM THEN
162 23 WRITE "***XAO6 MANAGER WAITING FOR SYSTEM RESOURCES...NO ACTION TAKEN."
163 24 EXIT XAKIL
164 25 ELSE
165 26 SET RETURN PARAMETER TO PROCESSOR ABENDED
166 27 INCREMENT MANAGER SUSPEND ADDRESS PAST SCHEDULE OF PROCESSOR
167 28 CALL BLIST TO REACTIVATE MANAGER
168 29 IF RETURNED BOTTOM WAS NOT MANAGER THEN
169 30 WRITE "**XAO7 FDS PROCESSOR 'NAME' SCHEDULED TO ABORT."
170 31 IF RETURNED BOTTOM WAS MANAGER THEN
171 32 SET NAME IN 'OFF' COMMAND
172 33 CALL MESS TO 'OFF' THE PROCESSOR
173 34 ENDIF
174 35 ENDIF
175 36 ENDIF
176 37 ENDIF
177 1 END XAKILL
179 1 BEGIN XABTM FIND BOTTOM AT
180 2 DO WHILE CURRENT IS IN GENERAL WAIT,
181 3 AND WAIT POINTER(P) HAS A SON ADDRESS,
182 4 AND SONS FATHER ID POINTS TO CURRENT
183 5 SET SON AS CURRENT
184 6 ENDDO
185 7 SET BOTTOM AS CURRENT
186 1 END XABTM
1 BEGIN XCNFG
2  
3 2 *01 FDS CONFIGURATION MANAGER
4 2 *01 INITIATES AN FDS SYSTEM FOR A REQUESTED TERMINAL OR
5 2 *01 TERMINATES AN FDS SYSTEM FOR A REQUESTED TERMINAL
6 2 *01 INITIATED VIA
7 2 *01 RU,FDS,LU,LU_ID,CNA SIZE,PARMS,OPTIONS
8 2 *02 INPUTS
9 2 *02 LOGICAL UNIT(LU) FOR THE REQUESTED TERMINAL,
10 2 *02 A PARM TO DENOTE INITIATION OR TERMINATION
11 2 *02 A ONE CHARACTER USERID
12 2 *02 A DEBUG OPTION INDICATOR
13 2 *02 THE NUMBER OF DNA TRACKS
14 2 *03 OUTPUTS
15 2 *03 INITIATION-
16 2 *03 A BLANK ID-SEGMENT WILL BE CONSTRUCTED FOR THE FDS MANAGER,
17 2 *03 AND FDS EXECUTIVE
18 2 *03 THE ERT FOR THE REQUESTED LU WILL BE CONNECTED
19 2 *03 TO THE FDS ATTENTION TASK
20 2 *03 THE FDS TABLE(FDSTAB) IN RESIDENT LIBRARY ROUTINE
21 2 *03 WILL BE UPDATED TO REFLECT THE INITIATION
22 2 *03 TERMINATION-
23 2 *03 THE FDS ID SEGMENTS WILL BE RETURNED TO BLANK STATUS
24 2 *03 THE ERT FOR THE LU WILL BE REINSTATED
25 2 *03 THE FDS TABLE(FDSTAB) IN RESIDENT LIBRARY ROUTINE
26 2 *03 WILL BE UPDATED FOR THE TERMINATION
27  
28 REPRODUCIBILITY OF THE
29 ORIGINAL PAGE IS POOR
29 2 * PART=LU, ID, DWA SIZE, PARM(ON OR OFF), OPTIONS
30 2 * SWITCH INPUT PARS AROUND SO THAT
31 2 * NOW PARM=LU,P2(ON OR OFF), ID, OPTS, DWA SIZE
32 2 * FOR COMPATABILITY TO BUILD 1.
33 2 CALL AMPAR(PARMS)
34 2 IF LU IS .LT. 0, OR
35 3 .GT. LUMAX(1653), OR
36 3 .EQ. 6(PRINTER), OR
37 3 THE DRIVER IS .NE. 0 OR 5, THEN
38 3 ISSUE MESSAGE "***XCO4 'LU' IS AN INVALID LU"
39 ELSE
40 4 IF PARM P2 IS OFF THEN
41 4 1 PERFORM XCOFF SIGN OFF
42 4 ELSE
43 4 2 PERFORM XCON SIGN ON
44 3 ENDIF
45 2 ENDIF
46 2 XCEXT
47 2 CALL EXEC PROGRAM TERMINATION
48 1 END XCMFG
1 BEGIN XCON
2 * SIGN ON A USER TO FDS
3  DO UNTIL VAILD USER ID (P3)
4    IF ID NOT A - 2 THEN
5      WRITE 'XCON ENTER VALID ID (A - 2)' 
6      READ RESPONSE
7  ENDIF
8  ENDDO
9  IF USER ID IS BEING USED THEN
10     WRITE "***XCON LU 'LU' IS CURRENTLY USING ID 'ID'- SIGN ON REJECTED"
11  EXIT :XCETA
12  ENDFD
13  IF FDS RESOURCE NUMBER NOT DEFINED, THEN
14     CALL RNRG (GLOBAL ALLOCATE, LOCAL SET)
15  ELSE
16     CALL RNRG (LOCAL SET)
17  ENDFD
18  IF NUMBER SIGNED ON (STBNK), THEN
19     ISSUE MESSAGE "***XCON FDS CURRENTLY AT MAX USER'S.
20  ELSE
21     DO FOR STBNK(NUMBER OF FDS ENTRIES)
22        IF ENTRY'S LU (STBNK) .EQ. REQUESTING LU(P1) THEN
23           ISSUE MESSAGE "***XCON 'LU' IS ALREADY SIGN ON TO FDS"
24           EXIT :XCETA
25        ELSE
26           IF THIS ENTRY IS AVAILABLE, THEN
27              SET AS CURRENT-ENTRY-ADDRESS
28          ENDFD
29        ENDFD
30     ENDDO
31     BECOME PRIVLEGED & DISABLED
32     CALL SLD
33  START-SEARCH WHILE NUMBER-FOUND .LT. NUMBER-NEEDED
34  SEARCH ID-SEGMENTS USING KEYS(V(1657)
35     IF XEXEC NOT FOUND AND THIS ID .EQ. XEXEC, THEN
36           SET ID ADDRESS OF XEXEC
37           INCREMENT NUMBER-FOUND
38     ELSE
39           IF XMGR NOT FOUND AND THIS ID .EQ. XMGR, THEN
40              SET ID ADDRESS OF XMGR
41              INCREMENT NUMBER-FOUND
42           ELSE
43              IF XATT NOT FOUND AND THIS ID .EQ. XATT, THEN
44                  SET ID ADDRESS OF XATT
45                  INCREMENT NUMBER-FOUND
46           ELSE
47                  IF FIRST-BLANK NOT FOUND AND THIS IS A BLANK ID, THEN
48                      SET ID ADDRESS OF FIRST-BLANK
49                      INCREMENT NUMBER-FOUND
50                  ELSE
51                      IF SECOND-BLANK NOT FOUND AND THIS IS A BLANK ID, THEN
52                          SET ID ADDRESS OF SECOND-BLANK
53                          INCREMENT COUNT
54                      ENDFD
55          ENDFD
56     ENDFD
57  WRITE ENDFD
58  ENDIF
59  ENDFD
60  EXIT IF THERE ARE NO MORE IDS
CALL BLRX, ENABLE
ISSUE MESSAGES "***XCO1 REPORTED THAT NAME*ID*SIGNON TERMINATED"
ENDLOOP

SET LU INQ STBLU
SET LU IN ASCII INTO STBLA
SET USER'S ID INTO STBD
SET ADDRESS OF FIRST-BLANK INTO STBLN
SET ADDRESS OF SECOND-BLANK INTO STBLX
INCREMENT ACTIVE COUNT(STBAC)
BUILD XPGMN & XEXMN
MOVE PRIORITY THRU DISC ADDRESS FROM XNCR TO FIRST-BLANK
TURN ON TN BIT
SET NAME TO XPGMN
MOVE PRIORITY THRU DISC ADDRESS FROM XEXEC TO SECOND-BLANK
TURN ON TN BIT
SET NAME TO XEXMN
LINK ATTENTION FUNCTION TO THE USER
DO FOR ANY TERMINAL EXCEPT SYSTEM'S CONSOLE
CALCULATE ERT OVERLAY
SAVE ERT VALUE IN STBD
SET ID ADDRESS OF XATTN INTO ERT
ENDIF
GET INPUT PARMs INTO ID OF XPGMN
SCHEDULE XPGMN VIA BLIST
CALL BLIST
CALL BLRX, ENABLE
IF FDS HAS A FATHER, THEN
CALL MESS 'OFF,FATHER'
ENDIF
ISSUE MESSAGE "***XCO1 REPORTED NAME*ID*SIGNON TERMINATED"
ENDSEARCH

:*XCTA
CALL RNRM (LOCAL CLEAR)
ENDIF
END
CALL EXEC TERMINATE
END XCON
Reproduction of the original page is poor.
*FORTRAN CALLING PROCEDURE

1 CALL XDLD (XDCLD)

** XDCLD DETERMINES WHICH OF THE FOLLOWING DIRECTIVES WAS REQUESTED AND CALLS THE APPROPRIATE HANDLER SUBROUTINE

1 INPUT

2 XE COMMON - MASSTA (BITS 10-13 CONTAIN A 0 INDEX INTO A LIST OF DIRECTIVES)

3

** INTERNAL VARIABLES

4 LIST - ORDERED LIST OF APPROPRIATE HANDLER ADDRESSES

5

** NOTES

6 USES .ENTR, XDST, XERMT

7 XDCLD IS DESIGNED TO BE THE MAIN ROUTINE FOR THE OVERLAY SEGMENT CONTAINING THE REFERENCED DIRECTIVES

8

**

9 BEGIN XDCLD

10 EXTRACT DIRECTIVE INDEX FROM MASSTA

11 CASE (:LIST:) INDEX

12 :LIST: CALL XDST

13 ENDCASE

14 CALL XERMT TO RETURN FROM SEGMENT

15 END XDCLD
40 1 *00 FORTRAN CALLING PROCEDURE
41 1 *00 CALL XELS (XDCLF)
42 1 *00
43 1 *00
44 1 *01 XDCLF DETERMINES WHICH OF THE FOLLOWING DIRECTIVES WAS
45 1 *01 REQUESTED AND CALLS THE APPROPRIATE HANDLER SUBROUTINE
46 1 *01 TQC
47 1 *01 SAVE
48 1 *01 RECALL
49 1 *01 DELETE
50 1 *01 RENAME
51 1 *01 COPY
52 1 *01 CLEAR
53 1 *01 OFF
54 1 *01
55 1 *01
56 1 *02 INPUT
57 1 *02 XE COMMON - MASSTA (BITS 10-13 CONTAIN A 1-8 INDEX INTO A LIST OF
58 1 *02 DIRECTIVES)
59 1 *02
60 1 *02
61 1 *04 INTERNAL VARIABLES
62 1 *04 LIST - ORDERED LIST OF APPROPRIATE HANDLER ADDRESSES
63 1 *04
64 1 *04
65 1 *05 NOTES
67 1 *05 :EXECUTE, :EXECUTE
68 1 *05 XDCLF IS DESIGNED TO BE THE MAIN ROUTINE FOR THE OVERLAY SEGMENT
69 1 *05 CONTAINING THE REFERENCED DIRECTIVES
70 1 *05
71 1 *05
72 1 *05
73 1 *05
74 1 *05
75 1 *05
76 1 *05
77 1 BEGIN XDCLF
78 2 EXTRACT DIRECTIVE INDEX FROM MASSTA AND DECREMENT BY 1
80 3 :TQC: CALL XDTC
81 3 :SAVE: CALL XDSC
82 3 :RECALL: CALL XDREC
83 3 :DELETE: CALL XDDELE
84 3 :RENAME: CALL XDREN
85 3 :COPY: CALL XDOP
86 3 :CLEAR: CALL XDCLF
87 3 :OFF: CALL XDFOF
88 2 ENDCASE
89 2 CALL XERTN TO RETURN FROM SEGMENT
90 1 END XDCLF
**FORTRAN CALLING PROCEDURE**

**XDCLU DETERMINES WHICH OF THE FOLLOWING DIRECTIVES WAS**

**REQUESSTED AND CALLS THE APPROPRIATE HANDLER SUBROUTINE**

**STORAGE, RESTORE, UNLOAD, LOAD, BATCH**

**INPUT**

**EXTERNAL VARIABLES**

**LIST - ORDERED LIST OF APPROPRIATE HANDLER ADDRESSES**

**NOTES**

**USES XSTR, XDRES, XDUNL, XDLOA, XDBAT, XERTN**

**XDCLU IS DESIGNED TO BE THE MAIN ROUTINE FOR THE OVERLAY SEGMENT**

**BEGIN XDCLU**

**EXTRACT DIRECTIVE INDEX FROM MASSTA AND DECREMENT BY 7**

**CASE (:STOR, :REST, :UNLO, :LOAD, :BATCH) INDEX**

**:STOR: CALL XDSTO**

**:REST: CALL XDRES**

**:UNLO: CALL XDUNL**

**:LOAD: CALL XDLOA**

**:BATCH: CALL XDBAT**

**ENDCASE**

**CALL XERTN TO RETURN FROM SEGMENT**

**END XDCLU**
FORTRAN CALLING PROCEDURE

CALL XDCL

******

XDCL PURGES ALL DATA FROM THE AWA. THE ONLY ELEMENTS REMAINING ARE PERMANENT SYSTEM TABLES AND DATA BASE FILES (CHAINS 1 & 6).

******

INPUT

XE COMMON - CTRLG, FLAG, LU
MANAGER - AWA HEADER AND TOC (SEE XMAMA)

******

OUTPUT

XE COMMON - REBUF, REOPTB, COMBUF-SCRATCH
XB COMMON - SCRATCH BEYOND XB(200)
XS COMMON - SCRATCH
MANAGER - COMMANDS TO PURGE AWA AND RESTORE CHAINS 0 AND 8

******

LOCAL VARIABLES

AVA - AWA HEADER AND TOC RECEIVED FROM MANAGER
MD1 - TOC CHAIN 1 HEAD
MD3 - TOC CHAIN 3 HEAD
MD6 - TOC CHAIN 6 HEAD
NEXT - INDEX TO NEXT ALLOCATABLE EIGHT WORD ENTRY IN QUEUE
NMAX - MAXIMUM SIZE OF QUEUE
ORG - BASE REFERENCE ADDRESS FOR AWA ADDRESSES
QUEUE - BUFFER FOR CONSTRUCTING AWA RESTORE REQUESTS FOR NON-DELETED ITEMS
TOCMAX - MAXIMUM SIZE OF TOC ACCOMMODABLE BY XDCL

******

NOTES

USES EXEC, IAND, PRTN, PURGE, XREQ, XRI6, XMMOV, XMMSG, XR06,
XRFN, XUDRG, XVAM

IN THE EVENT THAT A COMPACTED AWA TOC WILL NOT FIT IN THE ALLOCAT-
ED BUFFER SPACE AND ELEMENTS OF CHAIN 1 OR 6 EXTEND BEYOND IT, THE
CLEAR FUNCTION WILL NOT BE PERFORMED.

SHOULD THE RESTORATION OF CHAIN 1 OR 8 FAIL AFTER THE CLEAR
REQUEST HAS BEEN COMPLETED BY THE MANAGER, FBS WILL BE TERMINATED
IN ORDER TO PROTECT USER DISK FILES.
188 1 BEGIN XOCLE
189 2  RETRIEVE TOC
190 2  BUILD REQUEST TO CLEAR AWAY
191 2  DO UNTIL END OF PERMANENT SYSTEM TABLES CHAIN (CHAIN 1)
192 2  EXIT TO :ERR24: IF CHAIN POINTS BEYOND TOC BUFFER
193 2  EXIT TO :ERR48: IF RESTORATION REQUEST QUEUE IS FULL
194 3  BUILD REQUEST TO REALLOCATE TABLE
195 3  REQUEST MANAGER TO RETRIEVE TABLE (HOLD IN SAM)
196 3  BUILD REQUEST TO STORE TABLE INTO AWAY FROM SAM
197 2  ENDDO
198 2  DO UNTIL END OF DATABASE FILES CHAIN (CHAIN 8)
199 2  EXIT TO :ERR24: IF CHAIN POINTS BEYOND TOC BUFFER
200 2  EXIT TO :ERR48: IF RESTORATION REQUEST QUEUE IS FULL
201 2  BUILD REQUEST TO REALLOCATE TOC ENTRY
202 2  ENDDO
203 2  BUILD REQUEST TO TERMINATE LIST
204 2  DO UNTIL END OF DRDE CHAIN (CHAIN 3)
205 3  IF CHAIN POINTS BEYOND TOC BUFFER
206 3  THEN
207 4  OUTPUT XD13 'TOC TOO LARGE, DRDE PURGE INCOMPLETE'
208 4  EXIT PURGE LOOP
209 3  ENDF
210 3  PURGE FILE
211 2  ENDDO
212 2  DO FOR EACH BLOCK OF EIGHT REQUESTS
213 2  TRANSMIT BLOCK TO MANAGER
214 2  EXIT TO :ERR23: IF REQUESTS FAILED
215 2  ENDDO
216 1 EXIT XOCLE
217 2  :ERR23:
218 2  DO FROM FAILING REQUEST TO END OF LIST
219 3  IF REQUEST TO STORE
220 4  THEN
221 5  READ SAM TO FREE BUFFER AND CLASS NUMBER
222 5  ENDF
223 2  ENDDO
224 2  DO UNTIL END OF DATABASE FILE CHAIN (CHAIN 8)
225 3  IF FILE IS UTOD (TYPE 1)
226 4  THEN
227 5  CALL PURGE TO DELETE FILE
228 5  ENDF
229 2  ENDDO
230 2  PURGE ALL UTOD FILES
231 2  TERMINATE FDS WITH CLEAR FAILURE MESSAGE
232 2  :ERR48:
233 2  :ERR24:
234 2  DO FOR ALL STORE REQUESTS BUILT
235 3  READ SAM TO FREE BUFFER AND CLASS NUMBER
236 2  ENDDO
237 1 EXIT XOCLE WITH CLEAR FAILURE MESSAGE
238 1 END XOCLE
**FORTRAN CALLING PROCEDURE**

CALL XDCOP  

**INPUT**

COMMON XE - CARTRY, COMBUF, QVAL, REPTR, TOKENS  

**OUTPUT**

COMMON XE - COMPR, RERBUF  

**INTERNAL VARIABLES**

COMMON XE - (1) DATCLS: DATA CLASS CODE (STORED IN LEFT BYTE)
(2) DTYPE: TYPE CODE FOR DATA BASE FILES
(3) I: INDEX
(4) IDC: EXEC BUFFER
(148) ERR: FILE MANAGER ERROR RETURN
(149) INFG: ERROR MESSAGE NUMBERS
(150) FNAME: NEW FMGR FILE NAME
(153) WNAME: OLD FMGR FILE NAME
(154) WMNAME: NAME TO OLD NAME IN COMBUF
(155) WO NAME: NAME TO OLD NAME IN COMBUF
(156) TOCEN: TOC ENTRY FOR DATA BASE RENAME
(166) EO: EOF RETURN FROM FMGR

COMMON XB - (204) BUuff: USER BUFFER FOR EXEC I/O (1024)
(124A) FTYPE: FMGR FILE TYPE
(1245) PARM1: FIRST BYTE OF REPTR
(1246) PARM2: SECOND BYTE OF REPTR
(124J: PREFIX: PREFIX FOR FILE NAME
(1248) SCRTY: FILE SECURITY CODE (0 OR 8)
(1249) SIZE: FILE SIZE IN BLOCKS
(1251) IDXBY: EXEC BUFFER
BEGIN XDCOP
EXIT TO :SYNTAX: IF FIRST TOKENS ARE NOT "NAME"
SAVE POINTER TO CURRENT NAME
EXIT TO :SYNTAX: IF NEXT TOKEN IS HYPHEN, THEN
DECODE CLASS NAME
EXIT TO :CLASS: IF CLASS SPECIFIED IS NOT VALID (S,I,D,F,B)
ELSE
SET CLASS TO DATA ELEMENT
ENDIF
EXIT TO :SYNTAX: IF NEXT TOKENS ARE NOT "NAME"
SAVE POINTER TO NEW NAME
EXIT TO :SYNTAX: IF NEXT TOKEN IS NOT END-OF-MESSAGE
IF CLASS IS DATA BASE OR ORDE, THEN
EXIT IF NEW NAME IS MORE THAN 4 CHARACTERS :NAME ERR:
IF CLASS IS DATA BASE, THEN
EXIT TO :NAME ERR: IF NEW NAME STARTS WITH DOUBLE EXCLAMATION
ENDIF
BUILD AUA MANAGER REQUEST FOR TOC ENTRY FOR CURRENT NAME
BUILD AUA MANAGER REQUEST TO VERIFY NEW NAME
CALL XAEN
EXIT TO :TOCERR: IF RETURN INDICATES ERROR ON FIRST REQUEST
EXIT TO :TOCERR: IF NO ERROR RETURNED ON SECOND REQUEST
CALL EXEC TO GET TOC ENTRY
IF CLASS IS DATA BASE THEN
IF TYPE IS PDB, THEN
CALL XDBBA TO ADD NEW PDB TO LOG FILE (XPDB)
EXIT TO :FSER: IF FNMGR ERROR RETURNED
EXIT TO :TOCERR: IF NEW NAME IS DUPLICATE
EXIT TO :MAXERR: IF PDB MAX IS EXCEEDED
SET FILE PREFIX TO RIGHT BRACKET
ELSE
EXIT TO :INVLO: IF FILE IS NOB
SET FILE PREFIX TO *
ENDIF
SET FMGR FILE TYPE TO 1
SET SECURITY CODE TO 88
ELSE
SET FILE PREFIX TO /
SET SECURITY CODE = 0
SET FMGR FILE TYPE FROM TOC ENTRY
IF FILE IS TYPE 2, THEN
STORE RECORD LENGTH FROM TOC ENTRY
ENDIF
ENDIF
CALL XRFNM TO FORMAT NEW FILE NAME
CALL CREATE FOR NEW FILE
EXIT TO :FSER: IF FNMGR ERROR RETURNED
CALL XRFNM TO FORMAT OLD FILE NAME
CALL OPEN FOR OLD FILE
EXIT TO :FSER: IF FNMGR ERROR RETURNED
EXIT TO :TYPERR: IF TYPE IS NOT SAME AS IN TOC
DO UNTIL END-OF-FILE IS READ ON OLD FILE
CALL READF TO READ RECORD FROM OLD FILE
EXIT IF ERROR DETECTED TO :FSER
CALL WRITF TO WRITE RECORD TO NEW FILE
EXIT IF ERROR DETECTED TO :FSER
ENDIF
CALL close FOR OLD FILE
CALL close FOR NEW FILE
BUILD AND ISSUE AHA MANAGER REQUEST TO ALLOCATE TOC ENTRY

EXIT TO :TOCERR: IF ERROR IS INDICATED

ELSE

BUILD MANAGER REQUEST FOR CURRENT TOC ENTRY

BUILD MANAGER REQUEST FOR DATA RETRIEVAL

CALL XREQ

EXIT TO :TOCERR: IF ERROR IS INDICATED

CALL EXEC TO GET TOC ENTRY

BUILD MANAGER REQUEST TO ALLOCATE NEW TABLE

BUILD MANAGER REQUEST TO STORE TABLE

IF CLASS IS INTERFACE TABLE, THEN

CALL EXEC TO WRITE NEW TABLE NAME TO SAM

BUILD MANAGER REQUEST TO STORE NEW NAME IN TABLE

ENDIF

CALL XREQ

EXIT TO :TOCERR: IF ERROR IS INDICATED

ENDIF

EXIT XDOP

:SYNTAX: CALL XRMG -"SYNTAX ERROR ..." AND EXIT

:CLASS: CALL XRMG -"INVALID CLASS DESIGNATOR ..." AND EXIT

:NAMEERR: CALL XRMG -"NEW NAME IS INVALID ..." AND EXIT

:MAXERR: CALL XRMG -"AUTHORIZED LIMIT ..." AND EXIT

:MAXERR: CALL XRMG -"NOB CANNOT BE ..." AND EXIT

:FILERR: CALL XRMG -"FILE ACCESS ERROR ..." AND EXIT TO :END:

:TOCERR: CALL XRMG TO OUTPUT APPROPRIATE MESSAGE AND EXIT TO :END:

:TYPOERR: CALL XRMG -"INCONSISTENT FILE TYPE ..."

:END:

IF PDB HAS BEEN LOGGED IN XPD, THEN

CALL XDOPD TO DELETE PDB FROM XPD

ENDIF

IF A NEW FILE HAS BEEN BUILT, THEN

PURGE NEW FILE

CLOSE OLD FILE

ENDIF

END XDOP
FORTRAN CALLING PROCEDURES

CALL XODBX(NAME, IERR, ISIZE)
CALL XORBX(NAME, IERR)

********

XODBX, XORBX, AND XODDB PROVIDE THE ABILITY TO ADD AN MDB OR
PB TO MDB/PDB DIRECTORY (XPDB). TO VERIFY THAT AN MDB OR PDB IS
LOGGED IN XPDB AND RETRIEVE THE SIZE OF THAT MDB/PDB, AND TO
DELETE AN MDB OR PDB FROM XPDB, RESPECTIVELY

********

INPUT
NAME - 2 WORD INTEGER ARRAY CONTAINING THE QUALIFIED ASCII
MDB/PDB NAME
ISIZE - INTEGER WORD CONTAINING THE MDB/PDB SIZE IN BLOCKS (FOR
XODBX ONLY)

********

OUTPUT
IERR - INTEGER WORD CONTAINING RETURN CODE

<0 SMALL ERROR CODE RETURNED FOR XPDB ACCESS
0 NO ERRORS
1 DUPLICATE NAME OR NAME NOT FOUND, AS APPROPRIATE
2 PDB/PDB LIMIT EXCEEDED
3 ISIZE - INTEGER WORD CONTAINING THE MDB/PDB SIZE IN BLOCKS (FOR
XORBX ONLY)

********

COMMON USED

XE - RUAL
CARTDB
XS - XCO (WORDS 1-144)

********

NOTES

XRULK, XRLCK, .ENDB, OPEN, CLOSE, READF, AND WRITF ARE USED

WHEN WORKING WITH MDB'S, QUAL SHOULD BE SET TO 77 OCTAL

********
5-28

445 1 BEGIN XDDBA
446 2 STORE RETURN ADDRESS
447 2 CALL ENTER TO SET UP CALLING ARGUMENTS
448 2 CALL XLCK FOR EXCLUSIVE USE OF XDPB
449 2 CALL OPEN FOR EXCLUSIVE USE OF XDPB
450 1 EXIT TO :FILER: IF ERROR RETURNED
451 2 COMPUTE RECORD NUMBER FOR USER'S DIRECTORY (QUDL-77B)/2+1
452 2 CALL READ FOR RECORD COMPUTED
453 1 EXIT TO :FILER: IF ERROR RETURNED
454 2 DETERMINE PART OF RECORD TO BE USED
455 2 IF REQUEST IS FOR ADD, THEN
456 3 EXIT TO :MAXERR: IF CURRENT # OF ENTRIES + 1 > MAX ENTRIES
457 3 START SEARCH UNTIL ALL CURRENT ENTRIES ARE TESTED
458 3 EXIT TO :NAMEERR: IF ENTRY NAME MATCHES PARAMETER NAME
459 3 ENDOLOOP
460 3 ENDSSEARCH
461 3 STORE NAME AND SIZE IN ENTRY FOLLOWING LAST ENTRY
462 3 INCREMENT # OF CURRENT ENTRIES
463 ELSE
464 3 START SEARCH UNTIL ALL CURRENT ENTRIES, IF ANY, ARE TESTED
465 3 EXIT IF ENTRY NAME MATCHES PARAMETER NAME
466 3 ENDOLOOP
467 3 EXIT TO :NAMEERR:
468 3 ENDSSEARCH
469 3 IF REQUEST IS FOR VERIFY, THEN
470 4 STORE WORD 3 OF ENTRY IN ISIZE
471 4 ELSE
472 4 REPLACE ENTRY WITH LAST ENTRY
473 4 STORE ZEROS IN LAST ENTRY
474 4 DECREMENT # OF CURRENT ENTRIES
475 3 ENDF
476 2 ENDF
477 2 CALL WRITE TO WRITE RECORD TO XDPB
478 1 EXIT TO :RETURN:
479 2 :NAMEERR: SET IERR = 1 AND EXIT TO :RETURN:
480 2 :MAXERR: SET IERR = 2 AND EXIT TO :RETURN:
481 2 :FILERR: SET IERR = FNSA ERROR CODE
482 2 :RETURN: CALL CLOSE FOR XDPB
483 2 CALL XLCK TO RETURN RESOURCE #
484 1 END XDDBA
1 BEGIN XDELETE
2 DO WHILE END-OF-STATEMENT NOT REACHED PROCESSING EACH ELEMENT SPECIFIED
3 EXIT IF COMMA IS NOT NEXT LEXICAL ELEMENT :ERROR:
4 EXIT IF CLASS DESIGNATOR IS SPECIFIED, THEN
5 SET REQUESTED CLASS APPROPRIATELY (B, S, I, O, OR F)
6 ELSE
7 Exit REQUESTED CLASS TO BE (O)
8 ENDIF
9 EXIT IF DATA BASE TO BE DELETED, THEN
10 BUILD AND ISSUE AMA MANAGER REQUEST FOR TOC ENTRY
11 EXIT IF ELEMENT IS NOT A MASTER DATA BASE, THEN
12 EXIT IF ELEMENT IS A PERSONAL DATA BASE, THEN
13 CALL XPDB TO wipe THIS PDB FROM XPD
14 IF ERROR IS RETURNED, THEN
15 CALL XMSG - "FILE ACCESS ERROR #.. XPD"
16 SET ERROR FLAG
17 ENDIF
18 BUILD AND ISSUE AMA MANAGER REQUEST TO DELETE ELEMENT SPECIFIED
19 EXIT IF RETURN CODE INDICATES ELEMENT DOES NOT EXIST, THEN
20 CALL XMSG - "XXXXX NOT FOUND"
21 SET ERROR FLAG
22 ENDIF
23 EXIT IF CLASS IS DATA BASE (C), OR
24 CLASS IS XDDE (F), THEN
25 EXIT IF ERROR FLAG IS NOT SET, THEN
26 CALL XRMN TO CONSTRUCT FILE NAME
27 ISSUE R/E PURGE FOR THE FILE
28 IF RETURN CODE FROM PURGE, THEN
29 CALL XMSG - "PER ERROR XXXX XXXX"
30 ENDIF
31 EXIT IF
32 EXIT IF
33 ELSE
34 CALL XMSG - "... IS A MDB. NOT DELETED."
35 ENDIF
36 END DO
37 EXIT TO :RETURN:
38 :ERROR: CALL XMSG - "SYNTAX ERROR"
39 :RETURN:
40 1 END XDELETE
CALL XMIS (TOCLST)

XMIS PROCESSES A LIST OF ELEMENTS TO BE STORED/RESTORED BY
DECODING EACH ELEMENT AND FLAGGING IT IN TOCLST AS AN ELEMENT
TO BE STORED/RESTORED.

INPUTS IN CALLING SEQUENCE:

TOCLST - (INTEGER, 1200 WORDS) ADDRESS OF FIRST TOC ENTRY
OF A LIST OF POSSIBLE ELEMENTS' TOC ENTRIES.

INPUTS IN COMMON:

XB(151) ABLG, XB(201) NOTOC

OUTPUTS IN CALLING SEQUENCE:

TOCLST - (INTEGER, 1200 WORDS) LIST OF TOC ENTRIES; THOSE
THAT ARE TO BE STORED/RESTORED ARE FLAGGED.

OUTPUTS IN COMMON:

XB(151) ABLG, XB(157) TOTSIZ, XB(158) TOTWPD

INTERNAL XB COMMON USED:

XB(151) ABLG - (INTEGER, 1 WORD) ABORT FLAG
XB(152) EBLG - (INTEGER, 1 WORD) ERROR MESSAGE FLAG
XB(153) MSGNO - (INTEGER, 1 WORD) MESSAGE NUMBER
XB(157) TOTSIZ - (INTEGER, 1 WORD) TOTAL # BLOCKS OF DATA TO
BE STORED/RESTORED
XB(158) TOTWPD - (INTEGER, 1 WORD) TOTAL # WORDS OF DATA TO
BE STORED/RESTORED
XB(198) BECT - (INTEGER, 1 WORD) XTOM RETURN CODE
XB(199) BATCLS - (INTEGER, 1 WORD) CLASS OF DATA BEING SEARCHED
FOR IN LIST OF TOC ENTRIES
XB(200) HMPTR - (INTEGER, 1 WORD) POINTER TO NAME IN COMBUF
XB(201) NOTOC - (INTEGER, 1 WORD) NUMBER ENTRIES IN TOCLST

COMMON USED:

EQUIVALENCE

* (RE(13), LU )
* (RE(50), NASTMA), (RE(85), EOS )
* (RE(85), NAME ), (RE(92), HTPN )
* (RE(113), COMMA ), (RE(144), COMTRA)
* (RE(145), COMBUF)
BEGIN XDLIS
DO WHILE ERROR FLAG IS ON OR UNTIL RESPONSE IS CR
TURN ERRFLG OFF
:RTN1:
DO UNTIL EOS IS SENSED IN COMBUS
ERREXIT IF TOKEN IS NOT "NAME" TO :ERR1:
SAVE INDEX TO NAME FIELD
INCREMENT TO NEXT TOKEN
IF TOKEN IS A HYPHEN THEN
ERREXIT IF NEXT TOKEN IS NOT "NAME" TO :ERR1:
INCREMENT TO NEXT TOKEN
DECODE CLASS NAME (I, S, D, F)
ERREXIT IF CLASS SPECIFIED IS NOT VALID TO :ERR1:
SET CLASS TO CLASS SPECIFIED
ELSE
SET CLASS TO DATA ELEMENT
ENDIF
IF XDLIS CALLED FROM STORE THEN
ERREXIT IF PREFIX IS DOUBLE EXCLAMATION TO :ERR2:
ENDIF
ERREXIT IF NAME/CLASS ENTRY NOT FOUND IN TOC TO :ERR2:
CALL ARSET TO TURN STORE/RESTORE BIT ON
_INCREMENT TOTAL SIZE BY SIZE OF THIS ELEMENT
ENDDO
:RTN2:
IF ERROR FLAG IS ON THEN
CALL XCOM TO REPROMPT USER TO CONTINUE
ERREXIT IF RESPONSE IS X TO :ERR3:
ENDIF
EXIT XDLIS
:ERR1:
SET ERROR FLAG ON
CALL XRMSG TO DISPLAY SYNTAX ERROR
GO TO :RTN2:
:ERR2:
IF ERROR FLAG IS OFF THEN
TURN ERROR FLAG ON
CALL XRMSG TO DISPLAY NOT STORED/RESTORED MESSAGE
ENDIF
CALL EXEC TO DISPLAY ELEMENT NAME
GO TO :RTN1:
:ERR3:
SET ABLFG TO ABORT STORE/RESTORE OPERATION
END XDLIS
693 CDO   FORTRAN CALLING PROCEDURE
694 CDO
695 CDO   CALL XLST
696 CDO
697 CDO   ********
698 CDO
699 CDO   ********
700 CDO   PROCESS THE LIST DIRECTIVE TO LIST ALL TABLES AND/OR
701 CDO   DATA ELEMENTS SPECIFIED.
702 CDO
703 CDO   ********
704 CDO   ********
705 CDO   INPUT
706 CDO
707 CDO   COMMON XE = COMBUF, COMPTR, FLAGS, LU, TOKENS
708 CDO   ********
709 CDO   ********
710 CDO
711 CDO   OUTPUT
712 CDO
713 CDO   PRINT TO LOGICAL UNIT 6 OR TO LOGICAL UNIT 'LU'
714 CDO
715 CDO
716 CDO
717 CDO   INTERNAL VARIABLES
718 CDO
719 CDO   COMMON XE - BUFFER = PRINT LINE TO BE OUTPUT
720 CDO   BUFPTR = INDEX INTO BUFFER FOR NEXT ASCII DATA
721 CDO
722 CDO   ********
723 CDO   ********
724 CDO   COMMON XE = POINTR = CHAIN POINTER TO NEXT (OR 1ST) TOC
725 CDO
726 CDO
727 CDO
728 CDO   NOTES
729 CDO
730 CDO   USES ROUTINES - EXEC, XEINT, XILSD, XILST,
731 CDO
732 CDO
733 CDO
734 CDO   ********
1. BEGIN XDLST
2. IF DEVICE ID FIELD SPECIFIED, THEN
3. SET LU FOR LISTING AS INDICATED ON INRECTIVE
4. ELSE
5. SET LU FOR LISTING TO BE TERMINAL LU
6. ENDIF
7. DO UNTIL END-OF-STATEMENT IS REACHED
8. ERREXIT IF COMMA IS NOT SPECIFIED :ERROR9:
9. ERREXIT IF A NAME DOES NOT FOLLOW THE COMMA :ERROR9:
10. RETAIN NAME FOR XREM CALL
11. IF A CLASS DESIGNATOR IS SPECIFIED, THEN
12. SET CLASS (I, S, OR D) FOR XREM CALL
13. ELSE
14. USE DATA (I) CLASS IN XREM CALL
15. ENDF
16. CALL XREM TO RETRIEVE THIS TABLE OR DATA ELEMENT
17. ERREXIT IF NOT FOUND :ERROR9:
18. ERREXIT IF AWA SPACE NOT AVAILABLE FOR TABLE IN DMA :ERROR11:
19. CALL EXEC TO PERFORM CLASS READ OF DATA OR TABLE INTO
20. BOTTOM OF WORKING BUFFER
21. IF INTERFACE TABLE TO BE LISTED, THEN
22. CALL XMOV TO MOVE CHARACTERISTICS TO TOP OF WORKING BUFFER
23. READ SHORT PROMPTS FOR THIS PROCESSOR INTO WORKING BUFFER
24. CALL XEINT TO INITIALIZE LITERAL ENTRIES
25. INITIALIZE INTERFACE TABLE EDITOR COMMON FOR USE ITS LIST RTN.
26. INITIALIZE 'LISTFLG' TO INDICATE ENTIRE TABLE TO BE LISTED
27. CALL XLIST TO LIST THE INTERFACE TABLE
28. ELSE
29. IF SEQUENCE TABLE TO BE LISTED, THEN
30. CALL XMOV TO MOVE TABLE TO TOP OF WORKING BUFFER
31. CALL XLIST TO LIST SEQUENCE TABLE
32. ELSE
33. INITIALIZE INTERFACE TABLE EDITOR COMMON FOR USE ITS LIST RTN.
34. SET 'LISTFLG' TO INDICATE ONLY 1 DATA ELEMENT BEING LISTED
35. INITIALIZE PRINT BUFFER WITH NAME OF ELEMENT
36. CALL XLIST TO LIST THE DATA
37. ENDF
38. ENDF
39. INCREMENT TO NEXT TOKEN IN OPERAND LIST OF THE DIRECTIVE IMAGE
40. :ERROR10: CALL XRMNG -- '... NOT FOUND'
41. :ERROR11: CALL XRMNG -- '... CANNOT BE MOVE FROM DMA TO AWA'
42. ENDDO
43. EXIT TO :RETURN:
44. :ERROR9: CALL XRMNG -- 'SYNTAX ERROR'
45. :RETURN:
46. END XDLST
783 ♦ 1 CDDDDDDDDDD
784 ♦ 1 CDD
785 ♦ 1 CDO FORTRAN CALLING PROCEDURE
786 ♦ 1 CDO CALL XDFF
787 ♦ 1 CDD
788 ♦ 1 CDD
789 ♦ 1 CDD
790 ♦ 1 CDD XDFF CONFIRMS THE USER'S REQUEST FOR TERMINATION,
791 ♦ 1 CDD DELETES ALL ORDE AND UTDX FILES LOGGED IN THE ANA,
792 ♦ 1 CDD PERFORMS ABNORMAL TERMINATION, IF INDICATED, OR
793 ♦ 1 CDD RETURNS NORMAL PARS TO THE FDS MANAGER AND TERMINATES
794 ♦ 1 CDD Normally VIA RTE.
795 ♦ 1 CDD
796 ♦ 1 CDD
797 ♦ 1 CDD INPUT
798 ♦ 1 CDD
799 ♦ 1 CDD
800 ♦ 1 CDD COMMON XE - LU, FLAGS, QUAL, RERBUFF
801 ♦ 1 CDD
802 ♦ 1 CDD COMMON XB - ORG = ORIGIN ADDRESS OF ANA, USED TO CALCULATE
803 ♦ 1 CDD INDICES INTO 'ANA' FROM ADDRESS POINTERS
804 ♦ 1 CDD
805 ♦ 1 CDD OF TOC ENTRIES
806 ♦ 1 CDD
807 ♦ 1 CDD ANA = IMAGE OF ANA HEADER, CHAIN HEADS, AND
808 ♦ 1 CDD TOC RETRIEVED VIA XREQ
809 ♦ 1 CDD
810 ♦ 1 CDD INTERNAL VARIABLES
811 ♦ 1 CDD
812 ♦ 1 CDD
813 ♦ 1 CDD COMMON XS - POINTER = CHAIN POINTER TO NEXT (OR 1ST) TOC
814 ♦ 1 CDD ENTRY. MOST SIGNIFICANT (BIT 15) BIT
815 ♦ 1 CDD SET TO INDICATE END-OF-CHAIN
816 ♦ 1 CDD INDEX = VALUE COMPUTED FROM POINTER TO BE
817 ♦ 1 CDD FORTRAN INDEX INTO 'ANA' FOR NEXT TOC
818 ♦ 1 CDD ENTRY
819 ♦ 1 CDD
820 ♦ 1 CDD NOTES
821 ♦ 1 CDD
822 ♦ 1 CDD ROUTINE USED - EXEC, PURGE, XDSTA, XPIT, XREQ, XREXT, XR16,
823 ♦ 1 CDD XRMVT, XRMST, XRPMK, XRPMK, XRUPK, XICOM,
824 ♦ 1 CDD XUDBG, XVBAM
1 BEGIN XDOFF
2 PROMPT USER FOR TERMINATION CONFIRMATION
3 IF USER RESPOND GO AHEAD WITH TERMINATION THEN
4 CALL XREX TO REQUEST TOC AND CHAIN HEADS
5 IF CHAIN HEAD FOR ORED FILES IS NOT NEGATIVE THEN
6 DO UNTIL ORED CHAIN HEAD IS NEGATIVE
7 IF CHAIN POINTS BEYOND END OF TOC BUFFER THEN
8 OUTPUT 'XDI3 TOC TOO LARGE, PURGE INCOMPLETE'
9 EXIT DO
10 ENDIF
11 CALL XROFN TO CREATE FILE NAME 'XXXXX'
12 CALL PURGE TO SCRATCH FILE
13 SET ORED CHAIN HEAD TO TOC ENTRY CHAIN POINTER
14 ENDDO
15 ENDIF
16 IF CHAIN HEAD FOR DATA BASES IS NOT NEGATIVE, THEN
17 DO UNTIL DATA BASE CHAIN HEAD IS NEGATIVE
18 IF CHAIN Points BEYOND End OF TOC BUFFER THEN
19 OUTPUT 'XDI3 TOC TOO LARGE, PURGE INCOMPLETE'
20 EXIT DO
21 ENDIF
22 IF TYPE OF DATA BASE IS UTD, THEN
23 CALL XROFN TO CREATE FILE NAME 'XXXXX'
24 CALL PURGE TO SCRATCH FILE
25 ENDIF
26 SET DATA BASE CHAIN HEAD TO TOC ENTRY CHAIN POINTER
27 ENDDO
28 ENDIF
29 CALL XDATA TO OUTPUT USAGE STATISTICS
30 IF USER REQUESTED DEBUG SNAP THEN
31 CALL XDBG
32 ENDIF
33 IF USER REQUESTED ABEND DUMP THEN
34 CALL XVARH - NO RETURN FROM THIS CALL
35 ENDIF
36 SET PARAMETER 1 TO INDICATE TERMINATE EXEC
37 CALL XEXIT TO WAIT ON I/O COMPLETION, RETURN PARAMS AND TERMINATE EXEC
38 ENDF
39 RETURN
40 END XDOFF
FORTRAN CALLING SEQUENCE:

CALL XDBDE (DATBUF, DBDCB)

XDBDE HANDLES THE RESTORING OF DATA ELEMENTS, SEQUENCE TABLES
AND INTERFACE TABLES

INPUTS FROM CALLING SEQUENCE:

DATBUF - (INTEGER, 1400 WORDS) BUFFER USED TO READ IN RECORDS
OF DATA FROM DATA BASE FILE
DBDCB - (INTEGER, 144 WORDS) OPEN DATA BASE FILE DDB

INPUTS FROM XB COMMON:

XB(154) ALLFLG, XB(175) ENDBLK, XB(182) TOTSIZ,
XB(201) NOTOC

OUTPUTS IN CALLING SEQUENCE:

DATBUF, DBDCB

OUTPUTS IN XB COMMON

XB(151) ADFLG, XB(152) ERFLG, XB(165) TOCHOS,
XB(169) TOPBLC, XB(170) ENDBLK, XB(171) TOCNX,
XB(201) NOTOC

INTERNAL XB COMMON USED:

XB(151) ADFLG - ABORT FLAG
XB(152) ERFLG - ERROR MESSAGE FLAG
XB(154) ALLFLG - RESTORE ALL UTDB FLAG
XB(155) DEBUG - DEBUG FLAG
XB(159) FILNAME - DB FILE NAME
XB(162) DATBLK- BLOCK # WHERE DATA ITEM BEGINS
XB(163) DATMX- WORD INDEX INTO DATBUF WHERE DATA BEGINS
XB(164) IERR - ERROR FLAG FOR FMNR CALLS
XB(165) TOCHOS- # WORDS OF TOC IN TOCBUF (NOTOC +8) + 8
XB(166) NOBLKS- # BLOCKS TO READ
XB(167) LEN- # WORDS OF DATA TO MOVE
XB(168) DATEND- BLOCK # WHERE DATA ITEM ENDS
XB(169) TOPBLC- BLOCK # OF FIRST BLOCK IN DATBUF
XB(170) ENDBLK- BLOCK # OF LAST BLOCK READ
XB(171) TOCNX- INDEX INTO TOC OF CURRENT DATA ITEM
XB(175) IL - # WORDS OF DATA TO READ
XB(176) TOSCNR- DATA BASE TOC ENTRY
XB(201) NOTOC - # TOC ENTRIES IN TOCBUF
XB(201) TOCBUF- TOC BUFFER
FORTAN CALLING SEQUENCE:

CALL XRDF (DATBUF, DDDBC)

XRDF HANDLES THE RESTORING OF ORDE FILES FROM A MDD/UTDB FILE

INPUTS IN CALLING SEQUENCE:

DATBUF - (INTEGER, 1408 WORDS) BUFFER USED TO READ IN RECORDS OF ORDE FILES FROM DATA BASE FILE

DDDBC - (INTEGER, 144 WORDS) OPEN DATA BASE FILE DBC

INPUTS IN XR COMMON:

XB(152) ERFLAG, XB(154) ABFLG, XB(165) TOCDBS,
XB(160) TOTPDB, XB(170) ENDBLKS, XB(171) TOCDBX,
XB(182) TOTSIZ

OUTPUTS IN CALLING SEQUENCE:

DBDBC

OUTPUTS IN XR COMMON:

XB(151) ABFLG

INTERNAL XR COMMON USED:

XB(151) ABFLG - ABORT FLAG
XB(152) ERFLG - ERROR MESSAGE FLAG
XB(153) ALLFLG - RESTORE ALL UTDB FLAG
XB(155) DBG - DEBUG FLAG
XB(159) FLM - BB FILE NAME
XB(162) DATBLK - BLOCK # WHERE ORDE BEGINS
XB(163) DTPDDBX - WORD INDEX INTO DATBUF WHERE ORDE BEGINS
XB(164) ERR - ERROR FLAG FOR FNRG CALLS
XB(165) TOCDBS - # WORDS OF TOC IN TOCDBF (MOTO + 8)
XB(166) NBLKS - # BLOCKS OF UTDB/ORDE TO READ/WRITE
XB(167) LEM - # WORDS OF DATA TO MOVE
XB(169) TOPLBLK - BLOCK # OF FIRST BLOCK IN DATBUF
XB(170) ENDBLK - BLOCK # OF LAST BLOCK READ
XB(171) TOCDBX - INDEX TO CURRENT TOC ENTRY
XB(172) DBCFLG - DB FILE NAME
XB(173) IL - # WORDS OF DATA TO READ/WRITE
XB(201) TOCBUF - TOC BUFFER

COMMON USED:

987  CD0
988  CD0
989  CD0
990  CD0
991  CD0
992  CD0
993  CD0
994  CD0
995  CD0
996  CD0
997  CD0
998  CD0
999  CD0
1000 CD0
1001 CD0
1002 CD0
1003 CD0
1004 CD0
1005 CD0
1006 CD0
1007 CD0
1008 CD0
1009 CD0
1010 CD0
1011 CD0
1012 CD0
1013 CD0
1014 CD0
1015 CD0
1016 CD0
1017 CD0
1018 CD0
1019 CD0
1020 CD0
1021 CD0
1022 CD0
1023 CD0
1024 CD0
1025 CD0
1026 CD0
1027 CD0
1028 CD0
1029 CD0
1030 CD0
1031 CD0
1032 CD0
1033 CD0
1034 CD0
1035 CD0
1036 CD0
1037 CD0
1038 CD0
1039 CD0
1040 CD0
1041 CD0
1042 CD0
1043 CD0
1044 CD0
1045 CD0
1 BEGIN XRDF
2 DO WHILE THERE ARE TOC ENTRIES TO PROCESS
3 IF ALLFLG IS ZERO OR THE STORE/RESTORE BIT IS ON THEN
4 CALL RDRNF TO CREATE THE DRE FILE NAME
5 IF THERE WAS A CREATED ERROR THEN
6 IF ERROR FLAG IS ON THEN
7 SET ERROR FLAG ON
8 CALL XRMSG TO DISPLAY MAIN MESSAGE
9 ENDIF
10 CALL XRIG TO CONVERT ERROR CODE TO ASCII
11 CALL XRDF TO DISPLAY DRE NAME AND ERROR CODE
12 ELSE
13 CALL CLOSE TO CLOSE FILE
14 ERROR IF ERROR TO :ERR1:
15 CALL OPEN TO OPEN DRE FILE AS TYPE 1
16 ERROR IF OPEN ERROR TO :ERR1:
17 DO UNTIL ALL BLOCKS OF DRE HAVE BEEN PROCESSED
18 IF DATA FOR DRE IS IN DATBUF THEN
19 CALL WRITF DATA TO DRE
20 ERROR IF WRITF ERROR TO :ERR1:
21 ELSE
22 CALL READF TO READ NEXT BUFFER OF DATA
23 ERROR IF READF ERROR TO :ERR2:
24 SET INDICES INDICATING DATA IN BUFFER
25 ENDIF
26 ENDIF
27 ENDDO
28 CALL CLOSE TO CLOSE DRE FILE
29 ERROR IF CLOSE FAILED TO :ERR1:
30 BUILD DRE REQUEST TO ALLOCATE DRE IN AMA
31 IF AMA REQUEST BUFFER IS FULL THEN
32 CALL XRDRF TO MAKE REQUEST
33 EXIT XRDF IF XRDRF ERROR
34 ENDIF
35 ENDIF
36 ENDIF
37 ENDIF
38 ENDIF
39 :ERR1:
40 EXIT XRDF
41 ENDIF
42 :ERR2:
43 CALL XRMSG TO DISPLAY ERROR MESSAGE WITH FILE NAME
44 SET ABFLG TO SAY ABORT RESTORE
45 END XRDF
1106 1 C***********
1107 1 CDG
1108 1 CDG
1109 1 CDG
1110 1 CDG
1111 1 CDG
1112 1 CDG
1113 1 CDG
1114 1 CDG
1115 1 CDG
1116 1 CDG
1117 1 CDG
1118 1 CDG
1119 1 CDG
1120 1 CDG
1121 1 CDG
1122 1 CDG
1123 1 CDG
1124 1 CDG
1125 1 CDG
1126 1 CDG
1127 1 CDG
1128 1 CDG
1129 1 CDG
1130 1 CDG
1131 1 CDG
1132 1 CDG
1133 1 CDG
1134 1 CDG
1135 1 CDG
1136 1 CDG
1137 1 CDG
1138 1 CDG
1139 1 CDG
1140 1 CDG
1141 1 CDG
1142 1 C***********
1144 1 BEGIN XDEC
1145 2 SET STATUS FLAG INDICATING PROG & UTD0 FILES NOT OPEN
1146 3 ENIFIT IF "-" IS NOT NEXT TOKEN :ERROR:
1147 4 INCREMENT TO NEXT TOKEN
1148 5 ERNERR IF NEXT TOKEN IS NOT "NAME" :ERROR:
1149 6 ERNERR IF THIS NAME IS > 1 CHARACTERS :ERR14:
1150 7 ERNERR IF NAME BEGINS WITH DOUBLE EXCLAMATION :ERR16:
1151 8 RETAIN THIS NAME AS PD:
1152 9 RETAIN CURRENT USER IN (QUALIFIER)
1153 10 INCREMENT TO NEXT TOKEN
1154 11 IF NEXT TOKEN IS "=" THEN
1155 12 INCREMENT TO NEXT TOKEN
1156 13 ERNERR IF NEXT TOKEN IS NOT "NAME" :ERROR:
1157 14 ERNERR IF "NAME" IS GREATER THAN 1 CHARACTER :ERR15:
1158 15 ERNERR IF QUALIFIER < 'A' OR > 'Z' :ERR16:
1159 16 SAVE IN AT (ZERO FILS), RIGHT-JUSTIFIED FORMAT AS QUALIFIER
1160 17 INCREMENT TO NEXT TOKEN
1161 18 ENIF
1162 19 ERNERR IF NEXT TOKEN IS NOT A "NAME" :ERROR:
1163 20 INCREMENT TO NEXT TOKEN
1164 21 ERNERR IF NEXT TOKEN IS NOT A "NAME" :ERROR:
1165 22 ERNERR IF THIS NAME BEGINS WITH DOUBLE EXCLAMATION :ERR06:
1166 23 RETAIN THIS NAME AS UTD:
1167 24 INCREMENT TO NEXT TOKEN
1168 25 ERNERR IF NEXT TOKEN IS NOT EOS :ERROR:
1169 26 BUILD AWA REQUEST TO VERIFY EXISTENCE OF UTD:
1170 27 CALL AWA TO PROCESS AWA REQUEST
1171 28 ERNERR IF UTD0 DOES ALREADY EXIST :ERR22:
1172 29 CALL AWA0 TO VERIFY PROG AND RETRIEVE SIZE
1173 30 ERNERR IF PROG DOES NOT EXIST :ERR16:
1174 31 ERNERR IF FNER ERROR RETURNED :ERR44:
1175 32 CALL XRWM TO CULT PB FILE NAME
1176 33 CALL OPEN TO OPEN PB FILE
1177 34 ERNERR IF OPEN FAILED :ERR18:
1178 35 SET STATUS FLAG INDICATING PB FILE OPEN
1179 36 BUILD AWA REQUEST TO ALLOCATE UTD:
1180 37 CALL AWA TO PROCESS AWA REQUEST
1181 38 ERNERR IF ALLOCATE REQUEST FAILS :ERR21:
1182 39 SET STATUS FLAG TO INDICATE UTD0 FILE ALLOCATED IN AWA
1183 40 RESTORE CURRENT USER'S ED (QUALIFIER)
1184 41 CALL XRWM TO BUILD UTD0 FILE NAME
1185 42 CALL CREAT TO CREATE TYPE 1 UTD0 FILE
1186 43 ERNERR IF CREATE FAILED :ERROR:
1187 44 SET STATUS FLAG INDICATING UTD0 FILE NOW OPEN
1188 45 DO FOR EACH BUFFER OF DATA IN PROG FILE
1189 46 CALL READ TO READ 1 BUFFER FROM PROG FILE
1190 47 ERNERR IF READ ERROR :ERR18:
1191 48 CALL WRIT TO WRITE 1 BUFFER TO UTD0 FILE
1192 49 ERNERR IF WRITE ERROR :ERROR:
1193 50 ENDOO
1194 51 CALL CLOSE FOR PD:
1195 52 CALL CLOSE FOR UTD:
1196 53 EXIT XDEC
1197 54 xDEC
1198 1 :ERROR: ISSUE MESSAGE - "SYNTAX ERROR. EXTRANEOUS DATA"
1199 2 :ERROR: ISSUE MESSAGE - "ILLEGAL: UTD0 NAME (NOT FOUND OR TOO LONG)"
1200 3 xDEC
1200 2 :ERROR7: ISSUE MESSAGE - "UTDB FILE ACCESS ERROR ..."
1201 2 :ERROR9: ISSUE MESSAGE - "SYNTAX ERROR - ILLEGAL OR MISSING FIELD"
1202 2 :ERROR16: ISSUE MESSAGE - "INVALID PDB FILE NAME ..."
1203 2 :ERROR18: ISSUE MESSAGE - "PDB FILE ACCESS ERROR ..."
1204 2 :ERROR19: ISSUE MESSAGE - "USER ID IS INVALID FOR PDB/UTDB logging"
1205 2 :ERROR21: ISSUE MESSAGE - "AWA OVERFLOW - XXX NOT LOGGED"
1206 2 :ERROR22: ISSUE MESSAGE - "XXX ALREADY EXISTS"
1207 2 :ERROR44: ISSUE MESSAGE - "FILE ACCESS ERROR #-- XPD"b"
1208 2 :RETURN:
1209 2 IF STATUS FLAG INDICATES UTDB FILE IS OPEN, THEN
1210 3 PURGE UTDB
1211 2 ENDIF
1212 2 IF STATUS FLAG INDICATES UTDB IS LOGGED IN AW, THEN
1213 3 CALL XER TO DELETE UTDB FROM AW
1214 2 ENDIF
1215 2 IF STATUS FLAG INDICATES PDB FILE IS OPEN, THEN
1216 3 CALL CLOSE FOR PDB FILE
1217 2 ENDIF
1218 1 "ND XDREC
1220 1 ********
1221 1 CD0 1 FORTRAN CALLING PROCEDURE
1222 1 CD0 1 CALL XOREN
1223 1 CD0 1 ********
1224 1 CD1 1 XOREN PROCESSES A USER REQUEST TO RENAME A SEQUENCE TABLE,
1225 1 CD1 1 INTERFACE TABLE, DATA ELEMENT, DRDE, UTDB, OR PDB. ONLY
1226 1 CD1 1 THOSE TABLES OR ELEMENTS LOGGED IN THE USERS AWS ARE RENAMED.
1227 1 CD1 1 ********
1228 1 CD2 1 COMMON XE - CARTRG, COMBUF, CMPTSR, QUAL, RERPTR, TOKENS
1229 1 CD2 1 ********
1230 1 CD3 1 COMMON XE - RERBUF
1231 1 CD3 1 ********
1232 1 CD4 1 INTERNAL VARIABLES
1233 1 CD4 1 COMMON XS - (2) DTYPE: TYPE CODE FOR DATA BASE FILES
1234 1 CD4 1 (3) J: INDEX
1235 1 CD4 1 (148) ERR: FILE MANAGER ERROR RETURN
1236 1 CD4 1 (149) IMSC: ERROR MESSAGE NUMBERS
1237 1 CD4 1 (150) FNNAME: NEW FHR FILE NAME
1238 1 CD4 1 (153) FNMPT: POINTER TO NEW NAME IN COMBUF
1239 1 CD4 1 (154) UNAME: OLD FHR FILE NAME
1240 1 CD4 1 (157) ONMPT: POINTER TO OLD NAME IN COMBUF
1241 1 CD4 1 (158) TOTENT: TOC ENTRY FOR DATA BASE RENAME
1242 1 CD4 1 (166) TATELS: DATA CLASS CODE (STORED IN LEFT BYTE)
1243 1 CD4 1 COMMON XB - (201) IDCS: EXEC BUFFER
1244 1 CD4 1 ********
1245 1 CD5 1 EXTERNAL REFERENCES
1246 1 CD5 1 ROUTINES USED -
1247 1 CD5 1 EXEC
1248 1 CD5 1 JANO
1249 1 CD5 1 KEVT
1250 1 CD5 1 NAMF
1251 1 CD5 1 XDDBA
1252 1 CD5 1 XDOBD
1253 1 CD5 1 XREG
1254 1 CD5 1 XRMOV
1255 1 CD5 1 XRMSG
1256 1 CD5 1 XROR
1257 1 CD5 1 XRSFR
1  BEGIN XDREN
2   EXIT TO :SYNTAX: IF FIRST TOKENS ARE NOT "NAME"
3   SAVE POINTER TO CURRENT NAME
4   INCEREMNT TO NEXT TOKEN
5   IF TOKEN IS A RHYPHEN, THEN
6   DECODE CLASS NAME
7   EXIT TO :SYNTAX: IF NEXT TOKEN IS NOT NAME
8   EXIT TO :CLASER: IF CLASS SPECIFIED IS NOT VALID (S, I, D, F, B)
9   ELSE
10  SET CLASS TO 0
11  ENDIF
12  EXIT TO :SYNTAX: IF NEXT TOKENS ARE NOT "NAME"
13  SAVE POINTER TO NEW NAME
14  IF CLASS IS DATA BASE OR DDB, THEN
15  EXIT TO :TOOLNG: IF NEW NAME IS MORE THAN 4 CHARACTERS
16  ENDIF
17  IF CLASS IS DATA BASE, THEN
18  EXIT TO :TOOLNG: IF NEW NAME STARTS WITH DOUBLE EXCLAMATION
19  ENDIF
20  IF CLASS IS INTERFACE TABLE, THEN
21     CALL EXEC TO WRITE/READ NEW NAME
22     BUILD AWA MANAGER REQUEST TO CHANGE NAME IN TOC
23     BUILD AWA MANAGER REQUEST TO STORE NEW NAME IN TABLE
24     CALL XREN TO ISSUE REQUESTS
25     EXIT TO :CALGET: IF RETURN CODE INDICATES ERROR
26     ELSE
27         IF CLASS IS DATA BASE, THEN
28             BUILD AND ISSUE AWA MANAGER REQUEST FOR TOC ENTRY
29             EXIT TO :NAMEERR: IF RETURN INDICATES ERROR
30             EXIT TO :INVALID: IF DATA BASE IS NOT DDB
31             ENDIF
32             BUILD AND ISSUE AWA MANAGER REQUEST FOR NAME CHANGE
33             EXIT TO :NAMEERR: IF RETURN CODE INDICATES ERROR
34             IF CLASS IS DATA BASE OR DDB, THEN
35                 CALL XRF/N TO FORMAT FILE NAME
36                 CALL FILE MANAGER TO CHANGE DISC FILE NAME
37                 EXIT TO :UNDO: IF FILE MANAGER RETURNS ERROR
38                 IF FILE IS A PDB, THEN
39                     CALL XDDDT TO DELETE OLD PDB FROM XPDB
40                     EXIT TO :NAMEERR: IF ERROR RETURNED
41                     CALL XDDDT TO ADD NEW PDB NAME TO XPDB
42                     EXIT TO :TELEUSER: IF ERROR RETURNED
43                     ENDIF
44                     ENDIF
45                     ENDIF
46                     ENDIF
47                     ENDIF
48                     ENDIF
49                     ENDIF
50                     ENDIF
51                     ENDIF
52                     ENDIF
53                     ENDIF
54                     ENDIF
55                     ENDIF
56                     ENDIF
57                     ENDIF
58                     ENDIF
59                     ENDIF
60                     ENDIF
61                     ENDIF
62                     ENDIF
63                     ENDIF
64                     ENDIF
65                     ENDIF
66                     ENDIF
67                     ENDIF
68                     ENDIF
69                     ENDIF
70                     ENDIF
71                     ENDIF
72                     ENDIF
73                     ENDIF
74                     ENDIF
75                     ENDIF
76                     ENDIF
77                     ENDIF
78                     ENDIF
79                     ENDIF
80                     ENDIF
81                     ENDIF
82                     ENDIF
83                     ENDIF
84                     ENDIF
85                     ENDIF
86                     ENDIF
87                     ENDIF
88                     ENDIF
89                     ENDIF
90                     ENDIF
91                     ENDIF
92                     ENDIF
93                     ENDIF
94                     ENDIF
95                     ENDIF
96                     ENDIF
97                     ENDIF
98                     ENDIF
99                     ENDIF
100                    ENDIF
101                    ENDIF
102                    ENDIF
103                    ENDIF
104                    ENDIF
105                    ENDIF
106                    ENDIF
107                    ENDIF
108                    ENDIF
109                    ENDIF
110                    ENDIF
111                    ENDIF
112                    ENDIF
113                    ENDIF
114                    ENDIF
115                    ENDIF
116                    ENDIF
117                    ENDIF
118                    ENDIF
119                    ENDIF
120                    ENDIF
121                    ENDIF
122                    ENDIF
123                    ENDIF
124                    ENDIF
125                    ENDIF
126                    ENDIF
127                    ENDIF
128                    ENDIF
129                    ENDIF
130                    ENDIF
131                    ENDIF
132                    ENDIF
133                    ENDIF
134                    ENDIF
135                    ENDIF
136                    ENDIF
137                    ENDIF
138                    ENDIF
139                    ENDIF
140                    ENDIF
141                    ENDIF
142                    ENDIF
143                    ENDIF
144                    ENDIF
145                    ENDIF
146                    ENDIF
147                    ENDIF
148                    ENDIF
149                    ENDIF
150                    ENDIF
151                    ENDIF
152                    ENDIF
153                    ENDIF
154                    ENDIF
155                    ENDIF
156                    ENDIF
157                    ENDIF
158                    ENDIF
159                    ENDIF
160                    ENDIF
161                    ENDIF
162                    ENDIF
163                    ENDIF
164                    ENDIF
165                    ENDIF
166                    ENDIF
167                    ENDIF
168                    ENDIF
169                    ENDIF
170                    ENDIF
171                    ENDIF
172                    ENDIF
173                    ENDIF
174                    ENDIF
175                    ENDIF
176                    ENDIF
177                    ENDIF
178                    ENDIF
179                    ENDIF
180                    ENDIF
181                    ENDIF
182                    ENDIF
183                    ENDIF
184                    ENDIF
185                    ENDIF
186                    ENDIF
187                    ENDIF
188                    ENDIF
189                    ENDIF
190                    ENDIF
191                    ENDIF
192                    ENDIF
193                    ENDIF
194                    ENDIF
195                    ENDIF
196                    ENDIF
197                    ENDIF
198                    ENDIF
199                    ENDIF
200                    ENDIF
201                    ENDIF
202                    ENDIF
203                    ENDIF
204                    ENDIF
205                    ENDIF
206                    ENDIF
207                    ENDIF
208                    ENDIF
209                    ENDIF
210                    ENDIF
211                    ENDIF
212                    ENDIF
213                    ENDIF
214                    ENDIF
215                    ENDIF
216                    ENDIF
217                    ENDIF
218                    ENDIF
219                    ENDIF
220                    ENDIF
221                    ENDIF
222                    ENDIF
223                    ENDIF
224                    ENDIF
225                    ENDIF
226                    ENDIF
227                    ENDIF
228                    ENDIF
229                    ENDIF
230                    ENDIF
231                    ENDIF
232                    ENDIF
233                    ENDIF
234                    ENDIF
235                    ENDIF
236                    ENDIF
237                    ENDIF
238                    ENDIF
239                    ENDIF
240                    ENDIF
241                    ENDIF
242                    ENDIF
243                    ENDIF
244                    ENDIF
245                    ENDIF
246                    ENDIF
247                    ENDIF
248                    ENDIF
249                    ENDIF
250                    ENDIF
251                    ENDIF
252                    ENDIF
253                    ENDIF
254                    ENDIF
255                    ENDIF
256                    ENDIF
257                    ENDIF
258                    ENDIF
259                    ENDIF
260                    ENDIF
261                    ENDIF
262                    ENDIF
263                    ENDIF
264                    ENDIF
265                    ENDIF
266                    ENDIF
267                    ENDIF
268                    ENDIF
269                    ENDIF
270                    ENDIF
271                    ENDIF
272                    ENDIF
273                    ENDIF
274                    ENDIF
275                    ENDIF
276                    ENDIF
277                    ENDIF
278                    ENDIF
279                    ENDIF
280                    ENDIF
281                    ENDIF
282                    ENDIF
283                    ENDIF
284                    ENDIF
285                    ENDIF
286                    ENDIF
287                    ENDIF
288                    ENDIF
289                    ENDIF
290                    ENDIF
291                    ENDIF
292                    ENDIF
293                    ENDIF
294                    ENDIF
295                    ENDIF
296                    ENDIF
297                    ENDIF
298                    ENDIF
299                    ENDIF
300                    ENDIF
301                    ENDIF
302                    ENDIF
303                    ENDIF
304                    ENDIF
305                    ENDIF
306                    ENDIF
307                    ENDIF
308                    ENDIF
309                    ENDIF
310                    ENDIF
311                    ENDIF
312                    ENDIF
313                    ENDIF
314                    ENDIF
315                    ENDIF
316                    ENDIF
317                    ENDIF
318                    ENDIF
319                    ENDIF
320                    ENDIF
321                    ENDIF
322                    ENDIF
323                    ENDIF
324                    ENDIF
325                    ENDIF
326                    ENDIF
327                    ENDIF
328                    ENDIF
329                    ENDIF
330                    ENDIF
331                    ENDIF
332                    ENDIF
1386  BEGIN XREQ
1387    CALL XREQ TO PROCESS AWA REQUEST(S)
1388    IF AN ERROR RETURNED BY AWA MANAGER, THEN
1389      IF ERROR FLAG (ERFLG) IS ZERO, THEN
1390        TURN ON ERFLG INDICATING THAT MSG 234 HAS BEEN ISSUED
1391        CALL XRMSE TO OUTPUT MSG 234 - 'FOLLOWING ELEMENTS NOT RESTORED'
1392    ENDIF
1393    CALL EXEC TO WRITE ELEMENT NAME, CLASS AND REASON
1394    IF CLASS OF ELEMENT IS DDE, THEN
1395      CALL PURGE TO DELETE THE FILE
1396    ELSE, ELEMENT RESIDES IN AWA
1397      CALL EXEC TO FREE CLASS NO. AND SAM BUFFER
1398    ENDIF
1399    IF AWA REQUESTS EXIST IN REQBUF BEYOND FAILING REQUEST, THEN
1400      MOVE THESE REQUESTS TO TOP OF REQUEST BUFFER
1401    ENDIF
1402    ELSE
1403      SET REQPTR TO 1 INDICATING NO REQUESTS PRESENT
1404    ENDIF
1405    EXIT XREQ
1406    END XREQ
FORTRAN CALLING PROCEDURE

CALL XRDE (DATBUF)

XRDE PROCESSES THE RESTORE DIRECTIVE. THE SPECIFIED UDDB OR NDB FILE IS OPENED AND ITS TOC IS READ. XRDE IS CALLED TO MARK TOC ENTRIES FOR RESTORE. XRDE IS CALLED TO RESTORE DATA ELEMENTS AND TABLES. XRDF IS CALL TO RESTORE DATA'S.

INPUT

COMMON XE - LU, Flags, REPTR, REPBUF, TOKENS, CARTAG, COMPTR, COMBUF

CALLING SEQUENCE

DATBUF - 1408 WORD BUFFER USED TO READ UDDB FILE

OUTPUT

COMMON XE - REPBUF

COMMON XB - ABFLG = ABORT FLAG. NON-ZERO VALUE FROM XRDF. XRDF, XRDE, OR XRDEW INDICATES ABORT OF RESTORE

ALLFLG = SET NON-ZERO IF LIST OF ELEMENTS SPECIFIED

DEBUG = DEBUG BIT OF 'FLAGS' IN XE COMMON

ENDBLK = BLOCK # OF LAST BLOCK READ

ERFLG = SET NON-ZERO IF MESSAGE #54 ISSUED SO THAT IT IS ISSUED ONLY ONCE

FIKNAME = UDDB/NDB FILE NAME

TOCBUF = UDDB TOC ENTRIES, 8 WORDS EACH, MAX 12000 WORDS

TOCENTRY = INDEX TO NEXT UDDB TOC ENTRY

TOTSIZ = NO. OF BLOCKS REMAINING IN UDDB FILE

NOTES

Routines Called

CLOSE

EXEC

OPEN

READ

XRDF

XRDE

XRDEW

XRDE
1476 1 BEGIN XDRES
1477 2 SET ABFLG TO ZERO
1478 3 ERREXIT IF NEXT TOKEN IS NOT A COMMA :ERR09:
1479 4 ERREXIT IF FOLLOWING TOKEN IS NOT A NAME :ERR09:
1480 5 BUILD AWA REQUEST FOR TOC ENTRY RETRIEVE
1481 6 CALL AWA TO PROCESS AWA REQUEST
1482 7 ERREXIT IF AWA REQUEST FAILED :ERR10:
1483 8 IF DATA BASE FOUND IS A PDB :ERR33:
1484 9 IF DATA BASE IS A UDB, THEN
1485 10 CALL XDREFM TO CONSTRUCT QUALIFIED FILE NAME
1486 11 ENDIF
1487 12 CALL OPEN TO OPEN SPECIFIED FILE
1488 13 ERREXIT IF OPEN FAILED :ERR08:
1489 14 CALL READ TO READ FIRST RECORD OF DATA BASE FILE INTO TOCBUF
1490 15 ERREXIT IF READ FAILED :ERR08:
1500 2 ENDIF
1501 2 UPDATE TOTSIZ TO NUMBER OF BLOCKS REMAINING IN FILE (DECREMENT BY ENDBLK)
1502 2 CLEAR ERROR MESSAGE FLAG (ERRFLG)
1503 2 IF TOKEN IS EOS (i.e. NO LIST OF ELEMENTS), THEN
1504 3 SET ALLFLG TO ZERO INDICATING TO RESTORE ALL TOC ENTRIES
1505 2 ELSE
1506 2 SET ALLFLG NON-ZERO INDICATING TO RESTORE ONLY FLAGGED TOC ENTRIES
1507 2 CALL XDRLS TO PROCESS ELEMENTS SPECIFIED AND TO FLAG TOC ENTRIES
1508 2 EXIT XDRES IF ALLFLG SET BY XDRLS
1509 2 ENDIF
1510 2 CALL XDREO TO RESTORE AWA RESIDENT ELEMENTS
1511 1 EXIT XDRES IF ABFLG SET BY XDREO
1512 2 CALL ZDRDF TO RESTORE ORDE'S
1513 1 EXIT XDRES IF ABFLG SET BY ZDRDF
1514 2 CALL CLOSE TO CLOSE DATA BASE FILE
1515 2 ERREXIT IF CLOSE FAILED :ERR08:
1516 2 DO WHILE AWA REQUESTS REMAIN IN RERBUF
1517 3 CALL XDREO TO PROCESS AWA REQUESTS
1518 2 EXIT XDRES IF ABFLG SET BY XDREO
1519 2 ENDDO
1520 1 EXIT XDRES
1521 2 :ERR04: CALL XRMSG - 'SYNTAX ERROR. EXTRAMOS DATA'
1522 2 :ERR08: CALL XRMSG - 'FILE MANAGER ERROR .... .......
1523 2 :ERR09: CALL XRMSG - 'SYNTAX ERROR. MISSING OR ILLEGAL FIELD'
1524 2 :ERR10: CALL XRMSG - '....... NOT FOUND'
1525 2 :ERR33: CALL XRMSG - 'CANT NOT RESTORE A PDB'
1526 2 DO UNTIL ALL AWA REQUESTS IN RERBUF HAVE BEEN PROCESSED
1527 3 IF REQUEST IS TO STORE DATA, THEN
1528 4 CALL EXEC TO FREE THE SPECIFIED CLASS NO. AND SAM BUFFER
1529 3 ENDIF
**FORTRAN CALLING PROCEDURE**

**INPUT**
- COMMON xe - CARTRG, COMBUF, CMPTXR, FLAGS, LU, TOKENS

**OUTPUT**
- COMMON xe - REGBUF, REGPTR

**FILES**
- xxx (UTDB FILE SPECIFIED)
- xxx (PDB FILE SPECIFIED)

**INTERNAL VARIABLES**
- DCBPDB - DCB FOR THE PDB FILE; ALLOCATED IN X5 COMMON
- BCBUTD - DCB FOR THE UTD, ALLOCATED IN X5 COMMON;
- CCONTAINS 1152 WORD BUFFER USED TO READ THE
- UTD AND TO WRITE THE PDB.
1625 2 :ERR18: ISSUE MESSAGE "PRO FILE ACCESS ERROR ... "
1626 2 :ERR00: ISSUE MESSAGE "FILE MANAGER ERROR ... "
1627 2 :ERR20: ISSUE MESSAGE "AUTHORIZED LIMIT OF ... PRO'S ALREADY REACHED"
1628 2 :ERR21: ISSUE MESSAGE "AN A OVERFLOW. NOT LOGGED"
1629 2 :RETURN:
1630 2 IF STATUS FLAG INDICATES PRO FILE IS OPEN, THEN
1631 3 PURGE PRO FILE
1632 2 ENDIF
1633 2 IF FLAG INDICATES PRO IS IN ANA, THEN
1634 3 CALL REN TO DELETE PRO FROM ANA
1635 2 ENDIF
1636 2 IF FLAG INDICATES UTDB IS OPEN, THEN
1637 3 CLOSE UTDB
1638 2 ENDIF
1639 2 IF FLAG INDICATES PRO IS IN XPROB, THEN
1640 3 CALL ZDDB TO DELETE PRO FROM XPROB
1641 2 ENDIF
1642 2 ENDIF
1643 1 END XDSAV
164:  1 CRO  FORTH calling procedure
164:  2 CC
164:  3 CDR  CALL XDSTA (LU)
164:  4 CT
164:  5 C**-------
165:  1 CD1  OUTPUT AMA and DMA usage data and statistics on AMA compaction
165:  2 CT1  PHASES
165:  3 CT
165:  4 C**-------
165:  1 CD2  INPUT
165:  2 CC2  LU --- LOGICAL UNIT OF OUTPUT DEVICE
165:  3 CR2
165:  4 C**----- COMMON
165:  1 C22  ASZ  --- TOTAL SIZE OF AMA
165:  2 CD2  FRE  --- AMOUNT OF FREE SPACE IN AMA
165:  3 CD2  DSZ  --- TOTAL SIZE OF DMA (IN 64-WORD SECTORS)
165:  4 CD2  DFS  --- AMOUNT OF FREE SECTORS IN DMA
165:  5 CD2  HP  --- ARRAY OF THREE COMPACT PHASE COUNTS
165:  6 CD2
165:  7 C**----- OUTPUT
165:  1 CD3  USAGE AND STATISTICS TO LU
165:  2 C**----- NOTES
165:  1 C05  USES EXEC, XRIG
165:  2 C05
165:  3 C**-----
165:  1 *
165:  2 *
165:  3 *
165:  4 *
165:  5 *
165:  6 *
165:  7 Begin XDSTA
165:  2 FORMAT AND PRINT AMA USAGE DATA
165:  2 FORMAT AND PRINT DMA USAGE DATA
165:  2 FORMAT AND PRINT COMPACT PHASE STATISTICS
165:  3 End XDSTA
CO************
CO DO
CO CALL XDSTO (DATBUF)
CO************
CO
CO XDSTO IS THE STORE DIRECTIVE HANDLER. IT VERIFIES INPUTS ON
CO DIRECTIVE, BUILDS UTDB TOC, CREATES UTDB AND STORES UTDB TOC
CO ENTRY IN AWA.
CO
CO************
CO
CO INPUTS FROM CALLING SEQUENCE:
CO
databuf (integer 1480 words) - buffer used to read in AWA TOC.
CO
CO************
CO
CO INTERNAL XD COMMON USED:
CO
CO XE(151) ABFLG - (integer, 1 word) abort flag
CO XE(152) ERLG - (integer, 1 word) error message flag
CO XE(153) MSGNO - (integer, 1 word) message number to be displayed
CO XE(157) TOTSIZ- (integer, 1 word) total size of utdb file
CO XE(158) TOTWTRD- (integer, 1 word) total words in a drede file
CO XE(159) FILE - (integer, 3 words) utdb file name (xxxx)
CO XE(162) DBREC- (integer, 1 word) record # where data goes next
CO XE(164) UDBERR- (integer, 1 word) utdb file error flag
CO XE(165) UDBNM- (integer, 1 word) utdb name (xxx)
CO XE(201) NOTOC- (integer, 1 word) number of toc entries
CO XE(201) TOCBUF- (integer, 1200 words) utdb toc buffer
CO
CO************
CO
CO COMMON USED:
CO
CO EQUIVALENCE  (XE(1), LU )
CO + (XE(4), FLGS ), (XE(19), REQPTR )
CO + (XE(20), RERBUF , (XE(85), EGS )
CO + (XE(89), NAME ), (XE(113), COMMA )
CO + (XE(142), ICR ), (XE(144), COMPTR )
CO + (XE(145), COMBUP )
CO
CO RTE ROUTINES USED:
CO
CO close, creat, exec, kcvt, purge, writf
CO
CO FDS ROUTINES USED:
CO
CO xls, xdr, xrcpr, xrel, xnext, xrmov,
CO xmsg, xset, xssl, xsfh, xxbf, xubge
1737 1 BEGIN XSTO
1738 2 :BEGIN ABFGL TO ZERO (ABORT FLAG)
1739 2 EXECUT IF ABFGL NAME IS NOT VALID TO :ERR2:
1740 2 CALL XREQ TO MAKE MANAGER REQUEST
1741 2 EXECUT IF SIZE OF OCC > MAXIMUM SIZE TO :ERR2:
1742 2 DO FOR ALL DATA BASE CLASS ENTRIES
1743 2 EXECUT IF NAME SPECIFIED ALREADY EXISTS TO :ERR2:
1744 2 ENDO
1745 4 EXECUT IF NAME SPECIFIED ALREADY EXISTS TO :ERR2:
1746 4 ENDO
1747 2 EXECUT IF NAME SPECIFIED ALREADY EXISTS TO :ERR2:
1748 2 ENDO
1749 2 EXECUT IF NAME SPECIFIED ALREADY EXISTS TO :ERR2:
1750 2 IF whole AWA IS TO BE STORED THEN
1751 2 IF whole AWA IS TO BE STORED THEN
1752 5 DO FOR IT, ST, DE, ORDE AWA TOC ENTRIES
1753 5 DO FOR EACH ENTRY IN THIS CHAIN
1754 5 IF PREFIX IS NOT DOUBLE EXCLAMATION AND
1755 6 PREFERENCES IS NOT AN ANDERAND THEN
1756 6 SET STORE/RESTORE BIT ON IN TOC ENTRY
1757 6 INCREMENT TOTSIZE BY SIZE OF THIS ELEMENT
1758 5 ENDF
1759 3 ENDO
1760 3 ELSE
1761 3 EXECUT IF ABFGL IS NOT ZERO TO :ERR5:
1762 3 ENDF
1763 3 EXECUT IF ABFGL IS NOT ZERO TO :ERR5:
1764 3 SET NOTOC = 0 (NUMBER OF UDTOB TOC ENTRIES)
1765 3 DO FOR IT, ST, DE, ORDE CHAINS
1766 4 DO FOR EACH ENTRY IN THIS CHAIN
1767 4 IF STORE/RESTORE BIT IS ON THEN
1768 5 TURNS STORE/RESTORE BIT OFF
1769 5 BUILD UDTOB TOC ENTRY
1770 4 ENDF
1771 3 ENDO
1772 3 ENDO
1773 3 EXECUT IF THERE ARE NO UDTOB TOC ENTRIES (NOTOC=0) TO :ERR2:
1774 3 COMPUTE DATREC AS FIRST RECORD AVAILABLE FOR DATA
1775 2 CALL XITP " TO CREATE FILE NAME
1776 2 CALL TO CREATE UDTOB FOR TOTSIZE
1777 2 EXECUT IF TOTSIZE IS NOT ERROR TO :ERR3:
1778 2 CALL UDTOB TO WRITE UDTOB FILE
1779 2 EXECUT IF ABFGL IS A (ORDRE LARGER THAN SPECIFIED) TO :ERR5:
1780 2 EXECUT IF ABFGL IS B (ORDRE FILE ERROR) TO :ERR4:
1781 2 EXECUT IF ABFGL IS C (ORDRE FILE ERROR) TO :ERR3:
1782 2 CALL WRITE TO WRITE TOC RECORDS AT RECORD 1
1783 2 EXECUT IF ERROR IN WRITE TO :ERR3:
1784 2 EXECUT IF ERROR IN CLOSE TO :ERR3:
1785 2 EXECUT IF ERROR IN CLOSE TO :ERR3:
1786 2 BUILD REQUEST TO ALLOCATE UDTOB IN AWA
1787 2 CALL XREQ TO MAKE REQUEST
1788 2 EXECUT IF AWA OVERFLOW TO :ERR1:
1789 1 EXIT XSTO
1790 2 :ERR1:
1791 2 CALL XREQ TO DISPLAY MSGNO
1792 2 GO TO :ERR4:
1793 2 :ERR2:
1805 1 CD0  FORTRAN CALLING PROCEDURE
1807 1 CD0  CALL XDOTOC
1810 1 CD0  **********
1812 1 CD1  FDS AWA/DWA TABLE OF CONTENTS DIRECTIVE HANDLER. XDOTOC INTER-
1813 1 CD1  PRETS THE TOC DIRECTIVE, RETRIEVES THE INDICATED TOC (AWA OR
1814 1 CD1  DATA BASE FILE) AND FORMATS AND OPUTES THE REQUESTED ENTRIES.
1815 1 CD0  **********
1818 1 CD2  INPUT
1820 1 CD2  ** COMMON - COMBUF, COMPTR, FLAGS, LU
1821 1 CD2  ** MANAGER - AWA HEADER AND TOC (SEE MODULE XMWA)
1822 1 CD2  ** FILES - DATA BASE FILES AS APPROPRIATE
1824 1 CD2  **********
1827 1 CD4  OUTPUT
1829 1 CD4  ** COMMON - COMPTR, RBUF, REPTR
1830 1 CD4  ** PRINTED OUTPUT - AWA OR UTB TOC DISPLAY
1832 1 CD4  **********
1835 1 CD7  LOCAL VARIABLES
1837 1 CD4  ** ASIZ - TOTAL ALLOCATABLE SIZE OF AWA (SEE MODULE XMWA)
1839 1 CD4  ** ANA - FIRST PORTION OF AWA (HEADER AND TOC)
1841 1 CD4  ** CLASS - ARRAY OF ALLOCABLE CLASS DESIGNATORS FOR TOC DIRECTIVE
1843 1 CD4  ** CLS - CLASS NAME NUMBER BEING PROCESSED PLUS ONE
1845 1 CD4  ** CODE - CLASS NUMBERS CORRESPONDING TO ELEMENTS OF 'CLASS'
1847 1 CD4  ** EOC - END OF TOC NAME DESIGNATOR (~32768)
1849 1 CD4  ** FRE - CURRENT AMOUNT OF FREE SPACE IN AWA (SEE MODULE XMWA)
1851 1 CD4  ** NO - ARRAY OF TOC NAME HEADS (SEE MODULE XMWA)
1853 1 CD4  ** IDIM - TOC ENTRY DIM FIELD (WORD 8)
1855 1 CD4  ** LINE - BUFFER FOR CONSTRUCTING CURRENT OUTPUT IMAGE
1857 1 CD4  ** MULT - ARRAY OF MULTIPLIERS USED TO DETERMINE NUMBER OF WORDS
1859 1 CD4  ** IN COLUMNS OF TWO DIMENSIONAL DATA ELEMENTS (FUNCTION OF
1861 1 CD4  ** DATA TYPE)
1863 1 CD4  ** OPTION - ARRAY OF SELECTED CHAINS TO LIST IN TOC DISPLAY
1865 1 CD4  ** ORG - ORIGIN ADDRESS OF AWA (SEE MODULE XMWA)
1867 1 CD4  ** POS - VALUE OF 2 OR 22 INDICATING FIRST OR SECOND POSITION IN
1869 1 CD4  ** TOC DISPLAY LINE
1871 1 CD4  ** SIZE - TOC ENTRY SIZE FIELD (WORD 7)
1873 1 CD4  ** TOCMAX - MAXIMUM SIZE OF TOC WHICH CAN BE TOTALLY ACCOMODATED BY
1875 1 CD4  ** IRENAL BUFFER
1877 1 CD4  ** TYPE - TOC ENTRY SIZE FIELD (BYTE 2 OF WORD 1)
1879 1 CD4  ** TYPEID - ARRAY OF TYPE ID CODES FOR DE AND ORDE DISPLAYS
1881 1 CD4  ** UNIT - LOGICAL UNIT SELECTED FOR OUTPUT OF DISPLAY
1883 1 CD4  **********
1885 1 CD5  NOTES
1888 1 CD5  ** USES CLOSE, EXEC, IAND, IOR, KCVT, OPEN, READF, XDSTA, XREG,
1890 1 CD5  ** XRIG, XRMOV, XMSG, XROG, XRF, XUSB
1892 1 CD5  ** IN THE EVENT THE COMPACTED AWA TOC AND HEADER DATA WILL NOT FIT
1894 1 CD5  ** IN THE ALLOCATED BUFFER SPACE, AS MUCH OF EACH CHAIN AS POSSIBLE
1896 1 CD5  ** WILL BE DISPLAYED UNTIL THE LINK FIELDS LEAD BEYOND THE BUFFER.
1898 1 CD5  **
1899 1 CD5  ** IT IS ASSUMED THAT XDOTOC AND XDSTO USE THE SAME SIZE BUFFER FOR
1901 1 CD5  ** TOC MANIPULATION; THUS, A DATA BASE FILE MAY NOT HAVE A TOC TOO
1903 1 CD5  ** LARGE FOR THE XDOTOC INTERNAL BUFFER.
1 BECOM XDOTC
2 (S) FOR OUTPUT OF ALL USER CLASSES FROM AMA TO TERMINAL
3 IF NEXT TOKEN IS NOT EOS
4 THEN
5 IF TOKEN IS A HYPHEN
6 THEN
7 INCREMENT TO NEXT TOKEN
8 IF TOKEN IS NOT A CHARACTER T
9 THEN
10 EXIT TO :ERROR: IF TOKEN IS NOT A CHARACTER P
11 SET OUTPUT UNIT FOR LINE PRINTER
12 ENDIF
13 INCREMENT TO NEXT TOKEN
14 ENDIF
15 IF TOKEN IS NOT EOS
16 THEN
17 EXIT TO :ERROR: IF TOKEN IS NOT A COMMA
18 INCREMENT TO NEXT TOKEN
19 IF TOKEN IS NOT A COMMA
20 THEN
21 EXIT TO :ERROR: IF TOKEN DOES NOT INDICATE A VALID CLASS (O, F, I, S OR B)
22 SET OPTION FOR INDICATED CLASS
23 INCREMENT TO NEXT TOKEN
24 ENDIF
25 IF TOKEN IS NOT EOS
26 THEN
27 EXIT TO :ERROR: IF TOKEN IS NOT A COMMA
28 INCREMENT TO NEXT TOKEN
29 EXIT TO :ERROR: IF TOKEN IS NOT A FOUR CHARACTER NAME
30 INCREMENT TO NEXT TOKEN
31 EXIT TO :ERROR: IF TOKEN IS NOT EOS
32 CALL XREQ TO RETRIEVE DATABASE FILE TOC ENTRY
33 EXIT TO :ERROR: IF NOT LOGGED IN TOC
34 GET TYPE AND SET NAME IN HEADER
35 CONSTRUCT FILE NAME
36 READ FIRST DATABASE FILE TOC RECORD
37 INITIALIZE TOC HEAD TO APPEAR SIMILAR TO AMA TOC
38 READ SUBSEQUENT TOC RECORDS
39 EXIT TO :ERROR: IF FILE ACCESS FAILS
40 DO FOR EACH NON-EMPTY TOC CHAIN
41 INDEX TO TOC ENTRY POINTED TO BY CHAIN HEAD
42 IF NOT FIRST ENTRY IN TOC, i.e., A PREVIOUS NON-NULL CHAIN EXISTED
43 THEN
44 MARK PREVIOUS ENTRY AS AN END OF CHAIN
45 ENDIF
46 ENDDO
47 DO FOR EACH ENTRY IN TOC
48 IF NOT MARKED AS AN END OF CHAIN
49 THEN
50 STORE POINTER TO NEXT SEQUENTIAL TOC ENTRY IN CHAIN POINTER FIELD
51 ENDIF
52 ENDDO
53 EXCLUDE CHAIN 8 (DATA BASE FILES) FROM DISPLAY
54 ENDIF
55 ENDDO
56 IF REFERENCING AMA TOC
57 THEN
58 CALL XREQ TO RETRIEVE AMA TOC
59 ENDIF
60 ENDIF
IF DEBUG AND/OR TRACE FLAGS ARE SET
THEN
EXTEND OPTION TO INCLUDE SYSTEM CLASSES (0, 1, 5 AND 7)
ENDIF
ENDIF
OUTPUT TOC HEADER
DO FOR EACH CLASS INDICATED BY OPTION
OUTPUT CLASS HEADER
LOCATE CLASS CHAIN HEAD
DO UNTIL END OF CHAIN FOUND (-32768)
INDEX TO NEXT TOC ENTRY
IF CHAIN "OINTS WITHIN LIMIT OF BUFFER
THEN
FORMAT NAME & SIZE FIELDS
IF DEBUG AND/OR TRACE FLAGS SET
THEN
FORMAT I-DIM FIELD IN OCTAL
ENDIF
FORMAT TYPE FIELD IN INTEGER
IF CLASS 2 OR 3
THEN
FORMAT TYPE FIELD USING DATA TYPE TABLE
IF CLASS 2
THEN
FORMAT I-DIM & J-DIM FIELDS
ENDIF
ELSE IF CLASS 8
THEN
FORMAT TYPE FIELD USING FILE TYPE TABLE
ENDIF
ENDIF
ELSE PRINT 'DATA LOST' MESSAGE
EXIT PROCESSING FOR THIS CHAIN
ENDIF
PRINT ENTRY
ENDDO
ENDDO
IF PROCESSING AWA TOC
THEN
CALL XSTA TO DISPLAY AWA USAGE STATISTICS
ENDIF
EXIT XDOTC
:ERRORS: EXIT WITH INVALID OUTPUT DEVICE ID
:ERROR4: EXIT WITH SYNTAX ERROR
:ERRORS: EXIT WITH INVALID CLASS DESIGNATOR
:ERROR6: EXIT WITH INVALID UDB FILE NAME
:ERROR7: EXIT WITH UDB FILE ACCESS ERROR
1 END XDOTC
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1976</td>
<td>CALL XQWRT (DATBUF)</td>
</tr>
<tr>
<td>1977</td>
<td>XQWRT WRITES INTERFACE TABLES, SEQUENCE TABLES, DATA ELEMENTS AND ORDE FILES OUT TO A</td>
</tr>
<tr>
<td>1978</td>
<td></td>
</tr>
<tr>
<td>1979</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td></td>
</tr>
<tr>
<td>1981</td>
<td></td>
</tr>
<tr>
<td>1982</td>
<td>XQWRT WRITES INTERFACE TABLES, SEQUENCE TABLES, DATA ELEMENTS AND ORDE FILES OUT TO A</td>
</tr>
<tr>
<td>1983</td>
<td></td>
</tr>
<tr>
<td>1984</td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td></td>
</tr>
<tr>
<td>1986</td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td></td>
</tr>
<tr>
<td>1991</td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td></td>
</tr>
<tr>
<td>1996</td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td>COMMON USED:</td>
</tr>
<tr>
<td>2016</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td>EQUVALENCEx</td>
</tr>
<tr>
<td>2020</td>
<td>+ (X(1), FLAGS ), (X(195), REQPTR)</td>
</tr>
<tr>
<td>2021</td>
<td>+ (X(20), REQBUFF)</td>
</tr>
<tr>
<td>2022</td>
<td></td>
</tr>
<tr>
<td>2023</td>
<td></td>
</tr>
<tr>
<td>2024</td>
<td></td>
</tr>
<tr>
<td>2025</td>
<td></td>
</tr>
<tr>
<td>2026</td>
<td></td>
</tr>
<tr>
<td>2027</td>
<td></td>
</tr>
<tr>
<td>2028</td>
<td></td>
</tr>
<tr>
<td>2029</td>
<td></td>
</tr>
<tr>
<td>2030</td>
<td></td>
</tr>
<tr>
<td>2031</td>
<td></td>
</tr>
</tbody>
</table>
1 BEGIN XDMRT
2 SET WRDNO = 1 (WORD INDEX INTO DATREC WHERE ELEMENT BEGINS)
3 SET MORE = 0 (NUMBER OF DATA REQUESTS IN REBUF)
4 DO FOR ALL UTDB TOC ENTRIES UNTIL CLASS IS DRDE
5 BUILD REQUEST FOR DATA FROM AWAT
6 INCREMENT MORE BY 1
7 IF REQUEST BUFFER IS FULL (MORE=8) THEN
8 PERFORM READ TO REQUEST DATA FROM MANAGER AND HANDLE OUTPUT TO FILE
9 ENDF
10 IF THERE ARE REMAINING REQUESTS (MORE>0) THEN
11 SET NEXT REQUEST TO SAY END OF REQUEST LIST
12 PERFORM READ TO REQUEST DATA FROM MANAGER AND HANDLE OUTPUT TO FILE
13 IF THERE IS A PARTIAL DATA RECORD LEFT (WRDNO>1) THEN
14 PERFORM WRITE TO OUTPUT DATA TO UTDB FILE
15 ENDF
16 DO FOR EACH DRDE UTDB TOC ENTRY
17 STORE DATREC IN UTDB TOC ENTRY
18 CALL DRDFN TO CREATE FILE NAME
19 IF DRDE FILE IS TYPE 3 THEN
20 CALL OPEN TO OPEN FILE AS CORRECT TYPE
21 ERREXIT IF OPEN ERROR TO :ERR1:
22 DJ UNTIL EOF IS READ
23 CALL READ TO READ 1 RECORD
24 ERREXIT IF READ ERROR TO :ERR1:
25 STORE RECORD LENGTH AT FRONT AND REAR OF DATA
26 INCREMENT WRDNO BY LENGTH + 2
27 IF THERE IS ENOUGH DATA TO WRITE (WRDNO>128) THEN
28 PERFORM WRITE TO OUTPUT DATA TO UTDB FILE
29 ENDF
30 IF THERE IS REMAINING DATA (WRDNO>1) THEN
31 PERFORM WRITE TO OUTPUT DATA TO UTDB FILE
32 ENDF
33 SET DATREC TO NEXT AVAILABLE RECORD FOR DATA
34 ELSE
35 CALL OPEN TO OPEN FILE AS TYPE 1
36 ERREXIT IF OPEN ERROR TO :ERR1:
37 COMPUTE TOTAL SIZE OF FILE IN WORDS
38 DO UNTIL ALL DATA IS COPIED TO UTDB (SIZE=0)
39 IF SIZE IS LESS THAN LENGTH TO BE WRITTEN THEN
40 SET LENGTH = SIZE
41 ERREXIT IF WRITE DATA LENGTH ERROR TO :ERR3:
42 CALL READ TO READ LENGTH DATA
43 ERREXIT IF READ ERROR TO :ERR1:
44 CALL WRITF TO WRITE LENGTH DATA
45 ERREXIT IF WRITF ERROR TO :ERR3:
46 DECREMENT SIZE BY LENGTH IN WORDS WRITTEN
47 ENDF
48 ENDF
49 CALL CLOSE TO CLOSE DRDE FILE
50 ERREXIT IF CLOSE ERROR TO :ERR1:
51 ENDF
52 1 EXIT XDMRT
BEGIN READ:
2000 2 CALL XTOE TO MAKE REQUEST
2001 3 ERREJ If THERE IS AN ANA OVERFLOW TO :ERR4:
2002 3 DO FOR NUMBER OF REQUESTS (MORE) IN BUFFER
2003 4 CALL EXEC TO GET DATA FROM SAN
2004 4 SET DATREC AND WORDNO INTO THIS UTDB TO ENTRY
2005 4 INCREMENT WORDNO BY SIZE OF THIS ELEMENT
2006 4 IF THERE IS ENOUGH DATA TO WRITE (WORDNO>128) THEN
2007 5 PERFORM WRITE TO OUTPUT DATA TO UTDB FILE
2008 5 ENDIF
2009 4 ENDDO
2100 3 REINITIALIZE REQUEST BUFFER (MORE=0)
2101 3 END READ
2102 2 *
2103 2 *
2104 2 *
2105 2 *
2106 2 BEGIN WRITE
2107 3 COMPUTE LENGTH TO WRITE IN WORDS AND RECORDS
2108 3 IF THERE IS NOT AT LEAST 1 RECORD TO WRITE THEN
2109 4 SET LENGTH TO MINIMUM OF 1 RECORD
2110 3 ENDIF
2111 3 IF THIS IS A DRDE TYPE 3 FILE THEN
2112 4 ERREJ If THERE ARE MORE BLOCKS THAN SPECIFIED TO :ERR2:
2113 4 ENDIF
2114 3 CALL WRITF TO WRITE LENGTH DATA TO UTDB FILE
2115 3 ERREJ If WRITE ERROR TO :ERR3:
2116 3 DECREMENT DRDE BY LENGTH
2117 3 INCREMENT DATREC BY LENGTH/128
2118 3 IF THERE IS REMAINING DATA (WORDNO>1) THEN
2119 4 CALL XMOV TO MOVE REMAINING DATA UP IN BUFFER
2120 3 ENDIF
2121 2 END WRITE
2122 2 :ERR1:
2123 2 CALL XMSG WITH DRDE NAME AND RC
2124 2 SET ABFLG TO SET A DRDE ERROR
2125 2 GO TO :ERR3:
2126 2 :ERR2:
2127 2 CALL XMSG TO SAY DRDE FILE LARGER THAN SPECIFIED IN TOC
2128 2 SET ABFLG TO SET DRDE FILE ERROR
2129 2 :ERR3:
2130 2 CALL CLOSE TO CLOSE DRDE FILE
2131 2 SET ABFLG TO SAY UTDB FILE ERROR
2132 1 EXIT XDWRT
2133 2 :ERRA:
2134 2 SET ABFLG TO SAY ANA OVERFLOW
2135 2 DO FOR REMAINING REQUESTS
2136 2 CALL EXEC TO FREE CLASS NUMBER
2137 3 ENDDO
2138 1 END XDWRT
FORTRAN CALLING PROCEDURE

CALL HELDS (THREE WORD ARRAY CONTAINING 'HELD ')

C**EXECUTIVE INITIALIZATION SEGMENT MAIN ROUTINE

INPUT

COMMON XE - SUBSTA

OUTPUT

COMMON XB - INITIALIZED ACCORDING TO THE VALUE OF SUBSTA
(SEE APPROPRIATE INITIALIZATION SUBROUTINE)

NOTE?

USES XEIND, XEINI, XEINS, XEINX, XETM

BEGIN XECAL

CASE (:GLOBAL:,:DIRECT:,:EXECUT:,:SEREDT:,:INTEDT:) SUBSTA (-1 TO 3)

GLOBAL: CALL XEIND TO INITIALIZE GLOBAL COMMON

DIRECT: CALL XEIND TO INITIALIZE DYNAMIC COMMON FOR DIRECTIVES

EXECUT: CALL XEIND TO INITIALIZE DYNAMIC COMMON FOR EXECUTION CONTROL

SEREDT: CALL XEIND TO INITIALIZE DYNAMIC COMMON FOR SEQUENCE EDITING

INTEDT: CALL XEIND TO INITIALIZE DYNAMIC COMMON FOR INTERFACE EDITING

ENDCASE

END XECAL
1  C80        FORTRAN CALLING PROCEDURE
40    1  C80    CALL XEIND
41    1  C80    CALL XEIND
42    1  C80    CALL XEIND
43    1  C80    CALL XEIND
44    1  C80    CALL XEIND
45    1  C80    CALL XEIND
46    1  C80    CALL XEIND
47    1  C80    CALL XEIND
48    1  C80    CALL XEIND
49    1  C80    CALL XEIND
50    1  C80    CALL XEIND
51    1  C80    CALL XEIND
52    1  C80    CALL XEIND
53    1  C80    CALL XEIND
54    1  C80    CALL XEIND
55    1  C80    CALL XEIND
56    1  C80    CALL XEIND
57    1  C80    CALL XEIND
58    1  C80    CALL XEIND
59    1  C80    CALL XEIND
60    1  C80    CALL XEIND
61    1  C80    CALL XEIND
62    1  C80    CALL XEIND
63    1  C80    CALL XEIND
64    1  C80    CALL XEIND
65    1  C80    CALL XEIND
66    1  C80    CALL XEIND
67    1  C80    CALL XEIND
68    1  C80    CALL XEIND
69    1  C80    CALL XEIND
70    1  C80    CALL XEIND
71    1  C80    CALL XEIND
72    1  C80    CALL XEIND
73    1  C80    CALL XEIND
74    1  C80    CALL XEIND
FORTRAN CALLING PROCEDURE

CALL XEIME

***************

INITIALIZE XE COMMON FOR FDS START UP

***************

INPUT

COMMON XE - CARTNS, FLAGS, LU, GUAL

XLBD - LIBRARY DIRECTORY FILE

XPDB - PDB AND MDB DIRECTORY

***************

OUTPUT

COMMON XB - INITIALIZED FOR DIRECTIVE LEVEL (SEE XEIND)

AWA - LIBRARY DIRECTORY NAME TABLE

- MOD'S AND PDB'S LOGGED INTO AWA

***************

LOCAL

IDSP - DISPLACEMENT (NEGATIVE) BETWEEN ORIGIN OF XE COMMON AND

ID SEGMENT POINTER LIST

ABSOLUTE ADDRESS OF ORIGIN OF ID SEGMENT LIST

LENGTH OF LIBRARY DIRECTOY RECORD BEING PROCESSED

DISPLACEMENT (NEGATIVE) BETWEEN ORIGIN OF XE COMMON AND

ID SEGMENT NAME FIELD OF ID BEING PROCESSED

MAPPED ADDRESS OF ORIGIN OF XE COMMON

***************

USES CLOSE, EXEC, IAND, OPEN, FRNM, READF, XEINI, XRCPR, XREQ,

XXRT, XXIG, XXLOC, XXNOV, XXMSG, XXDBG, XXABN

***************

(*
BEGIN XEINE

108 1 BEGIN XEINE
109 2 INITIALIZE COMMON TO ZEROS
110 3 SET FILE (ART-DC) NUMBER
111 4 SET STATES TO DIRECTIVE LEVEL
112 5 SET TKMLNG
113 6 INITIALIZE TOKENS
114 7 READ LIBRARY DIRECTORY FIRST RECORD
115 8 EXIT TO :ERROR1: IF READ FAILED
116 9 RETRIEVE NAME OF PROCESSORS IN NOPROC
117 10 READ PROCESSOR NAME RECORD
118 11 EXIT TO :ERROR1: IF READ FAILED
119 12 CALL XER0 TO ALLOCATE AND STORE PROCESSOR DIRECTORY IN AMA
120 13 EXIT TO :ERROR1: IF REQUEST FAILED
121 14 DO FOR EACH ID SEGMENT IN SYSTEM
122 15 DO FOR EACH PROCESSOR IN LIBRARY DIRECTORY
123 16 IF NAMES ARE THE SAME
124 17 THEN
125 18 ELSE
126 19 EXIT IF EXISTANCE OF ID
127 20 ENDIF
128 21 ENDDO
129 22 IF ANY PROCESSORS NOT MARKED
130 23 THEN
131 24 LIST PROCESSORS MISSING ID SEGMENTS
132 25 EXIT XEINE WITH INITIALIZATION FAILURE
133 26 ENDIF
134 27 CALL OPEN TO OPEN PDB/MDB DIRECTORY ( XPDB)
135 28 EXIT TO :ERROR1: IF OPEN FAILED
136 29 CALL READ TO READ 1ST RECORD OF XPDB (LIST OF MDB'S)
137 30 EXIT TO :ERROR1: IF READ FAILED
138 31 PERFORM DLOG TO LOG MDB'S IN AMA
139 32 CALCULATE RECORD NO. OF PDB'S FOR THIS QUALIFIER
140 33 CALL READ TO READ THAT RECORD OF XPDB
141 34 CALCULATE CORRECT INDEX INTO BUFFER (EACH RECORD IS FOR 2 QUALIFIERS)
142 35 CALL CLOSE TO CLOSE XPDB
143 36 PERFORM DLOG TO LOG PDB'S IN AMA
144 37 IF THERE ARE REQUESTS IN THE AMA REQUEST BUFFER, THEN
145 38 CALL XER0 TO PROCESS THE AMA REQUESTS
146 39 ENDIF
147 40 CALL XENO TO INITIALIZE DYNAMIC COMMON FOR DIRECTIVE LEVEL
148 41 EXIT XEINE
149 42 BEGIN DLOG
150 43 DO WHILE THERE ARE DATA BASE FILES TO BE LOGGED
151 44 BUILD AN ENTRY IN AMA REQUEST BUFFER TO ALLOCATE THIS DATA BASE
152 45 IF 8 AMA REQUESTS HAVE BEEN BUILT, THEN
153 46 CALL XER0 TO PROCESS AMA REQUESTS
154 47 ENDIF
155 48 EXIT TO :ERROR1: IF A REQUEST FAILED
156 49 ENDDO
157 50 END DLOG
158 51 :ERROR1: LIBRARY INITIALIZATION ERROR TERMINATION
159 52 1 END XEINE
FORTRAN CALLING PROCEDURE

CALL XEINI

**********

INITIALIZE XE AND XB COMMON FOR INTERFACE TABLE EDITING

**********

INPUT

COMMON XE -

MASTER = MASTER STATE

TOKENS = LEXICAL TOKEN VALUES

NPROC = NUMBER OF ENTRIES IN LIBRARY DIRECTORY

COMPTR = INDEX OF NEXT TOKEN IN COMBUF

COMBUF = COMMUNICATIONS BUFFER

PRCNAM = PROCESSOR NAME FROM EXECUTION CONTROLLER

LIBRARY -

INTWAM = OLD INTERFACE TABLE TO BE EDITED

<XXXXX = DEFAULT INTERFACE TABLE

>XXXXX = PROMPT TABLE

**********

OUTPUT

COMMON XE -

SUBSTA = SUBSTATE -- SET TO 1 IF INIT. ERROR

BUFF = REQUEST BUFFER FOR AWA INPUTS

PRCNAM = PROCESSOR NAME ON INTE DIRECTIVE

COMMON XB -

WKBLNG = LENGTH OF WKBUF (CONSTANT)

WKBUF = WORKING BUFFER CONTAINING IN/E. TABLE WITH SHORT PROMPTS

DIRECT = SUPPORTED INTE DIRECTIVES

NUMDIR = NO. OF ENTRIES IN DIRECT

MINTAB = NAME TABLE NAME

NUMARG = NO. OF ARGUMENTS

ARG = NO. OF WORDS IN SPACES AND HEADER OF WKBUF

LITLEN = NO. OF WORDS IN LITERAL AREA

LITPTR = INDEX TO START OF LITERAL AREA

ISIZES - ARRAY MAPPING ARG. TYPE TO EFFECTIVE LENGTH

**********

NOTES

<XXXXX = DEFAULT INTERFACE TABLE FOR THIS PROCESSOR

>XXXXX = PROMPT TABLE FOR THIS PROCESSOR

**********

USES FILES -

**********

USES ROUTINES -

**********

BEGIN XEINI

IF CALLED AS A RESULT INTE DIRECTIVE, THEN
POSITION TO 1ST TOKEN AFTER 'INTE'

EXIT IF TOKEN IS NOT ':', :ERR1:

INCREMENT TO NEXT TOKEN

EXIT IF TOKEN IS NOT A NAME :ERR2:

SET PRCNAM TO THIS NAME

INCREMENT TO NEXT TOKEN

SET INTNAM TO 0

EXIT IF TOKEN IS NOT EOS (END-OF-STATEMENT), THEN

EXIT IF TOKEN IS NOT A COMMA :ERR2:

INCREMENT TO NEXT TOKEN

If TOKEN IS NOT EOS, THEN

IF TOKEN IS A NAME, THEN

SET INTNAM TO THE NAME

INCREMENT TO NEXT TOKEN

ENDIF

IF TOKEN IS NOT EOS, THEN

EXIT IF TOKEN IS NOT A COMMA :ERR2:

INCREMENT TO NEXT TOKEN

IF TOKEN IS A NAME, THEN

SET NEWTAB TO THE NAME

INCREMENT TO NEXT TOKEN

ENDIF

EXIT IF TOKEN IS NOT EOS :ERR2:

ENDIF

ELSE

INTNAM AND PRCNAM ARE INITIALIZED BY THE EXECUTION CONTROLLER

NEWTAB IS SET TO 'EINTAB'

MESSAGE THAT INTERFACE TABLE EDITOR HAS BEEN INVOKED

ENDIF

MAKE MANAGER REQUEST FOR LIBRARY DIRECTORY FILE

EXIT IF REQUEST IS UNSUCCESSFUL :ERR11:

START SEARCH UNTIL ALL OF DIRECTORY IS SEARCHED, OR

EXIT IF PROCESSOR PRCNAM IS FOUND

ELSE

INCREMENT TO NEXT ENTRY

ENDLOOP

EXIT :ERR8:

ENDSEARCH

SET IVENS TO VERSION NO. OF DIRECTORY ENTRY FOUND

IF INTNAM = 0, THEN

CONSTRUCT THE NAME OF THE DEFAULT INTERFACE TABLE AS "PRCNAM"

OPEN THE DEFAULT INTERFACE TABLE FILE

EXIT IF OPEN FAILED :ERR2:

READ THE DEFAULT INTERFACE TABLE

EXIT IF THE READ FAILED :ERR16:

READ LITERAL RECORD FROM DEFAULT INTERFACE TABLE FILE INTO BOTTOM OF WKBUFF

EXIT IF READ FAILED :ERR14:

ELSE

MAKE MANAGER REQUEST FOR INTNAM INTERFACE TABLE

EXIT IF ERROR OR COULD NOT FIND :ERR4:

ENDIF

EXIT IF VERSION NO. OF TABLE TO BE EDITED :ERR10:

CONSTRUCT NAME OF PRCNAM INTERFACE TABLE AS "PRCNAM"

OPEN THE PRCNAM TABLE

EXIT IF OPEN FAILED :ERR15:
279  2  READ 2ND RECORD OF THIS PROMPT TABLE
280  2  ERREXIT IF READ FAILED :ERR17:
281  2  MOVE LITERAL DATA ENTRIES UP FROM BOTTOM OF WKBUF TO AREA FOLLOWING
282  2  THE PROMPT TABLE
283  2  THIS IS DONE BY LOOPING THROUGH ALL ARGUMENTS TO FIND THE
284  2  LITERAL DISPL. THAT MATCH EACH LITERAL DATA ENTRY
285  2  THE TYPE OF EACH ARGUMENT FOUND TO HAVE LITERAL DATA IS USED IN THE
286  2  MOVING PROCESS TO DETERMINE THE NUMBER OF WORDS IN EACH DATA ELEMENT
287  2  AS WELL AS THE NUMBER OF WORDS TO LEAVE FOR INCOMPLETE ELEMENTS.
288  1  EXIT XEINI
289  2  :ERR2: "SYNTAX ERROR ... "
290  2  :ERR4: "INTNM" NOT FOUND
291  2  :ERR6: INVALID NAME FIELD
292  2  :ERR8: "PRCNAM" NOT IN DIRECTORY
293  2  :ERR10: VERSION OF INTNM DOES MATCH CURRENT VERSION
294  2  :ERR11: XEINI OP 1 INITIALIZATION ERROR
295  2  :ERR12: XEINI OP 2 INITIALIZATION ERROR
296  2  :ERR13: XEINI OP 3 INITIALIZATION ERROR
297  2  :ERR14: XEINI OP 4 INITIALIZATION ERROR
298  2  :ERR15: XEINI OP 5 INITIALIZATION ERROR
299  2  :ERR16: XEINI OP 6 INITIALIZATION ERROR
300  2  :ERR17: XEINI OP 7 INITIALIZATION ERROR
301  2  SET SUBSTATE TO DIRECTIVE LEVEL TO INDICATE THE ERROR
302  1  END XEINI
304 1 CD0 FORTRAN CALLING PROCEEDURE
305 1 CD0
306 1 CD0
307 1 CD0 CALL XEINS
308 1 CD0
309 1 CD0
310 1 CD0 INITIALIZE XE AND XB COMMON FOR SEQUENCE TABLE EDITING
311 1 CD0
312 1 CD0
313 1 CD0
314 1 CD0
315 1 CD0 INPUT
316 1 CD0 COMMON XE - CARTRG, COMBUF, COMPTR, LU, NOPROC, TOKENS
317 1 CD0
318 1 CD0
319 1 CD0
320 1 CD3 OUTPUT
321 1 CD3 COMMON XB - REQBUF, REQPTR, SUBSTA,
322 1 CD3
323 1 CD3 COMMON XB - DEBUG, DIRECT, NEWTAB, NUMDIR, NUMENT,
324 1 CD3 OLDTAB, PRMTAB, WKBNG, WKBUF, XLIBD
325 1 CD3
326 1 CD3
327 1 CD3
328 1 CD3 NOTES
329 1 CD3
330 1 CD3
331 1 CD3 USES FILES - XSPRM - SEQUENCE TABLE EDITOR PROMPT FILE
332 1 CD3
333 1 CD3
334 1 CD5 USES ROUTINES
335 1 CD5
336 1 CD5 EXEC
337 1 CD5 IAND
338 1 CD5 OPEN
339 1 CD5 READF
340 1 CD5 XRPR
341 1 CD5 XREQ
342 1 CD5 XREXT
343 1 CD5 XRMO
344 1 CD5 XRMSG
345 1 CD5 XUDBG
346 1 CD5
347 1 CD5
BEGIN XEMS

BUILD AWB REQUEST TO RETRIEVE "XLIBD"

ERREXIT IF TOKEN IS NOT COMMA :ERR02:

INCREMENT TO NEXT TOKEN

IF TOKEN IS NAME, THEN

BUILD AWB REQUEST TO RETRIEVE OLDTAB

RETAIN THIS NAME AS OLDTAB

INCREMENT TO NEXT TOKEN

ELSE

SET OLDTAB TO ZERO

ENDIF

ERREXIT IF TOKEN IS NOT COMMA :ERR02:

INCREMENT TO NEXT TOKEN

ERREXIT IF TOKEN IS NOT NAME :ERR02:

RETAIN THIS NAME AS NEWTAB

IF NEWTAB NOT EQUAL TO OLDTAB, THEN

BUILD AWB REQUEST TO VERIFY EXISTENCE OF NEWTAB

ENDIF

CALL XERG TO PROCESS AWB REQUEST(S)

IF NEWTAB NOT EQUAL TO OLDTAB, THEN

ERREXIT IF NO ERROR FROM XERG :ERR12:

ENDIF

IF OLDTAB NOT ZERO, THEN

ERREXIT IF 2ND REQUEST (RETRIEVE OLDTAB) FAILED :ERR04:

SET PROMPT MODE AS UPDATE

SET NO. ENTRIES AS OLDTAB SIZE / 7

READ OLDTAB INTO WORKING BUFFER

ERREXIT IF FIRST REQUEST (RETRIEVE "XLIBD") FAILED :ERR01:

READ "XLIBD INTO COMMON

ELSE

SET PROMPT MODE AS CREATE

SET NO. ENTRIES TO ZERO

ENDIF

OPEN, READ AND CLOSE FILE XSPRM

SET COUNT AND SEREDIT DIRECTIVES INTO XSP COMMON

SET SUBSTATE FLAG TO SEQ. EDIT. (=2)

EXIT XEMS

:ERR01: CALL XMSG - "INITIALIZATION ERROR ..."

:ERR02: CALL XMSG - "SYNTAX ERROR"

:ERR04: DEFAULT MESSAGE TO '...NOT FOUND'

IF ERROR WAS NO AWB SPACE THEN

SET MSG TO '...NO AWB SPACE'

ENDIF

CALL XMSG TO DISPLAY MESSAGE

:ERR12: CALL XMSG - '....ALREADY EXISTS'

END XEMS
FORTRAN CALLING PROCEDURE

CALL XEINT

INTERFACE TABLE LITERAL AREA INITIALIZATION

INPUT

COMMON XB - LITPTR, NUMARS, WKBLSN, WKBUF

NOTE: WKBUF IS INPUT WITH THE INTERFACE TABLE'S CHARACTERISTICS IN THE TOP AND THE "PACKED" LITERAL AREA IN THE BOTTOM.

OUTPUT

COMMON XB - LITPTR, LITOWN, NARL, W^KUF

NOTE: WKBUF IS OUTPUT WITH THE LITERAL AREAS IN THEIR "UNPACKED" FORM.

USES Routines

INCLUDE XEXIT
INCLUDE XRMOV
INCLUDE XRXXB
INCLUDE XRXXG
INCLUDE XRSET
BEGIN XEINT

DO UNTIL ALL LITERAL ENTRIES HAVE BEEN PROCESSED

START SEARCH UNTIL ALL ARGUMENTS SEARCHED

EXITIF 'DISP' FIELD FOR ARGUMENT = DISP. OF LITEAL ENTRY, AND

E-BIT IS ON

BUILD 3 WORD LITERAL ENTRY (3, I-SUB, J-SUB)

EXITIF 'DISP' FIELD FOR ARGUMENT = DISP. OF LITERAL ENTRY, AND

D-BIT IS ON

IF ARGUMENT IS COMPLETE (C-BIT ON), THEN

CREATE BIT MASK WORD(S) IN NEW LITERAL ENTRY AREA

ELSE

MOVE BIT MASK WORDS UP TO NEW LITERAL ENTRY AREA

ENDIF

DO UNTIL ALL BITS OF BIT MASK PROCESSED

DETERMINE NUMBER OF CONTIGUOUS BITS ON (OR OFF)

AND MOVE CORRESPONDING NUMBER OF DATA WORDS

(COR O'S) INTO NEW LITERAL ENTRY AREA

ENDDO

ORELSE

INCREMENT TO NEXT ARGUMENT

ENDLOOP

EREREIT IF NOT CALLED BY LIBRARY MAINTENANCE :ERRS:

SKIP TO NEXT LITERAL ENTRY

ENDSEARCH

ENDCO

EXIT TO :RETURN:

:ERRS: CALL XRMSC - 'SYSTEM INITIALIZATION ERR 5 '

:RETURN:

1 END XEINT
FORTRAN CALLING PROCEDURE

CALL XEINX

initialize xe and xb common for execution controller

common xe - combuf, comptr, flags, lu, massta, nopen, tokens

ava - sequence table, library directory name table

output

common xe - comptr, massta, rebuf, reptr, segend, segname, segptr

serstr, subs

common xb - libd, nopen2, seqlng, wkblng, wkbuf

uses exec, prtm, xreg, xrest, xrig, xmov, xmsg, xudeg, xvabn
494 1 BEGIN XEINX
495 2 IF INITIALIZATION FROM DIRECTIVE
496 3 THEN
497 4 IF DIRECTIVE IS NAME
498 5 THEN
499 6 EXIT TO :ERROR2: IF NEXT TOKEN IS NOT EOS
500 7 ELSE
501 8 IF DIRECTIVE IS AUTO
502 9 THEN
503 10 IF TOKEN IS A HYPHEN
504 11 THEN
505 12 INCREMENT TO NEXT TOKEN
506 13 EXIT TO :ERROR3: IF TOKEN IS NOT THE NAME 'T'
507 14 CHANGE EXECUTION CONTROL MODE TO AUTO-T
508 15 INCREMENT TO NEXT TOKEN
509 16 ENDIF
510 17 ENDIF
511 18 EXIT TO :ERROR2: IF NEXT TOKEN IS NOT A COMMA
512 19 INCREMENT TO NEXT TOKEN
513 20 EXIT TO :ERROR2: IF NEXT TOKEN IS NOT A NAME
514 21 STORE NAME IN SEGNAM
515 22 CALL XREG TO RETRIEVE SEQUENCE TABLE
516 23 EXIT TO :ERROR4: IF NON-ZERO RETURN CODE
517 24 INCREMENT TO NEXT TOKEN
518 25 SET SEGSIZ TO FIRST SEQUENCE NUMBER
519 26 SET SEGRAD TO LAST SEQUENCE NUMBER
520 27 IF TOKEN NOT EOS
521 28 THEN
522 29 EXIT TO :ERROR2: IF TOKEN NOT A COMMA
523 30 INCREMENT TO NEXT TOKEN
524 31 IF TOKEN IS AN INTEGER
525 32 THEN
526 33 STORE STARTING RANGE NUMBER
527 34 SEARCH SEQUENCE NUMBERS FOR STARTING VALUE
528 35 EXIT TO :ERROR13: IF NOT FOUND
529 36 INCREMENT TO NEXT TOKEN
530 37 ENSIF
531 38 IF TOKEN NOT END
532 39 THEN
533 40 EXIT TO :ERROR2: IF TOKEN NOT A COMMA
534 41 INCREMENT TO NEXT TOKEN
535 42 EXIT TO :ERROR2: IF TOKEN NOT AN INTEGER
536 43 ERROR TO :ERROR5: IF 'ENDING' SEG # < BEGINNING SEG #
537 44 SEARCH SEQUENCE NUMBERS FOR ENDING VALUE
538 45 EXIT TO :ERROR13: IF NOT FOUND
539 46 INCREMENT TO NEXT TOKEN
540 47 EXIT TO :ERROR2: IF TOKEN NO. :0S
541 48 ENSIF
542 49 ENSIF
543 50 SET SEGRAD TO SEGSIZ
544 51 ENSIF
545 52 ENSIF
546 53 INITIALIZE DYNAMIC COMMON WITH NUMBER OF PROCESSORS AND DIRECTORY NAME TABLE
547 54 EXIT TO :ERROR1: IF INITIALIZATION FAILS
548 55 1 EXIT KEIN
549 56 :ERROR1: INITIALIZATION FAILURE TERMINATION
550 57 :ERROR2: SET SUBSTA TO DIRECTIVE LEVEL & EXIT WITH SYNTAX ERROR
562 1 C0**********
563 1 C00 FORTRAN CALLING PROCEDURE:
564 1 C00 CALL XELS (SEGNUM) TO CALL SEGMENT FROM MAIN
565 1 C00 CALL XERTN TO RETURN TO MAIN PROGRAM
566 1 C00**********
567 1 C00 YELS ALLOWS A MAIN PROGRAM TO "CALL" A SEGMENT AND
568 1 C00 THE ENTRY POINT XERTN PASSES CONTROL BACK TO THE MAIN PROGRAM
569 1 C00**********
570 1 C00 INPUTS IN CALLING SEQUENCE:
571 1 C02 SEGNUM - (INTEGER, 3 WORDS) ARRAY CONTAINING NAME OF THE
572 1 C02 SEGMENT TO BE LOADED
573 1 C02**********
574 1 C05 SUBROUTINES AND FUNCTIONS CALLED:
575 1 C05 EXEC
576 1 C05**********
577 1 C05 NOTES
578 1 C05 1) XELS CONTAINS 2 ENTRY POINTS: XELS AND XERTN
579 1 C05 2) SEGMENT CALLED BY MAIN MUST BEGIN WITH A PROGRAM
580 1 C05 3) IN ORDER TO RETURN TO MAIN, A SEGMENT MUST "CALL XERTN"
581 1 C05 FOLLOWED BY A CALL TO MAIN WHICH IS NOT EXECUTED
582 1 C05 4) WHEN LOADER IS RUN, THE FOURTH PARAMETER MUST BE 1
583 1 C05 INDICATING TO LOAD MAIN PLUS SEGMENTS
584 1 C05**********
585 1 C05 BEGIN XELS
586 1 C05 :XELS:
587 2 :CALL .ENTR TO RETRIEVE PARAMETERS AND RETURN ADDRESS
588 2 :CALL .ADDRES OF THE SEGMENT NAME INTO EXEC PARAMETER LIST
589 2 :CALL EXEC TO LOAD AND PASS CONTROL TO SEGMENT
590 2 :XERTN:
591 2 :RETURN TO MAIN PROGRAM
592 2 END XELS
1 CDO  FORTRAN CALLING PROCEDURE
1 CDO
1 CDO  CALL XESCM
1 CDO
1 CDO  **********************
1 CDO
1 CDO  XESCM IS CALLED BY XEXEC AFTER COMPLETION OF ALL AUTOMATIC AND
1 CDO  SEMI-AUTOMATIC EXECUTIONS TO ASSURE THAT ALL DO AND SCAN PROCESSOR
1 CDO  CONTROL DATA IS PURGED FROM THE ANA AND ALL INCOMPLETE DATA BOX
1 CDO  FILES ARE PURGED FROM THE SYSTEM.
1 CDO
1 CDO  **********************
1 CDO  1 CDO  INPUT
1 CDO  1 CDO  XE COMMON - CARTRG
1 CDO  1 CDO  ANA - GOSTK, BSCTB (SEE DO AND SCAN PROCESSORS
1 CDO  1 CDO  **********************
1 CDO
1 CDO  OUTPUT
1 CDO  XE COMMON - RERBF, RERPTR
1 CDO  XB COMMON - SCRATCH
1 CDO
1 CDO  **********************
1 CDO
1 CDO  EXTERNAL ROUTINES
1 CDO  CLOSE, EXEC, PURGE, XREQ, XRIG, XMOV, XRUNG
1 CDO
1 CDO  **********************
1 CDO
1 CDO  1 *
1 CDO
1 CDO  1 *
1 CDO
1 CDO  1 *
1 CDO
1 CDO  1 EEGIN XESCM
1 CDO
1 CDO  2 BUILD REQUESTS TO DELETE GOSTK AND RETRIEVE BSCTB; THEN DELETE IT
1 CDO  2 CALL XREQ TO ATTEMPT REQUESTS
1 CDO  2 ISSUE MESSAGE XE18 FOR SUCCESSFUL DELETES
1 CDO  2 IF XREQ COMPLETED REQUESTS, I.E., BSCTB EXISTED, THEN
1 CDO  3 DO FOR EACH SCAN CONTROL ENTRY IN BSCTB
1 CDO  4 CALL CLOSE TO CLOSE THE DATA BOX FILE ASSOCIATED WITH THIS SCAN
1 CDO  4 CALL PURGE TO PURGE THE FILE
1 CDO  4 IF PURGE RETURNED AN ERROR, THEN
1 CDO  5 CALL XRMSG TO DISPLAY WARNING MESSAGE
1 CDO  4 ENDIF
1 CDO  4 BUILD REQUEST TO DELETE DATA BOX FROM TOC
1 CDO  5 ENDDO
1 CDO  5 CALL XREQ TO DELETE DATA BOXES FROM ANA TOC
1 CDO  2 ENDIF
1 CDO
1 CDO  1 END XESCM
655 1 CD1 FDS EXECUTIVE TASK MAIN PROGRAM, SCHEDULED BY FDS MANAGER.
656 1 CD1 ONE PROGRAM PER SIGNED-ON USER
657 1 CD1
658 1 CD1
659 1 CD1************
660 1 CD1 INPUT
661 1 CD1 SCHEDULING PARAMETERS - LU, CLASH, QUAL, FLAGS (SEE XE COMMON)
662 1 CD1
663 1 CD1************
664 1 CD3 OUTPUT
665 1 CD5 COMMON XE - COMBUF, COMPTR, NASST, SUBSTA, PLUS XEINE
666 1 CD5 INITIALIZATION
667 1 CD5 COMMON XB - INITIALIZATIONS FROM XEIND, XEINI, XEINS, XEINX
668 1 CD3
669 1 CD5************
670 1 CD5 NOTES
671 1 CD5 USES BMPAR, XDCLD, XDCLF, XELDS, XINTE, XRCPR, XRMSG, XSERF,
672 1 CD5 XTCOM, XTCMT
673 1 CD5
674 1 CD5 THE CALLS TO XELDS PROVIDE LINKAGE TO THE INITIALIZATION SEGMENT
675 1 CD5 XECAL AND DIRECTIVE SEGMENTS XDCLD AND XDCLF.
676 1 CD5
677 1 CD5 THE LOOP STRUCTURE ASSOCIATED WITH EXECUTION CONTROLS OCCURS
678 1 CD5 BECAUSE OF PARTITION SIZE LIMITATIONS WHICH PROHIBIT XXCMT FROM
679 1 CD5 CALLING XSERF AND XINTE DIRECTLY. LOGIC FLOW BETWEEN THESE
680 1 CD5 MODULES IS GOVERNED BY THE VALUE OF SUBSTA. CYCLING TERMINATES
681 1 CD5 WHEN NASST IS SET TO THE DIRECTIVE LEVEL.
682 1 CD5
683 1 CD1************
BEGIN EXEC
RETRIEVE SCHEDULING PARAMETERS AND SET LU, CLASMO, QUAL & FLAGS
CALL XEINI TO INITIALIZE GLOBAL COMMON
DO FOREVER -- TERMINATES INSIDE HANDLER FOR XOFF
CALL XTCM FOR INPUT OF DIRECTIVE
IF ERROR OR NOT A VALID DIRECTIVE NAME
THEN
ISSUE MESSAGE 06
ELSE IF NAME IS INTE
THEN
SET STATES TO INTE LEVEL
CALL XEINI TO INITIALIZE DYNAMIC COMMON
EXIT TO :RESET; IF ERROR
CALL XINTO TO EDIT TABLE
ELSE IF NAME IS SERE
THEN
SET STATES TO SERE LEVEL
CALL XEINS TO INITIALIZE DYNAMIC COMMON
EXIT TO :RESET; IF ERROR
CALL XSERE TO EDIT TABLE
ELSE IF NAME IS FOR SOME EXECUTION CONTROL OPTION
THEN
SET STATES TO APPROPRIATE EXECUTION CONTROL MODE
DO UNTIL MASSTA IS AT DIRECTIVE LEVEL
CALL XEINI TO INITIALIZE DYNAMIC COMMON
EXIT TO :RESET; IF ERROR
CALL XXCNT TO PERFORM EXECUTIONS
IF SUBSTA IS SET TO SERE LEVEL
THEN
CALL XEINS TO REINITIALIZE DYNAMIC COMMON
EXIT TO :RESET; IF ERROR
CALL XSERE TO SUPPORT EXECUTION CONTROL
ENDIF
IF SUBSTA IS SET TO INTE LEVEL
THEN
CALL XEINI TO REINITIALIZE DYNAMIC COMMON
EXIT TO :RESET; IF ERROR
CALL XINTO TO SUPPORT EXECUTION CONTROL
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
:RESET:
IF SUBSTA IS NOT DIRECTIVE LEVEL
THEN
CALL XEIND TO REINITIALIZE DYNAMIC COMMON
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDING
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDER
REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR
1 CDO FORTRAN CALLING PROCEDURE
2
3 CDO
4
5 CDO CALL XINTE
6
7 C************
8
9 CDO1 OVERLAY INTERFACE ROUTINE FOR INTERFACE TABLE EDITOR
10 CDO1
11 C************
12
13 CDO2
14
15 CDO2 INPUT
16
17 CDO2 COMMON XE -
18 CDO2 LU = USER'S LOGICAL UNIT NO.
19
20 CDO2 COMMON XB -
21 CDO2 DEBUG = DEBUG AND TRACE FLAG FOR INTERFACE
22 CDO2 TABLE EDITOR ROUTINES
23
24 C************
25
26 CDO5 NOTES
27
28 CDO5 USES Routines
29 CDO5 XINIX
30 CDO5 XUDBG
31 CDO5 XERTW
32
33 C************
34
35 C
36
37 C XINTE IS THE INTERFACE ROUTINE FOR THE INTERFACE TABLE EDITOR
38 C WHEN CALLED IN THE FDS EXECUTIVE'S OVERLAY STRUCTURE.
39 C
40 C
41 C BEGIN XINTE
42 C CALL XINIX TO EXECUTE INTERFACE TABLE EDITOR
43 C CALL XERTW TO RETURN TO XEXEC IN MAIN SEGMENT
44 C END XINTE
Fortran Calling Procedure

CALL XIMIX

Main Program for Interface Table Editor

Input

Common XE -
COMBUF = Terminal Communications Output Buffer
COMPT = Index to Next Token in COMBUF
BACKSL = Token for "/"
TOKENS = Identifying Values of Tokens in COMBUF

Common XB -
LITLEN = Length of Literal Area of Interface Table in WBUF
NARG = Index to Start of Short Prompt Area of WBUF
NEWTAB = Name of Interface Table to Be Created By This Edit. INPUT = 0 IF XINTER IS CALLED By FDS Library Maintenance Program.
NUMARG = Number of Arguments in This Interface Table
MKBUF = Working Buffer Containing Interface Table and Short Prompts with Literal in Expanded Form.

Output

Common XE -
MASTA = Master Status Set To 'Z' Level If Error Or If 'Z' Input

Common XB -
PRMTWO = Current Prompt Mode. Set To 5 (Modify Mode) If '/? Input
MKBUF = Working Buffer Containing Interface Table
WITH LITERALS IN COMPACTED FORM

AMA = XXXXX = New Interface Table and Literal Area (Not Done If Called 'B' Library Maintenance)

Notes

Uses Routines

EXEC
XREG
XILIT
XIPRM
XINPT
1 CD5
2 XBMV
3 CD5
4 XBMSE
5 CD5
6 XTCOM

********
1 *
2 * THE INTERFACE TABLE EDITOR IS ENTERED AS A RESULT OF THE 'INIT' DIRECTIVE
3 * OR FROM THE EXECUTION CONTROLLER TO COMPLETE AN INTERFACE TABLE. THE DIRECTIVE
4 * PROVIDES THE NAME OF THE TABLE TO BE EDITED AND THE NAME FOR THE NEW
5 * TABLE. THE EDITOR INTERACTS WITH THE USER IN ORDER TO ACQUIRE DATA VALUES
6 * OR VARIABLE NAMES FOR EACH OF THE PARAMETERS IN THE INTERFACE TABLE.
7 * NOTE: ALL INITIALIZATION, INCLUDING MBUFF (OLD INTERFACE TABLE),
8 * HAS BEEN PERFORMED BY XEINI.
9 *
10 BEGIN XEINI
11 1 IF DP (NO. OF PARAMETERS) > 0, THEN
12 2 SET ARGNO (NO. OF CURRENT ARGUMENT BEING PROCESSED) TO 0
13 3 PRMTND = 1 => CREATE M MODE
14 4 PRMTND = 3 => CREATE A MODE
15 5 PRMTND = 4 => CREATE CONTINUE MODE
16 6 PRMTND = 5 => MODIFY MODE
17 7 DO UNTIL 'EXIT' OR '=' IS ENTERED
18 8 CALL XIPRM TO CONSTRUCT A PROMPT BASED ON PRMTND, SIZE, TYPE, AND STATUS
19 10 OF NEXT ARGUMENT
20 11 CALL XTCOM TO PROMPT USER AND RETURN PARSED INPUT
21 12 IF 'X' WAS NOT ENTERED, THEN
22 13 IF '\' WAS ENTERED, THEN
23 14 SET PRMTND TO 5
24 15 ELSE
25 16 CALL XINTP TO PROCESS THE USER'S INPUT
26 17 ENDIF
27 18 ENDIF
28 19 END
29 20 END
30 21 IF 'X' WAS ENTERED, THEN
31 22 SET RETURN CODE INDICATING X
32 23 (I.E. MASSTA = 0)
33 24 ELSE
35 25 COMPRESS THE LITERAL LIST AREA
36 26 ENDIF
37 27 END
38 28 STORE INTERFACE TABLE AS NEWNAME
39 29 IF STORE INTO AW FAILED, THEN
40 30 SET MASSTA TO INDICATE DIRECTIVE LEVEL (=0)
41 31 ELSE
42 32 SET GOOD RETURN CODE
43 33 ENDIF
44 34 END XEINI
192 1 = CONSTRUCT PROMPT TO BE ISSUED
193 2 BEGIN XIPRN
194 3 DO UNTIL A PROMPT IS CONSTRUCTED
195 4 IF PRTHD = 5, THEN
196 5 CONSTRUCT A ";" PROMPT
197 6 ELSE
198 7 IF PRTHD = 4 (CONTINUE NODE), OR
199 8 PRTHD = 6 (CONTINUE HERE NODE), THEN
200 5 IF ARGNO IS A SCALAR, THEN
201 6 CONSTRUCT PROMPT AS \"ARG\"; OR \"ARG\"
202 7 ELSE
203 8 IF PRTHD NOT = 6, THEN
204 9 IF THERE ARE NO EMPTY SLOTS BEYOND LAST ENTERED (LAST)
205 10 SET PRTHD TO 4
206 11 COMPUTE CURRENT ELEMENT NO. (SUBSCRIPT) FROM CURRENT INDEX
207 12 ENDIF
208 13 IF ARGNO IS DOUBLE SUBSCRIPTED PARAMETER, THEN
209 14 COMPUTE I & J FROM ISUB AND 1-DIMENSION (IDIM)
210 15 CONSTRUCT PROMPT AS \"ARG=(I,J)\";
211 16 ELSE
212 17 CONSTRUCT PROMPT AS \"ARG=(ISUB)\"
213 18 ENDIF
214 19 ENDIF
215 20 ELSE
216 21 IF ARGNO = GP, THEN
217 22 SET PRTHD TO 5
218 23 ELSE
219 24 INCREMENT ARGNO TO NEXT PARAMETER
220 25 SET IFLAG TO 1, 2, OR 3 INDICATING I, 0, OR 0
221 26 SET LAST ENTERED INDICATOR (LAST) TO 0
222 27 IF PRTHD = 5, THEN
223 28 IF SOME DATA VALUE(S) OR PARM NAME EXISTS FOR ARGNO, THEN
224 29 CALL XILSD TO LIST DATA FOR THIS ARGUMENT
225 30 ENDIF
226 31 CONSTRUCT PROMPT AS \"ARG\"; OR 0 OR =0
227 32 ELSE
228 33 IF ARGNO IS MARKED INCOMPLETE, THEN
229 34 IF A PARTIAL LITERAL LIST EXISTS, OR
230 35 THIS ARGUMENT IS A SCALAR, THEN
231 36 COMPUTE ISUB AS FIRST EMPTY ELEMENT
232 37 IF DOUBLY SUBSCRIPTED PARAMETER, THEN
233 38 COMPUTE I & J FROM ISUB AND 1-DIMENSION (IDIM)
234 39 CONSTRUCT PROMPT \"ARG=(I,J)\"
235 40 ELSE
236 41 CONSTRUCT PROMPT \"ARG=(ISUB)\"
237 42 ENDIF
238 43 ELSE
239 44 CONSTRUCT PROMPT \"ARG\"; OR 0 OR =0
240 45 ENDIF
241 46 ENDIF
242 47 ENDIF
243 48 ENDIF
244 49 ENDIF
250 50 ENDIF
254 1 CDO  FOR CALLING PROCEDURE
255 1 CDO  CALL XIPMT
257 1 CDO  
258 1 CDO  ********
259 1 CDO  PROMPT DIRECTIVE PROCESSOR
260 1 CDO  
261 1 CDO  ********
263 1 CDO  INPUT
264 1 CDO  
266 1 CDO  COMMON IE - COMBUF, COMPTA, TOKENS
267 1 CDO  
268 1 CDO  ********
269 1 CDO  OUTPUT
270 1 CDO  
272 1 CDO  COMMON IX - ARGNO, PRMTND
273 1 CDO  
274 1 CDO  ********
275 1 CDO  XIPMT PROCESS IS THE PROMPT DIRECTIVE
276 1 CDO  BEGIN XIPMT
277 2  ERREXIT IF Token is NOT comma :ERROR2:
278 2  POSITION TO NEXT TOKEN
279 2  ERREXIT IF Token is NOT NAME :ERROR2:
280 2  ERREXIT IF Token(s) FOLLOW THE NAME :ERROR2:
281 2  IF Name is 'W', THEN
282 3  SET PRMTND TO 9
283 2  ELSE
284 3  IF Name is 'A', THEN
285 4  SET PRMTND TO 3
286 2  ELSE
287 4  ERREXIT :ERROR2:
289 2  ENDIF
290 2  EXIT TO :RETURN:
292 2  :ERROR2: CALL XNSG FOR 'INVALID SYNTAX'
293 2  :RETURN:
294 1 END XIPMT
* XILST PROCESSES THE LIST DIRECTIVE

1 BEGIN XILST

1 IF TOKEN IS '-', THEN

2  POSITION TO NEXT TOKEN

3 ERREXIT IF TOKEN IS NOT NAME : ERR02:

4 ERREXIT IF NAME IS NOT 'C', 'Y', OR 'A' : ERR02:

5 SET MODEFG TO INDICATE SPECIFIED MODE (C=1, V=2, A=3)

6 ELSE

7 SET MODEFG TO 2

8 ENSDF

9 IF TOKEN IS EOS, THEN

10 WRITE A HEADER LINE INDICATING TABLE NAME, PROCESSOR VERSION

11 AND STATUS

12 DO UNTIL ALL ARGUMENTS HAVE BEEN PROCESSED

13 IF MODEFG = 1 OR MODEFG = 3, THEN

14 CALL XICH N TO WRITE CHARACTERISTICS OF THIS ARGUMENT

15 ENSDF

16 IF MODEFG = 2 OR MODEFG = 3, THEN

17 CALL XILSD TO WRITE DATA VALUES OF THIS ARGUMENT

18 ENSDF

19 ENDDO

20 ELSE

21 DO UNTIL EOS IS REACHED

22 ERREXIT IF TOKEN IS NOT COMMA : ERR02:

23 ERREXIT IF NEXT TOKEN IS NOT NAME : ERR02:

24 SET ARGNO TO 1

25 START SEARCH DO UNTIL ALL ARGUMENTS HAVE BEEN PROCESSED

26 EXIT IF NAME = ARGNO'S NAME IN PROMPT TABLE

27 IF MODEFG = 1 OR MODEFG = 3, THEN

28 CALL XICH N TO WRITE CHARACTERISTICS OF THIS ARGUMENT

29 ENSDF

30 IF MODEFG = 2 OR MODEFG = 3, THEN

31 CALL XILSD TO WRITE DATA VALUES OF THIS ARGUMENT

32 ENSDF

33 ENDDO

34 ENDLOOP

35 PRINT MESSAGE THAT NAME IS NOT A VALID PARAMETER

36 ENSEARCH

37 INCREMENT TO NEXT TOKEN

38 ENDDO

39 EXIT TO :RETURN:

40 :ERR02: CALL XRMST TO WRITE 'INVALID SYNTAX'

41 :RETURN:

42 END XILST
1 COO FORTRAN CALLING PROCEDURE
2 COO CALL XISUB
3 COO
4 COO EFFCTVE SUBSCRIPT CALCULATION ROUTINE
5 COO
6 COO INPUT
7 COO COMMON X& - COMBUF, COMPIA, TOKENS
8 COO COMMON X& - IDIM, ISIZE, LENEFF
9 COO
10 COO OUTPUT
11 COO COMMON X& - IRET, ISUB
12 COO
13 COO USES ROUTINES
14 COO COMMON X& - XMSG
15 COO
16 COO
17 COO XISUB IS CALLED TO CALCULATE AN EFFECTIVE SUBSCRIPT (ISUB) FROM
18 COO THE INPUT SUBSCRIPT
19 COO BEGIN XISUB
20 COO INCREMENT TO NEXT TOKEN
21 COO ERREXIT IF TOKEN IS NOT INTEGER VALUE :ERR14:
22 COO IF IDIM FOR THIS ARGUMENT > 0, THEN
23 COO ERREXIT IF SPECIFIED INTEGER VALUE > IDIM :ERR16:
24 COO INCREMENT TO NEXT TOKEN
25 COO ERREXIT IF TOKEN IS NOT COMMA :ERR15:
26 COO INCREMENT TO NEXT TOKEN
27 COO ERREXIT IF TOKEN IS NOT INTEGER VALUE ;ERR14:
28 COO CALCULATE ISUB AS (J-1)*IDIM+1
29 COO ELSE
30 COO
31 COO BEGIN XISUB
32 COO INCREMENT TO NEXT TOKEN
33 COO ERREXIT IF TOKEN IS NOT RIGHT PARENTHEIS :ERR14:
34 COO EXIT TO RETURN:
35 COO :ERR14: CALL XMSG - 'INVALID SUBSCRIPT SYNTAX'
36 COO :ERR15: CALL XMSG - 'DOTLY SUBSCRIPTED - MUST SPECIFY BOTH'
37 COO :ERR16: CALL XMSG - 'INVALID SUBSCRIPT VALUE'
38 COO
39 COO END XISUB
FORTRAN CALLING PROCEDURE

CALL XIDAT

LITERAL DATA PROCESSOR

INPUT

COMMON XE - COMBUF, COMPTR, TOKENS

COMMON XB - CFLAG, COMPLE, DFLAG, IARG, IARG4,
ISIZE, ISISZ, STYPE, LINDEX, LITDSP,
LITOWN, LTSIZ, MODAY, NOX8TH, NOBITH,
PRMTND, SFLAG, WKBNG, WBUF

OUTPUT

COMMON XB - IBYTC, ISUB, LITDSP, LITOWN, NOX8TH,
PRMTND, WBUF

INTERNAL VARIABLES

COMMON XG -
IDISP = INDEX INTO WBUF OF LOCATION FOR THIS
LITERAL DATUM
NUMCLP = NUMBER OF CONSEQUENTIAL ELEMENTS TO BE
MARKED COMPLETED AS A RESULT OF THIS
STACK = PUSH-DOWN LIST (MAX. OF 4 ENTRIES)
DESCRIBING NESTED REPEAT GROUPS.
IDENTICAL ENTRY IS 3 WORDS:
WORD 1 = INDEX TO 1ST TOKEN (IN COMBUF)
WORD 2 = REPEAT COUNT
WORD 3 = FLAG INDICATING WHETHER
REPEAT GROUP IS PARENTHESEICALLY
GROUPED

STKPTR = INDEX TO NEXT ENTRY TO BE BUILT IN 'STACK'

NOTES

USES ROUTINES
XISUB
XBRIT
XRMOV
XRMSG
CD5 *XMMB
CD5 *XSET
CD5 *CD5
CD5 *CD5

**C**********
* XDAT PROCESSES THE INPUT LITERAL LIST

1 BEGIN XDAT
2 DO UNTIL EOS IS REACHED, OR
3 UNTIL AN ERROR OCCURS
4
5 IF THIS IS A DATA ELEMENT, THEN
6 (I.E. INTEGER, REAL, DOUBLE OR CHAR.)
7 IF ARG TYPE (ITYPE) IS INTEGER, REAL, OR DOUBLE, THEN
8 EREREIT IF DATA TYPE IS NOT SAME AS ITYPE :-ERR10:
9 SET LENGTH TO BE MOVED (LENMOV) TO EFFECTIVE LENGTH FOR DATA
10 OF ARG'S TYPE (LENEFF) (WILL BE 1, 2, OR 3 WORDS)
11 ELSE
12 IF ARG TYPE IS FREE, THEN
13 SET LENGTH TO BE MOVED (LENMOV) TO EFFECTIVE LENGTH
14 FOR DATA INPUT
15 ELSE THIS MUST BE CHARACTER DATA BEING INPUT
16 EREREIT IF ARGUMENT'S TYPE IS NOT CHARACTER DATA :-ERR10:
17 INCREMENT TOKEN POINTER TO COUNT OF CHARACTERS
18 COMPUTE NO. WORDS IN INPUT CHARACTER STRING
19 EREREIT IF NO. WORDS (LENMOV) > EFFECTIVE LENGTH OF
20 THIS ARGUMENT'S DATA (LENEFF) :-ERR10:
21 ENDFI
22
23 ENDFI
24 INCREMENT TOKEN POINTER TO THE DATA INPUT
25 VERIFY THAT SUFFICIENT SPACE EXISTS IN LITERAL AREA OF THIS
26 ARGUMENT FOR DATA INPUT (ISUB <= ISIZE - LENMOV + 1)
27 EREREIT IF INSUFFICIENT SPACE :-ERR11:
28 IF DATA DOES NOT EXIST FOR THIS ARGUMENT, THEN
29 ALLOCATE AND INITIALIZE A LITERAL AREA FOR THIS ARGUMENT
30 ENDFI
31 MOVE DATA FROM INPUT COMMUNICATIONS BUFFERS TO LITERAL AREA
32 SET NUMBER OF ELEMENTS COMPLETED (NUMCMP) TO 1 OR, FOR A FREE
33 ARGUMENT, TO LENMOV
34 IF LENMOV < LENEFF (ONLY POSSIBLE FOR CHARACTER DATA), THEN
35 MOVE LENMOV-LENEFF BLANKS INTO LITERAL AREA AS A FILL
36 ENDFI
37 INCREMENT TO NEXT TOKEN
38 INCREMENT EFFECTIVE SUBSCRIPT (ISUB) BY LENMOV
39 SET LAST ENTERED INDICATOR (LASTE) TO ISUB = 1
40 ELSE, TOKEN INDICATES NON-DATA ELEMENT
41 IF THIS IS A "(" , THEN
42 CALL ISUB TO CALCULATE EFFECTIVE SUBSCRIPT (ISUB)
43 BASED ON INPUT SUBSCRIPT, ARGUMENT TYPE (ITYPE) AND
44 SECONDARY DIMENSION (IDIM)
45 EXIT XDAT IF ERROR (IRETE < 0)
46 ELSE
47 IF THIS IS AN ",", THEN
48 CLEAR THE PARAMETER FIELD IN ARGUMENT'S CHARACTERISTICS
49 ELSE
50 MARK ONE ELEMENT OF THIS ARGUMENT AT ISUB AS INCOMPLETE
51 INCREMENT EFFECTIVE SUBSCRIPT (ISUB) BY EFFECTIVE
52 LENGTH OF ONE ELEMENT (LENEFF)
53 ENDFI
54 TURN OFF COMPLETE FLAGS FOR THIS ARGUMENT AND INTERFACE TABLE
55 INCREMENT TO NEXT TOKEN
56
ELSE
  IF TOKEN IS A REPEAT COUNT, THEN
    IF NEXT TOKEN IS ",", THEN
      SET PARENFLAG TO 1
    INCREMENT TO NEXT TOKEN
  ELSE
    SET PARENFLAG TO 0
  ENDIF
  SAVE REPEAT COUNT, TOKEN INDEX, AND PARENFLAG IN A PUSH-DOWN STACK
ELSE
  ERREXIT (INVALID FIELD) :ERROR:
ENDIF
ENDIF
INCREMENT TO NEXT TOKEN
DO UNTIL TOKEN IS NOT ")"
  IF PUSH-DOWN STACK IS NOT EMPTY, AND
  (PREVIOUS TOKEN WAS DATA, AND
  PARENFLAG OF TOP OF STACK ENTRY IS 0), OR
  (CURRENT TOKEN IS ")", AND
  PARENFLAG OF TOP OF STACK ENTRY IS 1), THEN
  GET REPEAT COUNT OF TOP OF STACK ENTRY
  DECREMENT REPEAT COUNT BY 1
  IF REPEAT COUNT > 0, THEN
    SET TOKEN INDEX TO INDEX ON PUSH-DOWN STACK
  ELSE
    REPLACE NEW REPEAT COUNT ON PUSH-DOWN STACK
  ENDIF
  POP (I.E. REMOVE) ENTRY FROM TOP OF STACK
  IF TOKEN IS ",", THEN
    INCREMENT TO NEXT TOKEN
  ELSE
    EXIT LOOP
  ENDFIND
ELSE
  IF PREVIOUS TOKEN WAS NOT A SUBSCRIPT, THEN
    IF THIS TOKEN IS NOT AN EOS, THEN
      ERREXIT IF TOKEN IS NOT A COMMA :ERROR:
    INC INCREMENT TO NEXT TOKEN
  ENDFIND
ENDFIND
ENDDO
IF PROMPT MODE NOT CONTINUE (<A), AND
  (THERE ARE EMPTY ELEMENTS BEYOND LASTE, OR
  PREVIOUS TOKEN WAS A COMMA), THEN
  RETAIN PROMPT MODE (SET MODAW TO PRMTMD)
  SET PROMPT MODE TO CONTINUE (<A)
ENDIF
IF ALL LITERAL SLOTS FILLED, THEN
  MARK ARGNO COMPLETE
  IF ALL ARGUMENTS ARE COMPLETE, THEN
    SET COMPLETE FLAG FOR INTERFACE TABLE
  ENDFIND
ELSE
  TURN OFF COMPLETE FLAG FOR INTERFACE TABLE
ENDFIND
615 1 EXIT TO :RETURN;
616 2 :ERROR: CALL XMSGS “INVALID SYNTAX”
617 2 :ERROR: CALL XMSGS “ONLY DATA VALID TO RIGHT OF .”
618 2 :ERROR: CALL XMSGS “DATA TYPE INCOMPATIBLE WITH TYPE OF ABSENCE”
619 2 :RETURN;
620 1 END XDAT

5-106
622  1 CD0  FORTRAN CALLING PROCEDURE
623  1 CD0
625  1 CD0  CALL XINPT
626  1 CD0
627  1 C*******
628  1 CD1
629  1 CD1  INTERFACE TABLE EDITOR'S INPUT PROCESSOR
630  1 CD1
631  1 C*******
632  1 CD2
633  1 CD2  INPUT
634  1 CD2
635  1 CD2  COMMON XE - COMBUF, COMPR, TOKENS
636  1 CD2
637  1 CD2  COMMON XB - DIRECT, IFLAG, ISUB, NRE, NDIR, NUMARG, PRNTMD, WBUF
638  1 CD2
639  1 CD2
640  1 C*******
641  1 CD3
642  1 CD3  OUTPUT
643  1 CD3
644  1 CD3  COMMON XB - ARMO, IRET
645  1 CD5
646  1 C*******
647  1 CD5
648  1 CD5  NOTES
649  1 CD5
650  1 CD5  USES ROUTINES
651  1 CD5
652  1 CD5  XIDAF
653  1 CD5  XIEF
654  1 CD5  XILS
655  1 CD5  XIPAM
656  1 CD5  XIPOF
657  1 CD5  XRCPR
658  1 CD5  XRNSG
659  1 CD5
660  1 CD5
661  1 C*******
*XINPT PROCESSES THE USER'S INPUT TEXT

BEGIN XINPT

IF PRMTD = 5, THEN

SET IFLAG OFF (-0)
ENDIF

ERREXIT IF TOKEN IS NOT A NAME :ERROR2:
SAVE NAME AND POSITION TO NEXT TOKEN
IF TOKEN IS "=", THEN
POSITION TO NEXT TOKEN
IF TOKEN IS 'B', THEN
SET IFLAG TO IO
POSITION TO NEXT TOKEN
ELSE
SET IFLAG TO I
ENDIF
ELSE
IF TOKEN IS 'G', THEN
SET IFLAG TO O
ENDIF
ENDIF
IF IFLAG NOT SET, THEN
CASE NAME (:EXIT; :PROMPT; :LIST;)
ERREXIT IF ANOTHER TOKEN FOLLOWS :ERROR2:
:EXIT: SET IRET C SO THAT PROMPTING LOOP TERMINATES
:PROMPT: CALL XIPMT TO PROCESS PROMPT DIRECTIVE
:LIST: CALL XILIST TO PROCESS LIST DIRECTIVE
ENDCASE
ENDIF
START SEARCH UNTIL I/F ENTRIES
EXIT IF NAME FOUND IN P nutrit T A B L E
SET ARG NO TO ENTRY NO.
SET ISUB TO 1
OR ELSE
INCREMENT TO NEXT PROMPT TABLE ENTRY
ENDCASE
ERREXIT :ERROR10:
ENDSEARCH
ERREXIT IF IFLAG IS NOT SAME AS I/O TYPE OF ARGUMENT :ERROR5:
ENDIF
IF NEXT TOKEN IS A NAME, THEN
CALL XIPAR TO PROCESS A PARAMETER FIELD
ELSE
ERREXIT IF IFLAG IS NOT I ("=") :ERROR8:
CALL XI DAT TO PROCESS DATA LIST
ENDIF
EXIT XINPT
EXIT TO :RETURN:
:ERROR2: CALL XMSG "INVALID SYNTAX"
:ERROR8: CALL XMSG "MUST USE PARAMETER NAME TO RIGHT OF & OR =& "
:RETURN:
END XINPT
FORTAN CALLING PROCEDURE

CALL XIPAR

*************

PROCESS AN INPUT PARAMETER NAME AND ANY ASSOCIATED SUBSCRIPT

*************

INPUT
COMMON XE - COMBUF, COMPTR, TOKENS

COMMON XB - DFLAG, IARG, IAREA, ICLASS,
LITDSP, SFLAG, WKBLINE, WKBUF

*************

OUTPUT
COMMON XB - IRETC, LITDWN, WKBUF

*************

NOTES

USES ROUTINES

XRMCC

XRMCG

XRSET

*************
1 * XIPAR PROCESSES A USER SPECIFIED PARAMETER FIELD
2 BEGIN XIPAR
3 IF A NAME IS SPECIFIED, THEN
4 INCREMENT TO NEXT TOKEN
5 IF TOKEN IS '(', THEN
6 PROCESS I AND J SUBSCRIPTS
7 ERREXIT IF INVALID SUBSCRIPTING :ERR14:
8 IF DOUBLY SUBSCRIPTED, THEN
9 SET S-FLAG IN ARGNO'S SPECS. FIELD
10 SET LITOSP IN ARGNO'S SPECS. TO NEXT LITERAL AREA SPACE (LITDOWN)
11 PUT ISUB AND JSUB INTO LITERAL AREA AT THIS SPOT
12 ELSE
13 SET LITOSP IN ARGNO'S SPECS TO ISUB
14 ENDIF
15 ELSE
16 SET LITOSP IN ARGNO'S SPECS TO 0
17 ENDIF
18 ERREXIT IF ORDE HAS MORE THAN 2 CHARACTERS :ERR18:
19 ERREXIT IF EXTRAN'GUS FIELD INPUT :ERR02:
20 TURN OF D-FLAG (SOME LITERAL DATA) IN ARGNO'S SPECS.
21 SET PAR'NAME INTO ARGUMENT'S CHARACTERISTICS
22 SET COMPLETE (AND &) FLAG IN ARGUMENT'S CHARACTERISTICS
23 IF ALL ARGUMENTS ARE COMPLETE, THEN
24 SET INTERFACE TABLE COMPLETE FLAG
25 ELSE
26 EL"." MUST BE A & INPUT
27 "REXIT IF NOT AN AMPERSAND (&) INPUT :ERR02:
28 CLEAR PAR'METER NAME IN ARGUMENT'S CHARACTERISTICS
29 SET ARGUMENT AND INTERFACE TABLE INCOMPLETE
30 ENDIF
31 EXIT TO :RETURN:
32 :ERR02: CALL XRMG - "INVALID SYNTAX"
33 :ERR14: CALL XRMG - "INVALID SUBSCRIPT SYNTAX"
34 :ERR18: CALL XRMG - "INVALID ORDE NAME"
35 :RETURN:
36 1 END XIPAR
FORTRAN CALLING PROCEEDURE

CALL XILSD

*******

LIST DATA OR PARAMETER VALUES FOR ONE ARGUMENT

*******

INPUT

*******

COMMON XE - .D

COMMON XA - A, A, DFLAG, IARG, IDIM,

AVIS, IFLAG, IUB, ITYPE,

LENFF, LITISP, LTSSIP, MARG,

M3XBM, MBBM, SFLAG, WBUF

*******

COMMON XS - BUFFER = LINE TO BE OUTPUT

BUFFTR = INDEX INTO BUFFER FOR NEXT ASCII DATA

KETRM = RETURN INDICATOR FOR INTERNAL Routines

*******

Notes

uses routines

exec

******

def
*XILSP WILL LIST THE DATA ASSOCIATED WITH ONE ARGUMENT
*IS RETURNED AS A PROMPT.
SET ARGUMENT NAME INTO BUFFER
USE IF-FLAG TO DETERMINE WHICH OF 'B', 'n', OR 'nB'
WILL GO INTO THE PRINT BUFFER
IF D-FLAG IS OFF INDICATING NO LITERAL DATA, THEN
IF A PARAMETER NAME IS SPECIFIED, THEN
PUT PARAMETER NAME INTO BUFFER
IF S-FLAG IS ON INDICATING TWO SUBSCRIPTS, THEN
COMPUTE AND CONVERT TO CHARACTER FORMAT EACH SUBSCRIPT
PUT SUBSCRIPT INTO BUFFER
ELSE
IF LITDSP OF ARGUMENT IS > 0, THEN
COMPUTE AND CONVERT THIS SUBSCRIPT
PUT SUBSCRIPT INTO BUFFER
ENDIF
WRITE OUT THE PRINT BUFFER BUILT
ENDIF
ELSE
LOCATE LITERAL LIST AND MASK
IF SYMBOLIC STRING, THEN
CALL FILES TO PRINT SYMBOLIC STRING
ELSE
DO UNTIL ALL ELEMENTS PROCESSED
DO UNTIL A BUFFER OF DATA HAS BEEN GENERATED, OR
UNTIL ALL ELEMENT'S PROCESSED
COMPUTE AND CONVERT THE SUBSCRIPT
IF MASK FOR ELEMENT INDICATES NO DATA, THEN
PUT "n" INTO BUFFER
ELSE
CONVERT THE DATA USING XR6, XR14, OR XR16
PUT DATA AND "," INTO BUFFER
ENDIF
ENDIF
IF ALL ELEMENTS OF THIS ARGUMENT HAVE BEEN PROCESSED, THEN
REMOVE THE TRAILING COMMA IN THE PRINT BUFFER
WRITE OUT THE PRINT BUFFER BUILT
ENDIF
ENDIF
ENDIF
*END XILSP
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>906</td>
<td>INTERNAL VARIABLES</td>
</tr>
<tr>
<td>907</td>
<td></td>
</tr>
<tr>
<td>908</td>
<td>CONTROL = CONTROL TABLE DESCRIBING DISPOSITION AND PROCESSING</td>
</tr>
<tr>
<td>909</td>
<td>FOR EACH OF THE TOKEN VALUES 1-32. EACH CONTROL TABLE</td>
</tr>
<tr>
<td>910</td>
<td>ENTRY IS 3 WORDS:</td>
</tr>
<tr>
<td>911</td>
<td></td>
</tr>
<tr>
<td>912</td>
<td></td>
</tr>
<tr>
<td>913</td>
<td></td>
</tr>
<tr>
<td>914</td>
<td></td>
</tr>
<tr>
<td>915</td>
<td></td>
</tr>
<tr>
<td>916</td>
<td></td>
</tr>
<tr>
<td>917</td>
<td></td>
</tr>
<tr>
<td>918</td>
<td></td>
</tr>
<tr>
<td>919</td>
<td>ENTRY</td>
</tr>
<tr>
<td>920</td>
<td>MC.</td>
</tr>
<tr>
<td>921</td>
<td>1</td>
</tr>
<tr>
<td>922</td>
<td>1</td>
</tr>
<tr>
<td>923</td>
<td>1</td>
</tr>
<tr>
<td>924</td>
<td>1</td>
</tr>
<tr>
<td>925</td>
<td>1</td>
</tr>
<tr>
<td>926</td>
<td>1</td>
</tr>
<tr>
<td>927</td>
<td>1</td>
</tr>
<tr>
<td>928</td>
<td>1</td>
</tr>
<tr>
<td>929</td>
<td>1</td>
</tr>
<tr>
<td>930</td>
<td>1</td>
</tr>
<tr>
<td>931</td>
<td>1</td>
</tr>
<tr>
<td>932</td>
<td>1</td>
</tr>
<tr>
<td>933</td>
<td>1</td>
</tr>
<tr>
<td>934</td>
<td>1</td>
</tr>
<tr>
<td>935</td>
<td>1</td>
</tr>
<tr>
<td>936</td>
<td>1</td>
</tr>
<tr>
<td>937</td>
<td>1</td>
</tr>
<tr>
<td>938</td>
<td>1</td>
</tr>
<tr>
<td>939</td>
<td>1</td>
</tr>
<tr>
<td>940</td>
<td>1</td>
</tr>
<tr>
<td>941</td>
<td>1</td>
</tr>
<tr>
<td>942</td>
<td>1</td>
</tr>
<tr>
<td>943</td>
<td>1</td>
</tr>
<tr>
<td>944</td>
<td>1</td>
</tr>
<tr>
<td>945</td>
<td>1</td>
</tr>
<tr>
<td>946</td>
<td>1</td>
</tr>
<tr>
<td>947</td>
<td>1</td>
</tr>
<tr>
<td>948</td>
<td>1</td>
</tr>
<tr>
<td>949</td>
<td>1</td>
</tr>
<tr>
<td>950</td>
<td>1</td>
</tr>
<tr>
<td>951</td>
<td>1</td>
</tr>
<tr>
<td>952</td>
<td>1</td>
</tr>
<tr>
<td>953</td>
<td>1</td>
</tr>
</tbody>
</table>
BEGIN XILSS

MOVE A CHARACTER INTO PRINT BUFFER AND INCREMENT BUFFER POINTER

DO UNTIL ALL TOKENS OF SYMBOLIC STRING HAVE BEEN PROCESSED

EXIT TO ERROR 1 IF TOKEN VALUE IS < 1 OR > 32

USE TOKEN VALUE TO RETRIVE 3 CONTROL WORDS (SIZE, FIELD, TOKSZ)

EXIT TO ERROR 2 IF FIELD = 0

IF SIZE < 0, THEN

SET SIZE TO VALUE IN WORD FOLLOWING THIS TOKEN IN THE SYMB. STRING

ENDIF

IF TOKSZ < 0, THEN

SET TOKSZ TO SIZE + 2

ENDIF

IF THERE IS NOT ROOM IN PRINT BUFFER FOR SIZE WORDS, THEN

CALL EXEC TO WRITE PRINT BUFFER TO INDICATED DEVICE

CLEAR PRINT BUFFER TO BLANKS

ENDIF

IF FIELD > 0, THEN

MOVE FIELD INTO CURRENT PRINT BUFFER POSITION

ELSE

CASE (ONE, TWO, THREE, FOUR, FIVE, SIX, EXIT), -FIELD

:ONE: CALL XRI6 WITH VALUE IN NEXT WORD OF SYMB. STRING

AND PUT RESULTS INTO PRINT BUFFER

:TW0: CALL XRI14 WITH VALUE IN NEXT 2 WORDS OF SYMB. STRING

AND PUT RESULTS INTO PRINT BUFFER

:THREE: CALL XRI18 WITH VALUE IN NEXT 3 WORDS OF SYMB. STRING

AND PUT RESULTS INTO PRINT BUFFER

:FOUR: MOVE THE NEXT 3 WORDS OF SYMB. STRING INTO PRINT BUFFER

:FIVE: MOVE SIZE WORDS FROM 2ND WORD PAST CURRENT TOKEN THE SYMB. STRING

:SIX: CALL XRI6 WITH VALUE IN NEXT WORD OF SYMB. STRING AND PUT RESULTS INTO PRINT BUFFER FOLLOWED BY AN "R"

:EXIT: PUT A CHARACTER INTO THE PRINT BUFFER, INCREMENT THE BUFFER

INDEX BY 1

CALL EXEC TO WRITE THE PRINT BUFFER TO INDICATED DEVICE

EXIT XILSS

ENDCASE

INCREMENT PRINT BUFFER INDEX BY SIZE
1015  1 CRO  FORTRAN CALLING PROCEDURE
1016  1 CRO  CALL XICHR
1017  1 CRO  CALL XICHR
1018  1 CRO  CALL XICHR
1019  1 CRO  CALL XICHR
1020  1 CRO  CALL XICHR
1021  1 CRO  CALL XICHR
1022  1 CRO  CALL XICHR
1023  1 CRO  CALL XICHR
1024  1 CRO  CALL XICHR
1025  1 CRO  CALL XICHR
1026  1 CRO  CALL XICHR
1027  1 CRO  CALL XICHR
1028  1 CRO  CALL XICHR
1029  1 CRO  CALL XICHR
1030  1 CRO  CALL XICHR
1031  1 CRO  CALL XICHR
1032  1 CRO  CALL XICHR
1033  1 CRO  CALL XICHR
1034  1 CRO  CALL XICHR
1035  1 CRO  CALL XICHR
1036  1 CRO  CALL XICHR
1037  1 CRO  CALL XICHR
1038  1 CRO  CALL XICHR
1039  1 CRO  CALL XICHR
1040  1 CRO  CALL XICHR
1041  1 CRO  CALL XICHR
1042  1 CRO  CALL XICHR
1043  1 CRO  CALL XICHR
1044  1 CRO  CALL XICHR
1045  1 CRO  CALL XICHR
1046  1 CRO  CALL XICHR
1047  1 CRO  CALL XICHR
1048  1 CRO  CALL XICHR
1049  1 CRO  CALL XICHR
1050  1 CRO  CALL XICHR
1051  1 CRO  CALL XICHR

* WRITE ARGUMENT CHARACTERISTICS
1046  1 CRO  BEGIN XICHR
1047  1 CRO  BEGIN XICHR
1048  1 CRO  BEGIN XICHR
1049  1 CRO  BEGIN XICHR
1050  1 CRO  BEGIN XICHR
1051  1 CRO  BEGIN XICHR

2 BUILD PRINT BUFFER WITH ARGUMENT NAME, SUBSCRIPTS, I/O TYPE AND
1049  1 CRO  DATA TYPE
1050  1 CRO  DATA TYPE
1051  1 CRO  DATA TYPE

2 WRITE OUT THE PRINT BUFFER
1050  1 CRO  WRITE OUT THE PRINT BUFFER
1051  1 CRO  WRITE OUT THE PRINT BUFFER
1053 1 CD0      FORTRAN CALLING PROCEDURE
1054 1 CD0      CALL XIEXT
1055 1 CD0
1056 1 CD0      CALL XIEXT
1057 1 CD0
1058 1 CD0
1059 1 CD0      ********
1060 1 CD0      ********
1061 1 CD1      EXTRACT VARIOUS FIELDS OF AN ARGUMENT'S CHARACTERISTICS
1062 1 CD1      AND PUT VALUES INTO COMMON
1063 1 CD1      ********
1064 1 CD1      ********
1065 1 CD2      INPUT
1066 1 CD2
1067 1 CD2
1068 1 CD2      COMMON XB - ARGNO, ISIZES, WKBUF
1069 1 CD2
1070 1 CD2
1071 1 CD3      OUTPUT
1072 1 CD3
1073 1 CD3
1074 1 CD3      COMMON XB - CFLAG, DFLAG, IARG, IARGA,
1075 1 CD3
1076 1 CD3      ICCLASS, IDIM, IOFLAG, ISIZE,
1077 1 CD3
1078 1 CD3
1079 1 CD3      ********
1080 1 CD5      NOTES
1081 1 CD5
1082 1 CD5
1083 1 CD5      USES ROUTINES
1084 1 CD5
1085 1 CD5      IAMD
1086 1 CD5      XREX
1087 1 CD5
1088 1 CD5      ********
1089 1 CD5      * EXTRACT THE VARIOUS VALUES AND FLAGS ASSOCIATED WITH THIS
1090 1 CD5      * ARGUMENT
1091 1 CD5      BEGIN XIEXT
1092 1 CD5      USING THE ARGUMENT NO. (ARGNO), LOCATE THIS ARGUMENT'S CHARACTERISTICS
1093 1 CD5      IN THE WORKING BUFFER
1094 1 CD5      EXTRACT EACH OF THE FIELDS INTO A WORD OF COMMON FOR GENERAL USEAGE
1095 1 CD5      END XIEXT
FORTRAN CALLING PROCEDURE

CALL XILIT

PACK LITERAL ENTRIES INTO FORMAT FOR STORAGE OF INTERFACE TABLE INTO AWA

INPUTS

COMMON X8 - ARGNO, DFLAG, IARG4, ISIZE,
LENFF, LITDGV, LITDWN, LITPTR,
LITSZ, MARG, NDXTMT, NOBITM
NUMARG, SFLAG, WXLNG, WBUF

OUTPUTS

COMMON X8 - LITDWN, LITLEM, LITPTR, WBUF

INTERNAL VARIABLES

COMMON XS - LITUP = INDEX INTO WBUF OF AREA FOR NEXT LITERAL ENTRY TO BE MOVED INTO
LITUMENT = INDEX INTO WBUF OF LITERAL ENTRY TO BE COMPRESSED AND MOVED

NOTES

USES ROUTINES

XIE  XH   XMOTO  XRMOV  XRMOV  XRNRXO  XRSET
1145 1 * PACK LITERAL AREA INTO FORMAT FOR STORAGE OF INTERFACE TABLE
1146 1 BEGIN XILIT
1147 2 DO UNTIL ALL LITERAL AREAS PROCESSED
1148 3 DO UNTIL ALL ARGUMENTS SEARCHED
1149 4 IF THIS LITERAL ENTRY BELONGS TO THIS ARGUMENT, THEN
1150 5 IF ALL ELEMENTS OF THIS ARGUMENT ARE COMPLETE, THEN
1151 6 MOVE ALL DATA FOR LITERAL ENTRY UP IN WORKING BUFFER
1152 7 ELSE
1153 8 DO UNTIL ALL BITS OF BIT MASK PROCESSED
1154 9 IF THE BIT IS ON, THEN
1155 10 MOVE CORRECT NUMBER OF WORDS (LENEFF) OF LITERAL
1156 11 UP IN THE WORKING BUFFER
1157 12 ENDF
1158 6 ENDDO
1159 5 ENDF
1160 5 CALCULATE NEW DISPLACEMENT AND SET IN LITDSP
1161 4 ENDF
1162 3 ENDDO
1163 2 ENDDO
1164 1 END XILIT
1 CD**********
2 CD
3 CD
4 CD
5 CD
6 CD
7 CD
8 CD
9 CD
10 CD
11 CD
12 CD
13 CD
14 CD
15 CD
16 CD
17 CD
18 CD
19 CD
20 CD
21 CD
22 CD
23 CD
24 CD
25 CD
26 CD
27 CD
28 CD
29 CD
30 CD
31 CD
32 CD
33 CD
34 CD
35 CD
36 CD
37 CD
38 CD
39 CD
40 CD
41 CD
42 CD
43 CD
44 CD
45 CD
46 CD
47 CD
48 CD
49 CD
50 CD
51 CD
52 CD
53 CD**********
55 1 BEGIN XLMAN
56 2 CALL RMPAR TO GET INPUT PARAMETERS
57 2 INITIALIZE COMMON TO ZERO
58 2 SET CLASS NUMBER TO ZERO
59 2 CALL EXEC TO GET A CLASS NUMBER
60 1 EXIT XLMAN IF SECURITY CODE IS NOT VALID
61 2 CALL XMOV TO INITIALIZE TOKENS IN COMMON
62 2 DO FOREVER
63 3 :PROMPT:
64 3 INITIALIZE MASTER AND SUBSTATE FLAGS
65 3 CALL XCCM TO PROMPT FOR OPTION
66 3 IF XCOM RETURN CODE IS NOT ZERO OR
67 4 FIRST TOKEN IS NOT AN INTEGER OR
68 4 INTEGER > 7 THEN
69 4 CALL XMGS TO WRITE INVALID RESPONSE
70 4 GO TO :PROMPT:
71 3 ENDF
73 4 :XLPRM:
74 4 SET NUMOR TO INTEGER
75 4 CALL XELS TO LOAD XLPRM TO CREATE SYSTEM PROMPT FILE
77 4 :XLPRM:
78 4 SET VALFLG TO SAY ORIGINAL XLPRM REQUEST
79 4 CALL XELS TO LOAD XLPRM TO ADD A PROCESSOR
80 4 DO UNTIL VALFLG SAYS EXIT (X)
81 5 CALL XELS TO LOAD XLLINT TO ENTER DEFAULT VALUES
82 5 CALL XELS TO LOAD XLPRO TO COMPLETE PROCESSING
83 4 ENDD
84 4 :XLDEL:
85 4 CALL XELS TO LOAD XLDEL TO DELETE A PROCESSOR
86 4 :XLMOD:
87 4 SET VALFLG TO SAY ORIGINAL XLMOD REQUEST
88 4 CALL XELS TO LOAD XLMOD TO MODIFY A PROCESSOR
89 4 DO UNTIL VALFLG SAYS EXIT (X)
90 5 CALL XELS TO LOAD XLLINT TO ENTER DEFAULT VALUES
91 5 CALL XELS TO LOAD XLMOD TO COMPLETE PROCESSING
92 4 ENDD
93 4 :XLMSG:
94 4 CALL XELS TO LOAD XLMSG TO ADD A MESSAGE
95 4 :XLOBF:
96 4 CALL XELS TO LOAD XLOBF TO HANDLE DATA BASE FILES
97 3 ENDCASE
98 2 ENDD
99 2 :EXIT:
100 2 CALL EXEC TO RELEASE CLASS NUMBER
101 1 END XLMAN
FORTRAN CALLING PROCEDURE:

CALL XELEDS ('XLPRM')

CREATES ONE OF THE SYSTEM PROMPT FILES DEPENDING ON "NUMBER"

INPUT FROM COMMON:

NUMBER - (INTEGER, 1 WORD) RESPONSE INDICATING WHICH SYSTEM PROMPT FILE TO CREATE:

1 = >XIPRM
2 = >XIPRM
3 = >XIPRM

INTERNAL VARIABLES:

COM3 - (INTEGER, 1 WORD) FIRST TOKEN IN COMBUF
COM4 - (INTEGER, 1 WORD) FIRST DATA IN COMBUF
MSG - (INTEGER, 1 WORD) CONTAINS APPROPRIATE MESSAGE NUMBER
WITH WHICH TO CALL XIRNSG
NAME - (INTEGER, 9 WORDS) 5 ELEMENT ARRAY, EACH ELEMENT IS A SYSTEM PROMPT FILE NAME
MODIR - (INTEGER, 1 WORD) NUMBER OF DIRECTIVES
NOK - (INTEGER, 1 WORD) NUMBER OF TOKENS
PROMS - (INTEGERS) PROM2, PROM3, PROM4 ARE ALL USER PROMPT ARRAYS

RTE FUNCTIONS AND SUBROUTINES USED:

KCVT, CLOSE, CREATE, PURGE, WATSF

FRS FUNCTIONS AND ROUTINES USED:

XMOV, XIRNSG, XTCOM

COMMON USED:

ERUVALANCE

+ (X(3), LICEU ), (X(7), NUMBR ),
+ (X(14), XCR ), (X(145), COMU7 ),
+ (X(145), NOK ), (X(147), COMU3 ),
+ (X(146), COMA ), (X(1), IBUF ),
+ (X(128), MODIR )
CD

CD          FORTRAN CALLING SEQUENCE:

CD          CALL RKLDS ('XLPRO')

CD          CD

CD          CD

CD          XLPRO WILL ADD A PROCESSOR TO THE LIBRARY DIRECTORY. IF

CD          THE PROCESSOR HAS A DEFAULT INTERFACE TABLE, IT WILL ALSO

CD          CREATE A PROMPT FILE AND INTERFACE TABLE FILE

CD          CD

CD          CD

CD          INPUT FROM COMMON:

CD          VALFLG - PROCESS CONTROL

CD          0 - ORIGINAL REQUEST TO ADD

CD          1 - COMPLETE PROCESSING

CD          CD

CD          CD

CD          INTERNAL VARIABLES:

CD          ABSTR - (INTEGER, 128 WORDS) CONTAINS THE ABSTRACT OF THE

CD          PROCESSOR IN LATER BUILDS. PRESENTLY IS A NULL RECORD.

CD          BLOCKS - (INTEGER, 1 WORD) NUMBER OF BLOCKS TO ALLOCATE

CD          TO INTERFACE TABLE FILE

CD          DIRECT - (INTEGER, 6 WORDS) 3 NAME ARRAY CONTAINING VALID

CD          DIRECTIVES FOR THE INTERFACE TABLE EDITOR

CD          NAMEY - (INTEGER, 3 WORDS) COMBINATION OF PROCESSOR

CD          NAME AND VERSION NUMBER

CD          PROM - (INTEGER, 3 WORDS) PROMPT FOR USER TO BE PROMPTED WITH

CD          TYPE - (.INTEGER, 1 WORD) TYPE OF PARAMETER USED IN

CD          CALCULATING SIZE

CD          VERS - (INTEGER, 1 WORD) VERSION NUMBER OF PROCESSOR

CD          CD

CD          CD

CD          RTE AND FNR ROUTINES USED

CD          EXEC, IAND, KEVT, CREAT, OPEN,

CD          READF, WRITF, CLOSE, PURGE

CD          CD

CD          CD

CD          FDS ROUTINES USED:

CD          XLFL, XLM, XLPL, XLPS

CD          XRKP, XMOV, XRMUS, XRPC, XRSET, XRPK, XTCON

CD          CD

CD          CD

CD          COMMON USED:

CD          EQUIVALENCE  (XE(3),  ISECU ),

CD          * (XE(3),  MASTA ),  (XE(66),  SUSTA ),

CD          * (XE(7),  WALTFL ),  (XE(16),  PRCPH ),

CD          * (XE(85),  EOSTOK ),  (XE(86),  INHAK ),

CD          * (XE(87),  MANTOK ),  (XE(113),  CONG ),

CD          * (XE(142),  CPC ),  (XE(145),  COMHAT ),

CD          * (X8(26),  VERS ),  (X8(37),  ARGW ),

CD          * (X8(38),  IT ),  (X8(90),  LITPTR ),
BEGIN XLPRO

IF THIS ENTRY IS THE ORIGINAL XLPRO ENTRY FOR THIS PROCESSOR THEN

INITIALIZE MASTER AND SUB STATES

PERFORM LIBL TO UPDATE LIBRARY DIRECTORY

IF PROCESSOR HAS AN INTERFACE TABLE THEN

SET RETN = 2

CALL XTCOM TO PROMPT FOR & PARAMETERS

ERREXIT IF XTCOM RETURN CODE IS NOT ZERO OR

ERREXIT IF & PARAMETERS NOT INTEGER 1-63 TO :PRMERR:

CREATE HEADEP ENTRY WITH &PARAMETERS AND PROCESSOR NAME

CALL XLINS TO WRITE INSTRUCTIONS FOR ENTERING SPECS

DO FOR & PARAMETERS

CALL XLSPS TO CREATE ONE PARAMETER ENTRY

ENDGO

SET CODES ARRAY TO ADD ABSTRACT AND PARAMETER DEFINITIONS

CALL XLFL TO CREATE >PROMPT FILE

SET RETN = 3

:PRMPT3:

CALL XTCOM FOR DEFAULT VALUES DECISION

ERREXIT IF RETURN CODE IS NOT ZERO OR

ERREXIT IF RESPONSE IS NOT YE OR NO TO :PRMERR:

IF RESPONSE WAS YE THEN

SET FLAG TO CALL INTERFACE TABLE EDITOR

EXIT XLPRO

ELSE

SET CODES ARRAY TO ADD ONLY ABSTRACT

CALL XLFL TO CREATE PROMPT FILE

PERFORM XLPRO - NO RETURN EXPECTED

ENDIF

ENDIF

CALL XLFL TO CREATE THE DEFAULT INTERFACE TABLE FILE

SET VALFLAG TO SET ORIGINAL REQUEST TO ADD A PROCESSOR

PERFORM XLPRO - NO RETURN EXPECTED

:PRMERR:

CALL XRMSG TC DISPLAY ERROR MESSAGE

GO TO (:PRMPT1;:PRMPT2;:PRMPT3), RETN

END XLPRO
1 BEGIN LIBD
2 CALL OPEN TO OPEN LIBRARY DIRECTORY
3 IF RETURN CODE SAYS FILE NOT FOUND THEN
4 SET RECORD 1 TO ALL ZEROS
5 ELSE
6 ERREXIT IF FILE ERROR TO :FILERR:
7 CALL READF AND CLOSE TO READ IN LIBRARY DIRECTORY
8 ERREXIT IF FILE ERROR TO :FILERR:
9 IF # PROCESSORS + 1 > 50 THEN
10 CALL XRMSG TO WRITE ERROR: TOO MANY PROCESSORS
11 EXIT XLPRO
12 ENDF
13 SET RE TN = 1
14 :PROMPT:
15 CALL XICOM TO PROMPT FOR PROCESSOR NAME, VERSION, INT TABLE
16 EXIT XLPRO IF RETURN CODE SAYS I ENTERED
17 ERREXIT IF RETURN CODE > ZERO OR
18 ERREXIT IF PROCESSOR NAME IS NOT 6-CHAR NAME TO :PRMERR:
19 CALL XRMOV TO MOVE PROCESSOR NAME INTO ENTRY
20 ERREXIT IF VERSION IS NOT INTEGER VALUE 0-127 TO :PRMERR:
21 CALL XRSET TO SET VERSION IN ENTRY
22 ERREXIT IF INTERFACE TABLE OPTION IS NOT YE OR NO TO :PRMERR:
23 SET IT BIT = 0
24 IF RESPONSE IS YES THEN
25 SET IT BIT = 1
26 ENDF
27 CALL XRSET TO SET BIT ON/OFF
28 ERREXIT IF PROCESSOR NAME ALREADY EXISTS TO :PRMERR:
29 INCREMENT # PROCESSORS BY 1
30 CALL XRMOV TO MOVE NEW ENTRY INTO XLIBD
31 IF # PROCESSORS > 1 THEN
32 CALL PURGE TO PURGE OLD FILE
33 ERREXIT IF RETURN CODE < ZERO TO :FILERR:
34 ENDF
35 CALL CREATE, WRITE AND CLOSE TO CREATE NEW LIBRARY DIRECTORY
36 ERREXIT IF FILE ERROR TO :FILERR:
37 ENDF
38 1 END LIBD
CD************
CALL XLCDB

CD********++++
XLCDB CREATES A NEW DATA BASE FILE (MDB/PDB) FROM AN OLD
DATA BASE FILE (MDB/PDB) AND DELETES THE OLD FILE

CD********+++ Inputs in COMMON:
XE(3) QUAL, XE(7) NUMBR, XE(8) SECU
XE(142) ICR, XB(5) OLDfil, XB(6) NEWfil
XB(9) TOTSZ

CD********+++ Internal Variables:
XB(17) FREC — FIRST RECORD NUMBER IN DATBUF
XB(18) LREC — LAST RECORD NUMBER IN DATBUF
XB(40) IDC0 - DCB FOR OLDfil
XB(56) IDC02 - DCB FOR NEWfil
XB(201) TOCBUF-BUFFER FOR COMPLETE DATA BASE TOC

CD********+++ RTE Routines Used:
CLOSE, CREAT, KCVT, OPE,
Purge, READF, WRITF

CD********+++ FDS Routines Used:
XDB0D, XREXT, XRMG

CD********+++ Common Used:

CD********+++ Equivalence
(XE(3), QUAL),
(XE(3), SECU),
(XB(3), OLDfil),
(XB(9), TOTSZ),
(XB(12), QUALif),
(XB(14), SIZE),
(XB(16), TOCPT),
(XB(18), LREC),
(XB(40), IDC0),
(XB(72), IDBU),
(XB(201), TOCBUF)

CD********+++
BEGIN XLCDB
CALL OPEN TO OPEN OLDFIL
ERREXIT IF OPEN ERROR TO :ERR3:
CALL READ TO READ IN HEADER RECORD
ERREXIT IF READ ERROR TO :ERR2:
COMPUTE NUMBER OF TOC RECORDS (NTOC) AS(# ENTRIES + 16)/ 16
IF NTOC > 1 THEN
CALL READ TO READ IN REMAINING TOC RECORDS
ERREXIT IF READ ERROR TO :ERR2:
ENDIF
SET # RECORDS LEFT TO READ (SIZE) AS TOTAL SIZE - NTOC
CALLcreat TO CREATE NEWFIL
ERREXIT IF CREAT ERROR TO :ERR2:
DO FOR EACH TOC ENTRY
IF REQUEST WAS FOR TOC THEN
ERREXIT IF NAME > 4 CHAR'S OR
ERREXIT IF CLASS IS DNE AND NAME > 2 CHAR'S TO :ERR1:
APPEND MOD CONVENTION TO FRONT OF NAME
ELSE
REMOVE MOD CONVENTION FROM NAME
ENDIF
ENDDO
CALL WRITE TO WRITE NEW TOC RECORDS TO NEWFIL
ERREXIT IF WRIT ERROR TO :ERR1:
SET TOCPTTR TO FIRST TOC ENTRY
SET FIRST RECORD NUMBER AND LAST RECORD NUMBER
CONV SIZE FROM BLOCKS TO WORDS
DO UNTIL ALL RECORDS ARE CUPIED (SIZE = 0)
SET LENGTH TO MAXIMUM SIZE OF 1 READ/WRITE (1024)
IF SIZE < LENGTH THEN
SET LENGTH TO SIZE
ENDIF
CALL READ TO READ LENGTH WORDS OF DATA
ERREXIT IF READ ERROR TO :ERR1:
DECREMENT SIZE BY LENGTH READ
UPDATE FIRST AND LAST RECORD NUMBERS
START SEARCH FOR TOCPTTR = TOCPTTR TO LAST TOC ENTRY
EXIT IF RECORD # IN TOC ENTRY > LAST RECORD NUMBER
IF CLASS IS AN INTERFACE TABLE THEN
COMPUTE INDEX INTO DATABUF FROM FIRST RECORD #, RECORD # IN TOC ENTRY
AM, INDEX IN TOC ENTRY
SET NAME IN INTERFACE TABLE TO NAME IN TOC ENTRY
ENDIF
ENDSEARCH
CALL WRIT TO WRITE LENGTH WORDS OF DATA TO NEWFIL
ERREXIT IF WRIT ERROR TO :ERR1:
ENDDO
CALL CLOSE TO CLOSE NEWFIL
CALL CLOSE TO CLOSE OLDFIL
CALL XDDBD TO DELETE OLDFIL FROM PDB LOG FILE
CALL PURGE TO PURGE OLDFIL FROM SYSTEM
EXIT XLCDB

:ERR1:
CALL CLOSE TO CLOSE NEWFIL
CALL PURGE TO PURGE NEWFIL

:ERR2:
CALL CLOSE TO CLOSE OLDFIL

:ERR3:
IF REQUEST WAS PDB TO MD5 THEN
SET GUAL TO SAY DELETE MD5 FILE
ELSE ( REQUEST WAS MD5 TO PDB )
SET GUAL TO SAY DELETE PDB FILE
ENDIF
CALL XDDBD TO DELETE MD5/PDB FROM LOG FILE
IF ERROR WAS FILE MANAGER THEN
CALL XRMG TO DISPLAY ERROR AND RETURN CODE
ELSE
CALL XRMG TO DISPLAY ERROR
ENDIF
END XLCDB
FORTRAN CALLING SEQUENCE:

CALL XLDOS ('XLDOS')

XLDOS IS THE DATA BASE FILES MAIN ROUTINE. IT DETERMINES WHICH
DATA BASE ROUTINE IS NEEDED AND GATHERS INPUTS NEEDED BY THAT
ROUTINE. HANDLES ALL PDB LOG FILE RELATED TRANSACTIONS.

INPUTS IN COMMON:

XE(3) ISECU, XE(7) NUMBR,
XE(85) TOKENS, XE(142) ICR

INTERNAL VARIABLES IN COMMON

XE(3) QUAL - USER QUALIFIER REQUIRED BY XDB ROUTINES AND XRFM
XE(8) SECU - TEMPORARY SLOT FOR ISECU DURING THIS OVERLAY
XB(3) OLDFL - OLD FILE NAME
XB(6) NEWFL - NEW FILE NAME
XB(9) TOTSIZ - SIZE OF OLDFL/NEWFL IN BLOCKS
XB(10) FILCHR - 4 BASE CHARACTERS OF FILE NAME
XB(12) QUALIF - QUALIFIER ENTERED BY USER

DATA ROUTINES USED:

KCVT, OPENW

FILE ROUTINES USED:

FILE(3), XDBRF, XDMIN, XLDB, XLRCH

COMMON USER:

ERIQUivalence (XE(3), ISECU),

+ (XE(3) QUAL ), (XE(7), NUMBR ),
+ (XE(8) SECU ), (XE(85), TOKENS),
+ (XE(142) ICR ), (XE(145) COMDB),
+ (XB(3) OLDFL ), (XB(6) NEWFL ),
+ (XB(9) TOTSIZ ), (XB(10), FILCHR),
+ (XB(12), QUALIF ), (XB(99), IERR ),
+ (XB(100), ISECB )
BEGIN XLDBF:
IF REQUEST IS TO CREATE/MODIFY LOG FILE THEN
CALL OPEN TO OPEN MDB/PDB LOG FILE
IF OPEN ERROR SAYS FILE NOT FOUND THEN
CALL XLPCR TO CREATE MDB/PDB LOG FILE
ELSE
ERREXIT IF OPEN ERROR TO :FILERR:
CALL XLPMOD TO MODIFY MDB/PDB LOG FILE
ENDIF
ELSE
DO UNTIL USER REQUESTS EXIT (?):
CALL XTCOM TO PROMPT USER FOR NAME AND USER ID
IF RESPONSE IS NOT EXIT (PERCENT) THEN
ERREXIT IF RESPONSE IS INVALID TO :ERR1:
SAVE 4 CHARACTER NAME AND ID IN COMMON
IF REQUEST WAS PDB TO MDB THEN
SET QUALIFIER TO SEARCH FOR PDB NAME
ELSE (REQUEST WAS FOR MDB TO PDB)
SET QUALIFIER TO SEARCH FOR MDB NAME
ENDIF
CALL XDOBV TO VERIFY EXISTENCE OF MDB/PDB DEPENDING ON QUALIFIER
ERREXIT IF NAME WAS NOT FOUND TO :ERR1:
ERREXIT IF FILE MANAGER ERROR TO :FILERR:
IF REQUEST WAS PDB TO MDB THEN
SET QUALIFIER TO ADD PDB TO LOG FILE
ELSE (REQUEST WAS MDB TO PDB)
SET QUALIFIER TO ADD PDB TO LOG FILE
ENDIF
CALL XDOBA TO ADD MDB/PDB NAME TO LOG FILE DEPENDING ON QUALIFIER
ERREXIT IF DUPLICATE NAME OF
ERREXIT IF MAXIMUM NUMBER OF ENTRIES EXIST TO :ERR1:
ERREXIT IF FILE MANAGER ERROR TO :FILERR:
IF REQUEST WAS PDB TO MDB THEN
CALL XRQFM TO SET OLDfil TO PDB NAME
SET NEWfil TO MDB NAME
ELSE (REQUEST WAS MDB TO PDB)
SET OLDfil TO MDB NAME
CALL XRQFM TO SET NEWfil TO PDB NAME
ENDIF
CALL XLDBG TO COPY OLDfil TO NEWfil
ENDIF
ENDIF
EXIT XLDBF:
1
:ERR1:
CALL XRMSG TO DISPLAY ERROR
RETURN TO PROMPT FOR ANOTHER 4 CHARACTERS AND USER ID
2
:FILERR:
CALL XRMSG TO DISPLAY FILE ACCESS ERROR
1 END XLDBF
1 CD**********
2 CD0
3 CD0  FORTRAN CALLING PROCEDURE FOR DELETE PROCESSOR
4 CD0
5 CD0  CALL XELDS ('XLDEL')
6 CD0
7 CD0
8 CD**********
9 CD0
10 CD0  XLDEL DELETES A PROCESSOR FROM THE LIBRARY DIRECTORY AND
11 CD0  THE PROMPT FILE. IF THE PROCESSOR HAS AN INTERFACE TABLE,
12 CD0  IT DELETES THE DEFAULT INTERFACE TABLE FILE ALSO.
13 CD0
14 CD0
15 CD0  INTERNAL VARIABLES:
16 CD0  COMMN - (INTEGER, 7 WORDS) IS A TEMPORARY WORK AREA
17 CD0  WHERE ENTRY IN LIBRARY DIRECTORY IS STRIPPED DOWN TO PROCESSOR NAME
18 CD0  PRNAME - (INTEGER, 2 WORDS) PROCESSOR NAME TO BE
19 CD0  DELETED.
20 CD0
21 CD0
22 CD0  RTE/ FMGR ROUTINES USED:
23 CD0  IAND, XCVT, CREAT, OPEN, READF, WRITF, CLOSE, PURGE
24 CD0  FDS ROUTINES USED:
25 CD0  XRCPR, XRSET, XRMOV, XRMSG, XRCK, XRSET, XRUPK, XTCON
26 CD0
27 CD0  XE AND XB COMMON USED
28 CD0  EQUIVALENCE (XE(3), XSECU),
29 CD0  + (XE(142), ICR), (XE(145), COMBUF),
30 CD0  + (XG(48), LIBD1), (XG(51), LIBD2)
31 CD0
32 CD0  CD**********
1 BEGIN XLDEL

2 BEGIN XLDEL

2 PROMPT:

CALL XTCOM TO PROMPT FOR PROCESSOR NAME

IF XTCOM RETURN CODE IS NOT ZERO OR
RESPONSE IS NOT A VALID PROCESSOR NAME THEN
CALL XRMSG TO WRITE 'ERROR MESSAGE
GO TO :PRMPT:

EXIT XLDEL IF RETURN CODE SAYS % ENTERED

ENDIF

CALL OPEN, READ AND CLOSE TO READ IN LIBRARY DIRECTORY

EREXIT IF FILE ERROR TO :FILERR:

IF PROCESSOR IS NOT IN LIBRARY DIRECTORY THEN
CALL XRMSG TO DISPLAY 'ERROR
GO TO :PRMPT:

ENDIF

DECIMATE # PROCESSORS BY 1

CALL PURGE TO PURGE OLD LIBRARY DIRECTORY

EREXIT IF RETURN CODE < ZERO TO :FILERR:

EXIT XLDEL IF #PROCESSORS IS ZERO

CALL CREAT, WRITF AND CLOSE TO RECREATE LIBRARY DIRECTORY

EREXIT IF FILE ERROR TO :FILERR:

CREATE THE PROMPT FILE NAME

CALL PURGE TO PURGE THE PROMPT FILE

EREXIT IF PURGE ERROR TO :FILERR:

CALL XRMSG TO SAY FILE PURGED SUCCESSFULLY

IF PROCESSOR HAD AN INTERFACE TABLE THEN

CREATE DEFAULT IT NAME

CALL PURGE TO PURGE DEFAULT IT

EREXIT IF RETURN CODE IS NOT ZERO TO :FILERR:

CALL XRMSG TO DISPLAY 'FILE :AGED' MESSAGE

ENDIF

GO TO :PRMPT:

:FILERR: CALL XRMSG TO WRITE file access error

1 END XLDEL
FORTRAN CALLING SEQUENCE:

CALL XLIFL

XLIFL CREATES THE DEFAULT INTERFACE TABLE FILE

INTERNAL VARIABLES

BLOCKS - (INTEGER, 1 WORD) # BLOCKS TO BE ALLOCATED TO THE FILE

FDS ROUTINES USED:

XEXT, XMSG

RTE ROUTINES USED:

CLOSE, CREAT, WRITF

COMMON USED:

EQUIVALENCE (X(0), IECU,)

+ (X(142), ICR, (X(90), LITPR,)

+ (X(91), LITLEN),

+ (X(96), HOPARM), (X(101), MEMP,)

+ (X(108), PMHS, (X(1), IERR,)

+ (X(2), IOCQ)
1 BEGIN XLIFL
2 EXTRACT LITERAL LENGTH FROM HEADER
3 COMPUTE # BLOCKS FOR THIS FILE
4 CALL CREATE TO CREATE DEFAULT INT TABLE FILE
5 ERREXIT IF RETURN CODE < ZERO TO :FILERR:
6 CALL WRITE TO WRITE LITERAL AND SPECS
7 ERREXIT IF RETURN CODE < ZERO TO :FILERR:
8 IF THERE IS A LITERAL RECORD THEN
9 CALL WRITE TO WRITE LITERALS
10 ERREXIT IF RETURN CODE < ZERO TO :FILERR:
11 EDF
12 CALL CLOSE
13 ERREXIT IF RETURN CODE < ZERO TO :FILERR:
14 CALL XMSG TO DISPLAY FILE NAME CREATED MESSAGE
15 EXIT XLIFL
16 :FILERR:
17 CALL XMSG TO WRITE FILE ACCESS ERROR
18 CALL XEXIT TO RETURN TO MAIN PROGRAM
19 END XLIFL
FCRTRAN CALLING SEQUENCE:

CALL XLINS

XLINS DISPLAYS INSTRUCTIONS FOR ENTERING PARAMETER SPECS

INTERNAL VARIABLES:

PROMPT - (INTEGER, 120 WORDS) ALL 7 LINES IN AN ARRAY TO BE DISPLAYED

RTE ROUTINES USED:

EXEC

COMMON USED:

EQUIVALENCE (RE(1), LU )

BEGIN XLINS

CALL EXEC TO DISPLAY ALL 7 LINES

END XLINS
FORTRAN CALLING PROCEDURE:
CALL XELDB (XLINT)

XLINT SEGMENT SETS UP COMMON TO CALL THE INTERFACE TABLE TO
ACCEPT DEFAULT VALUES FOR THE INTERFACE TABLE

FDS FUNCTIONS AND SUBROUTINES USED:
XEINT, XERTM, XINIT, XRMOV

COMMON USED:
EQUIVALENCE (XRM(5), MASSTA), (XRM(6), SUBSTA)
(XRM(7), NUMDIR), (XRM(10), MPRMT), (XRM(37), ARCH), (XRM(41), PMPMT)
(XRM(73), ISPsIZ), (XRM(89), LITDMN)
(XRM(90), LITFTR), (XRM(91), LITLEM)
(XRM(92), NAPES)
(XRM(96), NUMARG), (XRM(97), NEWTAB)
(XRM(100), MPRLMG), (XRM(101), MPRM ),
(XRM(140), END)
BEGIN XLINT

1 SET MASTER STATE AS EXEC
2 SET SUBSTATE AS INTERFACE TABLE EDITOR
3 SET UP LIST OF VALID DIRECTIVES
4 SET UP LIST FLAG TO GET ENTIRE LIST
5 SET CURRENT ARGUMENT TO ZERO
6 SET PROMPT MODE TO ALL
7 SET INDEX INTO PARTS OF SHORT PROMPTS
8 SET NEW TABLE NAME TO ZERO
9 INITIALIZE ARRAY OF TYPE LENGTHS
10 IF LITERAL LENGTH IS ZERO THEN
11 SET LITERAL POINTER TO FIRST AVAILABLE WORD
12 ELSE
13 SET LITERAL POINTER TO FIRST WORD OF LITERALS
14 CALL XEINT TO UNPACK LITERALS
15 ENDIF
16 CALL XIMIX TO ACCEPT DEFAULT VALUES
17 CALL XERAI TO RETURN
18 END XLINT
FORTRAN CALLING SEQUENCE:

CALL XMODS ('XMOD')

XLMOD ALLOWS A USER TO MODIFY THE VERSION, SPECS, AND PROMPTS FOR A PROCESSOR AS WELL AS ADD/DELETE PARAMETERS FROM AN INTERFACE TABLE.

RTE ROUTINES USED:
CLOSE, IAND, OPEN, POSMT, READF, WRITF

FD0 ROUTINES USED:
XIIF, XIINV, XLPIF, XLSF, XREPB, XREX,
XROV, XRMG, XRPK, XRSFT, XRUPK, XRCON

COMMON USED:

EQUIVALENCE (XEC(3), ISECU),
+ (XEC(5), HASSTA), (XEC(6), SUBSTA),
+ (XEC(7), VALFLG), (XEC(135), PUNAM),
+ (XEC(145), PRECW), (XEC(85), EDSTOK),
+ (XEC(86), INTTOK), (XEC(69), NAMTOK),
+ (XEC(97), HEFTOK), (XEC(142), ECH ),
+ (XEC(145), CONNH ), (XEC(260), VERS ),
+ (XEC(333), SFLAG ), (XEC(357), CFLAG ),
+ (XEC(377), ARENO ), (XEC(388), IT ),
+ (XEC(433), LEM ), (XEC(444), HEGFLE ),
+ (XEC(881), LITPS ), (XEC(907), LIPTBS ),
+ (XEC(982), IXMOX ), (XEC(963), HOPARK ),
+ (XEC(1001), WEBR ), (XEC(1015), LEB01 ),
+ (XEC(1016), LEB92 ), (XEC(1038), PARA )
BEGIN XLMOD
IF VALFLAG SAYS THIS IS AN ORIGINAL REQUEST TO MODIFY THEN
SET RTN = 1

CALL XLMOD TO DISPLAY SPEC INSTRUCTIONS
INITIALIZE MASTER AND SUB STATE FLAGS
CALL XCMD TO PROMPT FOR PROCESSOR NAME
EXIT XLMOD IF RETURN CODE SAYS X ENTERED
ERREXIT IF XCMN RETURN CODE NON-ZERO OR
ERREXIT IF INVALID PROCESSOR NAME (NOT 8CHAR NAME) TO :PRMERR:
CALL OPEN, READF AND CLOSE TO READ IN LIBRARY DIRECTORY
ERREXIT IF THERE IS A FILE ERROR TO :FILERR:
ERREXIT IF PROCESSOR IS NOT IN LIBRARY DIRECTORY TO :PRMERR:
SAVE INTERFACE TABLE BIT AND VERSION NUMBER
PERFORM VERSION TO UPDATE VERSION NUMBER
IF THE PROCESSOR HAD AN INTERFACE TABLE THEN
SET NEW VERSION NUMBER IN INTERFACE TABLE
CREATE DEFAULT INTERFACE TABLE NAME
CALL OPEN AND READF TO READ IN NEDR AND SPECS
IF THERE ARE LITERALS THEN
CALL READF TO READ IN LITERALS
ENDIF
CALL CLOSE TO CLOSE FILE
ERREXIT IF THERE WAS A FILE ERROR TO :FILERR:
CREATE PROMPT TABLE NAME
CALL OPEN, READF AND CLOSE TO READ IN SHORT PROMPTS
ERREXIT IF THERE WAS A FILE ERROR TO :FILERR:
CALL NAME TO RENAME PROMPT FILE >XLMAP
ERREXIT IF NAME ERROR TO :FILERR:
SET CODES ARRAY TO MODIFY/ABSTRACT AND NO CHANGES TO PARAMETER SPECS
PERFORM DELPRM TO DELETE PARAMETERS
PERFORM MODPRM TO MODIFY PARAMETERS
PERFORM ADDPRM TO ADD PARAMETERS
CALL XLPLF TO CREATE NEW PROMPT FILE
PERFORM DEFAULT TO ADD/MODIFY/DELETE ANY DEFAULT VALUES
ELSE
CALL NAME TO RENAME PROMPT FILE >XLMAP
ERREXIT IF NAME ERROR TO :FILERR:
SET CODES ARRAY TO MODIFY ABSTRACT ONLY
CALL XLPLF TO CREATE NEW PROMPT FILE
PERFORM XLMOD - NO RETURN EXPECTED
ENDIF
CALL PURGE TO PURGE OLD DEFAULT INTERFACE TABLE FILE
ERREXIT IF FILE ERROR TO :FILERR:
CALL XLMF TO CREATE NEW DEFAULT INTERFACE TABLE FILE
SET VALFILE TO SAY ORIGINAL REQUEST TO MODIFY
PERFORM XLMOD - NO RETURN EXPECTED

:PRMERR:
CALL XRMSG TO DISPLAY ERROR MESSAGE
GO TO (:PRM15,:PRM2,:PRM3,:PRM4,:PRM5,:PRM6,:PRM7),RTN

:FILERR:
CALL XRMSG TO DISPLAY FILE ERROR
END XLMOD

KEEPING OF THE
ORIGINAL EAGLE IS POOR
BEGIN VERSION

SET RTN = 2

CALL XCOM TO PROMPT USER FOR VERSION NUMBER

IF RETURN CODE IS NOT CR THEN

ERRE0: IF RETURN CODE IS NOT-ZERO CR

ERRE1: IF VERSION IS INVALID TO :PRM2:

IF INPUT VERSION IS NOT EQUAL TO OLD VERSION THEN

CALL XRSET TO PUT NEW VERSION IN ENTRY

CALL OPEN, WRITF, CLOSE TO UPDATE LIBRARY DIRECTORY

ERRE2: IF FILE ERROR TO :FILEERR:

ENDIF

ENDIF

END VERSION

BEGIN DELPRM

SET RTN = 3

END OF

BEGIN DELPRM

DO UNTIL RETURN CODE IS CR ENTERED

IF NUMBER OF PARAMETERS > 1 THEN

CALL XCOM TO PROMPT FOR DELETE PARAMETER NAME

IF RETURN CODE IS NOT CR ENTERED THEN

PERFORM RSPMD TO INTERPRET RESPONSE

PERFORM CHDAT TO CHECK FOR EXISTING DATA

SET ARGNO TH WORD IN CODES TO SAY 'DELETED'

CALL XREM0V TO MOVE DATA TO DELETE PARAMETER

DECREMENT # PARAMETERS BY 1

ENDIF

ELSE

CALL XRMS06 TO DISPLAY NOT PARAMETERS CAN BE DELETED

EXIT DELPRM

ENDIF

END OF

BEGIN MODPRM

SET RTN = 6

BEGIN MODPRM

DO UNTIL RETURN CODE IS CR ENTERED

CALL XCOM TO PROMPT FOR MODIFY PARAMETER NAME

IF RETURN CODE IS NOT CR ENTERED THEN

PERFORM RSPMD TO INTERPRET RESPONSE

PERFORM CHDAT TO CHECK FOR EXISTING DATA

SET ARGNO TH NON-DELETED WORD IN CODES TO SAY 'MODIFIED'

CALL XCLIPE TO PROMPT USER FOR SPECIFICATIONS

SET IT COMPLETE BIT OFF

ENDIF

ENDO

END OF

END MODPRM

END OF
959 1 BEGIN ADDPRM
960 2 SET RTN = 5
961 2 :PROMPT:
962 2 DO UNTIL RETURN CODE IS CR ENTERED
963 3 IF NUMBER OF PARAMETERS < 43 THEN
964 4 CALL XCTOM TO PAROMPT FOR ADD PARAMETER BEFORE/AFTER PARAMETER NAME
965 4 IF RETURN CODE IS NOT CR ENTERED THEN
966 5 PERFORM RSPND TO INTERPRET RESPONSE
967 5 5 SET ARGNO TH NON-DELETE WORD IN CODES TO SAY 'ADDED'
968 5 INCREMENT NUMBER OF PARAMETERS BY 1
969 5 CALL XRSN0 TO MOVE DATA TO MAKE SPACE FOR NEW PARAMETER
970 5 CALL XLSPS TO GET NEW SPECS FOR THIS PARAMETER
971 5 SET IT COMPLETE BIT OFF
972 4 ENDIF
973 3 ELSE
974 4 CALL XRSNSG TO DISPLAY NO MORE PARAMETERS CAN BE ADDED
975 3 EXIT ADDPRM
976 3 ENDIF
977 2 END00
978 1 END ADDPRM
979 1 +
980 1 *
981 1 *
982 1 BEGIN DEFAULT
983 2 SET RTN = 7
984 2 :PROMPT:
985 2 CALL XCTOM TO PROMPT FOR ADD/MODIFY/DELETE DEFAULT VALUES
986 2 CALL XCTOM TO PROMPT FOR ADD/MODIFY/DELETE DEFAULT VALUES
987 2 ERXEXIT IF RETURN CODE IS NON-ZERO TO :PRMEX:
988 2 IF RESPONSE IS YES THEN
989 3 SET WFLAG TO SAY CALL INTERFACE TABLE EDITOR
990 2 EXIT XLMOD
991 2 ENDIF
992 1 END DEFAULT
O V 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

W K O.

OH W J

K = W H W

►

3c0! r

W O K a

{~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,

z WN;ll KW %44_j

NWM M t {D

►

K i W i N or Ji F ^J

M0~tM-iO FNJJ =W1VF^10t 1--/^ iHm 0- CA"HW w4- 6-1090 0V7H O	1.4RM

r r r r r r r r r r r r r r r

1 2’O S S

W O K d

a

W {~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,

z WN;ll KW %44_j

NWM M t {D

►

K i W i N or Ji F ^J

M0~tM-iO FNJJ =W1VF^10t 1--/^ iHm 0- CA"HW w4- 6-1090 0V7H O	1.4RM

r r r r r r r r r r r r r r r

1 2’O S S

W O K d

a

W {~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,

z WN;ll KW %44_j

NWM M t {D

►

K i W i N or Ji F ^J

M0~tM-iO FNJJ =W1VF^10t 1--/^ iHm 0- CA"HW w4- 6-1090 0V7H O	1.4RM

r r r r r r r r r r r r r r r

1 2’O S S

W O K d

a

W {~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,

z WN;ll KW %44_j

NWM M t {D

►

K i W i N or Ji F ^J

M0~tM-iO FNJJ =W1VF^10t 1--/^ iHm 0- CA"HW w4- 6-1090 0V7H O	1.4RM

r r r r r r r r r r r r r r r

1 2’O S S

W O K d

a

W {~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,

z WN;ll KW %44_j

NWM M t {D

►

K i W i N or Ji F ^J

M0~tM-iO FNJJ =W1VF^10t 1--/^ iHm 0- CA"HW w4- 6-1090 0V7H O	1.4RM

r r r r r r r r r r r r r r r

1 2’O S S

W O K d

a

W {~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,

z WN;ll KW %44_j

NWM M t {D

►

K i W i N or Ji F ^J

M0~tM-iO FNJJ =W1VF^10t 1--/^ iHm 0- CA"HW w4- 6-1090 0V7H O	1.4RM

r r r r r r r r r r r r r r r

1 2’O S S

W O K d

a

W {~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,

z WN;ll KW %44_j

NWM M t {D

►

K i W i N or Ji F ^J

M0~tM-iO FNJJ =W1VF^10t 1--/^ iHm 0- CA"HW w4- 6-1090 0V7H O	1.4RM

r r r r r r r r r r r r r r r

1 2’O S S

W O K d

a

W {~D

Z Z<

~OwH

NW

H =H JJO OtDZ1•+i t ZtW WY

Vw

=1 K W

5F

V0,
CD************
CD1  FORTRAN CALLING PROCEDURE
CD2  CALL HELDS ('XLMGS')
CD3  XLMGS PROVIDES MAINTENANCE OF THE FDS MESSAGE FILE XRMGS
CD4  
CD5  
CD6  
CD7  
CD8  
CD9  
CD10  
CD11  
CD12  
CD13  
CD14  
CD15  
CD16  
CD17  
CD18  
CD19  
CD20  
CD21  
CD22  
CD23  
CD24  
CD25  
CD26  
CD27  
CD28  
CD29  
CD30  
CD31  
CD32  
CD33  
CD34  
CD35  
CD36  
CD37  
CD38  
CD39  
CD40  
CD41  
CD42  

LOCAL
CD4  AREA   - NUMERICAL AREA INDICATOR FOR MESSAGE
CD5  DIRECT  - MESSAGE DIRECTORY (SEE SDD 6.2.4.12)
CD6  I   - INDEX TO BEGINNING OF CURRENT MESSAGE ENTITY
CD7  IDC - FILE MANAGER DATA CONTROL BLOCK
CD8  IERR - FILE MANAGER & XTM RETURN CODE
CD9  INUMB - MESSAGE NUMBER WITHIN MESSAGE AREA
CD10  INBLK - BLOCK NUMBER WITHIN FILE
CD11  ICREAT - NEXT BLOCK AVAILABLE FOR ALLOCATION
CD12  IUPD - NUMBER OF BLOCK CONTAINING MESSAGE
CD13  IRPOS - MESSAGE LOCATION WITHIN 128 WORD BLOCK (1, 33, 65 OR 97)

CD5  NOTES
CD6  USES APOS, CLOSE, CREATE, EXEC, IAMD, KCVT, OPEN, READF, WRITF,
CD7  XER, XIG, XRMGS, XTMOM, XUBG
CD8  
CD9  WHEN REPLACING AN EXISTING MESSAGE, A NULL RESPONSE WILL LEAVE THE
CD10  EXISTING TEXT IN PLACE.
CD11  MESSAGE UPDATING MAY BE TERMINATED AT ANY TIME BY ENTERING A %
1067 1 BEGIN XLM SG
1068 2  OPEN XLM SG
1069 3  IF FILE NOT FOUND
1070 4   THEN
1071 5     OUTPUT 'MESSAGE FILE CREATION'
1072 6     DO FOR EACH OF THE 32 DIRECTORIES ENTRIES
1073 7     PROMPT FOR AREA ID AND MAXIMUM NUMBER OF MESSAGES FOR THIS ENTRY NUMBER
1074 8     IF RESPONSE WAS NULL
1075 9     THEN
1076 5     CLEAR ENTRY
1077 4     ELSE
1078 5     STORE ID
1079 6     COMPUTE AREA ORIGIN AND STORE
1080 7     CLEAR LAST MESSAGE NUMBER
1081 8     COMPUTE NUMBER OF BLOCKS AND STORE
1082 9   ENDIF
1083 5   ENSIDO
1084 6   CREATE A CLEARED FILE OF TOTAL REQUIRED SIZE
1085 7   ELSE
1086 3   READ DIRECTORY
1087 2  ENDIF
1088 2  DO UNTIL USER INPUTS X
1089 3  PROMPT FOR MESSAGE NUMBER
1090 3  SEPARATE AREA AND MESSAGE NUMBER AND COMPUTE BLOCK NUMBER AND MESSAGE LOC
1091 3  IF VALID AREA AND BLOCK NUMBER <= NUMBER OF BLOCKS
1092 3  THEN
1093 4   READ BLOCK
1094 4   IF FIRST WORD OF MESSAGE IS NOT NULL (MESSAGE ALREADY EXIST)
1095 4   THEN
1096 5    DISPLAY OLD MESSAGE TEXT
1097 4    ENDIF
1098 4    PROMPT FOR TEXT
1099 4    IF NON-NULL RESPONSE
1100 4    THEN
1101 5   STORE TEXT IN BLOCK
1102 5   REWRITE BLOCK
1103 5   IF MESSAGE NUMBER > LAST MESSAGE NUMBER
1104 5   THEN
1105 6    REPLACE LAST MESSAGE NUMBER WITH NEW NUMBER
1106 5    ENDIF
1107 4    ENDIF
1108 3  ELSE
1109 4  OUTPUT 'XL29 AREA INVALID OR NUMBER TOO LARGE'
1110 3  ENDIF
1111 2  ENDDO
1112 2  REWRITE DIRECTORY BLOCK
1113 1 END XLM SG
BEGIN XL CR

CALL CREAT TO CREATE MDB/PDB LOG FILE

ERREXIT IF CREATE ERROR TO :FILERR:

CALL LOG RECOR BUFFER TO ZEROS

SET # MDB FILES CURRENTLY USED TO ZERO

SET MAXIMUM NUMBER OF 20

CALL WRITE TO WRITE LOG RECORD TO LOG FILE

ERREXIT IF WRITE ERROR TO :FILERR:

DO FOR EACH REMAINING LOG RECORD

DO FOR THIS PAIR OF USER ID'S

CALL XTCON TO PROMPT FOR MAXIMUM ALLORED # PDB'S

SET MAXIMUM # PDB FILES TO RESPONSE

SET # PDB FILES CURRENTLY USED TO ZERO

ENDDO

CALL WRITE TO WRITE 1 PDB RECORD TO LOG FILE

ERREXIT IF WRITE ERROR TO :FILERR:

ENDDO

CALL CLOSE TO CLOSE FILE

ERREXIT IF CLOSE ERROR TO :FILERR:

EXIT XL CR

:FILEERR:

CALL XRMSG TO DISPLAY FILE ERROR

CALL CLOSE TO CLOSE FILE

END XL CR
**FORTRAN CALLING SEQUENCE:**

```fortran
CALL XLPFL (NOCOD, CODES)
```

**INPUTS IN CALLING SEQUENCE:**

- **NOCOD** - NUMBER OF CODES IN THE CODE ARRAY
- **CODES** - ARRAY OF CODES THAT REPRESENT:
  - CODES(1) PROCESSOR ABSTRACT
  - CODES(2) THRU CODES(NOCOD) PARAMETERS 1 THRU N

**INPUTS IN COMMON:**

- **XE(1) LU, XE(3) ISECU, XE(16) PROCNAM, XE(142) ICR, XE(96) NOPARM, XE(108) PARMS**

**RTE FUNCTIONS USED:**

- **CLOSE, CREAT, EXEC, OPEN, POSN, PUNGE, READF, WRITF**

**FDS FUNCTIONS USED:**

- **XERTN, XRPCP, ZPMOV, ZRMSG, XRPCK, XRUPK, XICOM**

**COMMON USED:**

- **EQUIVALENCE (XE(1), LU)**

```fortran
+ (XE(3), ISECU), (XE(77), VALFL6),
+ (XE(16), PROCNAM), (XE(85), TOKEMS),
+ (XE(142), ICR), (XE(144), COMPIR),
+ (XE(145), COMBUS), (XE(96), NOPARM),
+ (XE(108), PARMS)**
```
1227 1 BEGIN XLPFL
1228 2 COMPUTE SIZE OF FILE AS 8 PARAMETERS *3
1229 3 CALL CREAT TO CREATE PROMPT FILE
1230 4 ERREXIT IF CREATE ERROR TO :FILERR:
1231 5 STUFF SYNTAX RECORD 0 AND 8 PARAMETERS INTO LIST OF SHORT PROMPTS
1232 6 CALL WRTF TO WRITE SHORT PROMPT RECORD(S)
1233 7 ERREXIT IF WRTF ERROR TO :FILERR:
1234 8 IF ABSTRACT CODE IS MODIFY THEN
1235 9 CALL OPEN TO OPEN OLD PROMPT FILE >XTMP
1236 10 ERREXIT IF OPEN ERROR TO :FILERR:
1237 11 CALL READ TO READ EXISTING ABSTRACT
1238 12 ERREXIT IF READ ERROR TO :FILERR:
1239 13 CALL EXEC TO DISPLAY EXISTING ABSTRACT
1240 14 CALL XTCOM TO PROMPT USER TO MODIFY ABSTRACT
1241 15 IF RESPONSE IS CR (NO MODIFICATION) THEN
1242 16 CALL WRTF TO WRITE EXISTING ABSTRACT TO NEW FILE
1243 17 ERREXIT IF WRTF ERROR TO :FILERR:
1244 18 ELSE (NEW ABSTRACT WAS ENTERED)
1245 19 PERFORM FORMAT TO FORMAT RESPONSE INTO FILE
1246 20 ENDF.
1247 21 ELSE (ABSTRACT CODE IS AD)
1248 22 CALL XTCOM TO PROMPT USER TO ENTER NEW ABSTRACT
1249 23 PERFORM FORMAT TO FORMAT RESPONSE INTO FILE
1250 24 ENDF.
1251 25 IF PARAMETERS IS NOT ZERO THEN
1252 26 PERFORM EXPRM TO GET DEFINITIONS FOR EACH PARAMETER
1253 27 ENDF.
1254 28 CALL CLOSE TO CLOSE PROMPT FILE
1255 29 ERREXIT IF CLOSE ERROR TO :FILERR:
1256 30 IF THERE WAS AN OLD FILE THEN
1257 31 CALL CLOSE TO CLOSE OLD FILE >XTMP
1258 32 CALL PURGE TO PURGE OLD FILE >XTMP
1259 33 ENDF.
1260 1 EXIT XLPFL
1261 2 :FILERR:
1262 2 CALL XRMFG TO DISPLAY ERROR CODE
1263 2 CALL CLOSE TO CLOSE NEW FILE
1264 2 CALL CLOSE TO CLOSE OLD FILE >XTMP
1265 2 CALL PURGE TO PURGE OLD FILE >XTMP
1266 2 SET VALFNG = 0
1267 2 CALL XRTM TO RETURN TO MAIN ***NO RETURN TO HERE***
1268 1 END XLPFL
1270 1 BEGIN FORMAT
1271 1 INITIALIZE TOKEN POINTER AND TOTAL WORD COUNT
1272 DO UNTIL EOS IS DETECTED IN RESPONSE
1273 ERREXIT IF RESPONSE IS NOT CHARACTER STRING TO :ERR1:
1274 ERREXIT IF RESPONSE IS TOO LONG (>128 WORDS) TO :ERR1:
1275 CALL XMOV TO MOVE RESPONSE TO BUFFER
1276 SET CONTROL CHARACTERS IN BUFFER
1277 INCREMENT TOTAL WORD COUNT BY THIS RESPONSE
1278 ERREXIT IF NEXT RESPONSE IS NOT A COMMA TO :ERR1:
1279 INCREMENT TOKEN POINTER TO NEXT CHARACTER STRING
1280 ENDDO
1281 SET REMAINDER OF BUFFER TO NULL
1282 CALL WRITF TO WRITE NEW RESPONSE TO PROMPT FILE
1283 ERREXIT IF WRITF ERROR TO :FILEERR:
1284 1 END FORMAT
1285 1 :ERR1:
1286 1 CALL XRMSG TO DISPLAY INVALID RESPONSE
1287 1 RETURN TO REPROMPT USER FOR ANOTHER RESPONSE
1299 1 BEGIN EXTPRN
1300 2 DO FOR EACH ENTRY IN CODES ARRAY
1301 3 BEGIN CASE (:MOD:, :DEL:, :ADD:, CODES+)
1302 4 :MOD: (SPECS WERE MODIFIED)
1303 4 CALL READ TO READ EXISTING DEFINITION FROM XLTMP
1304 4 ERREXIT IF READF ERROR TO :FILERR:
1305 4 CALL EXEC TO DISPLAY EXISTING DEFINITION
1306 4 IF RESPONSE IS CR (NO RESPONSE) THEN
1307 5 CALL WRITF TO WRITE EXISTING DEFINITION TO NEW FILE
1308 5 ELSE (NEW DEFINITION WAS ENTERED)
1309 5 PERFORM FORMAT TO FORMAT DEFINITION INTO FILE
1310 4 ENDFIF
1311 4 :DEL: (SPECS WERE DELETED)
1312 4 CALL POSNT TO POSITION XLTMP OVER THIS ENTRY
1313 4 ERREXIT IF POSNT ERROR TO :FILERR:
1314 4 :ADD: (SPECS WERE ADDED)
1315 4 CALL XTOM TO PROMPT USER TO ENTER NEW DEFINITION
1316 4 PERFORM FORMAT TO FORMAT RESPONSE INTO FILE
1317 3 END CASE
1318 3 DO UNTIL RESPONSE IS CR (EXIT)
1319 4 CALL XTOM TO PROMPT USER FOR SHORT PROMPT TO MODIFY DEFINITION
1320 4 EXIT EXTPRN IF RESPONSE IS CR
1321 4 SEARCH SHORT PROMPTS FOR RESPONSE
1322 4 ERREXIT IF NOT FOUND TO :ERR:
1323 4 CALL READ TO READ EXISTING DEFINITION FROM NEW FILE
1324 4 ERREXIT IF READF ERROR TO :FILERR:
1325 4 CALL EXEC TO DISPLAY EXISTING DEFINITION
1326 4 CALL XTOM TO PROMPT USER TO MODIFY DEFINITION
1327 4 IF RESPONSE IS POSITIVE
1328 5 CALL POSNT TO REPOSITION NEW FILE BACK TO PREVIOUS RECORD
1329 5 ERREXIT IF POSNT ERROR TO :FILERR
1330 5 PERFORM FORMAT TO FORMAT NEW DEFINITION INTO FILE
1331 1 END EXTPRN
FORTRAN CALLING SEQUENCE:

CALL XLPNO

INPUTS FROM COMMON:

XE(1) LU, XE(3) QUAL, XE(8) SECU,
XE(142) ICA, XB(99) IERR, XB(100) INCB,

INTERNAL VARIABLES IN COMMON:

XE(50) UNO - NUMBER OF USER ID A-Z (1-26)
XB(51) RECNO - RECORD # WHERE UNO'S PDB LIST IS (2-14)
XB(52) PHAM - FILE NAME OF USER'S PDB (FULL 6 CHAR)
XB(53) DPB - NUM-ER OF FILES THAT MUST BE DELETED
WHEN NEW WAY < CURRENTLY IN USE
XB(244) OUTFU - AREA WHERE DISPLAY OF CURRENT FILES IS BUILT.

RTE ROUTINES USED:

CLOSE, EXEC, KEVT, PURGE, READ, WRITF

FDS ROUTINES USED:

XRCPA, XREXT, XRMOV, XRMSG,
XRQFN, XRQFR, XTCOM

COMMON USED:

EQUIVALENCE (XE(1), LU ), (XE(3), QUAL ),
(XE(8), SECU ),
(XE(142), ICA ),
(XE(144), COMPA ), (XE(145), COMBU ),
(XB(50), UNO ), (XB(51), RECNO ),
(XB(52), PHAM ), (XB(53), DPB ),
(XB(56), ATN ), (XB(57), MSE ),
(XB(99), IERR ), (XB(100), INCB ),
(XB(114), ITHS ), (XB(244), OUTFU ),
(XB(304), XNCD )
1387 1 BEGIN XLPNO
1388 2 DO UNTIL USER RESPONSE IS EXIT (3)
1389 3 CALL XTCOM TO PROMPT USER FOR USER ID
1390 4 ERREXIT IF RESPONSE IS INVALID TO :ERR1:
1391 5 SAVE USER ID AND NUMBER (1-26) IN COMMON
1392 6 CALL READ TO READ IN THE RECORD CONTAINING THIS ID
1393 7 ERREXIT IF READ ERROR TO :FILERR:
1394 8 DISPLAY MAX # ALLOWED FOR THIS USER AND # CURRENTLY USED
1395 9 CALL XTCOM TO PROMPT USER FOR NEW MAXIMUM
1396 10 IF RESPONSE IS A MODIFICATION (NOT CR) THEN
1397 11 ERREXIT IF RESPONSE IS NOT VALID TO :ERR1:
1398 12 COMPUTE DIFFERENCE AS CURRENTLY USED - NEW MAXIMUM
1399 13 IF DIFFERENCE <= 0 THEN
1400 14 CALL WRITE TO REWRITE UPDATED RECORD TO FILE
1401 15 ERREXIT IF WRITE ERROR TO :FILERR:
1402 16 ELSE
1403 17 DISPLAY LIST OF CURRENT FILES
1404 18 SET #PURGED = 0
1405 19 DO UNTIL DIFFERENCE <= 0 OR RESPONSE IS EXIT (PERCENT)
1406 20 CALL XTCOM TO PROMPT USER TO DELETE #OVER FILES
1407 21 DO FOR EACH FILE NAME IN RESPONSE
1408 22 SEARCH LIST FOR FILE NAME
1409 23 ERREXIT IF INVALID NAME TO :ERR1:
1410 24 MOVE FILE NAME TO PURGE LIST
1411 25 INCREMENT #PURGED BY 1
1412 26 DECREMENT DIFFERENCE BY 1
1413 27 COMPRESS OLD FILE NAME OUT OF LIST
1414 28 ENDDO
1415 29 ENDDO
1416 30 IF RESPONSE WAS NOT EXIT THEN
1417 31 CALL WRITE TO REWRITE UPDATED LOG RECORD
1418 32 ERREXIT IF WRITE ERROR TO :FILERR:
1419 33 DO FOR #PURGED FILES
1420 34 CALL XFRGN TO CREATE FILE NAME
1421 35 CALL PURGE TO PURGE FILE
1422 36 ENDDO
1423 37 ENDF
1424 38 ENDF
1425 39 ENDF
1426 40 CALL CLOSE TO CLOSE NDB/PDB LOG FILE
1427 41 EXIT XLPNO
1428 42 :ERR1:
1429 43 CALL XHMSG TO DISPLAY ERROR IN RESPONSE
1430 44 RETURN TO REISSUE LAST PROMPT
1431 45 :FILERR:
1432 46 CALL XHMSG TO DISPLAY FILE ERROR
1433 47 CALL CLOSE TO CLOSE NDB/PDB FILE
1434 48 ENDP
FORTRAN CALLING SEQUENCE FOR SPEC PROCESSOR:

CALL XLSPS

XLSPS PROCESSES INPUTS FOR ONE PARAMETER.

INTERNAL VARIABLES:

CLASS - (INTEGER, 1 WORD, XE(12)) IS THE PARAMETER CLASS
IPRTNT - (INTEGER, 1 WORD) IS THE INDEX INTO PARMS WHERE
THE SHORT PROMPT IS TO BE STORED
ISPEC - (INTEGER, 1 WORD) IS THE INDEX INTO PARMS WHERE
THE SPECS ARE TO BE STORED
MSG - (INTEGER, 1 WORD) IS THE CURRENT ERROR CONDITION
TYPE - (INTEGER, 1 WORD, XE(11)) IS THE PARAMETER TYPE

ROUTINES USED:
KCVT, XRCP, XRMOV, XRMSG, XRSET, XTCOM

XE COMMON USED:

EQUIVALENCE (XE(8), RETC ), (XE(9), RETN ),
+ (XEC(10), CLASS ), (XEC(11), TYPE ),
+ (XEC(145), COMBUF ), (XEC(137), ARGNO ),
+ (XEC(96), MOPARM ), (XEC(108), PARMS )
1474 1 BEGIN XLSPS
1475 2 CALL KCVT TO CONVERT PARAMETER NUMBER TO ASCII
1476 2 :PRMPT1:
1477 2 SET RETURN TO 1
1478 2 CALL XCOM TO PROMPT FOR PROMPT,CLASS,TYPE AND I/O FLAGS
1479 2 ERREXIT IF RETURN CODE IS NOT ZERO TO :PRMERR:
1480 2 INITIALIZE THIS SHORT PROMPT AND ENTRY
1481 2 ERREXIT IF SHORT PROMPT IS NOT VALID (6 CHAR NAME) TO :PRMERR:
1482 2 ERREXIT IF THIS IS A DUPLICATE SHORT PROMPT
1483 2 CALL XMOV TO MOVE SHORT PROMPT INTO SPECS
1484 2 ERREXIT IF CLASS IS NOT VALID TO :PRMERR:
1485 2 CALL XRSET TO SET CLASS IN ENTRY
1486 2 ERREXIT IF TYPE IS NOT VALID TO :PRMERR:
1487 2 ERREXIT IF CLASS IS 00 AND TYPE IS SYMBOLIC STRING TO :PRMERR:
1488 2 CALL XRSET TO SET TYPE IN ENTRY
1489 2 ERREXIT IF I/O FLAGS ARE NOT VALID TO :PRMERR:
1490 2 ERREXIT IF TYPE IS SYMBOLIC STRING AND I/O FLAGS ARE NOT INPUT TO :PRMERR:
1491 2 CALL XRSET TO SET I/O FLAGS IN ENTRY
1492 2 IF CLASS IS DATA ELEMENT THEN
1493 3 IF TYPE IS NOT SYMBOLIC STRING THEN
1494 4 SET RETURN TO 2
1495 4 :PRMPT2:
1496 4 CALL XCOM TO PROMPT FOR I AND J DIMENSIONS
1497 4 ERREXIT IF RETURN CODE IS NOT NORMAL TO :PRMERR:
1498 4 ERREXIT IF I DIMENSION IF NOT VALID TO :PRMERR:
1499 4 I: J DIMENSION IS NOT ENTERED THEN
1500 5 SET J DIMENSION TO 1
1501 4 ELSE
1502 5 ERREXIT IF J DIMENSION IS NOT VALID TO :PRMERR:
1503 5 IF J DIMENSION < 2 THEN
1504 6 SET J DIMENSION TO 1
1505 6 ELSE
1506 6 STORE I DIMENSION IN ENTRY
1507 6 :ENDIF
1508 4 :ENDIF
1509 4 SET SIZE =I DIMENSION * J DIMENSION * TYPE LENGTH
1510 4 ERREXIT IF SIZE IS NOT VALID TO :PRMERR:
1511 5 ELSE
1512 5 SET RETURN TO 3
1513 5 :ELSE:
1514 5 CALL XCOM TO PROMPT FOR MAXIMUM SIZE
1515 5 ERREXIT IF RETURN CODE IS NOT NORMAL TO :PRMERR:
1516 5 ERREXIT IF MAXIMUM SIZE IS NOT VALID TO :PRMERR:
1517 5 SET MAXIMUM SIZE INTO IDIM FIELD OF ENTRY
1518 3 :ENDIF
1519 2 :ENDIF
1520 1 EXIT XLSPS
1521 2 :PRMERR: CALL XRMSG TO DISPLAY ERROR MESSAGE
1522 2 GO TO (:PRMPT1,:PRMPT2,:PRMPT3), RETURN
1523 1 END XLSPS
### SYMBOL DEFINITION TABLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>1371</td>
</tr>
<tr>
<td>ADDPRM</td>
<td>939</td>
</tr>
<tr>
<td>CASE</td>
<td>1291</td>
</tr>
<tr>
<td>CMAT</td>
<td>1004</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>982</td>
</tr>
<tr>
<td>DEL</td>
<td>1308</td>
</tr>
<tr>
<td>DELPRM</td>
<td>922</td>
</tr>
<tr>
<td>ERR1</td>
<td>1429</td>
</tr>
<tr>
<td>ERR2</td>
<td>463</td>
</tr>
<tr>
<td>ERR3</td>
<td>199</td>
</tr>
<tr>
<td>ERR4</td>
<td>1285</td>
</tr>
<tr>
<td>ERR5</td>
<td>381</td>
</tr>
<tr>
<td>ERR6</td>
<td>466</td>
</tr>
<tr>
<td>ERR7</td>
<td>468</td>
</tr>
<tr>
<td>EXIT</td>
<td>99</td>
</tr>
<tr>
<td>EXPRM</td>
<td>1289</td>
</tr>
<tr>
<td>FILE</td>
<td>202</td>
</tr>
<tr>
<td>FILE1</td>
<td>384</td>
</tr>
<tr>
<td>FILE2</td>
<td>1261</td>
</tr>
<tr>
<td>FILE3</td>
<td>901</td>
</tr>
<tr>
<td>FILE4</td>
<td>658</td>
</tr>
<tr>
<td>FILE5</td>
<td>1432</td>
</tr>
<tr>
<td>FILE6</td>
<td>1169</td>
</tr>
<tr>
<td>FILE7</td>
<td>714</td>
</tr>
<tr>
<td>FORMAT</td>
<td>1270</td>
</tr>
<tr>
<td>LIBD</td>
<td>315</td>
</tr>
<tr>
<td>MOD</td>
<td>1297</td>
</tr>
<tr>
<td>MODRM</td>
<td>944</td>
</tr>
<tr>
<td>NDCHNG</td>
<td>1292</td>
</tr>
<tr>
<td>PRMERR</td>
<td>1521</td>
</tr>
<tr>
<td>PRMERR</td>
<td>898</td>
</tr>
<tr>
<td>PRMERR</td>
<td>310</td>
</tr>
<tr>
<td>PRMPT1</td>
<td>627</td>
</tr>
<tr>
<td>PRMPT1</td>
<td>328</td>
</tr>
<tr>
<td>PRMPT2</td>
<td>1476</td>
</tr>
<tr>
<td>PRMPT2</td>
<td>281</td>
</tr>
<tr>
<td>PRMPT3</td>
<td>1497</td>
</tr>
<tr>
<td>PRMPT3</td>
<td>1513</td>
</tr>
<tr>
<td>PROMPT</td>
<td>293</td>
</tr>
<tr>
<td>PROMPT</td>
<td>161</td>
</tr>
<tr>
<td>PROMPT</td>
<td>55</td>
</tr>
<tr>
<td>PROMPT</td>
<td>853</td>
</tr>
<tr>
<td>PROM2</td>
<td>907</td>
</tr>
<tr>
<td>PROM3</td>
<td>924</td>
</tr>
<tr>
<td>PROM4</td>
<td>944</td>
</tr>
<tr>
<td>PROM5</td>
<td>961</td>
</tr>
<tr>
<td>PROM6</td>
<td>984</td>
</tr>
<tr>
<td>RSPND</td>
<td>998</td>
</tr>
<tr>
<td>VERSION</td>
<td>905</td>
</tr>
<tr>
<td>XLCOB</td>
<td>410</td>
</tr>
<tr>
<td>XLDBF</td>
<td>95</td>
</tr>
<tr>
<td>XLDBF</td>
<td>537</td>
</tr>
<tr>
<td>XLDEL</td>
<td>84</td>
</tr>
<tr>
<td>XLDEL</td>
<td>626</td>
</tr>
<tr>
<td>XLIFL</td>
<td>699</td>
</tr>
<tr>
<td>XLINS</td>
<td>751</td>
</tr>
<tr>
<td>XLIST</td>
<td>788</td>
</tr>
<tr>
<td>XLMAN</td>
<td>32</td>
</tr>
</tbody>
</table>
101 DFS MANAGER. SCHEDULED BY DFS CONFIGURATION MANAGER. ONE PROGRAM
102 PER SIGNED ON USER.
103
104 **********
105 INPUT
106 *P2* DFS STATUS TABLE (SEE SDD 6.2.2)
107 *P2* DFS MANAGEMENT REQUESTS (SEE SDD 6.2.6.2)
108
110 OUTPUT
111 *P3* DFS MANAGER RESPONSES (SEE SDD 6.2.6.3)
112
114 NOTES
115 *P3* USMAG, XMAGA, XMPAI, XMISC, XMAGT, XUDMP
116
118 *P5* RMARG, SLIBG, SLIBR, ANES, CHUNM, XEEQ
119
121 *P5* XNWA IS A MANAGER GLOBAL DATA AREA CONTAINING THE AWA AND VARIOUS
122 CONTROL VALUES
123
126 1 BEGIN XMGR
127 2 *P1* INPUTS: *P1* LU
128 3 *P2* XVSTB ENTRY ADDRESS
129 4 *P3* USER ID
130 5 *P4* OPTIONS
131 6 *P5* NUMBER OF DWA TRACKS
132
135 CALL RM?AR
136 LOCK THE MANAGER INTO THE PARTITION
137
138 SET XVSTA( DFS STATUS TABLE ENTRY ADDRESS)
139
140 RESOLVE XVSTB ADDRESS
141
142 CALL XMAG INITIALIZE DWA FUNCTIONS
143
146 IF ERROR RETURN THEN
147 EXIT TO :XREMB
148
151 ENDF
152
153 GET EXEC'S CLASS NUMBER
154 GET PROCESSOR'S CLASS NUMBER
155 CALL SLIBR GET PRIVILEGED
156 SET EXEC'S CLASS NUMBER IN STBEC
157 SET PROCESSOR'S CLASS NUMBER IN STBPC
158
159 SET CURRENT TASK IN STBAM FROM EXEC ID STBEX
160 CALL SLIBX ENABLE
161 FORM EXEC'S NAME FROM 'XKEY' & ASCII LU(STBLA)
162
163 SCHEDULE EXEC WITH WAIT
164 CALL XMCSW
165
169 DO UNTIL EXEC REQUESTS TERMINATION OR ABENDS(P1=9 OR -32768)
170
173 1 IF REQUEST IS IN P1
174 2 IF REQUEST IS FOR AWA MANAGEMENT (P1=1) THEN
175 3 PERFORM XMAG
176 4 ELSE
177 5 IF REQUEST IS FOR SEQUENCE TABLE EXECUTION (P1=2) THEN
178 6 PERFORM 'XMAG'
179 7 ELSE
180 8 IF REQUEST NOT TERMINATION (P1=9 OR -32768) THEN
181 9 ISSUE MESSAGE '*** XMGR INVALID REQUEST (II) FROM XEXNM'
**REQUEST & NN IS LU**

*SET UP FOR FDS TERMINATION*

EXIT TO :XMEND

ENDIF

ENDIF

ENDDO

:XMEND

IF DUMP OPTION THEN

CALL XUDMP TO DUMP OUR PARTITION

ENDIF

RELEASE DWA TRACK SPACE

FREE EXEC'S & PROCESSOR'S CLASS NUMBERS

CALL RNRQ SET LOCK ON STATUS TABLE

UNLOCK THE MANAGER FROM THE PARTITION

CALL SLIBD DISABLE

CLEAR EXEC'S ID SEGMENT

CALCULATE EQT ADDRESS FOR ATTEMENT ID

CLEAR THE STATUS TABLE ENTRY

CALL SLIBD ENABLE

CALL RNRQ CLEAR THE LOCK ON THE STATUS TABLE

ISSUE MESSAGE '***XMNZ SIGN OFF FOR LU "NN"'

CALL SLIBD GET PRIVLEDGED AND DISABLED

CALL SI6ST TO MAKE US DORMANT

CLEAR OUR OWN ID

RESET EQT TO REMOVE ATTENTION ID

EXIT :SIEG TO THE RTE DISPATCHER

END **XMGR**
1  *D0  CALLING PROCEDURE
49  *D0  JSB  XMGST
54  *D0  ********
61  *D0  SEQUENCE TABLE EXECUTION FROM &SEQTB
69  *D0  ********
75  *D0  INPUT
82  *D0  XMNAW, XMPRM, XVSTA
90  "02  ********
99  *D0  OUTPUT
106  *D0  XMCP, XMPRM, XVSTA
115  "03  ********
123  "03  ********
131  "03  ROUTINES USED
139  "05  CMUND, EXEC, XMANG, XMKIL, XMPAW (XMSCH),
147  "05  XMSST, XMTP, $LIBR, $LIBR
1 BEGIN XMXGT
2 CALL XMSTF TO FIND BSEGTO TEC ENTRY
3 FIND ADDRESS OF BSEGTO
4 CALL XMSTO TO CONVERT ENDING SEQUENCE NUMBER INTO ENDING DISPLACEMENT
5 CALL XMSTO TO CONVERT STARTING SEQUENCE NUMBER INTO CURRENT DISPLACEMENT
6 COMPUTE CURRENT ENTRY ADDRESS
7 DO UNTIL THE LAST TABLE ENTRY IS EXECUTED OR
8 UNTIL THE TERMINATION ENTRY IS EXECUTED
9 IF PROCESSOR REQUIRES AN INTERFACE TABLE (WORD 3 BIT 8 IS SET) THEN
10 EXECUTE IF INTERFACE TABLE NOT SPECIFIED (WORD 4 = 0) (PARMS = 1)
11 CALL XMSTF TO SEARCH AWA FOR INTERFACE TABLE (CHAIN 4)
12 EXECUTE IF TABLE NOT FOUND (PARMS = 2)
13 IF TABLE NOT IN AWA THEN
14 CALL INDRT TO RETRIEVE FROM AWA
15 EXECUTE IF SPACE IS NOT FOUND
16 ENDF
17 EXECUTE IF INTERFACE TABLE NOT COMPLETE (WORD 3 BIT 8 CLEAR) (PARMS = 3)
18 EXECUTE IF PROCESSOR NAME FIELDS DIFFERENT (BYTES 3-7) (PARMS = 4)
19 EXECUTE IF VERSION FIELDS NOT EQUAL (WORD 3 BITS 9-15) (PARMS = 5)
20 ENDF
21 SEARCH SEGMENT TABLE FOR PROCESSOR TO BE SCHEDULED
22 EXIT IF:ERROR: IF NOT FOUND
23 CALL SLIBR TO BE PRIVILEGED
24 CALL CURRENT TASK TO THIS PROCESSOR
25 CALL SLIBO TO BE UNPRIVILEGED
26 IF PROCESSOR REQUIRES AN INTERFACE TABLE THEN
27 CALCULATE LENGTH OF INTERFACE TABLE HEADER AND SPECs
28 CALL EXEC TO CLASS I/O WRITE HEADER AND SPECs
29 ENDF
30 CALL EXEC TO SCHEDULE PROCESSOR WITH WAIT
31 CALL XASCH TO RETRIEVE PARAMETERS FROM PROCESSOR
32 CALL XMSTF TO FIND BSEGTO TEC ENTRY
33 FIND ADDRESS OF BSEGTO
34 COMPUTE CURRENT ENTRY ADDRESS
35 DO WHILE PROCESSOR REQUESTS AWA MANAGEMENT (PARM = 1)
36 CALL XAHG TO HONOR AWA REQUEST
37 ENDDO
38 CLEAR OUT CLASS BUFFERS FROM LAST PROCESSOR
39 EXIT IF:ERROR: IF PROCESSOR REQUESTED TERMINATION (PARM1 = 8)
40 EXIT IF:ERROR: IF PROCESSOR ABENDED (PARM1 = -32768)
41 IF REQUEST IS TO RESET CURRENT SEQUENCE ENTRY (PARM1 = 3) THEN
42 EXIT IF:ERROR: IF RESET NUMBER IS ZERO
43 CALL XMSTO TO CONVERT SEQUENCE NUMBER INTO CURRENT ENTRY — DISPLACEMENT
44 EXIT IF:ERROR: IF RESET SEQUENCE NUMBER IS NOT FOUR
45 IF TERMINAL ENTRY WAS JUST EXECUTED THEN
46 SET UP PARMS TO SHOW RESET SEQUENCE NUMBER
47 PERFORM :END: = **NO RETURN EXPECTED**
48 ENDF
49 ELSE
50 IF REQUEST IS NOT NORMAL COMPLETION (PARM1 = 0) THEN
51 DISPLAY ERROR MESSAGE — INVALID REQUEST
52 EXIT IF:ERROR: IF CURRENT AT IS DORMANT
53 CALL XMSTO TO SET PARAMETERS TO ABEND ASSOCIATED TASK
54 CALL XMPAN TO RESCHEDULE PROCESSOR
55 EXIT IF:ERROR: TO TERMINATE SEQUENCE
56 ENDF
57 ENDF
58 INCREMENT CURRENT DISPLACEMENT TO NEXT ENTRY
59 ENDDO
171  2  SET PARM1 = 0 (NORMAL COMPLETION)
172  2  PERFORM :END: - NO RETURN EXPECTED
173  2  :ERROR:  SET PARM1 = 1
174  2  :ERRORS:  SET PARM1 = 8 AND PARMS TO APPROPRIATE REASON CODE
175  2  :END:
176  2  CALL SLIBR TO BECOME PRIVLEDGED
177  2  SET CURRENT TASK IN PROG, STATUS TABLE, AND ANA TO EXEC
178  2  CALL SIBOX TO BECOME UN-PRIVLEDGED
179  2  CALL XMPAU TO POST EXEC AND WAIT FOR NEXT REQUEST
180  1  END XMKGT
CALLING PROCEDURE

JSB XMAFR
DEF "*3
DEF ADDR
DEF SIZE

PLACE A FE ON THE FE CHAINS AND MERGE WITH ANY ADJACENT FES

INPUT

ADDR - ADDRESS OF AREA BEING FREED
SIZE - SIZE OF AREA BEING FREED. IF LESS THAN 3 THE FOLLOWING
WORK(S) WILL ALSO BE FREED SUCH THAT THE MINIMUM FE SIZE
OF 3 WORDS IS MAINTAINED.

EXTERNAL SYMBOLS FROM XMAWA
XMBCP, XMFCP, XMFC, XMFR

OUTPUT (EXTERNAL SYMBOLS FROM XMAWA)
XMBCP, XMFCP, XMFC, XMFR

NOTES
USES .ENTR

**********
2131 BEGIN XMAFR
2141 SET NEW FE SIZE FIELD TO MAX(SIZE, 3)
2151 INCREMENT TOTAL FREE SPACE BY NEW FE SIZE
2161 INDEX TO FORWARD CHAIN POINTER (FCP) HEAD
2171 START SEARCH WHILE FCP NOT = END-OF-CHAIN (-32768)
2181 EXIT IF 'ADDR' < FCP VALUE
2191 SET NEW FE FCP TO CURRENT FCP VALUE
2201 SET CURRENT FCP VALUE TO 'ADDR'
2211 SET NEW FE BACKWARD CHAIN POINTER (BCP) TO NEXT FE BCP VALUE
2221 SET NEXT FE'S BCP VALUE TO 'ADDR'
2231 PERFORM MERGE TO ATTEMPT COMBINATION OF NEW FE AND NEXT FE
2241 OR ELSE
2251 INDEX TO NEXT FE FCP
2261 END LOOP
2271 SET NEW FE'S FCP VALUE TO CURRENT FE'S FCP VALUE (-32768)
2281 SET CURRENT FE'S FCP VALUE TO 'ADDR'
2291 SET NEW FE BCP TO BCP HEAD VALUE
2301 SET BCP HEAD TO 'ADDR'
2311 END SEARCH
2321 IF NEW FE BCP NOT = END-OF-CHAIN
2331 THEN
2341 PERFORM MERGE TO ATTEMPT COMBINATION OF PREVIOUS FE AND NEW FE
2351 END IF
2361 IF TOC SPACE FENCE IS WITHIN BOUNDARIES OF THE NEW
2371 (OR CONSOLIDATED) FE, THEN
2381 MOVE THE TOC SPACE FENCE TO BE ORIGIN OF THIS FE
2391 ENDF
2401 END XMAFR
2411 BEGIN MERGE
2421 IF FE 1 IS ADJACENT TO FE 2
2431 THEN
2441 INCREMENT FE 1 SIZE FIELD BY FE 2 SIZE FIELD
2451 SET FE 1 FCP TO VALUE OF FE 2 FCP
2461 IF FE 1 FCP NOT = END-OF-CHAIN
2471 THEN
2481 SET FE 3 BCP TO ADDRESS OF FE 1
2491 ELSE
2501 SET BCP HEAD TO ADDRESS OF FE 1
2511 ENDF
2521 ENDF
2531 END MERGE
255  1 400  CALLING PROCEDURE
256  1 400  JSB XNAGT
257  1 400  DEF +3
258  1 400  DEF OPTM
259  1 400  DEF SIZE
260  1 400  
261  1 400  ************
262  1 401  FIND A BLOCK OF FREE SPACE IN THE AREA AT LEAST 'SIZE' WORDS LARGE
263  1 401  
264  1 401  ************
265  1 402  INPUT
266  1 402  
267  1 402  QPM - INDICATOR OF WHICH FREE CHAIN TO SEARCH
268  1 402  0 = FORWARD POINTER CHAIN (FOR TDC SPACE)
269  1 402  1 = BACKWARD POINTER CHAIN (FOR DATA SPACE)
270  1 402  SIZE - NUMBER OF WORDS NEEDED (A MINIMUM OF 3 WORDS WILL BE
271  1 402  ALLOCATED EVEN IF 'SIZE' IS 1 OR 2)
272  1 402  
273  1 402  EXTERNAL SYMBOLS FROM XNAV
274  1 402  XMFCP, XMFNC, XMFR
275  1 402  
276  1 402  ************
277  1 403  OUTPUT
278  1 403  
279  1 403  A-REG - ADDRESS OF ALLOCATED BLOCK OR -32768 (OCTAL 100000)
280  1 403  
281  1 403  INDICATING NONE AVAILABLE
282  1 403  
283  1 403  EXTERNAL SYMBOLS FROM XNAV
284  1 403  XMFCP, XMFNC, XMFR
285  1 403  XMFK, XMFKK
286  1 403  
287  1 403  ************
288  1 405  NOTES
289  1 405  IF SOME FREE EXACTLY 'SIZE' WORDS OR >= SIZE+3 WORDS IS NOT FOUND AN
290  1 405  ERROR RETURN (A-REG = -32768) IS TAKEN
291  1 405  
292  1 405  USES .EMIR
293  1 405  
294  1 405  
295  1 405  
296  1 405  
297  1 405  XNAGT
298  1 405  XNAGT
299  1 405  XNAGT
300  1 405  XNAGT
301  1 405  XNAGT
302  1 405  XNAGT
303  1 405  XNAGT
304  1 405  XNAGT
305  1 405  XNAGT
306  1 405  XNAGT
1 BEGIN XMIRC
2 SET INDEX TO APPROPRIATE CHAIN HEAD, I.E., FEMC(frotch)
3 START SEARCH WHILE POINTER NOT = END-OF-CHAIN (-32768), AND
4 WHILE TOC SPACE FENCE HAS NOT BEEN CROSSED
5 EXIT IF FC SIZE = MAX('SIZE', 3)
6 DECREMENT TOTAL FREE SPACE BY MAX('SIZE',3)
7 DECHAIN FE
8 RETURN ADDRESS OF AREA
9 EXIT IF FC SIZE >= MAX('SIZE', 3) + 3
10 DECREMENT TOTAL FREE SPACE BY MAX(SIZE,3)
11 IF ALLOCATING FROM HEAD OF SPACE (OPTN = 0)
12 THEN
13 CREATE CHAIN POINTERS AND SIZE FIELDS IN BOTTOM OF SPACE
14 RECHAIN NEW FE
15 RETURN ADDRESS OF AREA
16 ELSE
17 CHANGE SIZE FIELD TO FC SIZE - MAX('SIZE',3)
18 COMPUTE AND RETURN ADDRESS OF AREA
19 ENDIF
20 OR ELSE
21 INDEX TO NEXT FE
22 END LOOP
23 SET RETURN CODE TO 0(NOT FOUND)
24 END SEARCH
25 IF TOC SPACE WAS FOUND AT THE TOC SPACE FENCE, THEN
26 INCREMENT TOC SPACE FENCE ON 'SIZE'
27 ENDF
END XMIRC
CALLING PROCEDURE

1 *00 PROVIDE AWA MANAGEMENT BASED ON REQUEST LIST (SEE FDS SDD TABLE 6.2 - III)
1 *01 PROVIDE AWA MANAGEMENT BASED ON REQUEST LIST (SEE TABLE 6.2 - III)
1 *02 CLSNO - CLASS I/O NUMBER CONTAINING REQUEST LIST
1 *02 REQUEST LIST (SEE TABLE 6.2 - III)
1 *02 ID SEGMENT PARAMETERS (SEE TABLE 6.2 - IV)
1 *03 REQUEST LIST FIELD 8
1 *05 NOTES
1 *05 USES EXEC,XMAPR,XMAG,XMAPK,XMIFN
1 *05 XWIN,XMDAL,XMDOA,XMDFT
1 *05 NEITHER XEXEC NOR THE PROCESSORS WILL MAKE AWA MANAGEMENT REQUESTS
1 *05 FOR DMA DATA (CLASS 3 & 5). ONLY THE MANAGER IS AWARE OF THE DMA
1 *05 AND IT WILL DUPLICATE INTERFACE TABLES AND SEQUENCE TABLES IN TO
1 *05 THE DMA AND COPY THEM BACK TO THE AWA AS NECESSARY.
380 1 BEGIN XMANG
381 2 RETRIEVE AWA MANAGEMENT REQUEST LIST FROM CLASS I/O NUMBER
382 3 CLEAR RETURN PARM1
383 4 CLEAR RETURN PARM2
384 5 CLEAR REWRITE FLAG
385 6 INITIALIZE TO FIRST REQUEST CODE
386 7 DO UNTIL END OF LIST (O) EIGHT REQUESTS PROCESSED OR PARM1 > ZERO
387 8 IF REQUEST FOR TFC (CODE 10)
388 9 THEN
389 10 IF REQUESTED SIZE .GT. TFC SIZE, THEN
390 11 CALL XMAG1 TO COLLAPSE AWA IF TOC TOO SCATTERED FOR XEXEC BUFFER SIZE
391 12 ENDIF
392 13 WRITE CHAIN HEADS, TOTAL FREE SPACE AND TOC TO CLASS I/O
393 14 STORE CLASS I/O NUMBER IN REQUEST WORD EIGHT
394 15 SET REQUESTED SIZE FROM TOC SIZE
395 16 SET REWRITE FLAG
396 17 ELSE
397 18 IF REQUEST TO CLEAR (CODE 17)
398 19 THEN
399 20 GET A(EDMA) FROM EDMA
400 21 SAVE HEADER AND DIRECTORY-SIZE
401 22 CLEAR EDMA THRU XMAG1
402 23 BUILD AN FE AT XMAG1 FOR AWA SIZE
403 24 CALL XMAG1 TO ALLOCATE A TOC ENTRY FOR EDMA
404 25 CHAIN IN TOC ENTRY TO EDMA
405 26 CALL XMAG1 TO ALLOCATE SPACE FOR EDMA
406 27 SET LOCATION, SIZE, & KEY IN THE TOC
407 28 SET DIRECTARY ADDRESS AT XMAG1
408 29 CLEAR THE DIRECTORY
409 30 SET LU, TRACK NUMBER, & NUMBER OF TRACKS IN THE DIRECTORY
410 31 ELSE
411 32 CALL XMAG1 TO SEARCH TOC FOR INDICATED ENTRY
412 33 CASE (:VERIFY, :VERALO, :VERALO, :RENAMF, :Delver, :Delver, :Store,)
413 34 (:RETRV, :RETRV) REQUEST CODE
414 35 :VERIFY:
415 36 IF ENTRY NOT FOUND
416 37 THEN
417 38 SET RETURN PARM1 AND PARM2 (2 & INDEX)
418 39 ENDIF
420 40 :VERALO:
421 41 IF ENTRY ALREADY EXISTS
422 42 THEN
423 43 IF ALLOCATE REQUEST (3)
424 44 THEN
425 45 SET RETURN PARM1 AND PARM2 (3 & INDEX)
426 46 ELSE
427 47 IF TYPE, SIZE AND I-DIM FIELDS DO NO MATCH
428 48 THEN
429 49 SET RETURN PARM1 AND PARM2 (4 & INDEX)
430 50 ENDIF
432 51 ELSE
433 52 CALL XMAG1 TO ALLOCATE TOC SPACE
434 53 IF CLASS EQ 3 OR 8, THEN
435 54 CHAIN IN NEW TOC ENTRY
436 55 SET DATA SPACE ADDRESS TO ZERO
437 56 ELSE
438 57 CALL XMAG1 TO ALLOCATE DATA SPACE
437 8 IF SPACE NOT AVAILABLE
438 8 THEN
439 9 SET RETURN PARM1 AND PARM2 (1 & INDEX)
440 9 ELSE
441 9 IF DATA ELEMENT (CLASS 2)
442 9 THEN
443 10 IF CHARACTER STRING (TYPE 4 - 8)
444 10 THEN
445 11 INITIALIZE AREA TO BLANKS
446 11 ELSE
447 11 INITIALIZE AREA TO ZEROS
448 11 ENDIF
449 9 ENDIF
450 9 CHAIN IN NEW TOC ENTRY
451 9 ENDIF
452 8 IF CLASS EQ 4 OR 6 AN SEQUENCE TABLE OR INTERFACE TABLE
453 8 THEN CALL XMDAL DMA ALLOCATION
454 8 ENDIF
455 7 ENDIF
456 6 ENDIF

:RENAME:
457 6 IF ENTRY NOT FOUND
458 6 THEN
459 7 SET RETURN PARM1 AND PARM2 (2 & INDEX)
460 7 ELSE
461 7 CALL XMTFN TO SEARCH TOC FOR NEW ENTRY AND DETERMINE CHAIN POSITION
462 7 IF ENTRY FOUND
463 7 THEN
464 8 SET RETURN PARM1 AND PARM2 (3 & INDEX)
465 8 ELSE
466 8 CALL XMAGT TO ALLOCATE NEW TOC ENTRY
467 8 IF SPACE NOT AVAILABLE
468 8 THEN
469 9 SET RETURN PARM1 AND PARM2 (1 & INDEX)
470 9 ELSE
471 9 COPY OLD ENTRY ATTRIBUTES INTO NEW ENTRY AND CHAIN IN TOC
472 9 DECHAIN OLD ENTRY
473 9 CALL XMARA TO RETURN OLD ENTRY TOC SPACE TO FE POOL
474 9 ENDIF
475 8 ENDIF
476 6 ENDIF

:DELIVER:
477 6 IF ENTRY FOUND
478 6 THEN
479 7 GENERATE KEY 1 LESS THAN FOUND KEY
480 7 CALL XMTFN FOR GENERATED KEY
481 7 DECHAIN TOC ENTRY
482 7 CALL XMARA TO RETURN TOC ENTRY SPACE TO FE POOL
483 7 CALL XMADA DMA DEALLOCATION
484 7 ENDIF
485 6 ELSE
486 6 IF DELETE REQUEST (5)
487 7 THEN
488 8 SET RETURN PARM1 AND PARM2 (2 & INDEX)
489 8 ENDIF
494 6 ENDF
495 6 :STORE:
496 6 IF ENTRY NOT FOUND
497 6 THEN
498 6 SET RETURN PARM1 AND PARM2 (2 & INDEX)
499 6 ELSE
500 7 IF (TOC TYPE > 0 AND INCONSISTENT WITH REQUEST TYPE) OR
501 7 DISPLACEMENT OR SPECIFIED SIZE < 0, OR
502 7 DISPLACEMENT + REQUESTED SIZE > ALLOCATED SIZE
503 7 THEN
504 7 SET RETURN PARM1 AND PARM2 (4 & INDEX)
505 7 ELSE
506 7 GET DATA FROM INDICATED CLASS I/O; STORE INTO AMA
507 7 FREE CLASS NUMBER
508 8 IF CLASS EQ 4 OR 6, THEN
509 9 CALL XMDRT DMA STORE DATA
510 8 ENDF
511 7 ENDF
512 6 ENDF
513 6 :RETRIEVE:
514 6 IF ENTRY NOT FOUND
515 6 THEN
516 6 SET RETURN PARM1 AND PARM2 (2 & INDEX)
517 6 ELSE
518 7 IF VALUES REQUESTED (8)
519 8 THEN
520 9 IF (TOC TYPE .NE. 0 AND .NE. REQUESTED-TYPE), OR
521 9 DISPLACEMENT OR SPECIFIED SIZE < 0, OR
522 9 DISPLACEMENT + SPECIFIED SIZE > ALLOCATED SIZE
523 9 THEN
524 9 SET RETURN PARM1 AND PARM2 (4 & INDEX)
525 8 ELSE
526 9 IF REQUESTED SIZE = ZERO
527 9 THEN
528 10 CALCULATE AMOUNT OF DATA TO RETRIEVE AS ACTUAL SIZE MINUS DISPLACEMENT
529 10 STORE COMPUTED SIZE IN REQUEST WORD SIX
530 9 ENDF
531 9 IF CLASS EQ 4 OR 6 AND TOC ADDRESS EQ 0, THEN
532 10 THE ELEMENT EXISTS ONLY ON THE DMA
533 10 CALL XMDRT MOVE INTO AMA
534 10 IF NO SPACE THEN
535 11 SET RETURN PARM1 AND PARM2 TO (1, INDEX)
536 10 EXIT TO :XMRX
537 10 ENDF
538 9 ENDF
539 9 WRITE VALUES TO CLASS I/O
540 9 STORE TYPE IN LOW BYTE OF REQUEST WORD 1
541 9 STORE CLASS NUMBER IN REQUEST WORD 8:CHT
542 9 SET NEWWRITE FLAG
543 8 ENDF
544 7 ELSE
545 8 WRITE TOC ENTRY TO CLASS I/O
546 8 STORE CLASS NUMBER IN REQUEST WORD 8:CHT
547 8 SET NEWWRITE FLAG
548 8 :XMRX
549 7 ENDF
CALLING PROCEDURE

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>564</td>
<td>*D0</td>
</tr>
<tr>
<td>565</td>
<td>*D0</td>
</tr>
<tr>
<td>566</td>
<td>*D0</td>
</tr>
<tr>
<td>567</td>
<td>*D0</td>
</tr>
<tr>
<td>568</td>
<td>*D0</td>
</tr>
<tr>
<td>569</td>
<td>*D0</td>
</tr>
<tr>
<td>570</td>
<td>---------</td>
</tr>
<tr>
<td>571</td>
<td>*D1</td>
</tr>
<tr>
<td>572</td>
<td>*D1</td>
</tr>
<tr>
<td>573</td>
<td>*D1</td>
</tr>
<tr>
<td>574</td>
<td>---------</td>
</tr>
<tr>
<td>575</td>
<td>*D2</td>
</tr>
<tr>
<td>576</td>
<td>*D2</td>
</tr>
<tr>
<td>577</td>
<td>*D2</td>
</tr>
<tr>
<td>578</td>
<td>*D2</td>
</tr>
<tr>
<td>579</td>
<td>*D2</td>
</tr>
<tr>
<td>580</td>
<td>---------</td>
</tr>
<tr>
<td>581</td>
<td>*D3</td>
</tr>
<tr>
<td>582</td>
<td>*D3</td>
</tr>
<tr>
<td>583</td>
<td>*D3</td>
</tr>
<tr>
<td>584</td>
<td>*D3</td>
</tr>
<tr>
<td>585</td>
<td>*D3</td>
</tr>
<tr>
<td>586</td>
<td>*D3</td>
</tr>
<tr>
<td>587</td>
<td>---------</td>
</tr>
<tr>
<td>588</td>
<td>*D5</td>
</tr>
<tr>
<td>589</td>
<td>*D5</td>
</tr>
<tr>
<td>590</td>
<td>*D5</td>
</tr>
<tr>
<td>591</td>
<td>---------</td>
</tr>
<tr>
<td>592</td>
<td>*</td>
</tr>
<tr>
<td>593</td>
<td>*</td>
</tr>
<tr>
<td>594</td>
<td>*</td>
</tr>
<tr>
<td>595</td>
<td>*</td>
</tr>
<tr>
<td>596</td>
<td>BEGIN</td>
</tr>
<tr>
<td>597</td>
<td></td>
</tr>
<tr>
<td>598</td>
<td>ISOLATE CLASS FROM KEY AND INDEX &quot;O A-PROPRIATE CHAIN HEAD</td>
</tr>
<tr>
<td>599</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>EXIT IF MATCH</td>
</tr>
<tr>
<td>601</td>
<td>RETURN ADDRESS OF ENTRY</td>
</tr>
<tr>
<td>602</td>
<td>EXIT IF 'KEYS' &lt; TOC ENTRY (EXPECTED ENTRY NOT IN CHAIN)</td>
</tr>
<tr>
<td>603</td>
<td>RETURN ADDRESS OF PREVIOUS ENTRY WITH INDIRECT BIT SET</td>
</tr>
<tr>
<td>604</td>
<td>OR ELSE</td>
</tr>
<tr>
<td>605</td>
<td>INDEX TO NEXT TOC ENTRY ON CHAIN</td>
</tr>
<tr>
<td>606</td>
<td>END LOOP</td>
</tr>
<tr>
<td>607</td>
<td>RETURN ADDRESS OF LAST (PREVIOUS) ENTRY WITH INDIRECT BIT SET</td>
</tr>
<tr>
<td>608</td>
<td>END SEARCH</td>
</tr>
<tr>
<td>609</td>
<td>END XMTFM</td>
</tr>
</tbody>
</table>
BEGIN XMPAY

*00 ENTRY: JSB XMPAY OR CALL XMPRM

*02 DEF RETURN ADDRESS

*03 INPUTS: XMPRM HAS REPLY

*02 OUTPUTS: XMPRM HAS REQUEST,

AND XVSTB IS UPDATED.

*03 DO UNTIL GOOD PARRS RECEIVED OR TOP AT TERMINATES

*03 IF ABORT CURRENT FLAG CM(- STBLU) THEN

*03 SET ABORT FLAG IN XMPRM

*03 TURN OFF ABORT CURRENT FLAG

ENDIF

*03 GET CURRENT TOP ASSOCIATED TASK(AT)

*03 SET IN PARM 1 FIELD OF MANAGER'S ID SEGMENT

*03 JSB XVPAW

*03 DEF *#3 RETURN

*03 DFC 0 MANAGER CALL

*03 DEF XMPRM PARM FIELD

*03 THIS IS AN IMPLIED WAIT

*03 :XMSCN GET CURRENT XVSTB ENTRY (XUSTA)

*03 IF THERE HAS BEEN A CALL TO PRTN (P1 FIELD IS NOT TOP AT) OR

*03 ID SEGMENT IS DORMANT OR

*03 ID SEGMENT IS NOT OUR SON THEN (TOP AT HAS TERMINATED)

*03 IF PARM IS NOT 0,3,8,9, OR -32768 THEN

*03 SET PARM TO -32768

*03 PRINT ERROR "INVALID REQUEST"

*03 ELSE

*03 SET PARM FIELD FROM MANAGER'S ID SEGMENT

*03 ENDIF

*03 ELSE (TOP AT IS STILL ACTIVE AND RETURNED VIP PAW)

*03 IF PARM NOT 1 OR 2 THEN

*03 CALL XMIL TO ABORT TOP AT

*03 PRINT ERROR "INVALID REQUEST"

*03 ELSE

*03 SET PARM FROM CURRENT ID SEGMENT

*03 ENDIF

ENDIF

ENDDO

1 END XMPAY
CALLING PROCEDURE
693 1 #00
694 1 #00 JSB XMDIN
695 1 #00
696 1 #00 FUNCTION
697 1 #01 INITIALIZE TOC
698 1 #01 & DMA FUNCTIONS
699 1 #01
700 1 #01 OUTPUT
701 1 #03 BREG O=COMPLETE
702 1 #03 MINUS= ERROR IN INITIALIZATION
703 1 #03
704 1 #03 NOTES
705 1 #05 USES EXEC DISC ALLOCATION,
706 1 #05 XMTFN, XMAGT
707 1 #05
708 1 #05 BEGIN XMDIN
709 2 GET NUMBER OF DMA TRACKS FROM P5
710 2 CALCULATE SIZE OF EDWA(3+6H) N IS # OF TRACKS
711 2 CALL XMTFN (EDWA)
712 2 CALL XMAGT (0,6) TOC ENTRY FOR EDWA
713 2 CALL XMAGT (1,SIZE) DATA AREA FOR EDWA
714 2 INITIALIZE & CHAIN EDWA TOC ENTRY
715 2 SET # OF TRACKS FOR EDWA
716 2 CLEAR EDWA
717 2 CALL EXEC (DISC TRACK ALLOCATION)
718 2 IF DISC ADDRESS .EQ. -1 TRACKS NOT AVAILABLE
719 2 THEN
720 3 ISSUE MESSAGE '***XMDIN "N" TRACKS NOT AVAILABLE'
721 3 SET ERROR RETURN
722 2 ELSE
723 2 SET DISC ADDRESS IN EDWA
724 2 SET ADDRESS OF EDWA FOR DMA MANAGEMENT
725 2 ENDF
726 2
727 1 END XMDIN
1 BEGIN XBDAL
2 * DMA ALLOCATION CALLING PROCEDURE
3 * JSB XBDAL
4
5 FUNCTION
6 * ALLOCATE DMA SPACE
7
8 INPUTS
9 A(TOC ENTRY OF THE ANA ELEMENT)
10 IN REG
11
12 OUTPUTS
13 UPDATE TOC ENTRY FOR ANA
14 RETURNS BREG= ZERO- ALLOCATION COMPLETE
15 = MINUS- ERROR CONDITION
16
17 NOT'S
18 CALLS XNRXX, XMBST
19
20 IF (A(DWA) .NE. 0) THEN
21 SET START-ADDRESS TO FIRST TRACK WORD IN DMA DIRECTOY
22 GET SIZE(IN WORDS FROM TOC ENTRY)
23 SIZE(IN SECTORS)=(SIZE+63/64)
24 DO FOR NUMBER OF TRACKS IN DMA OR DMA SIZE .GE. SIZE
25 SET BITNUM TO ZERO
26 DO UNTIL DMA SIZE .GE. SIZE OR BITNUM .GE. 96
27 CALL XNRXB (0,BITNUM,START-ADDRESS)
28 STARTBIT =BITNUM
29 CALL XNRXB (1,BITNUM,START-ADDRESS)
30 DMA-SIZE=BITNUM-STARTBIT
31 END-DO
32 IF BITNUM .GE. 96, THEN
33 START-ADDRESS=START ADDRESS+1 TRACK ADDRESS
34 ENDIF
35
36 IF (DWA) .NE. 0, THEN
37 * START-ADDRESS & BITNUM DEFINES THE TRACK & SECTOR
38 SET DISC ADDRESS & SIZE IN TOC ENTRY
39 CALL XPST (TOC-ENTRY)
40 SET NORMAL RETURN
41 ELSE
42
43 ISSUE MESSAGE "***XNOS NO DMA SPACE REMAINING"
44 SET ERROR RETURN
45 ENDIF
46
47 ENDIF
BEGIN XREFRT
+0 RETRIEVE CALLING PROCEDURE
+0 JSB XREFRT
+00 FUNCTION
+01 RETRIEVE DWA DATA INTO AWA
+0 T02 INPUTS
+0 T07 A(TOC ENTRY) IN REG
+0 OUTPUTS
+03 ADDRESS OF DATA IN THE TOC
+03 REG=0, RETRIEVE SUCCESSFUL
+03 MINUS, ERROR NO AWA DATA
+05 NOTES
+05 USES XMAST, XMDMA, EXEC(READ)
+05 IF NO DWA DIRECTORY, THEN
+05 SET ERROR CODE -5
+05 ELSE
+05 CALL XMAST, GET DATA SPACE
+05 IF NO SPACE, THEN
+05 SET ERROR CODE -1
+05 ELSE
+05 SET DATA ADDRESS IN TOC
+05 GET DISC ADDRESS
+05 READ DATA INTO AWA
+05 SET RETURN CODE TO 0
+02 ENDIF
+02 ENDIF
+0 END XREFRT
BEGIN XMST
CALLING PROCEDURE
*00 JSB XMST
*01 FUNCTION
*02 SET ALLOCATION & DEALLOCATION IN DMA DIRECTORY
*03 INPUTS
*04 AWA TOC ENTRY
*05 OUTPUTS
*06 UPDATES DMA DIRECTORY
*07 GET TRACK-ADDRESS FROM TOC
*08 SUBTRACT START OF TRACKS FROM DMA DIRECTORY FOR RELATIVE TRACK
*09 DIVIDE SECTOR ADDRESS BY 16(NUMBER OF BITS/WORD)
*10 QUOTIENT IS NUMBER OF RELATIVE WORDS
*11 REMAINDER IS BIT-POSITION(BP)
*12 WORD ADDRESS=DMA ADDRESS+RELATIVE TRACK+RELATIVE WORDS
*13 SUBTRACT BIT-POSITION FROM 16 GIVING NBFW(NUMBER BITS IN FIRST WORD)
*14 GET DATA SIZE, ADD 63, DIVIDE BY 64 GIVING NUMBER OF SECTORS(BITS)
*15 BITS=NBFW+REMAINING BITS(RB)
*16 IF RB .IE. 0, THEN
*17 SET NUMBER OF WORDS(NW) TO ZERO
*18 SET NUMBER OF BITS LAST WORD(NBLW) TO ZERO
*19 SET NBFW TO BITS
*20 ELSE
*21 DIVIDE RB BY 16
*22 SET NW TO QUOTIENT
*23 SET NBFW TO REMAINDER
*24 ENDIF
*25 L-WAD DATA POINTED TO BY WORD ADDRESS
*26 ROTATE LEFT (BP+NBFW-1 BITS) SAVE BP BITS & POSITION NBFW BITS
*27 EXCLUSIVE OR SIGN BIT (ON TO OFF; OFF TO ON)
*28 IF NBFW .GT. 1, THEN
*29 SHIFT RIGHT(ARITHMETIC BY NBFW-1 PROPAGATE BITS
*30 ENDIF
*31 IF DATA IS NEGATIVE, THEN
*32 SET FILL WORD TO -1
*33 ELSE
*34 SET FILL WORD TO 0
*35 ENDIF
*36 ROTATE RIGHT BY BP, RESET SAVED BITS
*37 STORE WHERE WORD-ADDRESS POINTS
*38 DO WHILE NW .GT. 0
*39 WORD-ADDRESS=WORD-ADDRESS+$
*40 SET FILL WORD INTO WHERE WORD-ADDRESS POINTS
*41 ENDWHILE
*42 IF NBFW .GT. 0, THEN
*43 W-WORD-ADDRESS=WORD-ADDRESS$
*44 LOAD DATA POINTED TO BY WORD-ADDRESS
*45 IF NBFW .GT. 1, THEN
*46 ROTATE LEFT NBFW-1 BITS
*47 ENDIF
*48 EXCLUSIVE OR SIGN BIT
*49 IF NBFW .GT. 1, THEN
*50 SHIFT RIGHT ARITHMETIC BY NBFW-1
*51 ENDIF
*52 STORE DATA WHERE WORD-ADDRESS POINTS
1 BEGIN XMAPK
2 *00 CALLING PROCEDURE
2 *00 JSB XMAPK
2 *00 DEF OPTION
2 *00 DEF PHASE 1 FLAG
2 *00 GENERAL COLLAPSE INTERFACE
2 *00 OUTPUT
2 *00 RETURNS AREA ADDRESS FOR AREA FOUND
2 *00 RETURNS -32768 AREA NOT FOUND
2 *00 NOTES
2 *00 USES XMPK1,XMPK2,XMPK3,XMSC
2 *05
2 IF PHASE1 FLAG NOT SET, THEN
2 CALL XMPK1 PURGE DWA ELEMENTS FROM AMA
2 PERFORM XMSC(OPTN,SIZE)
2 ELSE
2 SET RETURN CODE TO NOT FOUND
2 ENDIF
2 IF RETURN CODE IS NOT FOUND, THEN
2 IF OPTN=1 (BACKWARD CHAIN), THEN
2 CALL XMPK2 PACK TOC CHAIN
2 CALL XMSC(OPTN,SIZE)
2 ENDIF
2 IF RETURN CODE IS NOT FOUND, THEN
2 CALL XMPK3 PACK AMA DATA AREAS
2 PERFORM XMSC(OPTN,SIZE)
2 ENDIF
2 ENDIF
2 SET RETURN VALUE TO RETURN CODE
1 END XMAPK
BEGIN XMPK1

CALLING PROEDURE

JSB XMPK1

PHASE 1 OF COLLAPSE(PURGE DNA ELEMENTS)

OUTPUT

UPDATES ADDRESS FIELD IN TOC

FOR EACH DNA DATA ELEMENT

NOTES

USES XMAFR, XMAFD, XMD6, XMPK1

CURRENTLY ONLY CLASS 4 & 6 ELEMENTS

INTERFACE TABLES & SEQUENCE TABLES

ARE DNA ELEMENTS.

INCREMENT XMPK1 UPDATE PHASE 1 COUNT

IF DNA DIRECTORY ADDRESS NOT EQ 0, THEN

SAVE X & Y REGS

DO FOR ALL CLASS 4 ELEMENTS

CALL XMAFR(DATA, SIZE)

ENDBB

DO FOR ALL CLASS 4 ELEMENTS

CALL XMAFR(DATA, SIZE)

ENDBB

RESTORE X & Y REGS

ENDIF

END XMPK1
BEGIN XMPK2

CALLING PROCEDURE

JSB XMPK2

PHASE 2 OF COLLAPSE(COLLAPSE TOC ENTRIES)

OUTPUT

REORDERS TOC ENTRIES

UPDATES XMFNC( TOC FENCE ADDRESS)

NOTES

USES XMFNC,XMFCP,XMTFR,XMTCR,XMGR,XMPK2

INCREMENTS XMPK2 UPDATE PHASE 2 COUNT

DO WHILE XMFCNT.GT. XMFCP AND XMFCP.NE. -32768

ONLY IF THERE ARE FREE ELEMENTS AND THEY ARE IN THE TOC

WILL A TOC COMPRESS BE DONE.

GET FIRST-FREE(FREE) FROM XMFCP

GET OLDSZ FROM THE FREE ELEMENT

NWSIZ=OLDSZ-8

IF NWSIZ .LT. 0, THEN THE FREE ELEMENT WILL BE DEPLETED

GET NEXT-FREE FROM FIRST-FREE'S FCP

SET INTO XMFCP DELETE FROM THE FORWARD CHAIN

IF XMFCP.NE. -32768, THEN IF NOT THE LAST FREE ELEMENT

SET NEXT-FREE'S BCP TO -32768

ELSE

SET XMFCP TO -32768 DELETE FROM BACKWARD CHAIN

ENDIF

UPDATE LENGTH IN FREE ELEMENT

OLDSZ=NWSIZ

ENDIF

NWTOCR=(FIRST+8)+NWSIZ THE NEW TOC IS LAST 8 WORDS OF FIRST-FREE

OLTOC=XMFNC-8 OLD TOC IS THE ENTRY MOVE THE FENCE

TMPK=(OLTOC+1) CONSTRUCT A KEY TO FIND PREVIOUS ENTRY

CALL XMTFR(TMPK) FIND PREVIOUS

SET PROTOC FROM AREG

COPY TOC ENTRY FROM OLTOC TO NWTOCR

STORE NWTOCR ADDRESS INTO PROTOC'S CHAIN

CALL XMGR(OLTOC,TOC-SIZE) FREE THE OLD TOC ENTRY

END

END XMPK2
BEGIN XMPK3
  JSB XMPK3

FUNCTION
  COMPRESS ALL DATA AREA BY MIGRATING DATA AREAS TO THE HIGHER ADDRESSES AND FREE AREAS TO LOWER ADDRESSES

OUTPUT
  UPDATES XMBCP, XMFNC, XMFR, XMP3

NOTES
  USES XMFR

INCREMENT XMP3 (NUMBER OF PHASE 3)

IF XMBCP .NE. -32768, THEN THERE ARE FREE AREAS
  CALL XMPK2, INSURE TOC IS COMPRESSED
DO WHILE XMBCP .GT. XMFCN UNTIL 1 FREE AREA ADJACENT TO THE FENCE
  IF HCP OF LAST-FREE .EQ. -32768 ONLY 1 FREE AREA
  SET HIGH-WATER TO XMFCN
  ELSE
  MULTIPLE AREAS
  SET HIGH-WATER TO BCP OF LAST-FREE
  ENDF
EXIT IF TOC-ADDRESS .EQ. 0
  PERFORM XMNV(TOC-ADDRESS)
ENDO
ENDIF
END XMPK3
1064 1 BEGIN XMTSC  TOC SEARCH
1065 2 * FIND TOC ENTRY WHICH HAS DATA ADDRESS GREATER THAN
1066 2 * HIGH-WATER AND LESS THAN LOW-WATER, AND HAS A DATA
1067 2 * ADDRESS GREATER THAN ANY OTHER FOUND ON THIS SEARCH.
1068 2 * RETURN THE TOC-ADDRESS OR (NONE FOUND).
1069 2 * FIRST TOC ENTRY IS AT SYMBOL XMAWA, THE
1070 2 * LAST TOC-ENTRY IS AT XMFNC-8.
1071 2 TOC-ENTRY=XMAWA)
1072 2 TEST-AD=0; TEST-TOC=0
1073 2 DO UNTIL TOC-ENTRY .GE. XMFNC
1074 3 IF DATA ADDRESS IN TOC-ENTRY Z:
1075 4 ME 0, AND IS
1076 4 LT LOW-WATER, AND IS
1077 4 GT HIGH-WATER, AND IS
1078 4 GT TEST-AD,
1079 3 THEN
1080 4 TEST-AD=DATA ADDRESS
1081 4 TEST-TOC=TOC-ENTRY
1082 3 ENDF
1083 3 ADD 8 TO TOC-ENT
1084 2 ENDDO
1085 2 TOC-ADDRESS=TEST-TOC  RETURN 0 OR A TOC ADDRESS
1086 1 ENDO XMTSC
1088 1 BEGIN XMAV
1089 2 * MOVE THE DATA DEFINED BY THE TOC(WHICH IS
1090 2 * IMMEDIATELY ABOVE THE LAST FREE AREA) INTO
1091 2 * THE BOTTOM OF THE LAST FREE
1092 2 * UPDATE THE LENGTH OF THE RESULTING FREE AREA.
1093 2 GET DATA ADDRESS FROM THE TOC
1094 2 SAVE THE FIRST THREE WORDS OF THE DATA AREA
1095 2 CALL XMAFR(DATA ADDRESS,SIZE)
1096 2 GET FREE AREA FROM XBEP
1097 2 DECREASE FREE AREA LENGTH BY MAX(DATA SIZE,3)
1098 2 CALCULATE NEW ADDRESS FROM FREE AREA + FREE LENGTH
1099 2 MOVE DATA FROM DATA ADDRESS TO NEW ADDRESS
1100 2 MOVE SAVED FIRST THREE WORDS TO NEW ADDRESS
1101 2 UPDATE DATA ADDRESS IN TOC WITH NEW ADDRESS
1102 1 END XMAV
FORTRAN CALLING PROCEDURE

CALL XPAR (LU, INBUF, INTLG, MRBUF, IMUN, NAME, TYPE, SIZE, IDIN, DSPFT)

XPAR allows processors to obtain the attributes of the parameters referenced by the interface table.

**INPUT**
- **LU** - Logical unit number of user terminal
- **INBUF** - Input/output buffer of 7*(# parameters + 1) words, allocated within the calling program to hold the interface table header, first word must be zero only on first use to cause initialization.
- **INTLG** - Length of INTBF
- **MRBF** - Manager request buffer (64 words) used to communicate with the FDS manager. May be used as a scratch area by the processor except access processor service calls.
- **IMUN** - Relative number of parameter in interface table whose attributes are requested.

**OUTPUT**
- **NAME** - An alphanumeric name of up to six characters which identifies the data element or data where the data is to be obtained/stored. A zero entry indicates input data which is literal data stored within the interface table.
- **TYPE** - Data type code of the parameter
- **SIZE** - Total number of words of literal data
- **TOTAL NUMBER OF WORDS OF REFERENCED INPUT DRE**
- **TOTAL NUMBER OF WORDS OF REFERENCED OUTPUT DRE IF SUBSCRPTED OR ZERO IF NOT SUBSCRPTED**
- **TOTAL NUMBER OF BLOCKS OF REFERENCED INPUT DRE OR ZERO IF OUTPUT DRE**
- **IDIM** - Column length of a data element, maximum record size of an input dre or the length of a symbolic string. Zero if an unsubscribed output.
- **DSPT** - Displacement from the beginning of the data for subscribed data elements else zero.
- **FOR AN INPUT DRE THE RTE FILE MANAGER TYPE CODE IS RETURNED IN DSPFT.

EXTERNAL SYMBOLS
(SEE XPAR)

INTERNAL VARIABLES

**NOTES**
- USES ENTR, XPAR, XPAR, XPAR, XPAR, XPAR, XPAR, XPAR
- XPAR IS IMPLEMENTED AS A SINGLE MODULE CONTAINING THE ENTRY POINTS XPAR, XPAR AND XPAR
1 BEGIN XPATR
2 PERFORM XPMIC(XPGET) TO INITIALIZE GLOBALS AND INTERFACE TABLE
3 EXIT TO :XPMIC (XPGET) IF PARAMETER IS OUT OF RANGE
4 EXTRACT NAME FROM INTERFACE TABLE
5 SET DSPTT TO ZERO
6 IF LITERAL PARAMETER (NAME IS ZERO)
7 THEN
8 COPY TYPE, SIZE AND IDIM FROM INTERFACE TABLE
9 ELSE
10 IF SUBSCRIPTED (INTERFACE TABLE DISP OR $ FIELDS ARE NON-ZERO)
11 THEN
12 PERFORM XPMBC(XPGET) TO RETRIEVE TOC ENTRY AND COMPUTE DISPLACEMENT
13 STORE TYPE, SIZE, IDIM AND DSPTT
14 ELSE
15 IF DRE
16 THEN
17 PERFORM XPMFM(XPGET) TO QUALIFY FILE NAME
18 ENDIF
19 IF INPUT PARAMETER
20 THEN
21 CALL XPMRT TO RETRIEVE TOC ENTRY
22 COPY TYPE, SIZE, IDIM AND DSPTT FROM TOC ENTRY
23 ELSE
24 SET TYPE, SIZE AND IDIM TO ZERO
25 ENDIF
26 ENDIF
27 ENDIF
28 END XPATR
29
PROCESSING LOOP

M = VALUE OF \( ^{\cdot}p \), WHERE \( p \) IS THE NUMBER OF PARAMETERS IN THE INTERFACE TABLE

HNAMEF = ADDRESSES OF THE NAME FIELD (THIRD WORD) OF THE REQUEST

ARRAY RESTR

VARY = NEGATIVE OF THE NUMBER OF IN/OUT ARRAYS IN CALLING SEQUENCE

REQUEST = EIGHT WORD ARRAY USED FOR CONSTRUCTING AWA MANAGEMENT REQUESTS FOR XPRED

USRID = USER FILE IDENTIFIER CHARACTER (\( \ast \)IPARM(3) )

EXTERNAL VARIABLES (SEE XPRED)

XPCLS

XPLU

XPRID

**********

NOTES

**********

USES .ENTRY, EXEC, XPRED, XPIT, XVSTO

XPGET MUST BE INCLUDED IN PROCESSOR AT FDS BUILD TIME.

SPM INTBUF MUST BE USED BY XPGET, XPPUT, AND XPTRT AND NCEOS TO BE INITIALIZED ONLY ONCE BY ANY OF THE TIME: ROUTINES:

SINCE REQUESTS FOR INPUT DATA FROM THE AWA MAY BE PAID FOR UP TO EIGHT AT A TIME, XPGET RUNS MOST EFFICIENTLY WHEN PARAMETERS ARE REQUESTED IN MULTIPLES OF EIGHT.
FORTRAN CALLING PROCEDURE

CALL XPUT (LU, INBUF, INTLG, NBUF, N, IMUXS, OUT(1), ..., OUT(N))

185 1 100
186 1 100
187 1 100
188 1 100
189 1 100
190 1 100
191 1 100
192 1 100
193 1 100
194 1 100
195 1 100
196 1 100
197 1 100
198 1 100
199 1 100
200 1 100
201 1 100
202 1 100
203 1 100
204 1 100
205 1 100
206 1 100
207 1 100
208 1 100
209 1 100
210 1 100
211 1 100
212 1 100
213 1 100
214 1 100
215 1 100
216 1 100
217 1 100
218 1 100
219 1 100
220 1 100
221 1 100
222 1 100
223 1 100
224 1 100
225 1 100
226 1 100
227 1 100
228 1 100
229 1 100
230 1 100
231 1 100
232 1 100
233 1 100
234 1 100
235 1 100
236 1 100
237 1 100
238 1 100
239 1 100
240 1 100
241 1 100
243 1 BEGIN XPGET
244 2 SET FOR 'GET'
245 2 PERFORM ACCESS TO RETRIEVE DATA
246 1 END XPGET
247 1 BEGIN XPPUT
248 2 SET FOR 'PUT'
249 2 PERFORM ACCESS TO STORE DATA
250 1 END XPPUT
251 1 BEGIN ACCESS
252 2 PERFORM XPINI TO INITIALIZE GLOBALS AND INTERFACE TABLE
253 2 DO FOR EACH PARAMETER REQUESTED
254 3 IF SELECTED PARAMETER IS OUT OF RANGE
255 4 CALL XPREF TO PURGE QUEUED REQUESTS
256 5 EXIT TO :XPEI3:
257 5 ENDIF
258 5 IF INPUT/OUTPUT TYPE DOES NOT MATCH 'GET'/'PUT' PROCESSING
259 6 THEN
260 7 CALL XPREF TO PURGE QUEUED REQUESTS
261 8 EXIT TO :ERRI2:
262 8 ENDIF
263 8 IF OVER-RUNNING CALLING SEQUENCE
264 9 THEN
265 10 CALL XPREF TO PURGE QUEUED REQUESTS
266 11 EXIT TO :XPEI3:
267 11 ENDIF
268 11 IF PROCESSING FOR 'GET'
269 12 THEN
270 13 IF PARAMETER IS MEMORY RESIDENT DE (CLASS 2)
271 14 THEN
272 15 IF INPUT IN LITERAL FORM
273 16 THEN
274 17 BUILD REQUEST WITH INTERFACE TABLE NAME AND DISPLACEMENT
275 18 ELSE
276 19 IF NORMAL XPGET/PUT PROCESSING (XPGPF = -1)
277 20 THEN
278 21 IF SUBSCRIPTED (DISPLACEMENT > 0 OR DOUBLE SUBSCRIPT FLAG SET)
279 22 THEN
280 23 PERFORM XPSBC TO COMPUTE DISPLACEMENT = F(DIM, SUBS, TYPE)
281 24 ELSE
282 25 DISPLACEMENT IS ZERO
283 26 ENDIF
284 26 ELSE
285 27 USE DISPLACEMENT FROM INTERFACE TABLE ENTRY
286 28 ENDIF
287 28 BUILD REQUEST WITH DE NAME AND DISPLACEMENT
288 29 ENDIF
289 30 CALL XPREF TO QUEUE RETRIEVAL AND STORAGE OF INPUT
290 31 ELSE PARAMETER IS DDE (CLASS 3)
291 32 THEN
292 33 BUILD REQUEST FOR DDE ENTRY
293 34 CALL XPREF TO IMMEDIATELY RETRIEVE DDE ENTRY
294 35 PERFORM XPSFM TO CONSTRUCT AND STORE QUALIFIED FILE NAME
295 36 "STORE FILE ATTRIBUTES"
296 37 ENDIF
297 37 ELSE PROCESSING FOR 'PUT'
298 38 THEN
299 39 IF PARAMETER IS MEMORY RESIDENT DE (CLASS 2)
300 40 THEN
301 41 IF NORMAL XPGET/PUT PROCESSING (XPGPF = -1)
302 42 THEN
IF SUBSCRIPTED
THEN
PERFORM XPSBC TO COMPUTE DISPLACEMENT = F(DIM, SUBS, TYPE)
ELSE
DISPLACEMENT IS ZERO
BUILD REQUEST TO DELETE ANY EXISTING ORDE WITH THIS NAME
CALL XPREG TO QUEUE DELETION
BUILD REQUEST TO REALLOCATE ORDE
CALL XPREG TO QUEUE ALLOCATION
ENDIF
ELSE
USE DISPLACEMENT FROM INTERFACE TABLE ENTRY
ENDIF
BUILD REQUEST TO OUTPUT DATA TO CLASS I/O AND STORE DATA IN AHA
CALL XPREG TO QUEUE STORAGE OF DATA
ELSE PARAMETER IS ORDE (CLASS 3)
BUILD REQUEST TO DELETE ANY EXISTING ORDE WITH THIS NAME
CALL XPREG TO QUEUE DELETION
BUILD REQ: ST TO REALLOCATE ORDE WITH NEW ATTRIBUTES
EXIT TO :ERR13: IF FILE TYPE NOT 1-13, # BLOCKS < 1 OR MAX REC SIZE NOT 1-1200
CALL XPREG TO QUEUE REALLOCATION OF ORDE
ENDIF
ENDD}
CALL XPREG TO COMPLETE QUEUED REQUESTS
EXIT ACCESS
:ERR12: TERMINATE PROCESSOR FOR INPUT/OUTPUT TYPE INCONSISTENCY
:ERR13: TERMINATE PROCESSOR FOR INVALID PARAMETER REQUEST
:ERR14: CALL XPREG TO PURGE QUEUED REQUESTS
TERMINATE PROCESSOR FOR INVALID ORDE FILE TYPE, BLOCK COUNT OR MAX RECORD SIZE
BEGIN XPINI
  INITIALIZE GLOBAL VALUES FROM LU AND XVSDB
  IF INTERFACE TABLE BUFFER NOT INITIALIZED THEN
    RETRIEVE INTERFACE TABLE FROM MANAGER CLASS I/O NUMBER
    IF RETRIEVAL NOT SUCCESSFUL THEN
      TERMINATE PROCESSOR WITH 'XP10 PROCESSOR INITIALIZATION ERROR'
      ENDF
  EXIT TO XPE15: IF N < 0
END XPINI

BEGIN XPGEN
  SET / IN FIRST CHARACTER POSITION
  MOVE FOUR WORD NAME INTO MIDDLE POSITIONS
  LOCATE FIRST BLANK CHARACTER
  REPLACE BLANK WITH USER ID
END XPGEN

BEGIN XPSBC
  IF DOUBLE SUBSCRIPTED THEN
    BUILD REQUEST FOR SUBSCRIPTS STORED IN LITERAL BLOCK
    CALL XPEN TO QUEUE RETURN OF SUBSCRIPTS
  ELSE
    SINGLE SUBSCRIPT IS CONTAINED IN DISPLACEMENT FIELD & J-SUBSCRIPT IS = 1
  ENDIF
  BUILD REQUEST TO RETURN TOC ENTRY
  CALL XPER TO IMMEDIATELY RETRIEVE TOC ENTRY
  IF DOUBLE DIMENSIONED (IDIM > 0) THEN
    DISPLACEMENT = (IDIM*(JSUB-1) + JSUB-1) * # WORDS PER ELEMENT
  ELSE
    DISPLACEMENT = SIZE*(JSUB-1) + (JSUB-1) * # WORDS PER ELEMENT
  ENDF
END XPSBC

END ACCESS
FORTRAN CALLING PROCEDURE

CALL XPGTI (LU, INBUF, INTLNG, MBUFF, INUM, IN, SIZE, DISP)

XPGTI ALLOWS THE CALLING PROGRAM TO RETRIEVE A SPECIFIED AMOUNT
OF DATA FROM AN INPUT PARAMETER BEGINNING AT ANY POINT IN THE
PARAMETER ARRAY.

INPUT!
LU = LOGICAL UNIT NUMBER OF USER TERMINAL
INBUF = INPUT/OUTPUT BUFFER OF 7# PARAMETERS + 1) WORDS,
ALLOCATED WITHIN THE CALLING PROGRAM TO HOLD THE
INTERFACE TABLE HEADER. FIRST WORD MUST BE ZERO ONLY ON
FIRST USE TO CAUSE INITIALIZATION
INTLNG = LENGTH OF INBUF
MBUFF = MANAGER REQUEST BUFFER (54 WORDS) USED TO COMMUNICATE
WITH THE DFS MANAGER. MAY BE USED AS A SCRATCH AREA BY
THE PROCESSOR EXCEPT ACROSS PROCESSOR SERVICE CALLS
INUM = RELATIVE NUMBER OF PARAMETER IN INTERFACE TABLE FROM
WHICH TO RETURN DATA
SIZE = TOTAL NUMBER OF WORDS TO BE RETURNED
DISP = DISPLACEMENT FROM THE BEGINNING OF THE INPUT PARAMETER
AT WHICH TO BEGIN DATA RETRIEVAL. A VALUE OF ZERO
INDICATES THE BEGINNING OF THE AREA SPECIFIED BY THE USER
IN THE INTERFACE TABLE. NEGATIVE VALUES OF DISP MUST NOT
BE SPECIFIED, I.E., RETRIEVAL FROM AN AREA PREVIOUS TO
THE USERS SUBSCRIPTS IS NOT SUPPORTED

OUTPUT
IN = ARRAY OF AT LEAST SIZE WORDS INTO WHICH INPUT DATA IS TO
BE STORED

INTERNAL
BSDSP = DISPLACEMENT FROM THE REAL ORIGIN OF THE DATA ELEMENT AS
A RESULT OF USER SUBSCRIBING... DISP IS ADDED TO THIS
VALUE IN THE REQUEST TO THE MANAGER
GTPT = DISPLACEMENT INTO GTPT DEPENDING ON WHETHER GET (O) OR
PUT (I) PROCESSING IS INVOLVED
JGTPT = TWO WORD VECTOR CONTAINING 'JSB XPGET' OR 'JSB XPFUT'
INSTRUCTIONS USED TO DYNAMICALLY PRODUCE XPGET AND
XPFUT CALLS

NOTES
USES EXEC, .ENTR, XPATR, XPE13(XPATR), XPGET(XPATR), XPFUT(XPATR),
FORTRAN CALLING PROCEDURE

 CALL XPPTI (LU, INBUF, INTLG, MRBUF, INUM, OUT, SIZE, DISP)

 XPPTI ALLOWS THE CALLING PROGRAM TO STORE A SPECIFIED AMOUNT OF
 DATA INTO AN OUTPUT PARAMETER BEGINNING AT ANY POINT IN THE
 PARAMETER ARRAY

 INPUT

 LU - LOGICAL UNIT NUMBER OF USER TERMINAL

 INBUF - INPUT/OUTPUT BUFFER OF $\#$(# PARAMETERS + 1) WORDS.
 ALLOCATED WITHIN THE CALLING PROGRAM TO HOLD THE INTERFACE
 TABLE HEADER. FIRST WORD MUST BE ZERO ONLY ON INITIALIZATION

 INTLG - LENGTH OF INBUF

 MRBUF - MANAGER REQUEST BUFFER (64 WORDS) USED TO COMMUNICATE
 WITH THE FDS MANAGER. MAY BE USED AS A SCRATCH AREA BY
 THE PROCESSOR EXCEPT ACROSS PROCESSOR SERVICE CALLS

 INUM - RELATIVE NUMBER OF PARAMETER IN INTERFACE TABLE INTO
 WHICH TO STORE DATA

 OUT - ARRAY OF AT LEAST SIZE WORDS FROM WHICH OUTPUT DATA IS TO
 BE TAKEN

 SIZE - TOTAL NUMBER OF WORDS TO BE STORED

 DISP - DISPLACEMENT FROM THE BEGINNING OF THE OUTPUT PARAMETER
 AT WHICH TO BEGIN DATA STORAGE. A VALUE OF ZERO
 INDICATES THE BEGINNING OF THE AREA SPECIFIED BY THE USER
 IN THE INTERFACE TABLE. NEGATIVE VALUES OF DISP MUST NOT
 BE SPECIFIED, I.E., STORAGE INTO AN AREA PREVIOUS TO THE
 USERS SUBSCRIPTS IS NOT SUPPORTED.

 OUTPUT

 NONE

 INTERNAL

 SEE XPSTI

 NOTES

 XPPTI IS AN ENTRY POINT INTO XPSTI
1 BEGIN XPGT1
2 SET FOR GET PROCESSING
3 PERFORM XPXI TO RETRIEVE DATA
4 END XPGT1
5 BEGIN XPPTI
6 SET FOR PUT PROCESSING
7 PERFORM XPXI TO STORE DATA
8 END XPPTI
9 BEGIN XPGT1
10 SET XPSPF(XPGT1) TO BY-PASS NORMAL XGGET/XPPUT SUBSCRIPT PROCESSING
11 CALL XPATR TO ASSURE INITIALIZATION OF INBUF AND RETURN BASE DISPLACEMENT
12 EXIT TO :XPE13: (XPATR) IF INDICATED PARAMETER IS A D roc FILE
13 SAVE INTERFACE TABLE ENTRY CLASS/TYPb WORD, FLAG/DISP WORD AND SIZE WORD
14 IF LITERAL (NAME = 0)
15 THEN
16 EXIT TO :ERR15: IF DISP + SIZE > SAVED SIZE
17 ELSE
18 INCREMENT BASE DISPLACEMENT TO CONVERT TO SUBSCRIPT
19 SET TYPE FIELD TO FREE
20 ENDIF
21 CLEAR ENTRY SUBSCRIPT BIT
22 SET DISP FIELD TO SUM OF BASE DISPLACEMENT AND DISP
23 SET SIZE FIELD TO SIZE
24 CALL XPGET/XPPUT TO TRANSFER DATA
25 RESTORE ORIGINAL INTERFACE TABLE ENTRY
26 RESTORE XFGPF(XPGT1) TO NOMINAL VALUE
27 EXIT XPXI
28 :ERR15: TERMINATE PROCESSOR FOR ATTEMPT TO RETRIEVE TOO MUCH DATA
29 END XPXI
1 BEGIN XPREQ
2 MOVE REQUEST INTO BUFFER
3 IF OPTION IS TO TRANSFER DATA
4 THEN
5 IF REQUEST IS TO RETRIEVE DATA (8)
6 THEN
7 STORE ADDRESS IN TABLE
8 ELSE SHOULD BE A REQUEST TO STORE DATA (7)
9 OUTPUT DATA TO CLASS I/O
10 STORE CLASS NUMBER IN REQUEST WORD 8
11 ENDIF
12 ENDIF
13 INCREMENT POINTER
14 IF BUFFER FULL OR OPTION IS TO CLOSE NON-EMPTY BUFFER
15 THEN
16 CLOSE BUFFER
17 TRANSMIT BUFFER TO MANAGER
18 PAM MANAGER WITH REQUEST FOR AWA MANAGEMENT
19 RETRIEVE RETURN PARAMETERS
20 IF REWRITE FLAG SET (PARMS)
21 THEN
22 RETRIEVE REQUEST BUFFER
23 ENDIF
24 IF REQUESTS WERE SUCCESSFUL
25 THEN
26 DO FOR EACH REQUEST IN BUFFER
27 IF REQUEST TO RETURN DATA (8, 9 OR 16)
28 THEN
29 RETRIEVE AND STORE DATA IN ADDRESS CONTAINED IN TABLE
30 ENDIF
31 ENDDO
32 CLEAR POINTER AND LOCAL CLASS NUMBER
33 ELSE
34 OUTPUT FAILURE MESSAGE (XP11)
35 DO FOR EACH REQUEST IN BUFFER
36 IF REQUEST SUCCESSFUL FOR DATA RETRIEVAL OR UNSUCCESSFUL STORE
37 THEN
38 FREE CLASS I/O NUMBER AND SAM BUFFER
39 ENDIF
40 ENDDO
41 EXIT PROCESSOR WITH REQUEST FOR SEQUENCE TERMINATION
42 ENDIF
43 ENDIF
44 END XPREQ
FORTRAN CALLING PROCEDURE FOR PROCESSOR TC SPECIFIC TYPE

CALL XPDRS (LU,PROBLEM,PROMPT,TYPE,DATLEN,IDIM,DATA,RETS)

CALLS XPDRS, DECODES COMMUNICATIONS BUFFER LOOKING FOR A SPECIFIC
"TYPE" AND STORES IT IN "DATA" FOR "DATLEN" LOGICAL ELEMENTS

INPUTS FROM CALLING SEQUENCE:

LU - (INTEGER, 1 WORD) LOGICAL UNIT OF USER'S TERMINAL

PROBLEM - (INTEGER, 1 WORD) LENGTH IN WORDS OF THE CHARACTER

PROMPT - (INTEGER, PROBLEM WORDS) IS THE CHARACTER STRING

USED AS THE USER PROMPT

TYPE - (INTEGER, 1 WORD) CODE FOR THE DATA TYPE

EXPECTED AS THE USER'S RESPONSE AS FOLLOWS:

0 - FREE

1 - INTEGER

2 - REAL

3 - DOUBLE PRECISION

4 - CHARACTER STRING LENGTH 2

5 - CHARACTER STRING LENGTH 4

6 - CHARACTER STRING LENGTH 8

7 - CHARACTER STRING LENGTH 16

8 - CHARACTER STRING LENGTH 32

9 - CHARACTER STRING LENGTH 64

10 - CHARACTER STRING LENGTH 128

DATLEN - (INTEGER, 1 WORD) NUMBER OF LOGICAL ELEMENTS IN

THE DATA AREA. IF TYPE IS FREE, DATLEN IS THE

NUMBER OF WORDS.

IDIM - (INTEGER, 1 WORD) THE COLUMN LENGTH OF THE DATA

AREA. IF IT IS A 2 DIMENSION ARRAY, ELSE 1 OR 0

IF IT IS A VECTOR.

OUTPUTS FROM CALLING SEQUENCE:

DATA - (INTEGER, DIM DEPENDS ON DATLEN AND TYPE)

AREA TO CONTAIN THE USERS RESPONSE

RETS - (INTEGER, 1 WORD) RETURN CODE PASSED BACK TO

CALLER;

0 - NORMAL RETURN, DATA AREA CONTAINS USER'S RESPONSE

1 - USER ENTERED A COM. THERE IS NO RESPONSE.

2 - USER ENTERED A CR. THERE IS NO RESPONSE.

3 - USER PROMPT WAS TOO LONG. MAXIMUM LENGTH IS

34 CHARACTERS OR 17 WORDS.

4 - PARAMETER LIST IS INVALID. EITHER:

A. IDIM LESS THAN ZERO

B. DATLEN LESS THAN ONE

C.TYPE NOT SPECIFIED.

INTERNAL VARIABLES
667  1  CD4
668  1  CD4  DATPTR - INDEX IN WORDS INTO DATA ARRAY WHERE NEXT
669  1  CD4  ELEMENT IS TO BE STORED
670  1  CD4  EFFTYP - THE EFFECTIVE INDEX (TYPE+1) OF THE CHARACTER
671  1  CD4  STRING RESPONDER. IF TYPE IS FREE, IT IS THE
672  1  CD4  CLOSEST SUPPORTED LENGTH, ELSE IT IS THE LENGTH
673  1  CD4  SPECIFIED BY TYPE.
674  1  CD4  INDX - CONTAINS THE INDEX INTO MSGS ARRAY OF THE CURRENT
675  1  CD4  ERROR MESSAGE.
676  1  CD4  LEN - LENGTH OF CURRENT ERROR MESSAGE.
677  1  CD4  LENGTH - 9 WORD ARRAY REPRESENTING THE LENGTH IN WORDS
678  1  CD4  OF THE 9 DATA TYPES 0-8 RESPECTIVELY.
679  1  CD4  STKIND - COUNT OF NUMBER OF NESTED REPEATS AND INDEX
680  1  CD4  TO THE CURRENT TOIP OF THE STACK.
681  1  CD4  STKPRM - 4 WORD ARRAY FOR STACK OF PARENTHESIS FLAGS
682  1  CD4  STKREP - 4 WORD ARRAY FOR STACK OF REPEAT COUNTS
683  1  CD4  STKTOK - 4 WORD ARRAY FOR STACK OF TOKEN POINTERS
684  1  CD4  WHERE EACH REPEAT GROUP BEGINS
685  1  CD4  TOKPTR - POINTER TO CURRENT TOKEN BEING PROCESSED
686  1  CD4  TYPE1 - INDEX 1-9 INTO LENGTH ARRAY BASED ON 0-8 TYPE
687  1  CD4  TTYPE1 = TYPE +1
688  1  CD4
689  1  CD4  **********
690  1  CD5
691  1  C:5  SUBROUTINES AND FUNCTIONS CALLED:
692  1  D5  EXEC, XPRMS, ERRNO
693  1  CD5
694  1  CD5
695  1  CD5  POL Routines Included:
696  1  CD5  XPRMS, STRING, SUBSCR, REPET, ERRMSG
697  1  CD5
698  1  CD5
699  1  CD5  **********
BEGIN STRING
  IF TYPE IS NOT CHARACTER OR FREE PERFORM ERMSG
  DETERMINE EFFECTIVE LENGTH OF RESPONSE AS NEXT LARGER SUPPORTED LENGTH
  ERMSIF TYPE OF RESPONSE > TYPE REQUESTED AND
  IF TYPE IS NOT FREE THEN
  SET EFFECTIVE LENGTH = LENGTH REQUESTED
END IF
SET MESSAGE NUMBER TO XPO7
ERMSIF THERE IS NO ROOM IN DATA AREA FOR THIS ELEMENT PERFORM ERMSG
CALL XMKP TO MOVE BLANKS INTO DATA AREA FOR EFFECTIVE LENGTH
CALL XMKP TO MOVE CHARACTER STRING INTO DATA AREA FOR REAL LENGTH
SET PREVIOUS TOKEN IS DATA
INCREMENT POINTER IN DATA AREA
INCREMENT TO NEXT TOKEN
END STRING

1 BEGIN SUBSCR
INCREASE POINTER TO NEXT TOKEN
SET MESSAGE NUMBER TO XPO7
ERMSIF TOKEN IS NOT AN INTEGER TO PERFORM ERMSG
IF I-DIMENSION > 1 THEN
SET 1 TO INTEGER VALUE
INCREMENT POINTER TO NEXT TOKEN
ERMSIF TOKEN IS NOT AN INTEGER OR
ERMSIF NEXT TOKEN IS NOT A CLOSE PAREN TO PERFORM ERMSG
SET MESSAGE NUMBER TO XPO7
ERMSIF INTEGER > I-DIMENSION TO PERFORM ERMSG
ERMSIF SUBSCRIPT IS OUT OF RANGE TO PERFORM ERMSG
ELSE
ERMSIF NEXT TOKEN IS NOT A CLOSE PAREN TO PERFORM ERMSG
SET MESSAGE NUMBER TO XPO7
ERMSIF SUBSCRIPT IS OUT OF RANGE
END IF
ADJUST INDEX INTO DATA AREA ACCORDING TO SUBSCRIPT
INCREMENT POINTER BY 3 TOKENS
SET PREVIOUS TOKEN = SUBSCRIPT
SET MESSAGE NUMBER TO XPO7
ERMSIF TOKEN IS AN EBS OR
ERMSIF TOKEN IS A REPEAT OR
ERMSIF TOKEN IS A CLOSE PAREN OR
ERMSIF TOKEN IS A SUBSCRIPT TO PERFORM ERMSG
END SUBSCR
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 - NEGATIVE POWER</td>
</tr>
<tr>
<td>2</td>
<td>0 - SINGLE PRECISION</td>
</tr>
<tr>
<td>3</td>
<td>1 - DOUBLE PRECISION</td>
</tr>
<tr>
<td>4</td>
<td>FLGTYP - TYPE OF REAL NUMBER</td>
</tr>
</tbody>
</table>
| 5 | SUBROUTINES AND FUNCTIONS USED: EXEC, KCVT, XRMOV, XRPCK, XRUPK, OV
| 6 | PDL ROUTINES USED: XROM, TOKENS, QUOTE, DIGIT, DCOL, DECPT, EORD |
| 7 | INTEGR, REAL, DBL, INVAL, COMPD |

SUBROUTINES AND FUNCTIONS USED:
- EXEC
- KCVT
- XRMOV
- XRPCK
- XRUPK
- OV

PDL ROUTINES USED:
- XROM
- TOKENS
- QUOTE
- DIGIT
- DCOL
- DECPT
- EORD
- INTEGR
- REAL
- DBL
- INVAL
- COMPD
1  BEGIN XPROM
2    IF PROMPT IS NOT TOO LONG THEN
3      CALL XMOV "" MOVE PROMPT INTO OUTPUT AREA
4      CALL EXEC TO WRITE PROMPT
5      SET XPROM RETURN CODE TO NORMAL RETURN
6      SET CONTINUE FLAG OFF
7      INITIALIZE COMMUNICATIONS BUFFER
8
9    IF INPUT BUFFER IS FULL PERFORM COMFUL
10     IF COMMA FLAG IS A COMMA THEN
11       ERREXIT IF COMBUF IS FULL PERFORM COMFUL
12     ENDIF
13     IF COMMA FLAG IS ON THEN
14       ERREXIT IF COMBUF CANNOT HOLD TOKEN PERFORM COMFUL
15     ENDIF
16     STORE NULL FIELD TOKEN IN COMBUF
17     INCREMENT #WORDS IN COMBUF BY 1
18     INCREMENT #TOKENS IN COMBUF BY 1
19     ENDIF
20
21     SET COMMA FLAG ON
22     SET NEXT INPUT CHARACTER
23
24     SET CONTINUE FLAG OFF
25     PERFORM TOKENS
26     ENDIF
27     ENDIF
28     END
29
30     IF CONTINUE FLAG IS OFF THEN
31       IF CONTINUE FLAG IS ON THEN
32       SET XPROM RETURN CODE TO SAY USER ENTERED CR
33     ENDIF
34     ENDIF
35
36     END
37
38     IF CONTINUE FLAG IS OFF THEN
39     SET XPROM RETURN CODE TO SAY PROMPT IS TOO LONG
40     ENDIF
41
42     END
43  END XPROM
1 BEGIN QUOTE
2    GET NEXT CHARACTER
3    SET #CHARACTERS = 0
4    DO WHILE (INPUT CHARACTER IS NOT A QUOTE AND
5       INPUT BUFFER HAS NOT BEEN COMPLETELY SCANNED) OR
6       (INPUT CHARACTER IS A QUOTE AND
7       NEXT CHARACTER IS A QUOTE AND
8       INPUT BUFFER HAS NOT BEEN COMPLETELY SCANNED)
9       MOVE CHARACTER INTO TEMPORARY BUFFER (#CHARACTERS)
10      IF INPUT CHARACTER IS A QUOTE THEN
11         GET NEXT CHARACTER
12      ENDIF
13      GET NEXT CHARACTER
14      ERREXIT IF LENGTH OF CHARACTER STRING IS 0 OR
15      ERREXIT IF INPUT CHARACTER IS NOT A QUOTE PERFORM INVAL
16      EXTRACT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM COMFUL
17      STORE CHARACTER STRING TOKEN IN COMBUF
18      STORE NUMBER OF CHARACTERS IN COMBUF
19      CALL XRPCK TO CONVERT CHARACTERS FROM R1 TO A2 FORMAT
20      INCREMENT #WORDS IN COMBUF BY 2*#CHARACTERS + 1)
21      INCREMENT #TOKENS IN COMBUF BY 1
22      GET NEXT CHARACTER
23    END QUOTE
1059  1Begin Decpt
1060  2Convert integer value to double precision value
1061  2Get next character
1062  2If input buffer is not exhausted then
1063  3If input character is a digit then
1064  4Perform decimal
1065  4Add fractional part to double precision value
1066  4Error if integer overflow has occurred to perform overflow
1067  3Endif
1068  3If input character is an "E" or a "D" then
1069  4Perform Cord
1070  3Else
1071  4Perform real
1072  3Endif
1073  2Else
1074  3Perform real
1075  2Endif
1076  1End Decpt
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1121</td>
<td>BEGIN INTEGER</td>
</tr>
<tr>
<td>1122</td>
<td>ERREXIT IF NEXT TOKEN IS NOT A COMMA AND</td>
</tr>
<tr>
<td>1123</td>
<td>ERREXIT IF NEXT TOKEN IS NOT A CLOSED PAREM AND</td>
</tr>
<tr>
<td>1124</td>
<td>ERREXIT IF INPUT BUFFER IS NOT EXHAUSTED PERFORM INVAL</td>
</tr>
<tr>
<td>1125</td>
<td>ERREXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM COMFUL</td>
</tr>
<tr>
<td>1126</td>
<td>CONVERT DOUBLE TO INTEGER</td>
</tr>
<tr>
<td>1127</td>
<td>ERREXIT IF INTEGER OVERFLOW HAS OCCURRED TO PERFORM OVERFLOW</td>
</tr>
<tr>
<td>1128</td>
<td>IF NEGATIVE NUMBER FLAG IS ON THEN</td>
</tr>
<tr>
<td>1129</td>
<td>SET INTEGER = -INTEGER</td>
</tr>
<tr>
<td>1130</td>
<td>ENDF</td>
</tr>
<tr>
<td>1131</td>
<td>STORE INTEGER TOKEN IN COMBUF</td>
</tr>
<tr>
<td>1132</td>
<td>INCREMENT #WORDS IN COMBUF BY 2</td>
</tr>
<tr>
<td>1133</td>
<td>INCREMENT #TOKENS IN COMBUF BY 1</td>
</tr>
<tr>
<td>1134</td>
<td>END INTEGER</td>
</tr>
<tr>
<td>1135</td>
<td>*</td>
</tr>
<tr>
<td>1136</td>
<td>*</td>
</tr>
<tr>
<td>1137</td>
<td>BEGIN REAL</td>
</tr>
<tr>
<td>1138</td>
<td>ERREXIT IF NEXT TOKEN IS NOT A COMMA AND</td>
</tr>
<tr>
<td>1139</td>
<td>ERREXIT IF NEXT TOKEN IS NOT A CLOSED PAREM AND</td>
</tr>
<tr>
<td>1140</td>
<td>ERREXIT IF INPUT BUFFER IS NOT EXHAUSTED PERFORM INVAL</td>
</tr>
<tr>
<td>1141</td>
<td>ERREXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM COMFUL</td>
</tr>
<tr>
<td>1142</td>
<td>SET REAL = DOUBLE * 10 ** POWER</td>
</tr>
<tr>
<td>1143</td>
<td>ERREXIT IF INTEGER OVERFLOW HAS OCCURRED TO PERFORM OVERFLOW</td>
</tr>
<tr>
<td>1144</td>
<td>IF NEGATIVE NUMBER FLAG IS ON THEN</td>
</tr>
<tr>
<td>1145</td>
<td>SET REAL = -REAL</td>
</tr>
<tr>
<td>1146</td>
<td>ENDF</td>
</tr>
<tr>
<td>1148</td>
<td>STORE TOKEN IN COMBUF</td>
</tr>
<tr>
<td>1149</td>
<td>INCREMENT #WORDS IN COMBUF BY 3</td>
</tr>
<tr>
<td>1150</td>
<td>INCREMENT #TOKENS IN COMBUF BY 1</td>
</tr>
<tr>
<td>1151</td>
<td>END REAL</td>
</tr>
</tbody>
</table>
1153 1 BEGIN INVAL
1154 2 CALL KCVT TO CONVERT OCTAL CHARACTER NUMBER TO ASCII
1155 2 CALL EXEC TO WRITE ERROR MESSAGE
1156 2 PERFORM XPRDM TO DISPLAY ORIGINAL PROMPT - NO RETURN
1157 1 END INVAL
1158 1 *
1159 1 *
1160 1 *
1161 1 BEGIN COMFUL
1162 2 CALL EXEC TO WRITE ERROR MESSAGE
1163 2 PERFORM XPRDM TO DISPLAY ORIGINAL PROMPT - NO RETURN
1164 1 END COMFUL
1165 1 *
1166 1 *
1167 1 *
1168 1 BEGIN OVFLOW
1169 2 CALL KCVT TO CONVERT OCTAL TO ASCII
1170 2 CALL EXEC TO WRITE ERROR MESSAGE
1171 2 PERFORM XPRDM TO DISPLAY ORIGINAL PROMPT - NO RETURN
1172 1 END OVFLOW
FORTRAN CALLING PROCEDURE

CALL XPIIT (LU, RPARMS)

TERMINATE A PROGRAM AFTER WAITING ON ANY BUFFERED I/O TO LU TO COMPLETE THEN PASS PARAMETERS BACK TO THE FATHER TASK.

INPUT

LU - LOGICAL UNIT OF TERMINAL OR OTHER OUTPUT DEVICE FOR WHICH PRINT MAY NOT BE COMPLETED. A VALUE OF ZERO WILL INDICATE NO I/O WAIT

RPARMS - ARRAY OF FIVE PARAMETERS TO BE RETURNED TO FATHER TASK

OUTPUT

RPARMS IS RETURNED TO THE FATHER TASK

NOTES

USES EXEC, PRTN

XPIIT DOES NOT RETURN TO THE CALLER.

BEGIN XPIIT

IF LU IS NON-ZERO THEN WAIT ON ANY CLASS I/O TO COMPLETE (CLASS I/O CONTROL THEN GET)

END XPIIT
FORTRAN CALLING PROCEDURE

100 CALL XRBIT (BIT, BITNUM, STRING)

---

BIT SET/CLEAR ROUTINE FOR MULTIPLE WORD BIT STRINGS. SET THE
VALUE OF BIT NUMBER 'BITNUM' IN STRING 'STRING' TO 'BIT'.

---

INPUT

BIT - INTEGER VALUE OF ZERO OR ONE TO BE SET IN BIT NUMBER

BITNUM - INTEGER BIT NUMBER OF BIT TO BE SET/CLEARED (FIRST BIT OF
STRING IS BIT NUMBER ZERO)

STRING - BIT STRING OF AT LEAST 'BITNUM' BITS IN LENGTH INTO WHICH
'BIT' IS TO BE SET

OUTPUT

STRING - BIT NUMBER 'BITNUM' HAS THE VALUE OF 'BIT'

---

NOTES

USES .ENTR, XRSET

---

BEGIN XRBIT

TRANSFER CALLING SEQUENCE

COMPUTE ADDRESS OF WORD CONTAINING BITNUM

COMPUTE BIT NUMBER WITHIN WORD

CALL XRSET TO SET/CLEAR BIT

END XRBIT
INTEGER FUNCTION
XRCPR(LENGTH, ARRAY1, ARRAY2)

************
1 101 COMPAR E 'LENGTH' WORDS OF 'ARRAY1' TO 'ARRAY2'. IF THE ARRAYS ARE IDENTICAL A VALUE OF 0 IS RETURNED. IF THE COMPARE FAILS A VALUE IS RETURNED INDICATING WHETHER THE FIRST UNEQUAL ELEMENT OF ARRAY1 IS LESS THAN ARRAY2 (-1) OR GREATER (+1)

************
1 102 INPUT
2 102 LENGTH - POSITIVE INTEGER NUMBER OF WORDS TO BE COMPARED
3 102 ARRAY1 - ARRAY OF INTEGERS OR CHARACTERS TO BE COMPARED TO ARRAY2
4 102 ARRAY2 - ARRAY OF INTEGERS OR CHARACTERS TO BE COMPARED TO ARRAY1

************
1 103 OUTPUT
2 103 FUNCTION VALUE - 0, ARRAYS EQUAL
3 103 -1, ARRAYS NOT EQUAL, ARRAY1 < ARRAY2
4 103 +1, ARRAYS NOT EQUAL, ARRAY1 > ARRAY2

************
1 105 NOTES
2 105 USES .ENTR
3 105...

1 BEGIN XRCPR
2 TRANSFER CALLING SEQUENCE
3 SETUP COMPARE
4 COMPARE ARRAY1 AND ARRAY2
5 RETURN RESULT FLAG
6 END XRCPR
80  1  CDO  FORTRAN CALLING PROCEDURE.
81  1  CDO
82  1  CDO  CALL XRD18 (DOUBLE, ASCII)
83  1  CDO
84  1  C**********
85  1  C01  CONVERT A DOUBLE PRECISION REAL NUMBER TO AN ASCII STRING IN
86  1  C01  I0P18.1Y FORMAT
87  1  C01
88  1  C01  ********
89  1  C02  INPUT
90  1  C02
91  1  C02  DOUBLE - THREE WORD DOUBLE PRECISION REAL NUMBER TO BE CONVERTED
92  1  C02
93  1  C**********
94  1  C03  OUTPUT
95  1  C03  ASCII - NINE ASCII CHARACTER STRING REPRESENTATION OF
96  1  C03  'DOUBLE'
97  1  C03
98  1  C**********
99  1  C04  LOCAL
100  1  C04  D - WORKING LOCATION FOR ABSOLUTE VALUE OR 'DOUBLE'
101  1  C04  REPEATEDLY MODIFIED TO EXTRACT REMAINING DECIMAL DIGITS
102  1  C04
103  1  C**********
104  1  C05  NOTES
105  1  C05  USES DOUBLE, FLOAT, IA10, IDINT, IO8, KCVT, XREXT
106  1  C05
107  1  C**********
108  1  *
109  1  *
110  1  *
111  1  *
112  1  BEGIN XRD18
113  2  SET SIGN FIELD
114  2  MOVE ABSOLUTE VALUE OF 'DOUBLE' INTERNAL
115  2  COMPUTE EXPONENT
116  2  SET SIGN AND VALUE OF EXPONENT FIELD
117  2  REDUCE VALUE TO RANGE OF 1 < VALUE <10
118  2  EXTRACT FIRST DIGIT, MERGE WITH SIGN AND STORE FIELD
119  2  EXTRACT SECOND DIGIT, MERGE WITH DECIMAL AND STORE FIELD
120  2  DO FOR NEXT FIVE PAIRS OF DIGITS
121  3  MULTIPLY BY 100 TO EXTRACT PAIR
122  3  EXTRACT DIGITS AND STORE FIELD
123  2  EMD00
124  1  EMD XRD18
FORTRAN CALLING PROCEDURE

CALL XREQ

MAKE A WORK AREA MANAGEMENT REQUEST AND WAIT FOR RESPONSE

COMMON XE - CLASNO, FLAGS, LU, REQBUF, REQTPR

ID SEGMENT PARAMETERS RETURNED FROM THE MANAGER

INPUT

OUTPUT

CLASS I/O WRITE/READ TO CLASS 'CLASNO'

REQBUF AND RESPONSE IS PRINTED IF REQUESTED

COMMUNICATES WITH FDS MANAGER FATHER TASK

USES EXEC, IAND, XRMOV, XRMG, XR6G, XRSFL, XUDPL, XVPW

BEGIN XREQ

PERFORM TRACE

OUTPUT REQUESTS TO MANAGER

REQUEST AWA MANAGEMENT AND WAIT FOR RESPONSE

RETRIEVE MANAGER RESPONSE

PERFORM TRACE

RETURN RESPONSE IN REQTPR

END XREQ

BEGIN TRACE

IF TRACE REQUESTED

THEN

DO FOR EACH REQUEST

CALL XUDPL TO FORMAT LINE

OUTPUT LINE

ENDDO

OUTPUT PARM1 AND PARM2

ENDIF

END TRACE
1 *DO INTEGER FUNCTION
2 *DO XREST(START, LENGTH, SOURCE)
3 *DO
4 ********
5 *DO EXTRACT 'LENGTH' BITS OF 'SOURCE' BEGINNING WITH BIT 'START'
6 *DO AND RIGHT ADJUST
7 *DO
8 ********
9 *DO INPUT
10 *DO START - INTEGER INDICATING LEFT MOST BIT OF FIELD TO BE EXTRACTED
11 *DO (SIGN BIT = 0)
12 *DO LENGTH - POSITIVE INTEGER SIZE OF FIELD TO BE EXTRACTED
13 *DO SOURCE - WORD FROM WHICH FIELD IS TO BE EXTRACTED
14 *DO
15 ********
16 *DO NOTES
17 *DO USES .ENTR
18 *DO
19 *DO
20 *DO
21 *DO
22 *DO
23 *DO
24 *DO
25 *DO
26 *DO
27 *DO
28 *DO
29 *DO
30 *DO
31 *DO
32 *DO
33 *DO
34 *DO
35 *DO
36 *DO
37 *DO
38 *DO
39 *DO
40 *DO
41 *DO
42 *DO
43 *DO
44 *DO
45 *DO
46 *DO
47 *DO
48 *DO
49 *DO
50 *DO
51 *DO
52 *DO
53 *DO
54 *DO
55 *DO
56 *DO
57 *DO
58 *DO
59 *DO
60 *DO
61 *DO
62 *DO
63 *DO
64 *DO
65 *DO
66 *DO
67 *DO
68 *DO
69 *DO
70 *DO
71 *DO
72 *DO
73 *DO
74 *DO
75 *DO
76 *DO
77 *DO
78 *DO
79 *DO
80 *DO
81 *DO
82 *DO
83 *DO
84 *DO
85 *DO
86 *DO
87 *DO
88 *DO
89 *DO
90 *DO
91 *DO
92 *DO
93 *DO
94 *DO
95 *DO
96 *DO
97 *DO
98 *DO
99 *DO
100 BEGIN XREST
101 2 TRANSFER CALLING SEQUENCE
102 2 IF START NOT = 0
103 2 THEN
104 3 CONSTRUCT SHIFT
105 3 LOAD A WITH SOURCE
106 3 SHIFT BA LEFT START BITS
107 2 ELSE
108 3 LOAD A WITH SOURCE
109 2 ENOIF
110 2 SAVE A
111 2 CLEAR B
112 2 CONSTRUCT SHIFT
113 2 RESTORE A
114 2 SHIFT BA LEFT LENGTH BITS
115 2 MOVE RESULT FROM B TO A
116 1 END XREST
FORTRAN CALLING PROCEDURE

CALL XRE14 (REAL, ASCII)

COND
CONVERT A SINGLE PRECISION REAL NUMBER TO AN ASCII STRING IN
1PE14.6 FORMAT

INPUT
REAL - TWO WORD SINGLE PRECISION REAL NUMBER TO BE CONVERTED

OUTPUT
ASCII - SEVEN WORD ASCII CHARACTER STRING REPRESENTATION OF
'REAL'

LOCAL
R - WORKING LOCATION FOR ABSOLUTE VALUE OR 'REAL' REPEATEDLY
MODIFIED TO EXTRACT REMAINING DECIMAL DIGITS

NOTES
USES FLOAT, IAND, IFIX, IOR, KCVT, XREXT, XRSFL

BEGIN XRE14
SET SIGN FIELD
MOVE ABSOLUTE VALUE OF 'REAL' INTERNAL
COMPUTE EXPONENT
SET SIGN AND VALUE OF EXPONENT FIELD
REDUCE VALUE TO RANGE OF 1 <= VALUE < 10
EXTRACT FIRST DIGIT AND STORE FIELD
SET DECIMAL FIELD
DO FOR NEXT THREE PAIRS OF DIGITS
MULTIPLY BY 100 TO EXTRACT PAIR
EXTRACT DIGITS AND STORE FIELD
END XRE14
263 1 CD0  FORTRAN CALLING PROCEDURE
264 1 CD0
265 1 CD0  CALL XR16 (INTEGER, ASCII)
266 1 CD0
267 1*******
268 1 CD1  CONVERT A SIXTEEN BIT SIGNED BINARY INTEGER TO A SIX CHARACTER
269 1 CD1  ASCII STRING
270 1 CD1
271 1*******
272 1 CD2  INPUT
273 1 CD2  INTEGER - SIXTEEN BIT INTEGER TO BE CONVERTED
274 1 CD2
275 1*******
276 1 CD3  OUTPUT
277 1 CD3  ASCII - THREE WORD CHARACTER STRING REPRESENTATION OF 'INTEGER'
278 1 CD3
279 1 CD3
280 1*******
281 1 CD4  LOCAL
282 1 CD4  I  - INTERNAL LOCATION FOR 'INTEGER' REPEATEDLY MODIFIED TO
283 1 CD4  PRODUCE 'ASCII'
284 1 CD4  WRK  - SEVEN WORD WORKING BUFFER FOR CONSTRUCTION OF 'ASCII'
285 1 CD4
286 1*******
287 1 CD5  NOTES
288 1 CD5  USES XRMV AND XRPCX
289 1 CD5
290 1*******
291 1 *  
292 1 *  
293 1 *  
294 1 *  
295 1 BEGIN XR16
296 2  BLANK WORKING SPACE
297 2  CONSTRUCT 'ASCII' LEAST SIGNIFICANT DIGITS FIRST USING REMAINING
298 2  SET SIGN OF 'INTEGER' IN 'ASCII'
299 2  CALL XRPCX TO CONVERT FROM A1 TO A2 FORMAT
300 1 END XR16
INTEGER FUNCTION

*00  *00  XRLOC(A)
*00  *00
********
*01  RETURN THE 16-BIT MAPPED ADDRESS OF A
*01  *01  INPUT
*02  *02  A  - VARIABLE, ROUTINE, ETC. FOR WHICH THE ADDRESS IS DESIRED
*02  *02
********
*03  OUTPUT
*03  XRLOC - 16-BIT ADDRESS OF A
*03  *03
********
*05  NOTES
*05  NO EXTERNAL REFERENCES
*05  *05
********
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
1  *  
********
BEGIN XRLOC
1  TRANSFER CALLING SEQUENCE
2  LOAD THE ADDRESS OF THE CALLING PARAMETER
1  END XRLOC

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR
FORTRAN CALLING PROCEDURE

1  *00  CALL XRMV (LENSTH, SOURCE, OBJECT)
2  *00
3  *00
4  *00
5  *00
6  *00
7  *00
8  *00
9  *00
10  *00
11  *00
12  *00
13  *00
14  *00
15  *00
16  *00
17  *00
18  *00
19  *00
20  *00
21  *00
22  *00
23  *00
24  *00
25  *00
26  *00
27  *00
28  *00
29  *00
30  *00
31  *00
32  *00
33  *00
34  *00
35  *00
36  *00
37  *00
38  *00
39  *00
40  *00
41  *00
42  *00
43  *00
44  *00
45  *00
46  *00
47  *00
48  *00
49  *00
50  *00
51  *00
52  *00
53  *00
54  *00
55  *00
56  *00
57  *00
58  *00
59  *00
60  *00
61  *00
62  *00
63  *00
64  *00
65  *00
66  *00
67  *00
68  *00
69  *00
70  *00
71  *00
72  *00
73  *00
74  *00
75  *00
76  *00
77  *00
78  *00
79  *00
80  *00
81  *00
82  *00
83  *00
84  *00
85  *00
86  *00
87  *00
88  *00
89  *00
90  *00
91  *00
92  *00
93  *00
94  *00
95  *00
96  *00
97  *00
98  *00
99  *00
100  *00
101  *00
102  *00
103  *00
104  *00
105  *00
106  *00
107  *00
108  *00
109  *00
110  *00
111  *00
112  *00
113  *00
114  *00
115  *00
116  *00
117  *00
118  *00
119  *00
120  *00
121  *00
122  *00
123  *00
124  *00
125  *00
126  *00
127  *00
128  *00
129  *00
130  *00
131  *00
132  *00
133  *00
134  *00
135  *00
136  *00
137  *00
138  *00
139  *00
140  *00
141  *00
142  *00
143  *00
144  *00
145  *00
146  *00
147  *00
148  *00
149  *00
150  *00
151  *00
152  *00
153  *00
154  *00
155  *00
156  *00
157  *00
158  *00
159  *00
160  *00
161  *00
162  *00
163  *00
164  *00
165  *00
166  *00
167  *00
168  *00
169  *00
170  *00
171  *00
172  *00
173  *00
174  *00
175  *00
176  *00
177  *00
178  *00
179  *00
180  *00
181  *00
182  *00
183  *00
184  *00
185  *00
186  *00
187  *00
188  *00
189  *00
190  *00
191  *00
192  *00
193  *00
194  *00
195  *00
196  *00
197  *00
198  *00
199  *00

Fortran Calling Procedure

CALL XRMG (NUMBER, LOCATE, LENGTH, SOURCE)

insert 'LENGTH' words of text from 'SOURCE' into message

'NUMBER' beginning after 'LOCATE' words, concatenate to prefix.

truncate to eightt characters and output to user terminal.

Input

Number - integer message number of the form 'A-MM' where

A - area indicator as follows

1 - AS
2 - XD
3 - XV
4 - XI
5 - XS
6 - YT
7 - X
8 - XU
9 - DF
10 - SC

MN - message number or zero which indicates only 'LENGTH'

Words of 'SOURCE' to be output

Locate - integer number of words of message to precede

'SOURCE' (not used if 'M' of 'NUMBER' is zero)

Length - integer number of words of 'SOURCE' to be inserted into

message. Zero indicates no insertion

Source - array of characters to be inserted into message (not used

if 'LENGTH' is zero)

Common

Lu - users logical unit number

Output

Remainder of message

A-MM message(1-LOCATE) SOURCE(1-LENGTH)

Notes

Uses FDS system message file XRMG

Uses close, exec, iams, kcut, open, read, xmov, xudog
INTEGER FUNCTION

XMKDB(BIT, BITNUM, STRING)

***********

SEARCH BIT STRING 'STRING' BEGINNING AT BIT NUMBER 'BITNUM' FOR
THE NEXT OCCURRENCE OF BIT SETTING 'BIT'.

***********

INPUT

BIT - INTEGER VALUE THE LAST BIT OF WHICH IS TO BE COMPARED TO
BITS OF 'STRING' FOR A MATCH

BITNUM - UNSIGNED SIXTEEN BIT INTEGER INDICATING THE BIT NUMBER IN
'STRING' WITH WHICH TO BEGIN THE SEARCH (FIRST BIT OF
'STRING' IS BIT NUMBER ZERO)

STRING - BIT STRING TO BE SEARCHED. SEARCH WILL CONTINUE THROUGH
MEMORY UNTIL A VALUE OF 'BIT' IS DETECTED.

***********

OUTPUT

FUNCTION VALUE - BIT NUMBER OF NEXT OCCURRENCE OF 'BIT' >= 'BITNUM'

***********

NOTES

USES .ENTR

SEARCH WILL NOT TERMINATE UNTIL A VALUE OF 'BIT' IS DETECTED OR
ALL OF MEMORY HAS BEEN EXAMINED. Thus, Appropriate steps should
be taken to force a match at the end of the string.

A maximum bit string length of 65535 bits (4096 words) can be
meaningfully accommodated.
BEGIN

1. Compute starting word number
2. Compute starting bit number
3. Load starting word
4. Shift word until starting bit number is in sign bit
5. Initialize bit count
6. Do while sign bit is not equal to bit
7. If word not finished
   Then
     Shift word left
     Increment bit count
   Else
     Do until word with some 'bit' value found
   End
7. Load next word
8. Enddo
9. Clear bit count
10. Endif
11. Enddo
12. Return value of matching bit number

END
542 1   00  FORTRAN CALLING PROCEDURE
543 1   00  CALL XRO6 (BINARY, OCTAL)
544 1   00  
545 1   00  
546 1   00  
547 1   00  
548 1   00  CONVET A WORD FROM BINARY TO SIX CHARACTER OCTAL REPRESENTATION
549 1   00  
550 1   00  
551 1   00  
552 1   00  
553 1   00  INPUT
554 1   00  BINARY - BINARY WORD TO BE CONVERTED
555 1   00  
556 1   00  
557 1   00  
558 1   00  
559 1   00  OUTPUT
560 1   00  OCTAL - THREE WORD ARRAY CONTAINING OCTAL REPRESENTATION OF
561 1   00  "BINARY" IN ASCII FORMAT (C6)
562 1   00  
563 1   00  
564 1   00  
565 1   00  
566 1   00  NOTES
567 1   00  
568 1   00  USES .EXITR
569 1   00  
570 1   00  
571 1   00  
572 1   00  
573 1   00  
574 1   00  
575 1   00  
576 1   00  
577 1   00  
578 1   00  
579 1   00  
580 1   00  
581 1   00  
582 1   00  
583 1   00  
584 1   00  
585 1   00  
586 1   00  
587 1   00  
588 1   00  BEGIN XRO6
589 1   00  
590 1   00  TRANSFER CALLING SEQUENCE
591 1   00  
592 1   00  
593 1   00  
594 1   00  
595 1   00  
596 1   00  
597 1   00  
598 1   00  
599 1   00  
600 1   00  STORE PAIR OF DIGITS IN OCTAL (C6)
601 1   00  
602 1   00  
603 1   00  
604 1   00  .END XRO6
FORTRAN CALLING PROCEDURE

CALL XRPACK (LENGTH, UNPKED, PACKED)

********

********

********

********

********

********

********

********

********

********

BEGIN XRPACK

TRANSFER CALLING SEQUENCE

SET BYTE FLAG FOR HIGH BYTE

INITIALIZE PACKED POINTER

DO FOR EACH CHARACTER IN UNPKED

IF BYTE FLAG SET HIGH THEN

LOAD A WITH NEXT WORD OF UNPKED

SHIFT CHARACTER INTO HIGH BYTE

ELSE

INCLUSIVE OR NEXT WORD OF UNPKED INTO A

STORE A IN PACKED

INCREMENT POINTER

ENDIF

FLIP BYTE FLAG

ENDIF

IF BYTE FLAG SET LOW THEN

INCLUSIVE OR BLANK INTO LOW BYTE

ENDIF

STORE A IN PACKED

END XRPACK
FORTRAN CALLING PROCEDURE

1  *00  CALL XRQFN (PREFIX, NAME4, NAME6)
2  *00
3  *00
4  *********
5  *00
6  *00  XRQFN BUILDS A QUALIFIED FILE NAME OF UPTO SIX CHARACTERS IN
7  *00  LENGTH BY PREFixING THE INPUT ONE TO FOUR CHARACTER NAME WITH
8  *00  THE PREFIX CHARACTER AND APPENDING A USER QUALIFIER CODE TO THE
9  *00  END
10  *00
11  *00  *********
12  *00
13  *00  INPUT
14  *00  PREFIX - FILE TYPE PREFIX STORED IN R1 FORMAT
15  *00  NAME4 - ONE TO FOUR CHARACTER PACKED NAME TO BE QUALIFIED
16  *00  COMMON XE - QUAL
17  *00  *********
18  *00  OUTPUT
19  *00  NAME6 - THREE TO SIX CHARACTER PACKED QUALIFIED NAME
20  *00  *00
21  *00  *********
22  *00  NOTES
23  *00  *05  USES .ENTR
24  *00
25  *00
26  *00  *********
27  *00
28  *00
29  BEGIN XRQFN
30  *00  STORE PREFIX IN FIRST POSITION OF INTERNAL CHARACTER STRING
31  *00  MOVE NAME4 INTO NEXT FOUR POSITIONS
32  *00  STORE BLANK IN SIXTH POSITION
33  *00  LOCATE FIRST BLANK CHARACTER
34  *00  REPLACE BLANK WITH USER ID CHARACTER (QUAL)
35  *00  MOVE QUALIFIED NAME TO NAME6
36  *00  END XRQFN
FORTRAN CALLING PROCEDURE

CALL XRSET (START, LENGTH, SOURCE, OBJECT)

REPLACE LENGTH BITS OF OBJECT, BEGINNING WITH BIT START,
WITH THE RIGHT MOST LENGTH BITS OF SOURCE

INPUT
START - INTEGER INDICATING LEFT MOST BIT OF FIELD TO BE REPLACED
(SIGN BIT = 0)
LENGTH - POSITIVE INTEGER SIZE OF FIELD BEING REPLACED
SOURCE - WORD CONTAINING REPLACEMENT FIELD RIGHT ADJUSTED
OBJECT - FIELD INTO WHICH FIELD IS TO BE INSERTED
NOTES
USES .ENTR

BEGIN XRSET
TRANSFER CALLING SEQUENCE
CONSTRUCT SHIFT INSTRUCTIONS
SHIFT LENGTH BITS OF SOURCE INTO HIGH END OF CLEARED REGISTER
SHIFT REGISTER RIGHT START BITS TO PROPERLY POSITION FIELD
CONSTRUCT MASK AND CLEAR FIELD OF OBJECT
INCLUSIVE OR POSITIONED SOURCE FIELD INTO OBJECT
END XRSET
734 1 *D0
735 1 *D0
736 1 *D0 XRSFL(COUNT, SOURCE)
737 1 *D0 XSSFN(COUNT, SOURCE)
738 1 *D0
739 1 *********
740 1 *D1
741 1 *D1 SHIFT 'SOURCE' LEFT/RIGHT LOGICALLY 'COUNT' BITS
742 1 *D1
743 1 *********
744 1 *D2 INPUT
745 1 *D2 COUNT - POSITIVE INTEGER SPECIFYING NUMBER OF BITS TO SHIFT
746 1 *D2 SOURCE - WORD TO BE SHIFTED
747 1 *D2
748 1 *********
749 1 *D5 NOTES
750 1 *D5 USES .ENTR
751 1 *D5
752 1 *********
753 1 *
754 1 *
755 1 *
756 1 *
757 1 BEGIN XRSFL
758 2 SET FOR LEFT SHIFT
759 2 PERFORM SHIFT (FLAG, COUNT, SOURCE)
760 1 END XRSFL
761 1 BEGIN XRSFR
762 2 SET FOR RIGHT SHIFT
763 2 PERFORM SHIFT (FLAG, COUNT, SOURCE)
764 1 END XRSFR
765 1 BEGIN SHIFT
766 2 TRANSFER CALLING SEQUENCE
767 2 CONSTRUCT SHIFT INSTRUCTION
768 2 LOAD A WITH SOURCE
769 2 CLEAR B
770 2 SHIFT BA AS SPECIFIED
771 1 END SHIFT
773 1 *DO  FORTRAN CALLING PROCEDURE
774 1 *DO  CALL XRUNG (NAME6, NAME4)
775 1 *DO
776 1 *DO
777 1 **********
778 1 *D1  XRUNG REMOVES THE PREFIX AND SUFFIX QUALIFYING CHARACTERS FROM
779 1 *D1  A SIX CHARACTER FILE NAME
780 1 *D1
781 1 *D1
782 1 **********
783 1 *D2  INPUT
784 1 *D2  NAME6 - THREE TO SIX CHARACTER PACKED QUALIFIED NAME
785 1 *D2
786 1 **********
787 1 *D3  OUTPUT
788 1 *D3  NAME4 - ONE TO FOUR CHARACTER PACKED NAME WITH PREFIX AND SUFFIX
789 1 *D3  REMOVED
790 1 *D3
791 1 **********
792 1 *D5  NOTES
793 1 *D5  USES .ENTR
794 1 *D5
795 1 **********
796 1 *
797 1 *
798 1 *
799 1 *
800 1 BEGIN XRUNG
801 2 MOVE CHARACTERS 2-5 OF NAME6 INTO NAME4
802 2 IF SIXTH CHARACTER IS BLANK, THEN
803 3 LOCATE LAST NON-BLANK CHARACTER OF NAME4
804 3 BLANK THAT CHARACTER
805 2 ENDF
806 1 END XRUNG
FORTAN CALLING PROCEDURE

1 00 CALL XRUPK (LENGTH, PACKED, UNPKED, COUNT)

1 #01 CONVERT 'LENGTH' WORDS OF 'PACKED' FROM A2 FORMAT TO R1 FORMAT,
1 #01 REMOVING BLANKS NOT DELIMITED BY QUOTE MARKS, AND RETURN IN
1 #01 'UNPKED' WITH THE NON-BLANK CHARACTER COUNT IS 'COUNT'.

1 #01 INPUT
820 1 #02 LENGTH - POSITIVE INTEGER NUMBER OF WORDS IN PACKED

1 #02 PACKED - ARRAY OF CHARACTER DATA IN A2 FORMAT

1 #02 UNPKED - ARRAY OF NON-BLANK CHARACTERS IN R1 FORMAT

1 #03 COUNT - NUMBER OF CHARACTERS IN UNPKED

1 #03 NOTES
830 1 #05 USES .ENTR
831 1 #05 IF PACKED AND UNPKED ARE THE SAME ADDRESS SPACE UNPKED MAY OVERLAP

1 #05 PACKED
833 1 #05 ** CAUTION: XRUPK CANNOT HANDLE QUOTE MARKS WITHIN CHARACTER STRINGS.
835 1 #05
837 1 #05
839 1 BEGIN XRUPK
840 2 TRANSFER CALLING SEQUENCE
841 2 TURN ON BLANK REMOVAL
842 2 INITIALIZE COUNTER
843 2 DO FOR EACH WORD OF PACKED
844 2 LOAD A WITH NEXT WORD
845 2 ROTATE A 8 BITS
846 2 DO FOR EACH BYTE OF WORD
847 4 AND OFF HIGH BYTE
848 4 IF A = QUOTE MARK
849 4 THEN
850 5 CHANGE BLANK REMOVAL OPTION
851 4 ENDIF
852 4 IF BLANK REMOVAL IS ON
853 4 THEN
854 5 IF A NOT = BLANK
855 5 THEN
856 6 INCREMENT COUNT
857 6 STORE A IN UNPACKED
858 6 ENDIF
859 4 ELSE
860 5 INCREMENT COUNT
861 5 STORE A IN UNPACKED
862 4 ENDIF
863 4 RELOAD A WITH WORD
864 3 ENDDO
865 2 ENDDO
866 2 RETURN VALUE OF COUNT
867 1 END XRUPK
BEGIN XR15P
CONVERT WORD COUNT INTO CHARACTER COUNT
SET STORE INDEX TO FIRST POSITION IN CHARACTER STRING
CLEAR CHARACTER STRING FLAG
DO FOR EACH CHARACTER IN STRING
  IF CHARACTER IS SPECIAL CHARACTER INDICATING CHARACTER STRING 'THF':
    REPLACE CHARACTER STRING INDICATOR WITH QUOTE MARK
  FLIP CHARACTER STRING FLAG
  CLEAR BLANK FLAG
ELSE
  IF CHARACTER STRING FLAG IS CLEAR, THEN
    IF CHARACTER IS A BLANK, THEN
      IF BLANK FLAG IS SET (AT LEAST ONE PREDECESSING BLANK), THEN
        SKIP THIS CHARACTER (EXIT TO ENDDO)
      ELSE
        SET BLANK FLAG
      ENDIF
      ELSE
        CLEAR BLANK FLAG
    ENDIF
  ENDIF
ENDIF
STORE CHARACTER AT INDEXED POSITION
INCREMENT STORE INDEX
ENDDO
IF NUMBER OF CHARACTERS STORED "THF" 000
STORE ONE MORE BLANK
ENDIF
ENDIF
END XR15P
BEGIN XSERE
DO UNTIL a % or 'EXIT' IS ENTERED
CALL XSPK TO BUILD A PROMPT BASED ON PROMPT MODE FOR THE
NEXT TABLE ENTRY (INDICATED BY TABNDX)
CALL XTCOM TO ISSUE THE PROMPT AND RETURN RESPONSE
EXIT XSERE IF ERROR IN XTCOM :ERR10:
EXIT XSERE IF a WAS ENTERED
IF NOTHING (ONLY CR) ENTERED, THEN
IF PROMPT MODE IS NOT 'ALL', THEN
CALL XRMG - 'INVALID INPUT'
ENDIF
ELSE
CALL XSNPT TO PROCESS INPUT BASED ON PROMPT MODE,
CURRENT TABLE ENTRY (TABNDX), AND PROMPTED SEQUENCE
NUMBER (PRNUM)
ENDIF
ENDO
BUILD AWA REQUEST TO DELETE/VERIFY ABSENCE OF MDTAB
CALL XSPCK TO PACK THE TABLE BUFFER (REMOVE DELETED ENTRIES)
BUILD AWA REQUEST TO ALOCATE MDTAB
IF NUMBER OF TABLE ENTRIES (NNUMEN) > 0, THEN
BUILD AWA REQUEST TO STORE MDTAB
CALL XRGN TO PROCESS THE REQUESTS
IF THE ALOCATE REQUEST FAILED, THEN
CALL XRMG - 'MDTAB FULL, SEQUENCE TABLE NOT STORED'
CALL EXEC TO FREE CLASS NO. AND SAM BUFFER
ELSE
CALL XRMG - 'VACUOUS TABLE -- NOT STORED'
ENDIF
ENDIF
EXIT XSERE
:ERR10: CALL XRMG - 'SYSTEM ERROR'
END XSERE
FORTRAN CALLING PROCEDURE

CALL XSPRN

XSPRN BUILDS SEQUENCE TABLE EDITOR PROMPTS.

INPUT

COMMON XE - LU

COMMON XB - DEBUG, NUMENT, PRNTMD, TABMDX, WKBVF

OUTPUT

COMMON XB - PRMLEN, PROMPT, SERNO, TABMDX

USES ROUTINES

XRI6
XMOV
XMS6
XSET
XDEG
BEGIN XSPRM

1201 IF PROMPT MODE IS ALL, THEN
1202 IF NUMBER OF ENTRIES (NUMENT) > 0, AND
1203 ENTRIES EXIST BEYOND TABLE ENTRY INDEX (TABINDX), THEN
1204 DO UNTIL A NON-DELETED ENTRY IS FOUND
1205 INCREMENT TABLE ENTRY INDEX (TABINDX) TO NEXT ENTRY (+7)
1206 END DO
1207 BUILD PROMPT OF THE FORM 'MMMM-PROC.INT'
1208 SET PROMPT SEQUENCE NUMBER (SEINO) TO SEQUENCE NO. OF ENTRY
1209 ELSE
1210 SET PROMPT NO. TO CREATE
1211 END IF
1212 END IF
1213 IF PROMPT MODE IS CREATE, THEN
1214 SET TABLE ENTRY INDEX (TABINDX) TO NEXT ENTRY (NUMENT * 7 + 1)
1215 IF NUMBER OF TABLE ENTRIES (NUMENT) > 0, THEN
1216 IF SEQUENCE NO. OF LAST ENTRY > 32760, THEN
1217 CALL XPRSC = 'UNABLE TO BUILD SEQUENCE NO. > 32700'
1218 SET PROMPT MODE TO UPDATE
1219 ELSE
1220 SET PROMPT SEQUENCE NUMBER (SEINO) TO NEXT MULTIPLE OF 100
1221 SET SEQUENCE NUMBER OF LAST TABLE ENTRY
1222 END IF
1223 ELSE
1224 SET PROMPT SEQ. NO. (SEINO) TO BE 100
1225 END IF
1226 IF PROMPT MODE IS NOT UPDATE, THEN
1227 BUILD PROMPT OF THE FORM 'MMMM-
1228 END IF
1229 ELSE
1230 IF PROMPT MODE IS UPDATE, THEN
1231 BUILD PROMPT OF THE FORM
1232 SET PROMPT LENGTH TO 0 CAUSING # PROMPT TO BE ISSUED
1233 END IF
1234 END XSPRM
200 1 BEGIN XSNPT
201 2 IF PROMPT MODE IS UPDATE, THEN
202 3 IF TOKEN INPUT IS AN INTEGER, THEN
203 4 ERREXIT IF INTEGER < 1:ERROR:
204 5 RETAIN INTEGER AS SEQUENCE NO. (SEQNO)
205 6 INCREMENT TO NEXT TOKEN
206 7 ERREXIT IF INTEGER IS NOT "#":ERROR:
207 8 INCREMENT TO NEXT TOKEN
208 9 START SEARCH UNTIL NUMBER OF TABLE ENTRIES (NUMENT) SEARCHED
210 10 IF SEQUENCE NO. OF ENTRY = SEQUENCE NO. INPUT (SEQNO.)
211 11 SET TABLE ENTRY INDEX (TABNOX) TO THIS ENTRY
212 12 SET INSERT FLAG TO ZERO INDICATING REPLACEMENT OF ENTRY
213 13 SET TABLE ENTRY INDEX (TABNOX) TO THIS ENTRY
214 14 SET INSERT FLAG TO 1 INDICATING INSERT NEEDED
215 15 ELSE
216 16 INCREMENT TO NEXT TABLE ENTRY
217 17 ENDLOOP
218 18 SET TABLE ENTRY INDEX (TABNOX) TO NEXT ENTRY OF TABLE
219 19 SET INSERT FLAG TO 2 INDICATING EXTENSION TO END OF TABLE
220 20 ENDSEARCH
221 21 CALL XSENT TO REPLACE/INSERT/ADD TABLE ENTRY BASED ON INSERT FLAG
222 22 ELSE
223 23 ERREXIT IF TOKEN IS NOT A NAME :ERROR1:
224 24 START SEARCH LIST OF SEQ. EDIT. DIRECTIVES SEARCHED
225 25 IF NAME INPUT IS DIRECTIVE
226 26 SET INDEX TO DIRECTIVE LIST ENTRY
227 27 ELSE
228 28 INCREMENT TO NEXT DIRECTIVE
229 29 ENDLOOP
230 30 ERREXIT :ERROR2:
231 31 ENDSEARCH
232 32 CASE (XSLIS, XSDEL, XSPAN, XSNAM, XSMGR), INDEX
233 33 ENDIF
234 34 ELSE, PROMPT MODE IS NOT UPDATE
235 35 IF TOKEN IS #, THEN
236 36 SET PROMPT MODE TO UPDATE
237 37 ELSE
238 38 IF AN & IS INPUT, THEN
239 39 ERREXIT IF PROMPT MODE IS NOT UPDATE :ERROR1:
240 40 MARK THIS TABLE ENTRY AS DELETED
241 41 DO FROM END OF TABLE UNTIL A NONDELETED ENTRY IS FOUND
242 42 IF TABLE ENTRY IS MARKED FOR DELETION, THEN
243 43 DECREMENT NUMBER OF TABLE ENTRIES BY ONE
244 44 ENDIF
245 45 ENDIF
246 46 ELSE
247 47 IF PROMPT MODE IS ALL, THEN
248 48 SET INSERT FLAG TO TWO TO INDICATE EXTENSION OF TABLE
249 49 ELSE
250 50 SET INSERT FLAG TO ZERO TO INDICATE REPLACEMENT OF TABLE ENTRY
251 51 ENDIF
252 52 CALL XSENT TO BUILD ENTRY BASE ON INSERT FLAG
253 53 ENDIF
254 54 ENDIF
255 55 EXIT XSNPT
256 56 ELSE
257 57 :ERROR1: CALL XRMG - 'SYNTAX ERROR - MISSING OR EXTRANEOUS FIELD'
CD0 FORTHAN CALLING PROCEEDURE
CD0 CALL XSENT
CD0
CD0
CD0
CD0
CD0
CD0
CD0
CD0 XSENT CONSTRUCTS SEQUENCE TABLE ENTRIES
CD0 COMMON XE - COMBUF, COMPR, TOKENS, LU
CD0 COMMON XB - DEBUG, LIBD2, NAMENT, PRMTMD,
             _SEGNO, TABNDX, WBUF
CD0
CD0
CD0
CD0 OUTPUT
CD0 COMMON XE - COMPR
CD0 COMMON XB - INTNAM, NUMENT, PRNAM, PRMTMD,
            SEGNO, TABNDX, WBUF
CD0
CD0
CD0 NOTES
CD0 USES ROUTINES
CD0 XRCPR
CD0 XREXT
CD0 XRMV
CD0 XRM5
CD0 XRESET
CD0 XRICK
CD0 XUEGG
CD0
CD0 BEGIN XSENT
CD0 1 ERRXIT IF TOKEN INPUT IS NOT A NAME :ERO01:
CD0 3 RETAIN THIS NAME AS PROC. NAME
CD0 3 INCREMENT TO NEXT TOKEN
CD0 3 START SEARCH UNTIL ALL ENTRIES OF XLIBD SEARCHED
CD0 3 EXIT IF XLIBD ENTRY = PROC. NAME
CD0 3 ORELSE
CD0 4 INCREMENT TO NEXT ENTRY
CD0 3 ENDCOMP
CD0 3 ERRXIT :ERO03:
CD0 4 ENDSEARCH
CD0 3 IF COMMA IS NEXT TOKEN, THEN
CD0 3 ERRXIT IF INT. TABLE NOT REQUIRED FOR THIS PROCESSOR :ERO04:
CD0 3 increment to next token
CD0 3 ERRXIT IF NEXT TOKEN IS NOT A NAME :ERO01:
CD0 3 RETAIN THIS NAME AS INTERFACE TABLE NAME
CD0 3 INCREMENT TO NEXT TOKEN
CD0 3 ELSE
CD0 4 SET INTERFACE TABLE NAME TO ZERO
CD0 3 EXIT
322 3  ENDIF
323 3  ERREXIT IF NEXT Token IS NOT EGs :ERROR1:
324 3  IF INSERT FLAG DOES NOT INDICATE REPLACE, THEN (I.E. INSERT OR ADD)
325 4  IF NUMBER OF TABLE ENTRIES (NUMENT) = 150, THEN
326 5  CALL XSPAck TO PACK TABLE BUFFER (remove DELETED ENTRIES)
327 5  IF NUMBER OF TABLE ENTRIES STILL = 150, THEN
328 6  SET PROMPT MODE TO UPDATE
329 6  ERREXIT :ERROR5:
330 5  ENDIF
331 4  ENDIF
332 4  IF INSERT FLAG INDICATES INSERT (=1), THEN
333 5  IF ENTRY ABOVE INDEXED ENTRY (TABNDX) IS MARKED DELETED, THEN
334 6  SET TABLE ENTRY INDEX (TABNDX) TO BE THIS DELETED ENTRY
335 6  SET INSERT FLAG TO O INDICATING ENTRY REPLACEMENT
336 5  ELSE
337 6  SET MOVLEN = MIN (5, 150-NUMENT) + 7
338 7  DO FOR ALL TABLE ENTRIES FROM BOTTOM OF TABLE TO TABNDX
339 7  MOVE THE ENTRY DOWN MOVLEN WORDS
340 6  ENDDO
341 6  IF MOVLEN > 7 (I.E. MORE THAN 1 ENTRY), THEN
342 7  MARK FOLLOWING ENTRIES AS DELETED
343 6  ENDF1
344 5  ENDIF
345 4  ENDIF
346 4  SET SEQUENCE NO. FIELD OF ENTRY TO SEQUENCE NO. (SEQRno) INPUT/PROMPTED
347 3  ENDF1
348 3  MOVE PROC. NAME AND INT. NAME INTO TABLE ENTRY AT TABNDX
349 3  IF INSERT FLAG DOES NOT INDICATE ENTRY REPLACEMENT, THEN
350 4  INCREMENT NUMBER OF ENTRIES (NUMENT) BY 1
351 3  ENDF1
352 2  EXIT XSENT
353 3  :ERROR1: CALL XRSN3 - 'SYNTAX ERROR - MISSING OR EXTRANEous FIELD'
354 3  :ERROR3: CALL XRSN3 - ' ... IS NOT A VALID PROCESSOR NAME'
355 3  :ERROR4: CALL XRSN3 - ' ... DOES NOT USE AN INTERFACE TABLE'
356 3  :ERROR5: CALL XRSN3 - 'MAX. SIZE OF 150 SEQUENCE ENTRIES ALREADY REACHED'
357 2  END XSENT
CD0  FORTRAN CALLING PROCEDURE
CD0
CD0  CALL XSPCK
CD0
CD0  **********
CD1  CD1  XSPCK COMPACTS THE WORKING BUFFER BY REMOVING ALL SEQUENCE
CD1  CD1  TABLE ENTRIES MARKED FOR DELETION
CD1
CD1  **********
CD2  CD2  INPUT
CD2
CD2  CD2  COMMON K - LU
CD2
CD2  CD2  COMMON X - DEBUG, NUMENT, TABNOX, WBUF
CD2
CD2  **********
CD3  CD3  OUTPUT
CD3
CD3  CD3  COMMON X - NUMENT, TABNOX, WBUF
CD3
CD3  **********
CD4  CD5  NOTES
CD5
CD5  CD5  USES ROUTINES
CD5
CD5  CD5  XRNOW
CD5  CD5  XRMSSG
CD5  CD5  XUDOS
CD5
CD5  **********
BEGIN XSPCK

395 2 IF THE TABLE IS NOT EMPTY, THEN
396 3 DO UNTIL NUMBER OF ENTRIES (NUMENT) PROCESSED
397 4 IF THIS ENTRY IS MARKED DELETED, THEN
398 5 SET MOVE LENGTH (MOVLEN) TO 7
399 6 DO UNTIL A NON-DELETED ENTRY IS FOUND
400 7 INCREMENT MOVLEN BY 7
401 8 ENDDO
402 9 DO
403 10 MOVE MOVLEN WORDS BEGINNING WITH THE NON-DELETED ENTRY TO
404 11 THE DELETED ENTRY
405 12 DECREMENT NUMENT BY MOVLEN/7
406 13 IF TABLE INDEX (TABNOX) > INDEX TO DELETED ENTRY, THEN
407 14 DECREMENT TABLE INDEX (TABNOX) BY MOVLEN
408 15 ENDF
409 16 ENDDO
410 17 ENDF
411 18 END XSPCK
*FORTRAN CALLING PROCEDURE*

**CD0**

**CD0** CALL XSLST

---

**CD1**

**CD1** XSLST WILL LIST TO A SPECIFIED DEVICE THE SEQUENCE TABLE CONTAINED IN THE WORKING BUFFER

---

**CD2**

**CD2** INPUT

**CD2** COMMON XE - LU, REBUFF, SUBSTA

**CD2** COMMON XB - BEGNO, DEBUG, ENDNO, LISTLU,

**CD2** NEWTAB, NUMENT, WKBUFF

---

**CD3**

**CD3** NOTES

**CD3** USES Routines

**CD3** XRIQ

**CD3** XRMQ

**CD3** XRSET

**CD3** XUDBG

---

**CD5**

**CD5** BEGIN XSLST

**CD5** IF SUBSTATE FLAG INDICATES THAT SEQ. EDITOR NOT MAKING THIS CALL, THEN

**CD5** DETERMINE SIZE OF SEQ. TAB FROM AWA REQUEST BUFFER ENTRY

**CD5** SET LIMITS (BEGNO AND ENDNO) OF SEQ. ENTRIES LISTED

**CD5** SET TABLE NAME (NEWTAB) FROM AWA REQUEST BUFFER ENTRY

**CD5** ENDIF

**CD5** WRITE HEADER LINE - 'SEQUENCE TABLE XXXXX'

**CD5** IF SEQ. TABLE ENTRY IS NOT MARKED AS DELETED, THEN

**CD5** DO FROM BEGNO TO ENDNO

**CD5** MOVE PROC. NAME AND INT. NAME FROM ENTRY TO PRINT BUFFER

**CD5** IF INT. TABLE NAME = 0, THEN

**CD5** SET LENGTH OF PRINT TO BE 7 WORDS (14 CHARs.)

**CD5** ELSE

**CD5** SET LENGTH OF PRINT LINE TO BE 10 WORDS (20 CHARs.)

**CD5** ENDF

**CD5** CALL XRIS TO CONVERT SEQ. NO. OF TABLE ENTRY AND PLACE IN BUFFER

**CD5** WRITE PRINT BUFFER

**CD5** ENDF

**CD5** END XSLST
466 2 CD0 FORTRAN CALLING PROCEDURE
467 2 CD0
468 2 CD0 CALL XSCAN
469 2 CD0
470 2 CD0
471 2 CD0
472 2 CD0 XSCAN PERFORMS SYNTACTICAL PROCESSING FOR THE LIST AND
473 2 CD0 DELETE DIRECTIVES OF THE SEQUENCE TABLE EDITOR
474 2 CD0
475 2 CD0
476 2 CD0
477 2 CD0 INPUT
478 2 CD0
479 2 CD0
480 2 CD0 COMMON XE - COMBUF, COMPTA, LU, TOKENS
481 2 CD0
482 2 CD0 COMMON XB - DEBUG, NUMENT, TAPSIZ, WBUF
483 2 CD0
484 2 CD0
485 2 CD0
486 2 CD0 OUTPUT
487 2 CD0
488 2 CD0 COMMON XB - BEGNO, ENDMO, IRETC
489 2 CD0
490 2 CD0
491 2 CD0 NOTES
492 2 CD0
493 2 CD0
494 2 CD0 USES ROUTINES
495 2 CD0
496 2 CD0 XERMSG
497 2 CD0
498 2 CD0
499 2 CD0
BEGIN XSCAN
SET LIST LIMITS (BEGINO AND ENDMO) TO ZERO
IF NEXT TOKEN IS A COMMA, THEN
INCREMENT TO NEXT TOKEN
IF NEXT TOKEN IS AN INTEGER, THEN
ERREXIT IF VALUE IS < 1:ERRO6:
SET BEGIN LIMIT (BEGINO) TO THIS VALUE
INCREMENT TO NEXT TOKEN
ENDIF
IF NEXT TOKEN IS A COMMA, THEN
INCREMENT TO NEXT TOKEN
IF NEXT TOKEN IS AN INTEGER, THEN
ERREXIT IF VALUE IS < 1:ERRO6:
SET END LIMIT (ENDMO) TO THIS VALUE
INCREMENT TO NEXT TOKEN
ENDIF
ENDIF
END
IF NEXT TOKEN IS NOT EOS:ERRO1:
BEGIN LIMIT (BEGINO) TO 1 (BEGINO IS NOW A TABLE INDEX)
ELSE
START SEARCH FROM FIRST TO LAST SEQ. TABLE ENTRY
EXIT IF SEQ. NO. OF THIS ENTRY = BEGIN LIMIT (BEGINO)
SET BEGIN LIMIT (BEGINO) TO INDEX OF THIS ENTRY
ORELSE
INCREMENT INDEX TO NEXT TABLE ENTRY
ENDLOOP
ERREXIT:ERRO6:
END SEARCH
IF END LIMIT (ENDMO) = 0, THEN
SET END LIMIT (ENDMO) TO INDEX OF LAST TABLE ENTRY
ELSE
START SEARCH FROM BEGIN LIMIT (BEGINO) TO LAST TABLE ENTRY
EXIT IF SEQ. NO. OF THIS ENTRY = END LIMIT (ENDMO)
SET END LIMIT (ENDMO) TO INDEX OF THIS ENTRY
ORELSE
INCREMENT INDEX TO NEXT TABLE ENTRY
ENDLOOP
ERREXIT:ERRO6:
END SEARCH
END
IF RETURN CODE TO INDICATE NO ERROR
SET RETURN CODE TO INDICATE AN ERROR
EXIT XSLIE
:ERRO1: CALL XRMSG - "SYNTAX ERROR - MISSING OR EXTRANEOUS FIELD"
:ERRO6: CALL XRMSG - "INVALID SEQUENCE NUMBER"
:ERRO8: CALL XRMSG - "INVALID SEQUENCE NUMBER RANGE"
END XSCAN
FORTRAN CALLING PROCEDURE

CALL XSPT

**********

XSPT PROCESSES THE SEQUENCE TABLE EDITOR PROMPT DIRECTIVE

**********

INPUT

COMMON AE - COMBUF, COMPTR, LU, TOKENS

COMMON XB - DEBUG

**********

OUTPUT

COMMON XB - PRNTMD, TABN0X

**********

NOTES

USES ROUTINES

XRMSG

XUDSG

**********

BEGIN XSPT

ERREXIT IF TOKEN IS NOT COMMA :ERROR1:

INCREMENT TO NEXT TOKEN

ERREXIT IF TOKEN IS NOT A NAME :ERROR1:

INCREMENT TO NEXT TOKEN

ERREXIT IF TOKEN IS NOT EOS :ERROR1:

IF NAME IS 'M', THEN

SET PROMPT MODE TO CREATE

ELSE

ERREXIT IF NAME IS NOT 'A':ERROR9:

SET PROMPT MODE TO ALL

SET TABLE ENTRY INDEX (TABN0X) TO 0

END IF

EXIT XSPT

:ERROR1: CALL XRMSG - 'SYNTAX ERROR - MISSING OR EXTRANEOUS FIELD'

:ERROR9: CALL XRMSG - 'SYNTAX ERROR - INVALID QUALIFIER'

END XSPT
602 2 CDO FORTRAN CALLING PROCEDURE
603 2 CDO
604 2 CDO CALL XSDEL
605 2 CDO
606 2 CDO ********
607 2 CDO
608 2 CDO XSDEL IS THE SEQUENCE TABLE EDITOR DELETE DIRECTIVE PROCESSOR
609 2 CDO
610 2 CDO ********
611 2 CDO
612 2 CDO INPUT
613 2 CDO
614 2 CDO COMMON XE - LU
615 2 CDO
616 2 CDO COMMON XB - BEGNO, DEBUG, ENDMN, IRETC, NUMENT
617 2 CDO
618 2 CDO
619 2 CDO ********
620 2 CDO OUTPUT
621 2 CDO
622 2 CDO COMMON XB - NUMENT, WKBUF
623 2 CDO
624 2 CDO ********
625 2 CDO NOTES
626 2 CDO
627 2 CDO USES ROUTINES
628 2 CDO
629 2 CDO XSCAN
630 2 CDO
631 2 CDO XUXDG
632 2 CDO
633 2 CDO
634 2 CDO ********
635 2 CDO BEGIN XSDEL
636 3 CDO CALL XSCAN TO SCAN AND INTERPRET SEQ. LIMITS ON THE DIRECTIVE
637 3 CDO IF NO ERROR INDICATED, THEN
638 4 CDO DO FROM THE BEGIN LIMIT (BEGNO) TO THE END LIMIT (ENDMN)
639 5 CDO MARK THIS SEQ. TABLE ENTRY AS DELETED
640 4 CDO ENDDO
641 4 CDO DO FROM LAST TABLE ENTRY TO FIRST ENTRY, OR
642 5 CDO UNTIL A NON-DELETED ENTRY FOUND
643 5 CDO IF ENTRY IS MARKED DELETED, THEN
644 6 CDO DECREMENT NUMBER OF TABLE ENTRIES (NUMENT) BY ONE
645 5 CDO ENDDO
646 4 CDO ENDDO
647 3 ENDDO
648 2 ENDSDEL
FORTRAN CALLING PROCEDURE

CALL XSLIS

*****

XSLIS IS THE SEQUENCE TABLE EDITOR LIST DIRECTIVE PROCESSOR

*****

INPUT

COMMON XE - LU

COMMON XB - DEBUG, ETC

*****

NOTES

USES ROUTINES

XICAN

XSLST

XUDGS

*****

FORTRAN CALLING PROCEDURE

CALL X$WNUM

*****

X$WNUM IS THE SEQUENCE TABLE EDITOR NUMBER DIRECTIVE PROCESSOR

*****

INPUT

COMMON XE - COMBUF, COMPTA, LU, TOKENS

COMMON XB - DEBUG, NUMEB

*****

OUTPUT

COMMON XB - WKBUF

*****

NOTES

USES ROUTINES

XRMSE

XUDGS
2       CD5
2       CD5
116     BEGIN XSMUM
118     IF TOKEN IS NOT EOS :ERROR:
119     IF NUMBER OF TABLE ENTRIES (NUMENT) > 0, THEN
120     SET SEQUENCE NUMBER (SENO) TO 100
121     DO FOR ALL ENTRIES IN TABLE
122     IF TABLE ENTRY IS NOT MARKED DELETED, THEN
123     SET SEQ. NO. FIELD OF ENTRY TO SEQUENCE NUMBER (SENO)
124     INCREMENT SEQUENCE NUMBER (SENO) BY 100
125     ENDIF
126     ENDDO
127     EXIT XSMUM
129     :ERROR: CALL XRMSG - "SYNTAX ERROR - MISSING OR EXTRANEOUS FIELD"
130     END XSMUM
**FORTRAN CALLING PROCEDURE FOR TERMINAL COMMUNICATIONS:**

**CALL XICOM (PROMPT, PRMLEN, RETCOD)**

**INPUTS FROM CALLING SEQUENCE:**

- PROMPT - (INTEGER, PRMLEN WORDS) an array of PRMLEN words used as a prompt to the user.
- PRMLEN - (INTEGER, 1 WORD) the length in words of the prompt array. Maximum length is 38 words.

**OUTPUTS IN CALLING SEQUENCE:**

- RETCOD - (INTEGER, 1 WORD) is a completion code defined as follows:
  0 - normal return, buffer contains response
  1 - user responded as buffer contains response up to and including 1.
  2 - user entered a CR. there is no response
  3 - prompt was too long, maximum length is 76
  5 - user requested a continuation

**INTERNAL VARIABLES**

- COMM - an array containing the continuation message
- PREFIX - 4 characters used as prefixes to prompt.
- CORRESPONDING TO CODES IN XE(5)
- RETCIP - return code from XIPRM extended prompting
- RETCLA - return code from XICLAL lexical analysis
- SUFFIX - 1 character appended to end of prompt

**XE COMMON USED:**

.equivalence (IXE(1), LU), (IXE(2), ILCLASS)

**XS COMMON USED:**
EQUIVALENCE (XS(1), INBUF), (XS(81), NOIN),
+ (XS(82), OUTBUF), (XS(122), SSFLAG),
+ (XS(123), IMLEN)

INBUF - 80 WORD ARRAY THAT CONTAINS THE USER'S RESPONSE
IN R1 FORMAT
IMLEN - LENGTH OF ORIGINAL PROMPT PASSED TO XTPRM
NOIN - NUMBER OF WORDS OF INBUF ACTUALLY USED FOR THE
USER'S RESPONSE
OUTBUF - 40 WORD AREA WHERE PROMPT IS CONSTRUCTED AND
WRITTEN FROM.
SSFLAG - SYMBOLIC STRING INDICATOR FLAG
0 = NOT WITHIN A SYMBOLIC STRING
MZERO = VALUE REPRESENTING THE INDEX INTO COMBUF
WHERE LENGTH OF SS IS TO BE STORED

SUBROUTINES AND FUNCTIONS CALLED:
EXEC, KCVT, XRMV, XRMSG, XINUPK, XTLAN, XTPRM

PDL ROUTINES INCLUDED:
XTCOM, READSEG
87 1 BEGIN XTCOM
88 2 IF PROMPT IS NOT TOO LONG (76 CHAR) THEN
89 3 MOVE PREFIX CHARACTER FOR EXEC LEVEL INTO OUTPUT AREA
90 4 MOVE PROMPT INTO OUTPUT AREA
91 5 MOVE SUFFIX CHARACTER INTO OUTPUT AREA
92 6 :LOOP:
93 7 ISSUE WRITE TO PROMPT USER
94 8 INITIALIZE COMMUNICATIONS BUFFER
95 9 TURN SYMBOLIC STRING FLAG OFF
96 10 INITIALIZE LA RETURN CODE TO NORMAL RETURN
97 11 PERFORM READSEG TO READ INPUT AND CALL LEXICAL ANALYSIS
98 12 DO WHILE LEXICAL ANALYSIS (LA) RETURN CODE SAYS CONTINUE AND
99 13 (EXEC LEVEL IS NOT INTERFACE TABLE EDITOR OR
100 14 SYMBOLIC STRING FLAG IS ON)
101 15 CALL EXEC TO WRITE CONTINUATION MESSAGE
102 16 PERFORM READSEG TO READ INPUT AND CALL LEXICAL ANALYSIS
103 17 ENDDO
104 18 IF LA RETURN CODE SAYS ERROR IN RESPONSE THEN
105 19 CALL KCVT TO CONVERT OCTAL TO ASCII
106 20 CALL XRMG TO WRITE ERROR MESSAGE
107 21 GO TO :LOOP: TO DISPLAY ORIGINAL PROMPT
108 22 ENDFIF
109 23 IF LA RETURN CODE SAYS OVERFLOW/UNDERFLOW THEN
110 24 CALL KCVT TO CONVERT OCTAL TO ASCII
111 25 CALL XRMG TO WRITE ERROR MESSAGE
112 26 GO TO :LOOP: TO DISPLAY ORIGINAL PROMPT
113 27 ENDFIF
114 28 IF LA RETURN CODE SAYS EXTENDED PROMPTING WAS REQUESTED THEN
115 29 CALL XTPM FOR EXTENDED PROMPT
116 30 IF EXTENDED PROMPT (EP) RETURN CODE SAYS INVALID REQUEST THEN
117 31 CALL XRMG TO WRITE ERROR MESSAGE
118 32 ENDFIF
119 33 GO TO :LOOP: TO DISPLAY ORIGINAL PROMPT
120 34 ENDFIF
121 35 IF LA RETURN CODE SAYS COMPUT IS FULL THEN
122 36 CALL XRMG TO WRITE ERROR MESSAGE
123 37 GO TO :LOOP: TO DISPLAY ORIGINAL PROMPT
124 38 ENDFIFI
125 39 SET XTCOM RETURN CODE = LA RETURN CODE
126 40 ELSE
127 41 SET XTCOM RETURN CODE = PROMPT IS TOO LONG
128 42 ENDFIFI
129 43 END XTCOM
**FORTRAN CALLING PROCEDURE FOR LEXICAL ANALYSIS:**

```
CALL XLAM (RETC)
```

**OUTPUTS IN CALLING SEQUENCE:**

RETC - (INTEGER, 1 WORD) IS A COMPLETION CODE PASSED
BACK TO CALLER AS FOLLOWS:

- 0 - NORMAL RETURN. BUFFER CONTAINS RESPONSE.
- 1 - USER RESPONDED Y. BUFFER CONTAINS RESPONSE
UP TO AND INCLUDING X.
- 5 - USER REQUESTED A CONTINUATION.
- 6 - EXTENDED PROMPTING REQUEST WAS RECEIVED. BUFFER
CONTAINS RESPONSE UP TO AND INCLUDING THE REQUEST.
- 7 - COMMUNICATIONS BUFFER IS FULL.
- 1X - ERROR IN RESPONSE AT OR BEYOND CHARACTER XX.
- 2X - OVERFLOW/UNDERFLOW DETECTED AT OR BEYOND
CHARACTER XX.

**INTERNAL VARIABLES**

- COMLEN - LENGTH IN WORDS OF COMBUF =256
- DBLINT - DOUBLE PRECISION LOCATION TO ACCUMULATE AN
INTEGER VALUE
- DBLFD - DOUBLE PRECISION LOCATION TO ACCUMULATE AN INTEGER
AND FRACTIONAL VALUE FOR DOUBLE PRECISION OR REM.
- FLGCOM - COMMA FLAG
- 0 - LAST CHARACTER WAS NOT A COMMA
- 1 - LAST CHARACTER WAS A COMMA
- FLGEND - END LOOP FLAG
- 0 - CONTINUE LOOP
- 1 - END LOOP
- FLGNCE - NEGATIVE EXPONENT FLAG
- 0 - EXPONENT WAS POSITIVE
- 1 - EXPONENT WAS NEGATIVE
- FLGTPR - TYPE OF REAL VALUE
- 0 - SINGLE PRECISION
- 1 - DOUBLE PRECISION
- POWER - EXPONENT PART OF A REAL NUMBER
- RELFD - SINGLE PRECISION LOCATION FOR REAL VALUE
- SCFHR - 25 SPECIAL CHARACTER ARRAY CONTAINING
THE HI FORMAT REPRESENTATION FOR:

```
"-*/@#&=?!%^X5.YXR..: DEW
X IS A CLOSED BRACKET
Y IS AN OPEN Bracket
2 IS A BACK SLASH
```
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>206</td>
<td>CD4</td>
</tr>
<tr>
<td>207</td>
<td>CD5</td>
</tr>
<tr>
<td>208</td>
<td>CD5</td>
</tr>
<tr>
<td>209</td>
<td>CD5</td>
</tr>
<tr>
<td>210</td>
<td>CD5</td>
</tr>
<tr>
<td>211</td>
<td>CD5</td>
</tr>
<tr>
<td>212</td>
<td>CD5</td>
</tr>
<tr>
<td>213</td>
<td>CD5</td>
</tr>
<tr>
<td>214</td>
<td>CD5</td>
</tr>
<tr>
<td>215</td>
<td>CD5</td>
</tr>
<tr>
<td>216</td>
<td>CD5</td>
</tr>
<tr>
<td>217</td>
<td>CD5</td>
</tr>
<tr>
<td>218</td>
<td>CD5</td>
</tr>
<tr>
<td>219</td>
<td>CD5</td>
</tr>
<tr>
<td>220</td>
<td>CD5</td>
</tr>
<tr>
<td>221</td>
<td>CD5</td>
</tr>
<tr>
<td>222</td>
<td>CD5</td>
</tr>
<tr>
<td>223</td>
<td>CD5</td>
</tr>
<tr>
<td>224</td>
<td>CD5</td>
</tr>
<tr>
<td>225</td>
<td>CD5</td>
</tr>
<tr>
<td>226</td>
<td>CD5</td>
</tr>
<tr>
<td>227</td>
<td>CD5</td>
</tr>
<tr>
<td>228</td>
<td>CD5</td>
</tr>
</tbody>
</table>

**XE COMMON USED:**

- EQUIVALENCE (XE(85), TOKENS), (XE(145), COMBUF), (XE(145), NOTOK), (XE(145), NOWD3).

**XS COMMON USED**

- EQUIVALENCE (XS(1), INBUF), (XS(81), NOCHAR), (XS(122), FLAGSS), (XS(186), SCRATCH)

**SUBROUTINES AND FUNCTIONS CALLED**

- XRPCK, XRMOV

**PDL ROUTINES INCLUDED:**

- XITLM, COMMA, ALPHA, DIGIT, DCM, DECP, EORD, INTEGRA, REAL, DBL, REPET, INVAL, SCHARS, QUOTE
BEGIN XILAN
SET END FLAG OFF
SET LAST CHARACTER WAS A COMMA ON
INITIALIZE INDEX INTO INPUT BUFFER
INITIALIZE RETURN CODE TO NORMAL RETURN
DO WHILE END FLAG IS OFF
  IF INPUT BUFFER HAS BEEN COMPLETELY SCANNED THEN
    IF LAST CHARACTER WAS A COMMA OR A SYMBOLIC STRING IS STILL OPEN THEN
      SET RETURN CODE TO SAY CONTINUATION REQUESTED
    ENDIF
  SET END FLAG ON
ELSE
  ERREXIT IF COMBUF IS FULL PERFORM COMFUL
  IF INPUT CHARACTER IS A COMMA THEN
    PERFORM COMPA
  ELSE
    SET LAST CHARACTER WAS A COMMA OFF
    IF INPUT CHARACTER IS AN ALPHABETIC CHARACTER THEN
      PERFORM ALPHA
    ELSE
      IF INPUT CHARACTER IS A DIGIT THEN
        PERFORM DIGIT
      ELSE
        PERFORM SCHARS
    ENDIF
  ENDIF
ENDIF
ENDIF
ENDDO
STORE END OF BUFFER TOKEN IN COMBUF
INCREMENT #TOKENS BY 1
END XILAN
1
*
*
*
BEGIN COMMA
SET LAST CHARACTER WAS A COMMA ON
ERREXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM COMFUL
STORE COMMA TOKEN IN COMBUF
INCREMENT #WORDS IN COMBUF BY 1
INCREMENT #TOKENS BY 1
GET NEXT INPUT CHARACTER
END COMMA

1 BEGIN DCOL
2   SET INTEGER = 0
3   SET COUNTER = 0
4   DO WHILE CHARACTER IS A DIGIT AND
5     INPUT BUFFER IS NOT EXHAUSTED
6     SET INTEGER = (INTEGER + 10) * CURRENT CHARACTER - 48
7     EXIT IF OVERFLOW/UNDERFLOW IS DETECTED PERFORM OVFLOW
8     INCREMENT COUNTER BY 1
9     GET NEXT CHARACTER
10   END DO
11 END DCOL

1 BEGIN DECPT
2   CONVERT INTEGER VALUE TO DOUBLE PRECISION VALUE
3   GET NEXT INPUT CHARACTER
4   IF INPUT BUFFER IS NOT EXHAUSTED THEN
5     IF INPUT CHARACTER IS A DIGIT THEN
6       PERFORM DCOL
7       ADD FRACTIONAL PART TO DOUBLE PRECISION VALUE
8       EXIT IF OVERFLOW/UNDERFLOW IS DETECTED PERFORM OVFLOW
9     ENDIF
10    IF INPUT CHARACTER IS AN "E" OR A "D" THEN
11       PERFORM EORD
12     ELSE
13       PERFORM REAL
14     ENDIF
15     ELSE
16       PERFORM REAL
17     ENDIF
18   ENDIF
19 END DECPT
IF INPUT CHARACTER IS AN "E" THEN
SET TYPE FLAG TO "E"
ELSE
SET TYPE FLAG TO "D"
ENDIF
GET NEXT CHARACTER
ERREXIT IF INPUT BUFFER IS EXHAUSTED
SET NEGATIVE FLAG OFF
IF CHARACTER IS A - THEN
SET NEGATIVE FLAG ON
GET NEXT CHARACTER
ELSE
IF CHARACTER IS A + THEN
GET NEXT CHARACTER
ENDIF
ENDIF
ERREXIT IF INPUT BUFFER IS EXHAUSTED OR
ERREXIT IF CHARACTER IS NOT A DIGIT PERFORM INVAL.
PERFORM BCOL
IF NEGATIVE FLAG IS ON THEN
SET POWER = -POWER
ENDIF
IF TYPE FLAG IS "E" THEN
PERFORM REAL
ELSE
PERFORM DBL
ENDIF
1 END EORD
1 BEGIN INTEGER
2 ERREXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM CONFUL
3 CONVERT NUMBER TO INTEGER
4 ERREXIT IF OVERFLOW/UNDERFLOW IS DETECTED PERFORM OVFLOW
5 STORE INTEGER TOKEN IN COMBUF
6 INCREMENT WORDS IN COMBUF BY 2
7 INCREMENT TOKENS BY 1
8 END INTEGER

10 BEGIN REAL
12 ERREXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM CONFUL
13 SET REAL = DOUBLE PRECISION * 10 ** POWER
14 ERREXIT IF OVERFLOW/UNDERFLOW IS DETECTED PERFORM OVFLOW
15 STORE REAL TOKEN IN COMBUF
16 INCREMENT WORDS IN COMBUF BY 3
17 INCREMENT TOKENS BY 1
18 END REAL

20 BEGIN DBL
22 ERREXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM CONFUL
23 SET DOUBLE = DOUBLE PRECISION * 10 ** POWER
24 ERREXIT IF OVERFLOW/UNDERFLOW IS DETECTED PERFORM OVFLOW
25 STORE DOUBLE TOKEN IN COMBUF
26 INCREMENT WORDS IN COMBUF BY 4
27 INCREMENT TOKENS BY 1
28 END DBL

30 BEGIN REPEP
32 ERREXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM CONFUL
34 ERREXIT IF NUMBER OF TIMES TO REPEAT IS ZERO PERFORM INVAL
35 STORE REPEAT TOKEN IN COMBUF
36 INCREMENT WORDS IN COMBUF BY 2
37 INCREMENT TOKENS BY 1
38 GET NEXT CHARACTER
39 END REPEP
1 BEGIN SCARS
2    CHARACTER TABLE:
3          "-#&<>*=?()'*"NXYM;
4          NH ARE INVALID CHARACTERS HERE
5          X IS A CLOSED BRACKET
6          Y IS AN OPEN BRACKET
7          Z IS A BACK SLASH
8    SET J=1
9    START SEARCH WHILE J<#CHARACTERS IN TABLE
10   EXIT IF INPUT CHARACTER MATCHES CHARACTER (J) IN TABLE
11   INCREMENT J BY 1
12   ENDLOOP
13   PERFORM INVAL - NO RETURN
14   ENDSEARCH
15   SET NEXT FLAG ON
16   CASE J:(A:,F:,F:,F:,F:,F:,F:,F:,F:,B:)
17         *( ) = 1 S Y X N N
18         (::F:,F:,C:,C:,E:,B:,F:,E:,F:,F:,F:,F:,F:,INVAL:,INVAL:,F:)
19    :A:
20    SET NEXT FLAG OFF
21    PERFORM QUOTE
22    :B:
23    SET RETURN CODE TO SAY EXTENDED PROMPT REQUESTED
24    SET END FLAG ON
25    :C:
26    IF SYMBOLIC STRING FLAG IS OFF THEN
27       SET SYMBOLIC STRING FLAG TO CURRENT COMBUF INDEX + 1
28    ELSE
29       SET COMBUF SYMBOLIC STRING FLAG = CURRENT COMBUF INDEX - SYMBOLIC STRING FLAG
30    ENDIF
31    SET J = J+1 TO STORE SYMBOLIC STRING CLOSE TOKEN
32    SET SYMBOLIC STRING FLAG OFF
33    ENDIF
34    :D:
35    SET RETURN CODE TO SAY X ENTERED
36    SET END FLAG ON
37    :E:
38    IF FOLLOWING CHARACTER IS A DIGIT THEN
39       SET NEXT FLAG OFF
40       SET INTEGER = 0
41       SET POWER = 0
42       PERFORM DECPT
43       ENDIF
44    :F:
45    ENDCASE
46    IF NEXT FLAG IS ON
47       STORE TOKEN (J) IN COMBUF
48       INCREMENT #WORDS IN COMBUF BY 1
49       IF TOKEN IS BEGIN SYMBOLIC STRING THEN
50          INCREMENT #WORDS IN COMBUF BY 1
51          ENDIF
52       INC #TOKENS BY 1
53       GET NEXT CHARACTER
54 ENDSCARS
1 BEGIN QUOTE
2 GET NEXT CHARACTER
3 SET #CHARACTERS = 0
4 DO WHILE (INPUT CHARACTER IS NOT A QUOTE AND
5 INPUT BUFFER HAS NOT BEEN COMPLETELY SCANNED) OR
6 (INPUT CHARACTER IS A QUOTE AND
7 NEXT CHARACTER IS A QUOTE AND
8 INPUT BUFFER HAS NOT BEEN COMPLETELY SCANNED)
9 INCREMENT #CHARACTERS BY 1
10 IF INPUT CHARACTER IS A QUOTE THEN
11 GET NEXT CHARACTER
12 ENDF
13 GET NEXT CHARACTER
14 ENDDO
15 EXIT IF THERE IS NO ROOM IN COMBUF FOR THIS TOKEN PERFORM COMFUL
16 EXIT IF LENGTH OF CHARACTER STRING IS 0 OR
17 EXIT IF INPUT CHARACTER IS NOT A QUOTE PERFORM INVALID
18 STORE CHARACTER STRING TOKEN IN COMBUF
19 CALL XRPOK TO PACK CHARACTERS INTO COMBUF
20 INCREMENT #WORDS IN COMBUF BY 2 + (#CHARACTERS + 1)/2
21 INCREMENT #TOKENS BY 1
22 GET NEXT CHARACTER
23 1 END QUOTE
503 1 BEGIN INVAL
504 2 SET RETURN CODE TO SAY INVALID RESPONSE
505 2 SET END FLAG ON
506 1 EXIT XILAN
507 1 END INVAL
508 1 *
509 1 *
510 1 *
511 1 BEGIN COMFUL
512 2 SET RETURN CODE TO SAY COMBUF IS FULL
513 2 SET END FLAG ON
514 1 EXIT XILAN
515 1 END COMFUL
516 1 *
517 1 *
518 1 *
519 1 BEGIN OVFLOW
520 2 SET RETURN CODE TO SAY OVFLOW/UNDERFLOW
521 2 SET END FLAG ON
522 1 EXIT XILAN
523 1 END OVFLOW
FORTAN CALLING PROCEDURE

CDO CALL XTPRM

CDO XTPRM IS CALLED BY XTCOM TO PROVIDE EXTENDED PROMPTING WHEN A ? IS ENTERED FROM THE USER TERMINAL.

CDO

CDO INPUT

CDO XE COMMON - CARTRG, COMBUF, FLAGS, LU, NOPROC, PRCKAM, SUBSTA, TOKENS

CDO XB COMMON - AREPTR, WBUF (FROM INTERFACE TABLE EDITOR)

CDO XS COMMON - PRML (FROM XTCOM)

CDO VARIOUS FDS PROMPT FILES (SEE INTERNAL VARIABLE TABLE)

CDO LISTING OF APPROPRIATE EXTENDED PROMPTS

CDO

CDO INTERNAL VARIABLES

CDO CONTIN - CONTINUATION INDICATOR (1) FOR CURRENT TABLE ENTRY

CDO FILE - FILE NAME OF CURRENT TABLE ENTRY

CDO INDEX - INDEX TO CURRENT TABLE ENTRY

CDO L - RECORD NUMBER OF TEXT OR SYNTAX BLOCK CORRESPONDING TO FIRST LIST ITEM IN RECORD (1) (SEE SDD SECTION 6.2.4.3)

CDO MESSAGE - MESSAGE NUMBER, IF ANY, FOR CURRENT TABLE ENTRY

CDO N - NUMBER OF LIST ITEMS IN RECORDS (1) AND (2) (SEE SDD SECTION 6.2.4.3)

CDO RECORD - RECORD NUMBER WITH WHICH TO BEGIN PROCESSING FOR CURRENT TABLE ENTRY

CDO SEARCH - LIST SEARCH FLAG (1) FOR CURRENT TABLE ENTRY

CDO SIZE - LIST ITEM SIZE OF CURRENT TABLE ENTRY (NEGATIVE INDICATES LAST CHARACTER TO BE MASKED)

CDO TABLE - PROCESSING CONTROL TABLE FOR VARIOUS SYNTAX CONDITIONS

CDO AS FOLLOWS

CDO ENTRY SYNTAX CONDITION INDEX FILE REC SIZE SRC MESSAGE CONTINUE

CDO 1 I ?

CDO 2 I \?

CDO 3 I #

CDO 4 I $, $?, $?

CDO 5 I 

CDO 6 I $?, $?:, $?:

CDO 7 I /?

CDO 8 I \?

CDO 9 I /?

CDO 10 I ??, ???

CDO WHERE P AND R INDICATE A PROMPT AND RESPONSE
1 BEGIN XTPRM
2 PERFORM SETUP TO COMPLETE CONTROL TABLE AND INDEX TO APPROPRIATE ENTRY
3 DO UNTIL 'NO CONTINUE' FOUND (O IN ENTRY CONTINUE FIELD)
4 IF OPEN SUCCESSFUL, THEN
5   POSITION TO INDICATED STARTING RECORD AND READ
6   IF TABLE SIZE FIELD < 128 (NOT A LIST RECORD), THEN
7     IF SIZE > 0 (NO LAST CHARACTER MASKING & POSSIBLE SPANNING), THEN
8       READ SECOND RECORD AND APPEND TO FIRST RECORD DATA
9     EXIT TO :ERROR9: IF FAILURE
10    ENDIF
11   ELSE
12     SET SIZE POSITIVE
13     DO FOR EACH LIST ITEM (1-M)
14     BLANK LAST CHARACTER
15     ENDDO
16    ENDF
17   IF LIST SEARCHING IS INDICATED (SEARCH FIELD = 1), THEN
18      START SEARCH WHILE LIST ITEMS REMAIN TO BE EXAMINED
19      EXIT IF TOKEN LOCATED IN LIST
20      POSITION TO APPROPRIATE RECORD (I+L-1) AND READ
21      EXIT TO :ERROR9: IF FAILURE
22      IF TABLE SIZE FIELD > 0, THEN
23        CALL XRMG TO DISPLAY 'NOT VALID ...' MESSAGE
24      ENDIF
25      EXIT TO ENDDO
26    ENDS
27   ENDF
28   PERFORM DISPLAY
29   ELSE OPEN ERROR
30   IF FILE NOT FOUND AND TABLE MESSAGE NUMBER FIELD > 0
31      CALL XRMG TO DISPLAY 'NOT VALID ...' MESSAGE
32   ELSE
33     :ERROR9: CALL XRMG TO DISPLAY 'FILE MANAGER ERROR ...' MESSAGE
34    ENDF
35   ENDIF
36 ENDDO
37 END XTPRM
BEGIN SETUP

EXIT TO :ERROR2: IF FIRST TOKEN NOT ? OR NAME FOLLOWED BY ?

CASE (:x:: :x:: :x:: :x://) SUBSTA

:Xi

IF FIRST TOKEN = ?, THEN

SET TABLE INDEX TO FIRST ENTRY

ELSE

SET TABLE INDEX TO SECOND ENTRY

ENDIF

:Si

IF FIRST TOKEN = ?, THEN

SET TABLE INDEX TO FOURTH ENTRY

ELSE

SET TABLE INDEX TO SIXTH ENTRY

FORM SIXTH ENTRY FILE NAME FROM > AND TOKEN

ENDIF

:Fi

IF PROMPT LENGTH = 0, THEN

ASSUME TABLE INDEX OF THIRD ENTRY

ELSE

ASSUME TABLE INDEX OF FOURTH ENTRY

ENDIF

:Ti

IF FIRST TOKEN IS A NAME, THEN

INCREMENT TABLE INDEX BY TWO ENTRIES (NOW FIFTH OR SIXTH)

FORM SIXTH ENTRY FILE NAME FROM > AND TOKEN

ENDIF

:Sf

FORM TENTH ENTRY FILE NAME FROM > AND IT EDITOR CURRENT PROCESSOR NAME

ASSUME TABLE INDEX OF TENTH ENTRY

IF FIRST TOKEN = ?, THEN

IF PROMPT LENGTH = 0, THEN

CHANGE TABLE INDEX TO SEVENTH ENTRY

COPY FILE NAME FROM ENTRY TEN TO ENTRY EIGHT

ELSE

CHANGE TOKEN TO KEYWORD CURRENTLY BEING PROCESSED BY IT EDITOR

ENDIF

ELSE

IF PROMPT LENGTH = 0, THEN

CHANGE TABLE INDEX TO NINETH ENTRY

ENDIF

ENDIF

END CASE

EXIT SETUP

:ERROR2: EXIT XTPRM WITH INVALID REQUEST FOR EXTENDED PROMPTING

END SETUP
FORTRAN CALLING PROCEDURE

CALL XUD8G (I,U, ID)

XUD8G PROVIDES THE PROGRAMMER WITH A CALLABLE INTERACTIVE MEMORY DUMP AND/OR MODIFICATION FACILITY

INPUT (CALLING SEQUENCE)

L - LOGICAL UNIT TO INTERACT WITH IN INVOKING XUD8G OPTIONS

I - THREE WORD ASCII ARRAY USED AS A HEADER TO IDENTIFY XUD8G CALLER

INPUT (INTERACTIVE)

O - ASCII CHARACTER IDENTIFYING XUD8G OPTION

S - SNAP OUT (DUMP) MEMORY

M - MODIFY MEMORY

E - EXIT XUD8G

START: - OCTAL MEMORY ADDRESS IN THE USERS MAP OF FIRST WORD TO BE SNAPPED OR MODIFIED

STOP: - OCTAL MEMORY ADDRESS OF LAST WORD TO BE SNAPPED

OUTPUT UNIT: - LOGICAL UNIT NUMBER TO WHICH SNAP IS TO BE WRITTEN

VALUES: - ARRAY OF OCTAL VALUES TO BE STORED IN MEMORY

BEGINNING AT LOCATION "START", WHEN INPUT "VALUES" MUST NOT EXCEED 50 CHARACTERS. NULL FIELDS, I.E., SUCCESSIVE COMMAS, INDICATE WORDS OF ZERO TO BE STORED.

OUTPUT (TO 'L')

HEADER - "*** XUD8G FROM IDIIO"

PROMPTS - (SEE INPUT)

OUTPUT (TO 'OUTPUT UNIT')

HEADER - "*** XUD8G FROM IDIIO"

SNAP - 102 WORD DUMP FORMATTED LINE (SEE XUDPL)

BASE - NUMBER BASE FOR PROMPT AND CONVERSION PROCEDURE

CLASS - CLASS I/O NUMBER FOR TERMINAL INPUT

LENGTH - LENGTH OF CHARACTER STRING BEING MANIPULATED

LINE - EIGHT WORD BUFFER OF WORD TO BE SNAPPED

LUI - TERMINAL LOGICAL UNIT WITH INPUT ECHO BIT SET

LUO - LOGICAL UNIT FOR SNAP OUTPUT

OPTN - ONE CHARACTER EXECUTION OPTION CODE

ORIGIN - REFERENCE POINT FOR MEMORY ACCESS OFFSET COMPUTATION

ORG - ADDRESS OF 'ORIGIN'

STP = 'ORIGIN' RELATIVE END OF DATA TO BE SNAPPED

STR = 'ORIGIN' RELATIVE BEGINNING OF MEMORY TO BE ACCESSED

RTN = RETURN POINT INDICATOR FOR INTERNAL PROCEDURE 'PROMPT AND CONVERSION'

NOTES
100 1 CD1   GENERAL FILE DUMP PROGRAM FOR FILE MANAGER FILES
101 1 CD1
102 1 CD1
103 1 CD1 INPUT
104 1 CD1 NAME - NAME OF FM FILE TO BE DUMPED
105 1 CD1 IREC - LOGICAL RECORD NUMBER OF FIRST RECORD TO BE DUMPED
106 1 CD1 NREC - NUMBER OF LOGICAL RECORDS TO DUMP
107 1 CD1 FMT - RUN TIME FORMAT FOR RECORDS (MAXIMUM OF 72 CHARACTERS) OR
108 1 CD1 BLANK INDICATING THE DEFAULT OF OCTAL AND ASCII DUMP TYPE
109 1 CD1 OUTPUT FOR M1-33 FREE FORMATTING OF THE CHARACTERS OF INDICATING UNFORMATTED
110 1 CD1 OUTPUT
111 1 CD1 LU - LOGICAL UNIT NUMBER OF OUTPUT DEVICE
112 1 CD1
113 1 CD1
114 1 CD1 OUTPUT
115 1 CD1 FORMATTED DUMP OF THE INDICATE PORTION OF THE INDICATED FILE
116 1 CD1
117 1 CD1
118 1 CD1
119 1 CD1 NOTES
120 1 CD1 USES EXEC, MACO, OPEN, POINT, READF, BMPAR, XPR05, XRN05, XUAPL
121 1 CD1
122 1 CD1
123 1 CD1 ANY FILE WITH VARIABLE LENGTH RECORDS WILL BE DUMPED USING A
124 1 CD1 RECORD BUFFER OF 1024 WORDS THUS LIMITING THE MAXIMUM DUMPABLE
125 1 CD1 RECORD LENGTH.
126 1 CD1
127 1 CD1
128 1 CD1
129 1 * BEG @: 1 CD1
130 1 *
131 1 * DO FOREVER 1 CD1
132 1 * READ FILE NAME 1 CD1
133 1 * EXIT XUDPF IF NAME IS NULL 1 CD1
134 1 * READ INITIAL RECORD NUMBER 1 CD1
135 1 * READ NUMBER OF RECORDS TO DUMP 1 CD1
136 1 * READ DUMP FORMAT 1 CD1
137 1 * IF FORMAT IS NULL 1 CD1
138 1 * THEN 1 CD1
139 1 * SET DEFAULT OCTAL/ASCII FORMAT 1 CD1
140 1 * ENSIF 1 CD1
141 1 * READ LU OF PRINT DEVICE 1 CD1
142 1 * OPEN FILE 1 CD1
143 1 * IF SUCCESSFUL 1 CD1
144 1 * THEN 1 CD1
145 1 * DO FOR NUMBER OF RECORDS TO DUMP 1 CD1
146 1 * READ RECORD 1 CD1
147 1 * EXIT TO ERROR IF FAILED 1 CD1
148 1 * FORMAT AND PRINT RECORD 1 CD1
149 1 * ELSE 1 CD1
150 1 * ERROR: OUTPUT MESSAGE 1 CD1
151 1 * ENSIF 1 CD1
152 1 * ENSD0 1 CD1
153 1 * ERROR: OUTPUT MESSAGE 1 CD1
154 1 * ENSIF 1 CD1
155 1 * ENSD0 1 CD1
156 1 * ENSD0 1 CD1
FORTRAN CALLING PROCEDURE

CALL XUDPL (ADDRESS, LINE, BUFFER)

+++++++ 1

ORIGINAL IV

+++++++ 1

INPUT

ADDRESS - TWO WORD INTEGER ARRAY CONTAINING THE ABSOLUTE AND
RELATIVE ADDRESS TO BE FORMATTED WITH THE LINE

LINE - EIGHT WORD ARRAY TO BE CONVERTED TO OCTAL AND ASCII

FORMATTED AND SPACED INTO A LINE IMAGE

BUFFER - FIFTY-ONE WORD BUFFER TO HOLD FORMATTED PRINT LINE. MUST
BE BLANKED PRIOR TO FIRST CALL TO XUDPL AND NOT STORED

INTO BETWEEN CALLS TO XUDPL.

+++++++ 1

OUTPUT

BUFFER - FIFTY-ONE WORD BUFFER CONTAINING FORMATTED LINE

COLUMNS CONTENTS

FIRST ADDRESS

SECOND ADDRESS

21-32 OCTAL REPRESENTATION OF 'LINE'

87-102 ASCII REPRESENTATION OF 'LINE'

+++++++ 1

NOTES

USES XREXT, XRO6, XRSET

+++++++ 1

BEGIN XUDPL

CALL XRO6 TO CONVERT EACH WORD OF ADDRESS TO OCTAL

DO FOR EACH WORD OF LINE

CALL XRO6 TO CONVERT WORD TO OCTAL

DO FOR EACH BYTE OF WORD

IF BYTE < 40 OR BYTE > 1368

THEN

REPLACE BYTE WITH ASCII PERIOD

ENDIF

END DB

END XUDPL
FORTRAN MAIN PROGRAM XUFMT IS SCHEDULED BY XDUMP TO PRINT
A PARTITION DUMP WHICH HAS BEEN WRITTEN TO DISK

INPUT ICNTL - CONTROL WORD FOR EXEC CALL READS. CONTAINS THE
LU NUMBER OF WHERE DUMP IS ON DISK
TRACK - TRACK NUMBER OF A 4 TRACK GLOBALLY ALLOCATED AREA
CONTAINING THE DUMP

FORMATTED DUMP TO LU 6
EXTERNAL REFERENCES
EXEC
RMPAR
XRCPR
XMNV
XUDPL

BEGIN XUFMT
CALL RMPAR TO GET LU AND STARTING TRACK NOS.
READ 1ST TRACK -- 1ST 12 WORDS ARE HEADER
.COUNT OF ID-SEGS TO BE DUMPED
.UP TO 7 ID-SEGMENT ADDRESSES
.LOW AND HIGH BASE PAGE ADDRESSES
.LOW AND HIGH MAIN MEMORY ADDRESSES
.POINT TO 2ND SECTOR OF DUMP DATA
DO UNTIL ALL ID-SEGS PRINTED
DO UNTIL A EIGHT-WORD LINES PRINTED
PRINT 1 LINE AND INCREMENT POINTER AND ADDRESSES TO NEXT
ENDDO
INCREMENT TO NEXT SECTOR OF DUMP DATA
ENDDO
COMPUTE N, THE NO. OF 8-WORD LINES IN THE BASE PAGE DUMP
PERFORM COMPARE AND PRINT FUNCTION
COMPUTE NTRKS, NO. OF DISK TRACKS OF MAIN MEMORY TO BE READ
DO UNTIL NTRKS ARE READ
READ NEXT TRACK FROM DISK
COMPUTE N, THE NUMBER OF 8-WORD LINES TO DUMP
PERFORM COMPARE AND PRINT FUNCTION
ENDDO
RELEASE THE GLOBALLY ALLOCATED TRACKS
EXIT XUFMT
BEGIN COMPARE AND PRINT FUNCTION
DO UNTIL N LINES PROCESSED
IF NOT 1ST LINE, THEN
CALL XRCPR TO COMPARE WITH PREVIOUS LINE
IF LINES ARE IDENTICAL, THEN
IF THIS IS 1ST OF A SERIES, THEN
WRITE 'DUPLICATE LINE'
ENDIF
ELSE
CALL XUDPL TO FORMAT THE DUMP LINE
ENDIF
endif
ELSE
CALL XUDPL TO FORMAT THE DUMP LINE
ENDIF
ENDIF
ENDDO
1 ENTRY XVABN
2 TYPE 1A ROUTINE TO ABORT CURRENT ID AND
3 TYPE 1A ROUTINE TO ABORT CURRENT ID AND
4 REQUEST A DUMP FOR THE ID.
5 ENTRY XVABN
6 TYPE 1A ROUTINE TO DUMP FROM CURRENT ID
7 THE REQUESTED DATA AREAS AND THE CURRENT
8 ID SEGMENT AND ITS FATHER CHAIN SEGMNENTS
9 BEGIN XVABN ABEND
10 INPUTS: NONE
12 OUTPUTS: SETS CURRENT ID TO ABORTED STATE
15 EXTERNALS: $ABRT, $ABRD, $XER, $LIBR
16 CALL SLIBR - REENTRANT ROUTINE CALL
17 GET CURRENT ID FROM XERT (LOCATION 1717)
18 INCREMENT TO ADDRESS BOUNDARIES
19 GET ADDRESS BOUNDARIES
20 STORE = XUDMP PARAMETER LIST
21 PERFORM XUDMP DUMP THIS PARTITION
22 CALL SABRE ABORT THIS PARTITION
23 CALL SABRE FREE CURRENT REENTRANT DATA BLOCK
24 EXIT TO $XEN GO TO DISPATCHER
25 END XVABN
26 *
27 *
28 *
29 BEGIN XUDMP
30 ENTRY XUDMP
31 ADDRESS OF DOUBLE WORD CONTAINING START AND END ADDRESSES
32 DEFINING AREA TO BE DUMPED
33 OUTPUTS: OUTPUTS TO DISK
34 HEADER DATA CONTAINING NO. OF ID SEGMENTS TO FOLLOW
35 MEMORY IMAGE OF BASE PAGE SEGMENT FOR THIS PARTITION
36 MEMORY IMAGE AS SPECIFIED BY INPUT ADDRESS RANGE
37 SCHEDULES XVFMT TO FORMAT DATA FROM DISK TO PRINT
38 EXTERNALS: $LIBR, $LIBX, EXEC, XVFMT
39 CALL SLIBR TO NOTIFY OF REENTRANCY AND BECOME PRIVILEGED
40 RUN FATHER ID SEGMENT CHAIN SAVING ADDRESS OF EACH
41 CALL SLIBX TO BECOME NON-PRIVILEGED
42 CALL EXEC FOR A GLOBAL ALLOCATION OF DISK
43 CALL EXEC TO WRITE HEADER DATA USING CLASS I/O
44 DO UNTIL ALL ID SEGMENTS PROCESSED
45 WRITE ID SEGMENT TO DISK
46 END ID SEGMENT TO REMAINING PORTION OF THIS TRACK
47 DO UNTIL END ADDRESS REACHED
48 WRITE ONE SECTOR OF DATA
49 SCHEDULE XVFMT WITH Lu AND STARTING TRACK NO.
50 CALL SLIBX TO RETURN FROM REENTRANT ROUTINE
51 END XUDMP
58 1 *01 TYPE 1A ROUTINE TO CONTROL COMMUNICATION BETWEEN AND
59 1 EXECUTION OF FDS MANAGER AND ITS ASSOCIATED TASKS
60 1 EXECUTIVE, PROCESSORS, AND UTILITIES.
61 1 ENTRY XVPAW AND XVSTB
62 1 *02 INPUTS
63 1 *02 FROM AN ASSOCIATED TASK
64 1 *02 CALL XVPAW(PARMS)
65 1 *02 ASSEMBLY FORM
66 1 *02 JSB XVPAW
67 1 *02 DEF *+2 RETURN ADDRESS
68 1 *02 DEF PARMS A(PARMS)
69 1 *02 WHERE PARMS ARE P1,P2,P3,P4,P5
70 1 *02 P1 IS THE SERVICE REQUEST
71 1 *02 0= NORMAL TERMINATION (P2-P5 NOT USED)
72 1 *02 1= WORK AREA REQUEST (P2-P5 NOT USED)
73 1 *02 2= EXECUTE A SEQUENCE TABLE
74 1 *02 (P2-P4 HAS TABLE NAME)
75 1 *02 (P5 INDICATES EXECUTION CONTROL IN CLASS I/O BUFFER)
76 1 *02 3= RESET SEQUENCE POINTERS
77 1 *02 (P2 HAS SEQUENCE NUMBER)(P3-P5 NOT USED)
78 1 *02 8= TERMINATE SEQUENCE (P2-P5 NOT USED)
79 1 *02 9= TERMINATE FDS FUNCTION (P2-P5 NOT USED)
80 1 *02 -32767= ABNORMAL TERMINATION OF ASSOCIATED TASK
81 1 *02 FROM AN FDS MANAGER
82 1 *02 ASSEMBLY FORM
83 1 *02 JSB XVPAW
84 1 *02 DEF (RETURN POINT)
85 1 *02 OCT 0
86 1 *02 DEF PARMS A(FDS MANAGER RESPONSE)
87 1 *02 OUTPUTS
88 1 *03 TO AN FDS MANAGER
89 1 *05 REQUEST PARMS (P1-P5) MOVED INTO ASSOCIATED TASK ID-SEGMENT
90 1 *05 RETURN ADDRESS IS MOVED INTO ID-SEGMENT WORD 9(XSUSP)
91 1 *05 CURRENT ID-SEGMENT IS MOVED INTO GENERAL WAIT VIA SLIST
92 1 *05 MANAGER IS ACTIVATED VIA SLIST
93 1 *05 TO AN ASSOCIATED TASK
94 1 *05 RESPONSE PARMS (P1-P5) MOVED INTO ASSOCIATED TASK ID-SEGMENT
95 1 *05 RETURN ADDRESS IS MOVED INTO ID-SEGMENT WORD 9(XSUSP)
96 1 *05 CURRENT ID-SEGMENT IS MOVED INTO GENERAL WAIT VIA SLIST
97 1 *05 ASSOCIATE TASK IS ACTIVATED VIA SLIST
103 1 BEGIN XVPAM
104 2 CALL BLIBR... BECOME PRIVILEGED
105 3 SET STOP-ID FROM XERT (OCT 17/77)
106 4 IF THIS IS A MANAGER RESPONSE
107 5 THEN SET UP TO ACTIVATE ASSOCIATED TASK AND SUSPEND MANAGER
108 6 CALL SEQUENCE IS RETURN, O, A (PARMS)
109 7 PERFORM XVPAM POST REQUESTOR AND WAIT
110 8 ELSE SET UP TO ACTIVATE MANAGER AND SUSPEND ASSOCIATED TASK
111 9 CALL SEQUENCE IS RETURN, O, A (PARMS)
112 10 PERFORM XVPAM POST MANAGER AND WAIT
113 11 ENDD
114 2 * STOP-ID HAS ID-SEGMENT TO BE SUSPENDED.
115 2 * AWAKEN-ID HAS ID-SEGMENT TO BE ACTIVATED.
116 2 * CALL SLIST (SCHEDULE, AWAKEN-ID)
117 2 MOVE RETURN ADDRESS TO XSUSP OF STOP-ID.
118 2 CALL SLIST (WAIT, STOP-ID)
119 1 EXIT TO :XENR RTE DISPATCHER
120 1 END XVPAM
122 1 BEGIN XPVPAW
123 2 * DETERMINE REQUESTORS MANAGER BY USING FATHER ID NUMBER
124 2 * FIELD IN ID SEGMENTS AS A BACKWARD CHAIN
125 2 * SET TARGET-ID FROM CURRENT-ID-SEGMENT
126 2 DO WHILE FATHER-ID-NUMBER .NE. 0 OR FATHER IS WAITING
127 3 COMPUTE FATHER-ID-SEGMENT FROM FATHER-ID-NUMBER IN TARGET-ID
128 3 PERFORM MGRFND(FATHER-ID-SEGMENT,COUNT)
129 3 EXITIF COUNT .GT. 0
130 3 SET TARGET-ID TO FATHER-ID-SEGMENT
131 2 ENDIF
132 2 IF FATHER-ID-NUMBER .EQ. 0, OR FATHER NOT WAITING THEN
133 3 CALL $SYM(12,*XV03,SEGMENT-NAME) *XV03,NAME* REQUESTING PROG
134 3 PERFORM PUMP
135 3 EXIT TO $LBBX TO ENABLE AND REDISPATCH
136 2 ENDIF
137 2 SET AWAKEN-ID FROM FSD-ENTRY STBMG
138 2 GET REQUEST PARMS MOVE INTO ID-SEGMENT
139 2 SET STBFS FROM CURRENT-ID
140 1 FND XPVPAW
141 1 *
142 1 *
143 1 *
144 1 *
145 1 BGN: XPVPAW
146 2 * DETERMINE IF CALLER IS A VALID FDS MANAGER
147 2 *
148 2 * PERFORM MGRFND (CURRENT-ID,COUNT)
149 2 * COUNT WILL BE 0 FOR NO MATCH.
150 2 * COUNT NOT EQUAL ZERO IMPLIES A MATCH
151 2 * AND FSD-ENTRY HAS MATCHING FDSTAB ENTRY ADDRESS
152 2 IF COUNT .LT. 0 THEN CALLER IS NOT A FDS MANAGER
153 2 CALL $SYM(12,*XV01,SEGMENT-NAME) *XV01 PROGRAM* REQUESTING PROG.
154 2 PERFORM PUMP
155 2 EXIT TO $SKEF THE DISPATCHER
156 2 ENDIF
157 2 FDSTAB-ENTRY HAS ENTRY FOR RESPONDING MANAGER
158 2 * SET AWAKEN-ID-SEGMENT FROM CURRENT-ASSOCIATED-TASK
159 2 IF CURRENT-ID NOT WAIT THEN IT WAS NOT ON WAIT LIST
160 3 CALL $SYM(12,*XV02,SEGMENT-NAME) *XV02 PROGRAM* ASSOCIATED PROG.
161 2 PERFORM PUMP
162 2 EXIT TO $SKEF THE DISPATCHER
163 2 ENDIF
164 2 AWAKEN-ID-SEGMENT-STAB
165 2 IF MANAGER HAS REQUEST FOR ABORT, THEN
166 3 CALL SBART FOR CURRENT AF
167 2 ENDIF
168 2 MOVE FDS MANAGERS INPUT PARMS TO ASSOCIATED TASK ID SEGMENT
169 2 MOVE PARMS TO ID-SEGMENT WORDS 2-6
170 1 END XPVPAW
172 1 BEGIN PDUMP
173 2 * SET A CALL TO XVABM AT CALLER'S SUSPEND POINT
174 2 * END PDUMP
175 1 *
176 1 *
177 1 *
178 1 *
179 1 *
180 1 *
181 1 *
182 1 BEGIN MGRFDN INPUT IS TARGET-ID
183 2 GET XVSTB TABLE OF ACTIVE FDS MANAGERS
184 2 SET COUNT TO NUMBER IN TABLE NUMBER MANAGERS
185 2 SET FSTAB-ENTRY TO FIRST-ENTRY(STBES)
186 2 DO WHILE COUNT .GT. 0
187 3 EXIT IF TARGET-ID .EQ. FDS-MANAGER-ID-SEGMENT
188 3 SET FSTAB-ENTRY TO FSTAB-ENTRY + STBEL
189 3 SET COUNT = COUNT - 1
190 2 ENDDO
191 2 * FSTAB-ENTRY HAS FOUND MANAGER ENTRY IF COUNT .GT. 0
192 2 * COUNT = 0 MEANS NO MATCH
193 1 END MGRFDN
194 1 *
195 1 *
196 1 *
197 1 *
198 1 BEGIN XVSTB
199 2 N EQU 3 EQUATE FOR NUMBER OF ENTRIES IN XVSTB
200 2 STBNM DEF N NUMBER-IN-TABLE
201 2 STBNC DEF N NUMBER-ACTIVE
202 2 STBNS DEF N STB RESOURCE NUMBER
203 2 STBES EQU ENTRY START
204 2 STBLU DEF 0 LUCAL
205 2 STBRE DEF 0 LUCASL
206 2 STBGM DEF 0 A(FDS-MANAGER-ID-SEGMENT)
207 2 STBEX DEF 0 A(FDS-EXECUTIVE-ID-SEGMENT)
208 2 STBEC DEF 0 CLASS NUMBER FOR EXEC
209 2 STBAT DEF 0 A(CURRENT-ASSOCIATED-TASK)
210 2 STBPC DEF 0 CLASS-NUMBER FOR PROCESSOR
211 2 STBGR DEF 0 ENTRY-AREA
212 2 STBEE EQU ENTRY -9D
213 2 STBEL EQU STBEE-STBES ENTRY LENGTH
214 2 STBNE EQU N NUMBER-OF-ENTRIES
215 2 ORG STBES
216 2 REP N DO FOR N ENTRIES
217 2 REP STBEL DEFINE STORAGE WORDS FOR AN ENTRY
218 2 DEF 0
219 1 END XVSTB
Fortran Calling Procedure:

CALL IXAUT

IXAUT Handles Automatic Execution Without Trace

Inputs in Common:

XE(5) MASTA, XE(10) SENV, XE(11) SEEND, XE(12) SEGPTA,
XE(14D)TABEND, XB(1) NDPROC, XB(2) LIBD,
XB(249) SENNO, XB(250) SENLEN, XB(251) SERTAB

Outputs in Common:

XE(5) MASTA, XS(1) FLGTAB

Common Used:

Equivalence (XE(5), MASTA)
+ (XE(10), SENV), (XE(11), SEEND)
+E (XE(12), SEGPTA), (XE(14D), TABEND),
+E (XB(1), NDPROC), (XB(2), LIBD),
+E (XB(253), RESIND), (XE(245), SENNO),
+E (XB(250), SENLEN), (XB(251), SERTAB),
+E (XS(1), FLGTAB)

Routines Called:

EXEC, EXXEXT, EXRMV, EXMSG, EXXEEE, EXSIO, EXTHMP

Rte Routines Called:

IOE
BEGIN XXAUT
IF ENTRY IS FROM A DIRECTIVE THEN
SET MASTER STATE TO INDICATE REENTRY
DO FOR EACH ENTRY IN THIS SEQUENCE TABLE
SEARCH LIBRARY DIRECTORY FOR THIS PROCESSOR
EREREIT IF PROCESSOR NOT FOUND TO :ERR1:
STUFF INTERFACE TABLE 6IT AND VERSION INTO SEQUENCE TABLE ENTRY
ENDDO
CALL XXSTO TO STORE REVISIRED SEQUENCE TABLE IN AXA AS &SEQTB
ELSE (I AM BEING REENTERED FROM INT.)
CALL XXMP TO SET UP TERNARY EN-8Y W:HEN &INTAB
CALL XXEKE TO EXECUTE FROM TEMPORARY ENTRY
IF SELECTED SEQUENCE NUMBER IS NOT REQUESTED THEN
EXIT XXAUT IF TERMINAL ENTRY WAS JUST EXECUTED
SET STARTING ENTRY TO NEXT ENTRY
ENDIF
ENDIF
DO UNTIL TERMINAL ENTRY IS EXECUTED AND IT IS NOT A REQUEST FOR RESET
CALL XXEKE TO EXECUTE REMAINDER OF TABLE
ENDDO
EXIT XXAUT
:ERR1:
CALL XXMSG TO DISPLAY INVALID PROCESSOR NAME
END XXAUT
**Fortran Calling Sequence:**

```
132 1 CO************
133 1 CO
134 1 CO FORTRAN CALLING SEQUENCE:
135 1 CO
136 1 CO CALL XDEC (RETC)
137 1 CO
138 1 CO************
139 1 CO
140 1 CO XDEC DECODES A RESPONSE OF PROCESSOR NAME (.INT TABLE NAME)
141 1 CO INTO A SEQUENCE TABLE ENTRY.
142 1 CO
143 1 CO************
144 1 CO
145 1 CO INPUTS IN COMMON:
146 1 CO
147 1 CO XE(85) TOKENS, XE(145) COMBUF, XB(1) NOPROC, XB(2) LIBD
148 1 CO
149 1 CO************
150 1 CO
151 1 CO OUTPUTS IN CALLING SEQUENCE:
152 1 CO
153 1 CO RETC - RETURN CODE (O IS NORMAL RETURN)
154 1 CO
155 1 CO OUTPUTS IN COMMON:
156 1 CO
157 1 CO XE(16) PRCNAM, XB(251) SEQTAB
158 1 CO
159 1 CO************
160 1 CO
161 1 CO COMMON USED:
162 1 CO
163 1 CO EQUIVALENCE (XE(16), PRCNAM),
164 1 CO (+ XE(85), TOKH), (XE(144), TOKPTR),
165 1 CO (+ XE(145), COMBUF),
166 1 CO (+ XB(1), NOPROC), (XB(2), LIBD ),
167 1 CO (+ (XB(251), SEQTAB)
168 1 CO
169 1 CO FDS ROUTINES USED:
170 1 CO
171 1 CO XRCPR, XREXT, XRM0V, XRM5G
172 1 CO
173 1 CO RTE ROUTINES USED:
174 1 CO
175 1 CO
176 1 CO
177 1 CO************
```
179 1 BEGIN XXDEC
180 2 INITIALIZE RETURN CODE TO ZERO
181 3 SET SEQUENCE ENTRY TO ZEROS
182 4 ERREXIT IF FIRST TOKEN IS NOT A PROCESSOR NAME TO :ERR1:
183 5 SEARCH LIBRARY DIRECTORY FOR PROCESSOR
184 6 ERREXIT IF NAME NOT FOUND TO :ERR1:
185 7 MOVE PROCESSOR NAME, IT BIT AND VERSION INTO SEQUENCE ENTRY
186 8 IF AN INTERFACE TABLE NAME WAS ENTERED THEN
187 9 MOVE INTERFACE TABLE NAME INTO SEQUENCE ENTRY
188 10 ENDIF
189 11 ERREXIT IF LAST TOKEN IS NOT EOS TO :ERR1:
190 12 ERREXIT IF INTERFACE TABLE IS SPECIFIED WHEN NOT NEEDED TO :ERR1:
191 13 IF AN INTERFACE TABLE IS REQUIRED BUT NOT SPECIFIED THEN
192 14 SET INTERFACE TABLE IN SEQUENCE ENTRY TO 'SINTAB'
193 15 ENDIF
194 16 EXIT XXDEC
195 17 :ERR1:
196 18 CALL XRNUG TO DISPLAY ERROR
197 19 SET RETURN CODE TO SAY ERROR
198 20 END XXDEC
1 CD************
2 CD
3 CD FORTRAN CALLING PROCEDURE:
4 CD
5 CALL XXDEF
6 CD************
7 CD************
8 CD1 XXDEF READS IN THE DEFAULT INTERFACE TABLE FOR A PROCESSOR
9 CD1 AND STORES IT IN THE AWA AS BINTAB
10 CD1
11 CD************
12 CD2 INPUTS FROM COMMON:
13 CD2
14 CD2 PROCNAME - (INTEGER, 3 WORDS) NAME OF PROCESSOR IN SERTAB
15 CD2 FOR WHICH NO INTERFACE TABLE WAS SUPPLIED
16 CD2
17 CD************
18 CD4 INTERNAL VARIABLES:
19 CD4
20 CD4 DEFTAB - (INTEGER, 1200 WORDS) ARRAY WHERE MAXIMUM SIZE
21 CD4 DEFAULT INTERFACE TABLE CAN BE READ INTO
22 CD4 DEFNAME - (INTEGER, 3 WORDS) ARRAY WHERE INTERFACE TABLE
23 CD4 NAME IS CREATED FROM PROCESSOR NAME
24 CD4
25 CD************
26 CD5 COMMON USED:
27 CD5
28 CD5 EQUIVALENCE (XE(5), MASSTA), (XE(6), SUBSTA),
29 CD5 + (XE(13), INTMAP), (XE(16), PRCHNM),
30 CD5 + (XE(19), RECPTR), (XE(20), RECBUF),
31 CD5 + (XE(142), IRC ),
32 CD5 + (XS(6), TMPTAB), (XS(14), DEFNAME),
33 CD5 + (XS(18), LEN1 ), (XS(19), LEN2 ,
34 CD5 + (XS(20), IDC), (XS(201), RETC )
35 CD5
36 CD5 RTE ROUTINES USED:
37 CD5
38 CD5 CLOSE, EXEC, KCVT, OPEN, READF
39 CD5
40 CD5 FDS ROUTINES USED:
41 CD5
42 CD5 XERTM, XREQ, XREXT, XRM0V, XRM0G, XRACK, XRUPK
43 CD5
44 CD5
45 CD5
46 CD5
249 1 BEGIN XXDEF
250 2 CREATE DEFAULT INTERFACE TABLE NAME FROM PROCESSOR NAME
251 2 CALL OPEN TO OPEN FILE
252 2 ERREXIT IF OPEN ERROR TO :FILERR:
253 2 CALL READ TO READ RECORD 1
254 2 ERREXIT IF READ ERROR TO :FILERR:
255 3 IF THERE IS LITERAL DATA THEN
256 3 CALL READ TO READ RECORD 2
257 3 ERREXIT IF READ ERROR TO :FILERR:
258 3 ENDIF
259 2 CALL CLOSE TO CLOSE FILE
260 2 ERREXIT IF CLOSE ERROR TO :FILERR:
261 2 CALL XREG TO ALLOCATE & STORE GINTAB
262 2 ERREXIT IF RETURN CODE IS NOT ZERO TO :MGERR:
263 3 IF DEFAULT TABLE IS INCOMPLETE THEN
264 3 SET INTERFACE TABLE NAME IN XE TO GINTAB
265 3 SET SUBSTATE TO INTERFACE TABLE EDITOR
266 3 CALL XERTN TO IMPLICITLY CALL INT EDITOR **NO RETURN**
267 3 ENDIF
268 1 EXIT XXDEF
269 2 :FILERR:
270 2 CALL CLOSE TO CLOSE FILE
271 2 CALL XRMSTO DISPLAY FILE ACCESS ERROR
272 1 EXIT XXDEF
273 2 :MGERR:
274 2 CALL XRMSTO DISPLAY SPACE ERROR
275 2 CALL EXEC TO FREE CLASS NUMBER
276 2 SET MASTER STATE TO DIRECTIVE LEVEL
277 2 CALL XERTN TO RETURN TO EXEC **NO RETURN**
278 1 END XXDEF
280 1 C***********
281 1 C*** FORTRAN CALLING PROCEDURE:
282 1 C***
283 1 C*** CALL xxexe
284 1 C***
285 1 C***********
286 1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1 C1
BEGIN X{xek}
INITIALIZE RESET INDEX TO ZERO
IF TABLE FLAG SAYS SETTAB IN XD THEN
CALL XVPAM WITH SEQ# START AND END
ELSE -- AM BEING CALLED TO EXECUTE A TEMPORARY SEQUENCE TABLE
CALL XVPAM WITH SEQUENCE POINTER AS START AND END
ENDIF
CALL RMPAR TO RETRIEVE RETURN PARAMETERS
IF THE MANAGER DETECTED A I ERROR TRYING TO EXECUTE THE SEQUENCE THEN
SET SEQUENCE POINTER TO SEQUENCE # IN ERROR
FIND ENTRY IN ERROR AND SAVE IN XE
CASE ERROR (:INIT1,:ERR1,:INT2,:ERR2,:RESET,:ERR4,:ERR5,:ERR6:)
:INIT1: INTERFACE TABLE NOT SPECIFIED
CALL XDEF TO READ UP DEFAULT TABLE
CALL XTMP TO SET UP A TEMPORARY ENTRY
PERFORM X{xek} TO EXECUTE **NO RETURN**
:ERR1: SET MESSAGE TO INTERFACE TABLE NOT FOUND
:INT2: INTERFACE TABLE NOT COMPLETE
SET SUBSTATE TO INTERFACE TABLE EDITOR
CALL XRTN TO RETURN TO EXEC **NO RETURN**
:ERR2: SET MESSAGE TO INT TABLE NOT FOR PROCESSOR
:ERR3: SET MESSAGE TO VERSIONS DO NOT MATCH
:RESET: RESET REQUESTED ON TERMINATION
CONVERT SEQUENCE # TO INDEX
SAVE INDEX AND SEQUENCE NUMBER IN COMMON
EXIT X{xek}
:ERR4: SET MESSAGE TO RESET SEQ# NOT FOUND
:ERR5: SET MESSAGE TO PROCESSOR ABENDED
:ERR6: SET MESSAGE TO AMA OVERFLOW
ENDCASE
ENDCASE
CALL XRMSG TO DISPLAY ERROR
FORMAT SEQUENCE ENTRY INTO ASCII
CALL XRMSG TO DISPLAY SEQUENCE TABLE ENTRY IN ERROR
IF MODE IS SEMI-AUTO AND ENTRY IS NOT AN OVERRIDE THEN
RESET OLD INDEX TO RE-EXECUTE THIS ENTRY
ENDIF
IF MODE IS AUTO THEN
SET MASTER STATE TO ZERO
CALL XRTN TO ABORT SEQUENCE *** NO RETURN ***
ENDIF
ENDIF
RETURN
1 END X{xek}
1 BEGIN XXMAN
2 IF ENTRY IS FROM A DIRECTIVE THEN
3 SET HASSTA TO INDICATE RE-ENTRY
4 DO UNTIL PERCENT IS ENTERED
5 :PROMPT: CALL XTOCOM TO PROMPT FOR PRNAME, "ITNAME"
6 IF PERCENT IS NOT ENTERED THEN
7 ERREXIT IF CR ENTERED TO :PROMPT:
8 CALL XXDEC TO DECODE RESPONSE
9 ERREXIT IF INVALID RESPONSE TO :PROMPT:
10 SET SER #5 IN 2E . J ZERO
11 SET #ENTRIES IN SERTAB IN XE TO 1
12 CALL XXST0 TO STORE SEQUENCE TABLE
13 IF IT NAME IN SERTAB IS $INTAB THEN
14 PROCESSOR REQUIRES AN IT THEN
15 CALL XXDEF TO READ UP DEFAULT INTERFACE TABLE
16 ENDF
17 CALL XXEXEC TO EXECUTE SERTAB
18 ENDIF
19 ELSE
20 CALL XXMP TO SET UP TO EXECUTE A TEMPORARY TABLE
21 CALL XXEXEC TO EXECUTE ENTRY
22 PERFORM XXMAN **NO RETURN**
23 ENDIF
24 RETURN
25 :PMERR: CALL XRMSGC TO DISPLAY ERROR
26 PERFORM XXMAN **NO RETURN**
27 END XXMAN
443 1 COMMON CALLING SEQUENCE:
444 1 CALL XSEM
445 1
446 1
447 1
448 1
449 1
450 1
451 1
452 1
453 1
454 1
455 1
456 1
457 1
458 1
459 1
460 1
461 1
462 1
463 1
464 1
465 1
466 1
467 1
468 1
469 1
470 1
471 1
472 1
473 1
474 1
475 1
476 1
477 1
478 1
479 1
480 1
481 1
482 1
483 1
484 1
485 1
486 1
487 1
488 1
489 1
490 1
491 1
492 1
493 1
494 1
495 1
496 1
497 1
498 1
499 1
500 1
501 1

EXTERNAL COMMON USED:

443 1 XE(159) EXEND - ENDING SEQUENCE NUMBER USED TO
444 1 TERMINATE SEQUENCE
445 1
446 1
447 1 XE(161) CURNOD - INDEX TO THE CURRENT ENTRY BEING EXECUTED
448 1
449 1
450 1 XE(233) RESIND - INDEX TO RESET CURRENT INDEX TO
451 1
452 1
453 1 XE(236) ASCENT - ASCII SEQUENCE TABLE ENTRY USED TO PROMPT
454 1 THE USER
455 1
456 1
457 1 XE(246) OLDIND - INDEX TO THE LAST ENTRY EXECUTED IN THE
458 1 SEQUENCE TABLE
459 1
460 1
461 1

COMMON USED:

443 1 EQUIVALENCE
444 1 (XE(5), MASSTA).
445 1
446 1
447 1
448 1
449 1
450 1
451 1
452 1
453 1
454 1
455 1
456 1
457 1
458 1
459 1
460 1
461 1
462 1
463 1
464 1
465 1
466 1
467 1
468 1
469 1
470 1
471 1
472 1
473 1
474 1
475 1
476 1
477 1
478 1
479 1
480 1
481 1
482 1
483 1
484 1
485 1
486 1
487 1
488 1
489 1
490 1
491 1
492 1
493 1
494 1
495 1
496 1
497 1
498 1
499 1
500 1
501 1

FDS ROUTINES USED:

443 1 SRCPR, XREX, XRIO, XRDNY.
444 1
445 1
446 1
447 1
448 1
449 1
500 1
501 1

443 1 XSTC, XTRMP, XTECM.
444 1
445 1
446 1
447 1
448 1
449 1
500 1
501 1
```
1 BEGIN XSEM:
2 IF ENTRY IS FROM A DIRECTIVE THEN
3 SET MASTER STATE TO SAT REENTRY
4 DO FOR # ENTRY, IN SEQUENCE TABLE
5 SEARCH LABEL FOR PROCESSOR NAME
6 ERROR IF NOT FOUND TO :ERR1:
7 MOVE IT BIT AND VERSION INTO SEQUENCE TABLE ENTRY
8 ENDDO
9 SAVE ORIGINAL ENDING SEQUENCE #
10 SET CURRENT SEQUENCE # TO BEGINNING SEQUENCE #
11 SET FLAT TO SAT EXECUTE ENTIRE SEQUENCE
12 CALL XISTO TO STORE SESENT IN AWA
13 ELSE (I AM BEING REENTERED FROM INTRE)
14 CALL XETMP TO SET UP TEMPORARY EXECUTI"N
15 CALL XERE TO EXECUTE ONLY THE FIRST ENTRY OF SESENT
16 IF RESET WAS REQUESTED THEN
17 RESET CURRENT SEQUENCE # TO NEW SEQUENCE #
18 ELSE
19 EXIT XSEM IF TERMINAL ENTRY WAS JUST EXECUTED
20 SET CURRENT SEQUENCE # TO NEXT SEQUENCE #
21 ENDIF
22 ENDIF
23 DO UNTIL TERMINAL ENTRY IS EXECUTED AND IT IS NOT A REQUEST TO RESET
24 CALL XETOM TO PROMPT USER WITH CURRENT ENTRY
25 EXIT XSEM IF RESPONSE IS X
26 IF RESPONSE IS CR THEN
27 IF THIS IS AN OVERRIDE WITH DEFAULT INTERFACE TABLE THEN
28 CALL XDJE TO READ UP DEFAULT TABLE
29 ENDIF
30 SET BEGINNING SEQUENCE # TO CURRENT SEQUENCE #
31 SET ENDING SEQUENCE NUMBER TO BEGINNING SEQUENCE #
32 CALL XERE TO EXECUTE
33 SET CURRENT SEQUENCE # TO NEXT SEQUENCE #
34 ELSE
35 IF RESPONSE WAS A SEQUENCE # THEN
36 ERRSEX IF NUMBER IS ZERO TO :ERR1:
37 SEARCH SEQUENCE TABLE FOR SEQUENCE #
38 ERROR IF NUMBER IS NOT FOUND TO :ERR1:
39 SET CURRENT SEQUENCE # TO SEQUENCE # REQUESTED
40 ELSE
41 IF RESPONSE IS AN AMPSAND THEN
42 SET CURRENT SEQUENCE # TO NEXT SEQUENCE #
43 ELSE (RESPONSE MUST HAVE BEEN AN OVERRIDE)
44 CALL XEDC TO DECODE PROCESSOR NAME, IT NAME RESPONSE
45 IF R.SPONSE IS VALID THEN
46 CALL XISTO TO STORE OVERRIDING ENTRY
47 SAVE CURRENT SEQUENCE # IN OLD SEQUENCE #
48 SET CURRENT SEQUENCE # TO FIRST ENTRY
49 ENDIF
50 ENDIF
51 ENDDO
52 ! EXIT XSEM
53 :ERR1:
54 CALL XINSF TO DISPLAY ERROR MESSAGE
55 ENDDO
56 ! EXIT XSEM
```
CALLING PROCEDURE FOR SXSTO:

CALL SXSTO

SXSTO SETS UP TO REQUEST MANAGER TO STORE SECTO AND HANDLE ANY ERRORS RETURNED BY THE MANAGER

INPUTS FROM COMMON:

XH(250) SERLEN, XH(251) SECTAB, XS(13) FLGTAB

OUTPUTS TO COMMON:

XH(5) MASSTA

COMMON USED:

EQUIVALENCE (XH(5), MASSTA),

+ (XH(19), REQPTR), (XH(20), REQBUF),

+ (XH(250), SECTAB), (XH(251), SECTAB),

+ (XH(13), FLGTab)

FDS ROUTINES USED:

XREQ, XRMOV, XRMSC, XERTM

RTE ROUTINES USED:

EXEC
1 BEGIN XXSTO
2 SET CLASS NUMBER TO ZERO
3 IF TABLE FLAG SAYS STORE ENTIRE TABLE THEN
4 CALL EXEC TO WRITE ENTIRE TABLE
5 CALL MOV TO MOVE REQUEST BUFFER TO LENGTHS IN XN
6 ELSE
7 CALL EXEC TO WRITE ONLY FIRST ENTRY
8 CALL MOV TO MOVE STORE REQUEST INTO REQUEST BUFFER
9 ENSIF
10 CALL EXEC TO REQUEST MANAGER TO STORE .SEERTO
11 IF RETURN CODE IS NOT ZERO THEN
12 CALL XMSG TO WRITE SPACE ERROR
13 CALL EXEC TO RELEASE CLASS #
14 SET MASTER STATE TO ZERO
15 CALL XERTN TO RETURN TO EXEC **NO RETURN**
16 ENSIF
17 RETURN
18 1 END XXSTO
FORTM CALLING SEQUENCE:

CALL XTMP

XTMP SETS UP A ONE ENTRY SEQUENCE TABLE USING RINTAB AND
STORES IT IN THE ANA TO EXECUTE WHEN EXECUTION WITH A
TEMPORARY ENTRY IS NECESSARY

INPUTS FROM COMMON:

XE(12) SEPTRA, XB(250) SELLEN, XB(251) SECTAB

OUTPUTS TO COMMON:

XE(6) SUBSTA, XB(249) SEBAO, XB(250) SELLEN,
XB(251) SECTAB, XS(13), FLTDB

COMMON USED:

EQUIVALENCE (XE(6), SUBSTA), (XE(12), SEPTRA),
+ (XE(19), RERPTR), (XE(20), REBUF),
+ (XB(249), SEBAO), (XB(250), SELLEN),
+ (XB(251), SECTAB), (XS(6), TMPTAB),
+ (XS(13), FLTDB)

FDS ROUTINES USED:

XRED, XMOV, XSTO

RTE ROUTINES USED:

EXEC
ASSGN - DATA ASSIGNMENT PROCESSOR
SCHEDULED BY FDS

ASSGN ALLOWS THE FDS USER TO COMPUTE VALUES AND STORE THEM IN AN EXISTING DATA ELEMENT IN THE ANA. ASSGN SUPPORTS EXTENDED FORTRAN TYPE MIXED-MODE EXPRESSIONS AND FUNCTIONS AND ALLOWS REPEATED EVALUATIONS IN ORDER TO COMPUTE AND STORE MULTIPLE VALUES.

INPUTS FROM THE MANAGER:

LU - LOGICAL UNIT OF USER'S TERMINAL
DEBUG - FLAGS FOR DEBUG

INPUTS FROM THE INTERFACE TABLE:

EXP - SYMBOLIC STRING CONTAINING DATA ASSIGNMENT (SEE BELOW FOR BACKUS-NAUR DEFINITION OF VALID SYNTAX)

OUTPUTS TO THE ANA:

THE COMPUTED VALUE(S) IS STORED INTO THE SPECIFIED DATA ELEMENT

INTERNAL VARIABLES:

BLANK COMMON - ASGCOMMON DIMENSIONED BY 2300 WORDS DEFINED AS FOLLOWS:

NAME DIMENSION START DESCRIPTION
PARMS 5 1 PARMS(1) = LU, PARMS(2) = DEBUG FLAGS
TOKERS 32 6 IDENTIFYING NUMBERS FOR TOKENS
STWIDE 1 38 SYMBOL TABLE WIDTH
STLONG 1 39 SYMBOL TABLE LENGTH
LASTSTY 1 40 LAST SYMBOL TABLE ENTRY DEFINED
SYNTAB 12,81 41 SYMBOL TABLE (WORDS 1-8 = TOC ENTRY OR APPLICABLE INFORMATION, WORDS 9-11 = VALUE, WORD 12 = 1 FOR INDEX, = 2 FOR SUBSCRIPTED DATA ELEMENT)
SSTING 247 1013 RESULT STACK USED DURING POST-RESULT STRING EVALUATION (EACH EN- STRING)
RESULT 4,35 1260 RESULT STACK ASSEMBLY DATA TYPE = 1-2-3 FIXED DATA -1 SYMBOL TABLE INDEX
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CARD</td>
</tr>
<tr>
<td>1</td>
<td>DCD</td>
</tr>
<tr>
<td>1</td>
<td>CNDS</td>
</tr>
<tr>
<td>4,9</td>
<td>OPRNDS</td>
</tr>
<tr>
<td>1</td>
<td>1400</td>
</tr>
<tr>
<td>4,9</td>
<td>SCRTHC</td>
</tr>
<tr>
<td>1</td>
<td>1436</td>
</tr>
<tr>
<td>8</td>
<td>REST</td>
</tr>
<tr>
<td>7</td>
<td>OPINFO</td>
</tr>
<tr>
<td>1</td>
<td>1487</td>
</tr>
<tr>
<td>1</td>
<td>NUMWDS</td>
</tr>
<tr>
<td>9</td>
<td>DATTPS</td>
</tr>
<tr>
<td>1</td>
<td>1497</td>
</tr>
<tr>
<td>1</td>
<td>EXPPTR</td>
</tr>
<tr>
<td>1</td>
<td>CLSREP</td>
</tr>
<tr>
<td>1</td>
<td>CLSTRM</td>
</tr>
<tr>
<td>9</td>
<td>MAPWDS</td>
</tr>
<tr>
<td>1</td>
<td>DECLAS</td>
</tr>
<tr>
<td>4,4</td>
<td>RNGSTR</td>
</tr>
<tr>
<td>1</td>
<td>1527</td>
</tr>
<tr>
<td>1</td>
<td>SYNTAX</td>
</tr>
<tr>
<td>7,36</td>
<td>FNCTBL</td>
</tr>
<tr>
<td>64</td>
<td>MBUFF</td>
</tr>
</tbody>
</table>

Note: Stacks used in the ASSGN processor are sized for the maximum possible and overflow is not tested.

Table Definitions:
### Syntax - Table Containing Information for Processing Each Token

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Token</th>
<th>Input Length</th>
<th>Output Length</th>
<th>Precedent Type</th>
<th>Valid Precedent</th>
<th>Operator Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>39</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>41</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>43</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>47</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>54</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>DFA</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0: 2</td>
<td>0</td>
</tr>
</tbody>
</table>

**Note:** Reproducibility of the original page is poor.
<table>
<thead>
<tr>
<th>NAME(Words 1-3)</th>
<th>#OPANDS</th>
<th>TYPE</th>
<th>OP TYPE</th>
<th>FIRST TYPE</th>
<th>SECOND TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>0</td>
<td>REAL</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>ALOG</td>
<td>0</td>
<td>REAL</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>ALOGT</td>
<td>0</td>
<td>REAL</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>AMOD</td>
<td>1</td>
<td>REAL</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>ATAN</td>
<td>0</td>
<td>REAL</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>ATAN2</td>
<td>1</td>
<td>REAL</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>0</td>
<td>REAL</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATN</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATN2</td>
<td>1</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DDBL</td>
<td>0</td>
<td>DOUBLE</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCO3</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DINT</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLOG</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLOGT</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DCOS</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DSINT</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DSINT2</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DTAN</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DTANH</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DBINH</td>
<td>0</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td>DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DBIN</td>
<td>0</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DBIN</td>
<td>0</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DBIN</td>
<td>0</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DBIN</td>
<td>0</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DBIN</td>
<td>0</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DBIN</td>
<td>0</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>

**FMCTBL - 7X36 - ABLE CONTAINING INFORMATION FOR PROCESSING FUNCTIONS**
260 1 CD5 <CONSTANT> ::= INTEGER //
261 1 CD5 SINGLE PRECISION REAL //
262 1 CD5 DOUBLE PRECISION REAL
263 1 CD5 <RANGE> ::= <RANGE> <LIMITS> //
264 1 CD5 <RANGE> //
265 1 CD5 <LIMITS> ::= INDEX=INTEGER,INTEGER
266 1 CD5
267 1 CD5
268 1 CD5
269 1 CD5
270 1 CD*********
DD DSP - DATA BOX DISPLAY PROCESSOR

- SCHEDULED BY FD

DD DSP PROCESSES THE DATA PRODUCED BY SCANNING FUNCTIONAL PROCESSORS. IT GENERATES A DIGITAL MATRIX DISPLAY CONTAINING THE VALUES OF ANY PARAMETER OR PAIR OF PARAMETERS OVER THE SCANNED SPACE.

BECAUSE OF THE SIZE OF THIS PROCESSOR, IT WAS DIVIDED INTO THREE OVERLAYS AS FOLLOW:

XDIM - READS AND EDITS INTERFACE TABLE

XDP1 - READS OR DE A, VALIDATES NAMES IN INTERFACE TABLE AGAINST NAMES IN DATABOX SCANNED. WITH SUBROUTINE XDPK IT DEPENDS CONSTRAINT MASKS FOR ARRAYS.

XDP2 - PROMPTS USER IF REQUIRED AND OUTPUTS REQUESTED PAGE ARRAYS OF UP TO TWO VARIABLES TO THE SPECIFIED LU DEVICES WITH SUBROUTINE XDPD.

DD DSP MERELY CALLS THESE OVERLAYS IN THE PROPER ORDER AND EXITS.

INPUTS TO DD DSP FROM INTERFACE TABLE:

- DATABOX - DATA BOX FILE NAME
- NOVAR - DISPLAY VARIABLE NAME LIST SET UP BY USER
- REIP - DISPLAY VARIABLE SCALE LIST SET UP BY USER
- VIODEF - CONSTRAINT VARIABLE DEFINITION LIST

INPUTS TO DD DSP FROM ORDE FILE:

RECORD 1:

(1) - NAME OF FD PROCESSOR СоздАTING FILE
(4) - INTERFACE TABLE VARIABLE NAME FOR THIS FILE
(7) - NAME OF FD PROCESSOR UPDATING FILE
(3) ASCII WORDS OF BLANKS

RECORD 2:

(1) - NO OF ENTRIES IN SUMMARY TABLE
(2) - X Scan Variable (6 CHAR)
(5) - X FI.SY SUBSCRIPT (INT OR ZERO)
(6) - XSECOND SUBSCRIPT (INT OR ZERO)
(7) - X UNITS (6 CHAR)
(10) - X CENTROID (REAL)
(12) - X INCREMENT (REAL)
(14) - X NUMBER OF STEPS (INTEGER 1-5)
(15) - YSCAN VARIABLE (A CHAR)
(18) - Y FIRST SUBSCRIPT (INT OR ZERO)
(19) - Y SECOND SUBSCRIPT (INT OR ZERO)
(20) - Y UNITS (A CHAR)
(23) - Y CENTROID (REAL)
(25) - YE INCREMENT (REAL)
(27) - Y NUMBER OF STEPS (INTEGER 1-5)

DEPENDANT VARIABLE NAME AND UNITS IN RECORDS 32 48 AND 5

SUMMARY TABLE RECORDS

- EACH SUMMARY TABLE CONTAIN VALUE FOR EACH
- DEPENDANT VARIABLE SCANNED (UP TO 32 VALUES
- INCLUDING ERROR FLAG WHICH IS FIRST VALUE
- IN SUMMARY TABLE

OUTPUT FROM DBDSP

- DISPLAY IS OUTPUT TO LU IDENTIFIED IN THE INTERFACE TABLE

1 CD 4 COMBUF - BUFFER TO RECEIVE ENCODED USER RESPONSE - XPRM
1 CD 4 DBX - NAME OF DATA BOX TO BE DISPLAYED BY DBDSP
1 CD 4 DATBUF - COMMON BUFFER FOR SCAN SUMMARY DESCRIPTORS
1 CD 4 ID1 - POINTER TO DEP. DISPLAY VARIABLE NAME LIST FOR FIRST VAR
1 CD 4 ID2 - POINTER TO DEP. DISPLAY VARIABLE NAME LIST FOR 2ND VAR
1 CD 4 JSAVE - LIST OF VIOLATED CONSTRAINTS BUILT BY XMSK (MAX OF 8)
1 CD 4 JSCMN - FIRST SUBSCRIPT FOR NAME OF X SCAN VARIABLE (INT OR O)
1 CD 4 JSCMN - SECOND SUBSCRIPT FOR NAME OF X SCAN VARIABLE (INT OR O)
1 CD 4 LINF - NAME OF COMMON AREA USED FOR INTERFACE TABLE
1 CD 4 LST - LENGTH IN WORDS OF CHAR STRING USED FOR USER PROMPT -XPRM
1 CD 4 XMASK - ARRAY CONTAINING MASKS FOR UP TO 8 CONSTR (4 X 121 SIZE)
1 CD 4 NVAR - NAME LIST FOR VARIABLES SCANNED BY SCAN/ENDSCM
1 CD 4 NC - NUMBER OF CONSTRAINTS INPUT BY USER (INTEGER)
1 CD 4 MCREL - LIST OF CONSTRAINT RELATIONS INPUT BY USER
1 CD 4 MVAR - LIST OF CONSTRAINT VARIABLE NAMES INPUT BY USER (32 MAX)
1 CD 4 MVARC - NUMBER OF DEP DISP VARIABLE PAIRS FOR PAGED OUTPUT (1-16PR)
1 CD 4 MVAR - LIST OF DEP DISP VARIABLE PAIR SCALE FACTORS FOR OFP
1 CD 4 MSERR - SET OF INDICATORS FOR CONSTRAINTS VIOLATED-G/HOT - D
1 CD 4 MSTEP - NUMBER OF STEPS ON EITHER SIDE OF X CENTROID (0 TO 5)
1 CD 4 XMCORD - LIST OF X VAR VALUES FOR X COORDINATES (1 - 11 REAL)
1 CD 4 XSCMN - NAME OF X VAR SCANNED TO BE PLACED ON DISPLAY (6 CHAR)
397 1 CD 4 XUNITS  - NAME OF X VAR UNITS TO BE PLACED ON DISPLAY (6 CHAR)
398 1 CD 4 YCORD - LIST OF X VAR VALUES FOR Y COORDINATES (1 - 11 REAL)
399 1 CD 4 YUNITS - NAME OF Y VAR SCANNED TO BE PLACED ON DISPLAY (6 CHAR)
400 1 CD 4 YUNITS - NAME OF Y VAR UNITS TO BE PLACED ON DISPLAY (6 CHAR)
401 1 CD 4 ZTABLE - TABLE IN COMMON FOR SMTAB VARIABLE NAMES AND UNITS
402 1 CD 4 NAMVAL - UNITS LIST FOR VARIABLES SCANNED BY SCAN/ENDSCN
403 1 CD 4 SMTAB - VALUES FOR SCAN VARIABLE(S) - 1 TO 32 VALUES/RECORD
404 1 CD 4 PARS - COMMUNICATION BUFFER FOR RMPAR - LU, USER ID, FLAGS
405 1 CD 4 LU - LOGICAL UNIT # FOR XPRM CALLING SEQUENCE - USER LOCATION
406 1 CD 4 LUDP - BDP/P WILL OUTPUT DISPLAY TO THIS USER SUPPLIED LU
407 1 CD 4 PROMPT - TABLE IN COMMON TO COMMUNICATE WITH XPRM
408 1 CD 4 DEBUG
409 1 CD 4 SELECT - SELECT =O PROMPT ; SELECT NOT O RUN ALL DISPLAYS TO O/P
410 1 CD 4 CARTDC - CARTDC USED TO LOCATE DATA BOX
411 1 CD 4 CVLUE -
412 1 CD 4 ICNO -
413 1 CD 5
414 1 CD 5
415 1 CD 5
416 1 CD 5
417 1 CD 5
418 1 CD 5
419 1 CD 5
420 1 CD 5
421 1 CD 5
DEFIN IS AN FDS PROCESSOR SCHEDULED BY THE MANAGER

DEFIN ALLOCATES DATA ELEMENTS IN THE ANA THAT WERE SPECIFIED
BY THE PARAMETER KEYWORD DEFINE. IF THE DATA ELEMENT ALREADY
EXISTS, IT IS DELETED AND REALLOCATED. DATA ELEMENTS ARE
INITIALIZED TO NULL (CHARACTER STRINGS TO BLANKS).

INPUTS FROM THE MANAGER:

LU - LOGICAL UNIT OF THE USER'S TERMINAL
DEBUG - FLAGS FOR DEBUG

INPUTS FROM THE INTERFACE TABLE:

DEFINE - SYMBOLIC STRING CONTAINING DATA ELEMENT NAME(S),
OPTIMAL I AND J DIMENSIONS AND A REQUIRED TYPE

OUTPUTS TO THE ANA:

SET OF DATA ELEMENT(S) REQUESTED

INTERNAL VARIABLES:

INTERNAL TABLE HEADER
LENGTH OF SYMBOLIC STRING
POSITION WITHIN THE SYMBOLIC STRING
BUFFER FOR VALID NAMES TO BE ALLOCATED
BUFFER AREA FOR APGET AND XPRET USE
NUMBER OF ENTRIES IN THIS ANA REQUEST
NUMBER OF NAMES IN SYMBOLIC STRING
TOKEN POSITION FOR NEXT NAME
SYMBOLIC STRING INPUT TO DEFINE

EXTERNAL ROUTINES USED:

EXEC, IAMD, KCRT, KMPAR, XPIET, XPRE6, XPXIT, XUB6G, XIXFT, XIXS6

COPYRIGHT OF THE
ORIGINAL PAGE IS OWNED
1 BEGIN DEF
2 CALL RMPAR TO GET LU AND DEBUG FLAGS
3 CALL AGET TO RETRIEVE SYMBOLIC STRING
4 GET STRING LENGTH FROM INTERFACE TABLE HEADER
5 STARTSEARCH WHILE TOKEN-POSITION LT. STRING-LENGTH, OR
6 WHILE CURRENT-TOKEN NE. END-OF-STRING
7 PERFORM ZIDPN TO FIND THE NEXT NAME IN THE SYMBOLIC STRING
8 PERFORM ZIDPM TO PROCESS THE CURRENT NAME
9 EXIT IF THERE WAS AN ERROR RETURN FROM ZIDPM
10 CALL ZINGS TO DISPLAY SYNTAX ERROR AND POSITION IN SYMBOLIC STRING
11 SET PROCESSOR RETURN CODE TO ABEND
12 INCREMENT TO NEXT ELEMENT IN THE SYMBOLIC STRING
13 ENDLOP
14 SET PROCESSOR RETURN FOR NORMAL EXIT
15 ENDSRARY
16 SET OPTION SO XPRER WILL DO A QUEUE REQUEST
17 DO FOR NUMBER OF REQUESTS IN REQUEST BUFFER (NAMARP)
18 IF THIS IS LAST REQUEST, THEN
19 SET OPTION TO CLOSE REQUEST BUFFER
20 ENDIF
21 COMPUTE INDEX TO THIS REQUEST
22 CALL XPRER TO QUEUE THIS REQUEST
23 ENDSGO
24 CALL XPRIT TO RETURN TO THE MANAGER
25 1 END DEF
527 1 BEGIN XIDPM
528 2 SET IDIM AND JDIM TO 1
529 3 IF TOKEN IS NOT A NAME THEN
530 4 SET ERROR CODE
531 5 ELSE
532 6 MOV NAME INTO REQUEST
533 7 INCREMENT TO NEXT TOKEN
534 8 IF THERE ARE SUBSCRIPTS (TOKEN IS A LEFT PAREN) THEN
535 9 INCREMENT TO NEXT TOKEN
536 10 IF TOKEN IS NOT AN INTEGER OR
537 11 TOKEN IS NOT ZERO THEN
538 12 CALL XIMSG TO DISPLAY ERROR "INVALID IDIM"
539 13 EXIT TO :PNERR1:
540 14 ENDIF
541 15 SET IDIM TO THIS TOKEN
542 16 INCREMENT TO NEXT TOKEN
543 17 IF THERE ARE TWO SUBSCRIPTS (TOKEN IS A COMMA) THEN
544 18 INCREMENT TO NEXT TOKEN
545 19 IF TOKEN IS NOT AN INTEGER OR
546 20 TOKEN IS NOT ZERO THEN
547 21 CALL XIMSG TO DISPLAY ERROR "INVALID JOIN"
548 22 EXIT TO :PNERR1:
549 23 ENDIF
550 24 SET JDIM TO THIS TOKEN
551 25 INCREMENT TO NEXT TOKEN
552 26 ENIF
553 27 IF TOKEN IS NOT A RIGHT PAREN THEN
554 28 CALL XIMSG TO DISPLAY ERROR "INVALID SUBSCRIPT DELIMITER"
555 29 EXIT TO :PNERR1:
556 30 ENDIF
557 31 INCREMENT TO NEXT TOKEN
558 32 IF TOKEN IS NOT A BEGIN TYPE FIELD SLASH THEN
559 33 CALL XIMSG TO DISPLAY ERROR "INVALID OR MISSING TYPE FIELD"
560 34 EXIT TO :PNERR1:
561 35 ENDIF
562 36 INCREMENT TO NEXT TOKEN
563 37 IF TOKEN IS NOT A NAME THEN
564 38 CALL XIMSG TO DISPLAY ERROR "INVALID OR MISSING TYPE FIELD"
565 39 EXIT TO :PNERR1:
566 40 ENDIF
567 41 INCREMENT TO NEXT TOKEN
568 42 STARTSEARCH FOR ALL VALID TYPES
569 43 EXIT IF TYPE MATCHES THE TYPE IN THE SYMBOLIC STRING
570 44 SET TYPE AND CLASS IN REQUEST
571 45 COMPUTE SIZE AS IDIM * JDIM * LENGTH OF TYPE
572 46 IF SIZE IS TOO LARGE (>1200 WORDS) THEN
573 47 CALL XIMSG TO DISPLAY ERROR "DATA ELEMENT IS TOO LARGE"
574 48 EXIT TO :PNERR1:
575 49 ENDIF
576 50 ENDLOOP
577 51 CALL XIMSG TO DISPLAY ERROR "INVALID OR MISSING TYPE FIELD"
578 52 EXIT TO :PNERR1:
579 53 ENDOSEARCH
580 54 INCREMENT TO NEXT TOKEN
581 55 IF TOKEN IS NOT END OF TYPE FIELD SLASH OR
582 56 NEXT TOKEN IS NOT A COMMA THEN
583 57 CALL XIMSG TO DISPLAY WARNING "TYPE NOT TERMINATED BY A SLASH"
584 58 ENDIF
641 1 BEGIN ENDS
642 2 CALL RMRF TO GET INPUTS FROM MANAGER
643 2 SAVE REQUEST TO RETRIEVE $SCTB FROM ANA
644 2 CALL XVPMAM TO REQUEST ANA MANAGEMENT
645 2 ERREXIT IF $SCTB NOT FOUND TO :ERR4:
646 2 CALL EXEC TO READ IN $SCTB
647 2 SAVE REQUEST TO RETRIEVE SUMTAB IN REGBUF
648 2 CALL XVPMAM TO REQUEST ANA MANAGEMENT
649 2 ERREXIT IF NOT FOUND TO :ERR4:
650 2 CALL EXEC TO READ IN SUMTAB
651 2 IF SUMMARY TABLE IS LARGER THAN 32 ENTRIES THEN
652 2 SET SIZE OF SUMTAB TO 32 ENTRIES
653 2 ENDF
654 2 CALL WRITF TO WRITE SUMTAB TO DATBOX
655 2 ERREXIT IF WRITF ERROR TO :ERR4
656 2 IF THERE IS 1 VARIABLE AND XCUR IS CENTROID OR
657 3 THERE ARE 2 VARIABLES AND XCUR IS CENTROID AND YCUR IS CENTROID THEN
658 3 CALL READ TO READ HEADER RECORD
659 3 ERREXIT IF READF ERROR TO :ERR4
660 3 UPDATE NUMBER OF SUMMARY TABLE ENTRIES
661 3 CALL WRITF TO WRITE UPDATED HEADER
662 3 ERREXIT IF WRITF ERROR TO :ERR4:
663 3 CALL CLOSE TO CLOSE DATBOX
664 3 SAVE REQUEST TO DELETE ABS $SCTB IN REGBUF
665 3 IF THERE ARE REMAINING SCANS IN $SCTB THEN
666 4 CALL EXEC TO WRITE REMAINING $SCTB
667 4 SAVE REQUEST TO ALLOC AND STORE VALUES FOR NEW $SCTB
668 3 ENDF
669 3 SET RETURN PARAMETER TO NORMAL RETURN
670 3 ELSE
671 3 PERFORM SETTY
672 2 ENDF
673 2 CALL XVPMAM TO REQUEST ANA MANAGEMENT
674 2 CALL XPXIT TO TERMINATE WITH RETURN PARAMETERS
675 1 EXIT ENDS
676 2 :ERR4:
677 2 CALL XIMSG TO DISPLAY ERROR
678 2 CALL XPXIT TO ABEND PROCESSOR
679 1 END ENDS
1 BEGIN SETXY
2 IF XCUR IS END STEP THEN
3 IF THERE IS 1 VARIABLE THEN
4 SET X TO CENTROID
5 CALL POSNT TO POSITION FILE TO CENTROID RECORD
6 ERREXIT IF POSNT ERROR TO :ERR4:
7 ELSE
8 IF YCUR IS END STEP THEN
9 SET XCUR TO ZERO
10 SET X TO CENTROID
11 SET Y TO CENTROID
12 CALL POSNT TO POSITION TO CENTROID RECORD
13 ERREXIT IF POSNT ERROR TO :ERR4:
14 ELSE
15 SET XCUR TO (-XSTEP)
16 IF XSTEP IS ZERO AND Y IS CENTROID THEN
17 INCREMENT YCUR BY 1
18 CALL WRITF TO WRITE DUMMY AS CENTROID RECORD
19 ERREXIT IF WRITF ERROR TO :ERR4:
20 ENDIF
21 COMPUTE X AS (XCENT + XINCR + FLOAT (XCUR))
22 COMPUTE Y AS (YCENT + YINCR + FLOAT (YCUR))
23 ENDIF
24 ENDIF
25 ELSE
26 INCREMENT XCUR BY 1
27 IF THERE IS 1 VARIABLE AND X IS THE CENTROID OR
28 THERE ARE 2 VARIABLES AND X IS THE CENTROID AND Y IS THE CENTROID THEN
29 INCREMENT XCUR BY 1
30 CALL WRITF TO WRITE DUMMY AS CENTROID RECORD
31 ERREXIT IF WRITF ERROR TO :ERR4:
32 ENDIF
33 COMPUTE X AS (XCENT + XINCR + FLOAT (XCUR))
34 IF THERE ARE 2 VARIABLES THEN
35 COMPUTE Y AS (YCENT + YINCR + FLOAT (YCUR))
36 ENDIF
37 ENDIF
38 ENDIF
39 DO FOR # SCAN VARIABLES
40 CALL EXEC TO WRITE VARIABLE
41 SAVE REQUEST TO STORE VALUES FOR VARIABLE IN RERBUF
42 ENDDO
43 CALL EXEC TO WRITE $SCNTB
44 SAVE REQUEST TO STORE NEW $SCNTB
45 SET RETURN PARAMETERS TO RESET SEQUENCE NUMBER
46 END SETXY
SCN IS AN FDS PROCESSOR SCHEDULED BY THE MANAGER

SCAN PROCESSOR EXECUTES A SERIES OF PROCESSORS ITERATING ON XSCAN AND TSCAN (IF ENTERED) VALUES COMPUTED USING THE CENTROID, THE INCREMENT, AND THE CURRENT STEP NUMBER, CREATING A DATA BOX FILE.

INPUTS FROM MANAGER:
- LU - LOGICAL UNIT OF THE USER'S TERMINAL
- USEID - USER ID CODE
- FLAGS -FLAGS FOR DEBUG
- ENTDIS -DISPLACEMENT OF THIS SCAN ENTRY IN BSEGTB

INPUTS FROM INTERFACE TABLE:
- PROCON-CARTRIDGE # FOR DATA BOX FILE
- SUMMARY-TABLE
- *DATBOX-NAME OF DATA BOX ENTERED
- NVAR - NUMBER OF SCAN VARIABLES
- *SCAN - NAME OF X SCAN VARIABLE
- XUNIT - UNIT OF X VARIABLE
- XCENT - CENTROID OF X VARIABLE
- ZINC - INCREMENT FOR X
- XSTEPS - NUMBER OF STEPS FOR X
- *TSCAN - NAME OF Y SCAN VARIABLE
- YUNIT - UNIT OF Y VARIABLE
- YCENT - CENTROID OF Y VARIABLE
- TINC - INCREMENT FOR Y
- YSTEPS - NUMBER OF STEPS FOR Y

* - ACTUALLY OUTPUTS, ONLY NAME ENTERED IS AN INPUT TO SCAN

OUTPUTS TO MANAGER
- XZRET - RETURN CODE TO MANAGER

OUTPUTS TO WORK AREA (CAW):
- DATBOX-FILE WHERE SUMMARY TABLE IS WRITTEN
- XSCAN - X SCAN VARIABLE
- YSCAN - Y SCAN VARIABLE
- BSCNTB - SCAN CONTROL TABLE (173 WORDS PER ACTIVE SCAN)

1. SUMMARY-TABLE
2. *DATBOX-NAME OF DATA BOX ENTERED
3. NVAR - NUMBER OF SCAN VARIABLES
4. *SCAN - NAME OF X SCAN VARIABLE
5. XUNIT - UNIT OF X VARIABLE
6. XCENT - CENTROID OF X VARIABLE
7. ZINC - INCREMENT FOR X
8. XSTEPS - NUMBER OF STEPS FOR X
9. *TSCAN - NAME OF Y SCAN VARIABLE
10. YUNIT - UNIT OF Y VARIABLE
11. YCENT - CENTROID OF Y VARIABLE
12. TINC - INCREMENT FOR Y
13. YSTEPS - NUMBER OF STEPS FOR Y

* - ACTUALLY OUTPUTS, ONLY NAME ENTERED IS AN INPUT TO SCAN
789 1 CD3  (18) XSTEP # STEPS OF X
790 1 CD3  (19) XCUR CURRENT X STEP NUMBER
791 1 CD3  (20) YSCAN NAME OF Y VARIABLE
792 1 CD3  (23) YDISPL DISPLACEMENT FOR Y
793 1 CD3  (24) YCENT CENTROID OF Y
794 1 CD3  (26) YINC INCREMENT FOR Y
795 1 CD3  (28) YSTEP # STEPS OF Y
796 1 CD3  (29) YCUR CURRENT Y STEP NUMBER
797 1 CD3  (30) IDCB 144 WORD DCB FOR DATBOX
798 1 CD********

RTE SUBROUTINES USED:
800 1 CD5  CLOSE, CREAT, EXEC, FLOAT, LAND,
801 1 CD5  KCVT, MOD, POSINT, PURGE, RMPAR,
802 1 CD5  WRITF
803 1 CD5  FDT SUBROUTINES USED:
804 1 CD5  XPATR, XPGT, XPPUT, XPYIT, XRCPR,
805 1 CD5  XNEXT, XMOV, XRDFM, XRUNG, XUDBG,
806 1 CD5  XPAW, XIMSG
BEGIN SCAN
CALL RMPAR TO RECEIVE INPUTS FROM MANAGER (LU, FLAGS, ENTRY DISPLACEMENT)
CALL XPGET TO GET PROCON AND # SCAN VARIABLES
ERR EXIT IF # SCAN VARIABLES < 1 OR > 2 TO :ERR2:
GET SUMMARY TABLE NAME AND DISPLACEMENT FROM INTERFACE TABLE
ERR EXIT IF SINTAB IS A LITERAL TO :ERR3:
ERR EXIT IF DISPLACEMENT IS NOT AN ELEMENT BOUNDARY TO :ERR3:
GET DATA BOX NAME FROM INTERFACE TABLE
DO FOR # SCAN VARIABLES
CALL XPATR TO GET NAME AND DISPLACEMENT
CALL SUBSCRIPT FROM DISPLACEMENT AND IDIN
CALL XPGET TO GET UNITS, CENTROID, INCR, # STEPS
ERR EXIT IT # STEPS < 0 OR > 2 TO :ERR3:
ENDDO
SAVE REQUEST TO RETRIEVE VALUES FOR ESER7B AND ECSNTB
CALL XPAM TO REQUEST AW4 MANAGEMENT
CALL EXEC TO READ IN ESER7B
IF ECSNTB NOT FOUND THEN
SET # SCAN TO ZERO
ELSE
SET # SCAN TO (TOTAL SIZE OF ECSNTB / SIZE OF ONE SCAN ENTRY)
CALL EXEC TO READ IN ECSNTB
ERR EXIT IF # SCAN IS MAXIMUM ALLOWED (4) TO :ERR3:
ERR EXIT IF THIS DATBOX NAME IS ALREADY IN USE TO :ERR3:
ENDIF
ERR EXIT IF THIS IS THE LAST ENTRY IN ESER7B TO :ERR3:
GET THE SEQUENCE NUMBER OF THIS SCAN FROM ESER7B
IF THE DISPLACEMENT OF THIS SCAN IS ZERO THEN
SEARCH ESER7B FOR THE SEQUENCE NUMBER
ERR EXIT IF THIS SCAN IS THE LAST ENTRY IN ESER7B TO :ERR3:
IF THIS IS A SEMI OVERRIDE (2 PROCESSOR NAMES NOT EQUAL) THEN
SET RESET NUMBER TO THIS ENTRY SEQUENCE NUMBER
ELSE
SET RESET NUMBER TO NEXT ENTRY SEQUENCE NUMBER
ENDIF
ELSE
SET RESET NUMBER TO NEXT ENTRY SEQUENCE NUMBER
ENDIF
ENDIF
COMPUTE SIZE OF DATBOX FILE = (2 + (2 + XSTEPS + 1) + (2 + YSTEPS + 1) + 1) / 2
COMPUTE CENTROID RECORD NUMBER = SIZE + 3
DO FOR # SCAN VARIABLES
COMPUTE BEGINNING VALUE = (CENT + INCR + FLOAT (CUR STEPS))
ENDDO
CALL XPPUT TO STORE DATBOX AND SCAN VARIABLES
CALL CREAT TO CREATE DATBOX FILE
IF FILE ALREADY EXISTS THEN
CALL PURGE TO PURGE FILE
ERR EXIT IF PURGE ERROR TO :ERR2:
CALL CREAT TO CREATE FILE
ENDIF
ERR EXIT IF CREAT ERROR TO :ERR2:
CALL WRIT3 TO WRITE HEADER RECORD TO DATBOX
ERR EXIT IF WRITE ERROR TO :ERR2:
CALL POSMT TO POSITION FILE TO FIRST DATA RECORD
ERR EXIT IF NEXT ERROR TO :ERR2:
CALL EXEC TO WRITE ECSNTB
SAVE REQUEST TO DELETE/VERIFY ABSENT ECSNTB IN REQBUF
SAVE REQUESTS TO ALLOC AND STORE VALUES FOR NEW ECSNTB
CALL XPAM TO REQUEST AW4 MANAGEMENT
873 2 ERREXIT IF NO ANA SPACE TO :ERR1:
874 2 CALL $EXIT TO EXIT NORMALLY
875 1 EXIT SCAN
876 2 :ERR1:
877 2 IF THERE ARE MORE ACTIVE SCANS (# SCANS > 0) THEN
878 3 5 CALL EXEC TO READ IN NEW $SCNTB
879 3 5 CALL EXEC TO WRITE OUT ORIGINAL $SCNTB
880 3 5 SAVE REQUESTS TO ALLOC AND STORE VALUES FOR ORIGINAL $SCNTB
881 2 ENDIF
882 2 :ERR2:
883 2 CALL CLOSE TO CLOSE DATBOX
884 2 CALL PURGE TO PURGE DATBOX
885 2 SET VALUE FOR XPPUT
886 2 SAVE REQUEST TO DELETE DATBOX FROM ANA
887 2 CALL XPPAN TO REQUEST ANA MANAGEMENT
888 2 :ERR3:
889 2 CALL XIMSG TO DISPLAY ERROR
890 2 CALL XEXIT TO ABEND SCAN
891 1 END SCAN
1 BEGIN VICHR
2 2 SET # WORDS TO BE STORED IN OBJECT = # WORDS/ELEMENT FOR OBJECT
3 2 * CASE (RESULT DATA TYPE) :CHSTR:, :CHRFRE:, :CHRFRE:, :FIXERR:, 4 2 :FIXERR:, :FIXERR:
5 2 * :CHSTR:
6 2 3 DETERMINE # WORDS IN CHARACTER STRING
7 2 3 MOVE CHARACTER STRING TO RESULT LOCATION
8 2 3 IF # WORDS IN STRING < # WORDS/ELEMENT FOR OBJECT, THEN
9 2 4 BLANK FILL AFTER CHARACTER STRING
10 2 3 ENDIF
11 2 :CHRFRE:
12 2 3 IF TOP ENTITY ON RESULT STACK IS A DISPLACEMENT (TYPE = -2), THEN
13 2 4 POP DISPLACEMENT FROM RESULT STACK
14 2 4 ELSE
15 2 4 SET DISPLACEMENT = 0
16 2 4 ENDIF
17 2 4 IF RESULT OPERAND IS FREE, THEN
18 2 5 SET # WORDS TO BE RETRIEVED TO # WORDS/ELEMENT FOR OBJECT
19 2 5 ELSE CHARACTER = CHARACTER
20 2 5 SET # WORDS TO BE RETRIEVED TO # WORDS/ELEMENT FOR OBJECT
21 2 5 ENDIF
22 2 5 CALL XPRET TO RETRIEVE DATA FROM RESULT OPERAND AT DISPLACEMENT DETERMINED
23 2 5 IF # WORDS RETRIEVED IS < # WORDS TO BE STORED, THEN
24 2 6 BLANK FILL REMAINING DATA
25 2 6 ENDIF
26 2 6 ENDIF
27 2 6 EMDCASE
28 2 1 EXIT VCHR
2 2 :FIXERR:
2 2 2 SET MESSAGE TO BE OUTPUT TO "CHARACTER DATA ELEMENT CANNOT BE SET EQUAL TO
2 2 2 NUMERICAL DATA"
2 2 3 CALL XING'S TO OUTPUT MESSAGE TO USER
2 2 3 CALL XLIST'S TO LIST SYMBOLIC STRING
2 2 3 CALL XEXIT'S TO EXIT PROCESSOR
2 1 END VCHR
XZDIM - DDBSP INPUT PROCESSOR

XZDIM IS CALLED TO INTERPRET THE VARIOUS INTERFACE TABLE INPUTS
(MOSTLY SYMBOIC STRINGS) AND BUILD DATA ARRAYS FROM THEM.

ALL INPUT COMES FROM THE 26 INTERFACE TABLE ARGUMENTS

OUTPUT

COMMON

MCVARL, NCRELL, CVALUE, MDVARS, MDVRUL, NC

NOTES

USES ROUTINES

EXEC

XPGET

XPIX

XMOV

XLLIS

963 1 C3-------
964 1 C90
965 1 C93
966 1 C90
967 1 C90
968 1 C90
969 1 C90
970 1 C90
971 1 C90
972 1 C90
973 1 C90
974 1 C90
975 1 C90
976 1 C90
977 1 C90
978 1 C90
979 1 C90
980 1 C90
981 1 C90
982 1 C90
983 1 C90
984 1 C90
985 1 C90
986 1 C90
987 1 C90
988 1 C90
989 1 C90
990 1 C90
991 1 C90
992 1 C90
993 1 C90
994 1 C90
995 1 C90
996 1 C90
997 1 C90

XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
XZDIM
FORTRAN CALLING SEQUENCE:

CALL ZIDFT (ARRAY, START, END, TOKEN, INDEX)

ZIDFT searches 'ARRAY' beginning at 'START' and ending at 'END'
for a specified 'TOKEN'. It returns either the position of
the 'TOKEN' or the 'END' value in 'INDEX'.

INPUTS FROM CALLING SEQUENCE:

1. ARRAY - SYMBOLIC STRING TO BE SEARCHED
2. END - ENDING INDEX INTO 'ARRAY'
3. START - BEGINNING INDEX INTO 'ARRAY'
4. TOKEN - TOKEN TO BE SEARCHED FOR

OUTPUTS TO CALLING SEQUENCE:

1. INDEX - POSITION OF 'TOKEN' IN SYMBOLIC STRING IF FOUND,
   OR 'END' IF NOT FOUND
1083 1 BEGIN XZDFT
1084 2 SET INDEX TO START
1085 3 DO UNTIL INDEX .GE. END
1086 4 IF ARRAY(INDEX) .EQ. TOKEN, THEN
1087 5 ELSE SET END TO INDEX
1088 6 CASE ARRAY(INDEX)+1 (  
1089 7 ADD1: 0 END OF STRING
1090 8 ADD2: 1 INTEGER
1091 9 ADD3: 2 REAL
1092 10 ADD4: 3 DOUBLE
1093 11 ADD1: 4 UNKNOWN
1094 12 ADD4: 5 NAME
1095 13 ADD1: 6 UNKNOWN
1096 14 ADD1: 7 UNKNOWN
1097 15 CALST: 8 CHARACTER STRING
1098 16 ADD1: 9 UNKNOWN
1099 17 ADD1: 10 +
1100 18 ADD1: 11 -
1101 19 ADD1: 12 *
1102 20 ADD1: 13 /
1103 21 ADD1: 14 <
1104 22 ADD1: 15 >
1105 23 ADD1: 16 =
1106 24 ADD1: 17 &
1107 25 ADD1: 18 %
1108 26 ADD1: 19 ?
1109 27 ADD1: 20 ?
1110 28 ADD1: 21 !
1111 29 ADD1: 22 )
1112 30 ADD2: 23 START OF SYMBOLIC STRING
1113 31 ADD1: 24 END OF STRING
1114 32 ADD1: 25 \n
REPRODUCIBILITY OF THE ORIGIINAL PAGE 18
<table>
<thead>
<tr>
<th>CD</th>
<th>1</th>
<th>CD</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1139</td>
<td></td>
<td>1140</td>
<td></td>
</tr>
<tr>
<td>1141</td>
<td></td>
<td>1142</td>
<td></td>
</tr>
<tr>
<td>1143</td>
<td></td>
<td>1144</td>
<td></td>
</tr>
<tr>
<td>1145</td>
<td></td>
<td>1146</td>
<td></td>
</tr>
<tr>
<td>1147</td>
<td></td>
<td>1148</td>
<td></td>
</tr>
<tr>
<td>1149</td>
<td></td>
<td>1150</td>
<td></td>
</tr>
<tr>
<td>1151</td>
<td></td>
<td>1152</td>
<td></td>
</tr>
<tr>
<td>1153</td>
<td></td>
<td>1154</td>
<td></td>
</tr>
<tr>
<td>1155</td>
<td></td>
<td>1156</td>
<td></td>
</tr>
<tr>
<td>1157</td>
<td></td>
<td>1158</td>
<td></td>
</tr>
<tr>
<td>1159</td>
<td></td>
<td>1160</td>
<td></td>
</tr>
<tr>
<td>1161</td>
<td></td>
<td>1162</td>
<td></td>
</tr>
<tr>
<td>1163</td>
<td></td>
<td>1164</td>
<td></td>
</tr>
<tr>
<td>1165</td>
<td></td>
<td>1166</td>
<td></td>
</tr>
<tr>
<td>1167</td>
<td></td>
<td>1168</td>
<td></td>
</tr>
<tr>
<td>1169</td>
<td></td>
<td>1170</td>
<td></td>
</tr>
<tr>
<td>1171</td>
<td></td>
<td>1172</td>
<td></td>
</tr>
<tr>
<td>1173</td>
<td></td>
<td>1174</td>
<td></td>
</tr>
<tr>
<td>1175</td>
<td></td>
<td>1176</td>
<td></td>
</tr>
<tr>
<td>1177</td>
<td></td>
<td>1178</td>
<td></td>
</tr>
<tr>
<td>1179</td>
<td></td>
<td>1180</td>
<td></td>
</tr>
<tr>
<td>1181</td>
<td></td>
<td>1182</td>
<td></td>
</tr>
<tr>
<td>1183</td>
<td></td>
<td>1184</td>
<td></td>
</tr>
<tr>
<td>1185</td>
<td></td>
<td>1186</td>
<td></td>
</tr>
<tr>
<td>1187</td>
<td></td>
<td>1188</td>
<td></td>
</tr>
<tr>
<td>1189</td>
<td></td>
<td>1190</td>
<td></td>
</tr>
<tr>
<td>1191</td>
<td></td>
<td>1192</td>
<td></td>
</tr>
<tr>
<td>1193</td>
<td></td>
<td>1194</td>
<td></td>
</tr>
<tr>
<td>1195</td>
<td></td>
<td>1196</td>
<td></td>
</tr>
<tr>
<td>1197</td>
<td></td>
<td>1198</td>
<td></td>
</tr>
</tbody>
</table>

**XDP1** - DATA BOX DISPLAY OVERLAY - PREPARES DATA FOR DISPLAY
- SCHEDULED BY DBDSP

**XDP1** READS IN THE SCAN CONTROL INFORMATION AND THE SUMMARY TABLES PRODUCED BY SCAN/ENDSCAN FOR FURTHER PROCESSING. IN ADDITION, XDP1 VERIFIES THE DISPLAY DEPENDANT VARIABLE NAMES AND THE CONSTRAINT VARIABLE NAMES ARE CONTAINED IN THE DEPENDANT VARIABLE NAME LIST GENERATED DURING THE SCAN. XDP1 ALSO GENERATES CONSTRAINT MASKS WITH SUBROUTINE XDMMK FOR SUBSEQUENT DISPLAY BY XDP2 AND XDP3.

**INPUTS FROM THE DATA BOX**

**RECORD 1**
- (1) - NAME OF FDS PROCESSOR CREATING FILE
- (2) - INTERFACE TABLE VARIABLE NAME FOR THIS FILE
- (3) - NAME OF FDS PROCESSOR UPDATING FILE
- (4) - INTERFACE TABLE VARIABLE NAME ( 3 ASCII WORDS OF BLANKS )
- (5) - INTERFACE TABLE VARIABLE NAME FOR THIS U-DATE ( 3 ASCII WORDS OF BLANKS )

**RECORD 2**
- (1) - NO OF ENTRIES IN SUMMARY TABLE
- (2) - X SCAN VARIABLE ( 6 CHAR )
- (3) - X FIRST SUBSCRIPT ( INT OR ZERO )
- (4) - X SECOND SUBSCRIPT ( INT OR ZERO )
- (5) - X UNITS ( 6 CHAR )
- (6) - X CENTROID ( REAL )
- (7) - X INCREMENT ( REAL )
- (8) - X NUMBER OF STEPS ( INTEGER 1-5 )
- (9) - Y SCAN VARIABLE ( 6 CHAR )
- (10) - Y FIRST SUBSCRIPT ( INT OR ZERO )
- (11) - Y SECOND SUBSCRIPT ( INT OR ZERO )
- (12) - Y UNITS ( 6 CHAR )
- (13) - Y CENTROID ( REAL )
- (14) - Y INCREMENT ( REAL )
- (15) - Y NUMBER OF STEPS ( INTEGER 1-5 )

DEPENDENT VARIABLE NAMES AND UNITS IN RECORDS 3, 4, 8, 9

**SUMMARY TABLE RECORDS**
- EACH SUMMARY TABLE CONTAIN VALUE FOR EACH
DEPANANT VARIABLE SCANNED (UP TO 32 VALUES
INCLUDING ERROR FLAG WHICH IS FIRST VALUE
IN SUMMARY TABLE)

OUTPUT FROM XZDP!

MAST TABLES CONTAINING THE CONSTRAINT MASKS FOR EACH
CONSTRAINT WHICH WAS VIOLATED AND INDICATION OF
WHETHER OR NOT ANY CONSTRAINT WAS VIOLATED FOR EACH
ARRAY COORDINATE

LIST OF VALID DISPLAY DEPENDANT VARIABLES FOR EACH
PAGE (UP TO 16 PAGES)

LIST OF VALID CONSTRAINT VARIABLES WHICH WERE VIOLATED
(UP TO 8 CONSTRAINTS)

DATA BOX DISPLAY ARRAY VALUES FOR ALL VARIABLES

MASK1 - ARRAY CONTAINING CONSTRAINTS 1 THROUGH

DATA BOX - NAME OF DATA BOX TO BE DISPLAYED BY DDSP

NAME LIST FOR VARIABLES SCANNED BY SCAN/ENDSCAN

NUMBER OF CONSTRAINTS INPUT BY USER (INTEGER)

LIST OF CONSTRAINT RELATIONS INPUT BY USER

LIST OF CONSTRAINT VARIABLE NAMES INPUT BY USER (32 MAX)

NUMBER OF DEP DISP VARIABLE PAIRS FOR PAGED OUT PUT (1-16PR)

NUMBER OF DEP DISP VARIABLE PAIR SCALE FACTORS FOR

SET OF INDICATORS FOR CONSTRAINTS VIOLATED=1 NOT 0

NUMBER OF STEPS ON EITHER SIDE OF X CENTROID (0 TO 5)

NUMBER OF STEPS ON EITHER SIDE OF Y CENTROID (0 TO 5)

NAME OF X VALUES FOR X COORDINATES (1 - 11 REAL)

NAME OF X VAR SCANNED TO BE PLACED ON DISPLAY (6 CHAR)

NAME OF X VAR UNITS TO BE PLACED ON DISPLAY (6 CHAR)

LIST OF X VAR VALUES FOR Y COORDINATES (1 - 11 REAL)

NAME OF Y VAR SCANNED TO BE PLACED ON DISPLAY (6 CHAR)

NAME OF Y VAR UNITS TO BE PLACED ON DISPLAY (6 CHAR)

TABLE IN COMMON FOR SUMTAB VARIABLE NAMES AND UNITS

UNITS LIST FOR VARIABLES SCANNED BY SCAN/ENDSCAN

VALUES FOR SCAN VARIABLES(s) - 1 TO 32 VALUES/RECORD

COMMUNICATION BUFFER FOR RMAP - LU, USER ID, FLAG

LOGICAL UNIT # FOR XROM CALLING SEQUENCE - USER LOCATN

DBDSP - DBDSP WILL OUTPUT DISPLAY TO THIS USER SUPPLIED LU

PROMPT - TABLE IN COMMON TO COMMUNICATE WITH XPRDM

SELECT =0 PORMPT; SELECT NOT 0 RUN ALL DISPLAYS TO O/P
XZDP2 - DATA BOX DISPLAY OVERLAY - PROMPTS USER FOR DISPLAY
DESIRED, THEN FORMATS AND DISPLAYS DATA ACCORDINGLY

- SCHEDULED BY DBDSP

XZDP2 ISSUES PROMPTS TO THE USER REQUESTING PAGE NUMBER,
STARTING ROW NUMBER, NUMBER OF ROWS, AND DESIRED
LU. SELECT OPTION CAN BE SET WHEN ENTERING
DDDSP IN THE INTERFACE TABLE TO OUTPUT ALL PAGES
TO NAMED LU.
XZDP2 UTILIZES SUBROUTINE XZDOT TO ACTUALLY FORMAT
AND DISPLAY THE SCAN RESULTS.

INPUTS TO XZDP2 FROM XZD1

- MASK TABLES CONTAINING THE CONSTRAINT MASKS FOR EACH
- CONSTRAINT WHICH WAS VIOLATED AND INDICATION OF
- WHETHER OR NOT ANY CONSTRAINT WAS VIOLATED FOR EACH
- ARRAY COORDINATE

LIST OF VALID DISPLAY DEPENDANT VARIABLES FOR EACH
PAGE (UP TO 16 PAGES)

DATA BOX DISPLAY ARRAY VALUES FOR ALL VARIABLES

OUTPUT FROM XZDP2

DISPLAY FORMAT SHOWN IN DOCUMENTATION IS SENT TO
NAME LU DEVICE

COMBUF - BUFFER TO RECEIVE ENCODED USER RESPONSE - XROM
DATBOX - NAME OF DATA BOX TO BE DISPLAYED BY DBDSP
DATBUF - COMMON BUFFER FOR SCAN SUMMARY DESCRIPTORS

IDV1 - POINTER TO DEP. DISPLAY VARIABLE NAME LIST FOR FIRST VAR
IDV2 - POINTER TO DEP. DISPLAY VARIABLE NAME LIST FOR 2ND VAR.
ISAVE - LIST OF VIOLATED CONSTRAINTS BUILT BY XZMSK (MAX OF B)
IXSCN1 - FIRST SUBSCRIPT FOR NAME OF X SCAN VARIABLE (INT OR O)
IXSCV2 - SECOND SUBSCRIPT FOR NAME OF X SCAN VARIABLE (INT OR O)
IXSCW2 - SECOND SUBSCRIPT FOR NAME OF Y SCAN VARIABLE (INT OR O)
LBUF - NAME OF COMMON AREA USED FOR INTERFACE TABLE

TOTAL LENGTH OF COMMON AREA - XROM

MAST - ARRAY CONTAINING MASKS FOR UP TO 4 COMMON X, 128 SIZE)

NAMVAR - NAME LIST FOR VARIABLES SCANNED BY SCAN/ENDSCAN

NC - NUMBER OF CONSTRAINTS INPUT BY USER (INTEGER)
NCREL - LIST OF CONSTRAINT RELATIONS INPUT BY USER

NCVARS - LIST OF CONSTRAINT VARIABLE NAMES INPUT BY USER (32 MAX)
1461   1 CD 4 NDVARL - LIST OF DEP LISP VARIABLE PAIRS FOR PAGED OUTPUT(1-16PR) XZDP2
1462   1 CD 4 NDVRCY - NUMBER OF DEP LISP VAR IN NDVARL LIST (INTEGER) XZDP2
1463   1 CD 4 NDVRLU - LIST OF DEP LISP VARIABLE PAIR SCALE FACTORS FOR O/P XZDP2
1464   1 CD 4 MSKERR - SET OF INDICATORS FOR CONSTRAINTS VIOLATED=O/HOT 0 XZDP2
1465   1 CD 4 NSTEP - NUMBER OF STEPS ON EITHER SIDE OF X CENTROID (0 TO 5) XZDP2
1466   1 CD 4 NSTEP - NUMBER OF STEPS ON EITHER SIDE OF Y CENTROID (0 TO 5) XZDP2
1467   1 CD 4 RETC - RTN CODE FROM XPRDMX; 0=NRML, 1=RTN TO EXEC, 2=NL FBR, 3=ERR XZDP2
1468   1 CD 4 STRING - CHAR STRING CONTAINING USER PROMPT MESSAGE XZDP2
1469   1 CD 4 TCOORD - LIST OF X VAR VALUES FOR X COORDINATES (1 - 11 REAL) XZDP2
1470   1 CD 4 XSCNMN - NAME OF X VAR SCANNED TO BE PLACED ON DISPLAY (6 CHAR) XZDP2
1471   1 CD 4 XUNITS - NAME OF X VAR UNITS TO BE PLACED ON DISPLAY (6 CHAR) XZDP2
1472   1 CD 4 TCOORD - LIST OF X VAR VALUES FOR Y COORDINATES (1 - 11 REAL) XZDP2
1473   1 CD 4 YSCNMN - NAME OF Y VAR SCANNED TO BE PLACED ON DISPLAY (6 CHAR) XZDP2
1474   1 CD 4 YUNITS - NAME OF Y VAR UNITS TO BE PLACED ON DISPLAY (6 CHAR) XZDP2
1475   1 CD 4 LIST - TABLE IN COMMON FOR SUNTAB VARIABLE NAMES AND UNITS XZDP2
1476   1 CD 4 NVARUL - UNITS LIST FOR VARIABLES SCANNED BY SCAN/ENDSCN XZDP2
1477   1 CD 4 SUNTAB - VALUES FOR SCAN VARIABLE(S) - 1 TO 32 VALUES/RECORD XZDP2
1478   1 CD 4 PARMS - COMMUNICATION BUFFER FOR RMPAR - LU, USER ID, FLAGS XZDP2
1479   1 CD 4 LU - LOGICAL UNIT # FOR XPRDM CALLING SEQUENCE - USER LOCAT W XZDP2
1480   1 CD 4 LUSP - DBDSP WILL OUTPUT DISPLAY TO THIS USER SUPPLIED LU XZDP2
1481   1 CD 4 PROMPT - TABLE IN COMMON TO COMMUNICATE WITH XPRDM XZDP2
1482   1 CD 4 SELECT - SELECT =0 PROMPT ; SELECT NOT 0 RUN ALL DISPLAYS TO O/P XZDP2
1483   1 CD 4 WITHOUT PROMPT XZDP2
1484   1 CD 4 CARTBG - CARTAGE USED TO LOCATE DATA BOX XZDP2
1485   1 CD ******** USES ROUTINES XZDP2
1486   1 CD 5 XPRDM, EXEC, XZDOT XZDP2
1487   1 CD 5 XPRDM, EXEC, XZDOT XZDP2
1488   1 CD 5 XPRDM, EXEC, XZDOT XZDP2
1489   1 CD 5 XPRDM, EXEC, XZDOT XZDP2
1490   1 CD 5 XPRDM, EXEC, XZDOT XZDP2
1491   1 CD 5 XPRDM, EXEC, XZDOT XZDP2
1492   1 CD ******** XZDP2
1591 1 CD******
1592 1 CD0
1593 1 CD0 FORTRAN CALLING PROCEDURE
1594 1 CD0 CALL XZISP (STRING, LEN)
1595 1 CD0
1596 1 CD******
1597 1 CD1
1598 1 CD1 XZISP REMOVES DUPLICATE (I.E. CONSECUTIVE) BLANKS FROM
1599 1 CD1 A CHARACTER STRING AND FILLS THE VACATED TRAILING WORDS
1600 1 CD1 WITH BLANKS
1601 1 CD1
1602 1 CD******
1603 1 CD2
1604 1 CD2 INPUT
1605 1 CD2
1606 1 CD2 CALLING SEQUENCE
1607 1 CD2 STRING - INPUT CHARACTER STRING
1608 1 CD2 LEN - NUMBER OF WORDS IN STRING
1609 1 CD2
1610 1 CD2
1611 1 CD******
1612 1 CD3
1613 1 CD3 OUTPUT
1614 1 CD3
1615 1 CD3 CALLING SEQUENCE
1616 1 CD3 STRING - CHARACTER STRING WITH ALL FIELDS OF CONSECUTIVE
1617 1 CD3 BLANKS REDUCED TO 1 BLANK AND TRAILING BLANK FILL
1618 1 CD3 LEN - NO. OF WORDS IN STRING PRIOR TO TRAILING BLANK FILL
1619 1 CD3
1620 1 CD3
1621 1 CD******
1622 1 CD4
1623 1 CD4 NOTES
1624 1 CD4
1625 1 CD4 USES Routines
1626 1 CD4
1627 1 CD4 XRISP
1628 1 CD4
1629 1 CD******
1630 1 BEGIN XZISP
1631 2 CALL XRISP TO REMOVE DUPLICATE BLANKS FROM STRING
1632 2 DO WHILE THERE ARE TRAILING WORDS IN STRING
1633 2 SET THIS TRAILING WORD TO BLANKS
1634 2 ENDDO
1635 1 END XZISP
FORTRAN CALLING SEQUENCE:

CALL XIFCL (LU)

XIFCL SEARCHES XVSTB LOOKING FOR LU SO THAT THE
PROCESSOR'S CLASS NUMBER CAN BE EXTRACTED. IT
SAVES THE CLASS NUMBER IN EXTERNAL XPCLS.

INPUTS FROM CALLING SEQUENCE:

LU - LOGICAL UNIT OF TERMINAL BEING USED.

NOTE: CALLING PROGRAM MUST HAVE XPCLS DEFINED
AS AN EXTERNAL REFERENCE.

CALL .ENTRY TO RESOLVE PARAMETER ADDRESS
START SEARCH FOR ALL XVSTB ENTRIES
EXIT IF THIS ENTRY'S LU IS MINE
END SEARCH
GET CLASS NUMBER FROM XVSTB
SAVE CLASS NUMBER IN XPCLS
END XIFCL
FORTRAN CALLING PROCEDURE:

CALL XIFNC(ENTRY)

XIFNC IS USED BY THE ASSIGN ROUTINE XZPS2 TO EVALUATE FUNCTION OPERATIONS

INPUTS

ENTRY - FUNCTION TOKEN CURRENTLY BEING PROCESSED
FROM ASGCOM - LU, STRNG, OPRNDS, OPIINFO

OUTPUTS TO ASGCOM
RESULT, RSLTPT, OPRNDS

EXTERNAL REFERENCES
FDI - XPXIT, XRMOV, XILSS, XZMSG

RTE - ABS, AINT, ALOG, ALGOT, AMOD, ATAN, ATAN2, COS, DABS, DATAN,
DATN2, DBLE, DCOS, DINT, DEXP, DLOG, DLGOT, DMOD, DSI-W, DSIM, DSQRT,
EXP, FLPOT, IABS, IINT, IFIX, ISIGN, MOD, OVF, SIGN, SIM, SNGLE, SQRT,
TAN, TANH
FORTRAN CALLING PROCEDURE:

CALL XIFRE

XIFRE IS USED BY THE ASSGN ROUTINE XPS2 TO PROCESS DATA ASSIGNMENTS
FOR FREE-TYPE OBJECT DATA ELEMENTS
BEGIN XFR

/ CASE (RESULT DATA TYPE) :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

/ : FREFI : FREFI :

END XFR

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR
FORTRAN CALLING PROCEDURE:

CALL XIFXO

XIFXO IS USED BY THE ASGRT ROUTINE XZPS2 TO PROCESS DATA ASSIGNMENTS
FOR FIXED-TYPE OBJECT DATA ELEMENTS

INPUTS FROM ASGCOM
LU,SYMTAB,SSTRNG,RESULT,RSLTPT,DATYPES,CLSTAN

OUTPUTS TO ASGCOM
RSLTPT,REQST,NUMWDS,OPFRNS

FDS - XPREQ,XPXIT,XRMV,XILSS,XIMSG.XIPCS
RTE - IAND
1815 1 BEGIN XIFX
1816 2  SET # WORDS TO BE STORED IN OBJECT = OBJECT DATA TYPE
1817 3  CASE (RESULT DATA TYPE ) :
1818 3 3 1
1819 3 3 2
1820 3 3 3
1821 3 3 3.1 :FIXRE:
1822 3 3 3.2 ERREX TO :NOCHAR: IF TYPE IN SYMBOL TABLE FOR RESULT OPERAND IS NOT FREE
1823 3 3 3.3 IF TOP ENTRY ON RESULT STACK IS A DISPLACEMENT (TYPE = -2), THEN
1824 3 3 4 POP DISPLACEMENT FROM RESULT STACK
1825 3 3 3 ELSE FREE ELEMENT HAS NOT BEEN SUBSCRIPTED
1826 3 3 4 SET DISPLACEMENT = 0
1827 3 3 4 ENDF
1828 3 3 5 POP RESULT OPERAND FROM RESULT STACK (SYMBOL TABLE INDEX)
1829 3 3 5 CALL XPREQ TO RETRIEVE DATA FROM RESULT OPERAND AT DISPLACEMENT DETERMINED
1830 3 3 5 (# WORDS RETRIEVED = OBJECT DATA TYPE)
1831 3 3 :FIXFIX:
1832 3 3 SET TARGET TYPE TO OBJECT DATA TYPE
1833 3 3 CALL XPCS TO POP RESULT VALUE, CONVERT IF NECESSARY, AND SET UP FOR STORE
1834 3 3 ENDCASE
1835 3 3 EXIT XIFX
1836 3 :NOCHAR:
1837 3 2 SET MESSAGE TO BE OUTPUT TO "NUMERICAL DATA ELEMENT CANNOT BE SET EQUAL TO
1838 3 2 CHARACTER DATA"
1839 3 2 CALL XIMSG TO OUTPUT MESSAGE TO USER
1840 3 2 CALL XLIST TO LIST SYMBOLIC STRING
1841 3 2 CALL XPIIT TO EXIT PROCESSOR
1842 1 END XIFX
1844 1 CDO  FORTRAN CALLING PROCEDURE
1845 1 CDO
1846 1 CDO
1847 1 CDO  CALL XILSS (LU, STRING, INDEX)
1848 1 CDO
1849 1 C********
1850 1 C1
1851 1 C1  XILSS IS CALLED TO LIST A SYMBOLIC STRING AND AN INDICATOR TO A
1852 1 C1  PARTICULAR TOKEN IN THAT STRING
1853 1 C1
1854 1 C********
1855 1 C2  INPUT
1856 1 C2
1857 1 C2  LU  - LOGICAL UNIT NO. FOR OUTPUT OF STRING
1858 1 C2
1859 1 C2  STRING - SYMBOLIC STRING TO BE LISTED
1860 1 C2  INDEX - SUBSCRIPT INTO STRING OF THE TOKEN TO BE INDICATED
1861 1 C2
1862 1 C********
1863 1 C3
1864 1 C3  OUTPUT
1865 1 C3
1866 1 C3  THE SYMBOLIC STRING IS OUTPUT TO THE LU FOLLOWED BY A LINE CONTAINING
1867 1 C3  AN INDICATOR (UP ARROW) TO THE DESIGNATED TOKEN.
1868 1 C3
1869 1 C********
<table>
<thead>
<tr>
<th>ENTRY</th>
<th>INTEGER</th>
<th>REAL</th>
<th>DOUBLE</th>
<th>NAME</th>
<th>CHAR. STR.</th>
<th>TOKEN FOLLOWING 2 TOKEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

5=369
1924 1 C*******
1925 1 CD5
1926 1 CD5  NOTES
1927 1 CD5  USES ROUTINES
1928 1 CD5
1929 1 CD5
1930 1 CD5  EXEC
1931 1 CD5  XRD
1932 1 CD5  XRE14
1933 1 CD5  XRD18
1934 1 CD5  XRM5G
1935 1 CD5  XRMOV
1936 1 CD5
1937 1 C*******
1938 2 BEGIN XILSS
1939 2 MOVE A ' CHARACTER INTO PRINT BUFFER AND INCREMENT BUFFER POINTER
1940 2 DO UNTIL ALL TOKENS OF SYMOLIC STRING HAVE BEEN PROCESSED
1941 2 EXIT TO ERROR 1 IF TOKEN VALUE IS < 1 OR > 32
1942 3 USE TOKEN VALUE TO RETRIEVE 3 CONTROL WORDS (SIZE, FIELD, TOKSIZ)
1943 2 EXIT TO ERROR 2 IF FIELD = 0
1944 3 IF SIZE < 0, THEN
1945 4 SET SIZE TO VALUE IN WORD FOLLOWING THIS TOKEN IN THE SYMBOL STRING
1946 3 ENDF
1947 3 IF TOKSIZ < 0, THEN
1948 4 SET TOKSIZ TO SIZE + 2
1949 3 ENDF
1950 3 IF THERE IS NOT ROOM IN PRINT BUFFER FOR SIZE WORDS, THEN
1951 4 CALL EXEC TO WRITE PRINT BUFFER TO INDICATED DEVICE
1952 4 SET BUFFER POINTER TO 1ST POSITION FOR DATA
1953 4 IF INDICATED TOKEN PROCESSED, THEN
1954 5 OUTPUT LINE WITH INDICATOR
1955 4 ENDF
1956 3 ENDF
1957 3 IF FIELD > 0, THEN
1958 4 MOVE FIELD INTO CURRENT PRINT BUFFER POSITION
1959 3 ELSE
1960 4 CASE (ONE, TWO, THREE, FOUR, FIVE, SIX, EXIT), -FIELD
1961 5 SOME:  CALL XRD WITH VALUE IN NEXT WORD OF SYMOLIC STRING
1962 5 AND PUT RESULTS INTO PRINT BUFFER
1963 5 ;TWO:  CALL XRE14 WITH VALUE IN NEXT 2 WORDS OF SYMOLIC STRING
1964 5 AND PUT RESULTS INTO PRINT BUFFER
1965 5 ;THREE: CALL XRD18 WITH VALUE IN NEXT 3 WORDS OF SYMOLIC STRING
1966 5 AND PUT RESULTS INTO PRINT BUFFER#1
1967 5 ;FOUR:  MOVE THE NEXT 3 WORDS OF SYMOLIC STRING INTO PRINT BUFFER
1968 5 ;FIVE:  MOVE SIZE WORDS FROM 2ND WORD PAST CURRENT TOKEN THE SYMOLIC STRING
1969 5 ;SIX:  CALL XRD WITH VALUE IN NEXT WORD OF SYMOLIC STRING AND PUT RESULTS
1970 5 INTO PRINT BUFFER FOLLOWED BY AN "R"
1971 5 ;EXIT:  PUT A ' CHARACTER INTO THE PRINT BUFFER, INCREMENT THE BUFFER
1972 5 INDEX BY 1
1973 5 CALL EXEC TO WRITE THE PRINT BUFFER TO INDICATED DEVICE
1974 5 IF INDICATED TOKEN HAS BEEN PROCESSED, AND
1975 6 INDICATOR LINE NOT YET OUTPUT, THEN
FORTAN CALLING PROCEDURE

1984 1 CD0 CALL XIMSG (CONT, NUMBER, LOCATE, LENGTH, SOURCE)
1988 1 CD0
1989 1 CD0
1990 1 CD0
1991 1 CD1 INSERT 'LENGTH' WORDS OF TEXT FROM 'SOURCE' INTO MESSAGE
1992 1 CD1 'NUMBER' BEGINNING AFTER 'LOCATE' WORDS, CONCATENATE TO PREFIX,
1993 1 CD1 TRUNCATE TO EIGHTY CHARACTERS AND OUTPUT TO USER TERMINAL
1994 1 CD1
1995 1 CD1
1996 1 CD1
1997 1 CD2 INPUT - THREE CONTROL WORDS CONTAINING THE TERMINAL LU AND THE
1998 1 CD2 DEBUG CONTROL FLAG. IF CONT(3) BIT 12 IS ON, XUDBG WILL
1999 1 CD2 BE CALLED AFTER THE MESSAGE IS OUTPUT
2000 1 CD2
2001 1 CD2 NUMBER - INTEGER MESSAGE NUMBER OF THE FORM 'ANN' WHERE
2002 1 CD2 A - AREA INDICATOR AS FOLLOWS
2003 1 CD2 1 - AS
2004 1 CD2 2 - XA
2005 1 CD2 3 - XE
2006 1 CD2 4 - XI
2007 1 CD2 5 - XS
2008 1 CD2 6 - XT
2009 1 CD2 7 - XX
2010 1 CD2 8 - XL
2011 1 CD2 9 - DF
2012 1 CD2 10 - SC
2013 1 CD2 NN - MESSAGE NUMBER OR ZERO WHICH INDICATES ONLY 'LENGTH'
2014 1 CD2 WORDS OF 'SOURCE' TO BE OUTPUT
2015 1 CD2 LOCATE - INTEGER NUMBER OF WORDS OF MESSAGE TO PRECEDE
2016 1 CD2 'SOURCE' (NOT USED IF 'NN' OF 'NUMBER' IS ZERO)
2017 1 CD2 LENGTH - INTEGER NUMBER OF WORDS OF 'SOURCE' TO BE INSERTED INTO
2018 1 CD2 MESSAGE. ZERO INDICATES NO INSERTION
2019 1 CD2 SOURCE - ARRAY OF CHARACTERS TO BE INSERTED INTO MESSAGE (NOT USED
2020 1 CD2 IF 'LENGTH' IS ZERO)
2021 1 CD2
2022 1 CD2
2023 1 CD3 OUTPUT UP TO EIGHTY CHARACTER LINE OF TEXT TO UNIT 'LU' OF THE FORM
2024 1 CD3 *ANN* MESSAGE(1-LOCATE) SOURCE(1-LENGTH) REMAINDER OF MESSAGE
2025 1 CD3
2026 1 CD3
2027 1 CD3
2028 1 CD3
2029 1 CD5 NOTES
2030 1 CD5 USES FOS SYSTEM MESSAGE FILE JXIMSG
2031 1 CD5 USES CLOSE, EXEC, IAND, KCVT, OPEN, READF, XRMOV, XUDBG
2032 1 CD5
2033 1 CD5
1 BEGIN XZMSG
2 SEPARATE NUMBER INTO AREA AND MESSAGE NUMBER
3 SET NUMBER IN PREFIX
4 READ MESSAGE DIRECTORY RECORD
5 IF AREA VALID
6 THEN
7 SET AREA CODE IN PREFIX
8 IF MESSAGE NUMBER > 0
9 THEN
10 IF VALID MESSAGE NUMBER
11 THEN
12 COMPUTE MESSAGE RECORD NUMBER
13 READ RECORD
14 CALL XRMV TO MOVE LOCATE WORDS FROM RECORD INTO BUFFER ENDIF
15 ELSE
16 EXIT TO :ERROR:
17 ENDF
18 ELSE
19 CALL XRMV TO MOVE LENGTH WORDS FROM SOURCE INTO BUFFER (MAX OF 40 TOTAL WORDS)
20 IF MESSAGE NUMBER > 0
21 THEN
22 CALL XRMV TO MOVE REMAINING RECORD INTO BUFFER (MAX OF 40 TOTAL WORDS)
23 ENDF
24 ELSE
25 SET AREA IN PREFIX
26 DEBUR: CALL XRMV TO MOVE 'XZMSG ERROR' INTO BUFFER
27 CALL XRMV TO MOVE LENGTH WORDS OF SOURCE INTO BUFFER (MAX OF 40 TOTAL WORDS)
28 ENDF
29 OUTPUT BUFFER TO USER'S TERMINAL
30 IF DEBUG IS REQUESTED
31 THEN
32 CALL XUDMP
33 ENDF
34 END IF
35 END XZMSG
FORTRAN CALLING PROCEDURE:

CALL XIOPR(ENTRY)

XIOPR IS USED BY THE ASSGM ROUTINE XIPS2 TO EVALUATE MATH OPERATIONS

INPUTS

ENTRY - OPERATOR TOKEN CURRENTLY BEING PROCESSED

FROM ASSGM - LU, SSTRNG, SYNTAX, DATYPE, RSLTI, CLSTM, MAPPDS, RESULT

OUTPUTS TO ASSGM

RESULT, OPRANDS, REGST, RSLTI

INTERNAL VARIABLES

MAPOP - MAPS OPERATOR TOKENS FOR EXECUTION

EXTERNAL REFERENCES

FDX - XPRED, XPXIT, XRMV, XLSS, XMSG

BTE - IAND, OUF
BEGIN XOPR

:ADD:
PERFORM ADDITION
SET RESULT'S DATA TYPE AS APPROPRIATE

:SUBTR:
PERFORM SUBTRACTION
SET RESULT'S DATA TYPE AS APPROPRIATE

:MULT:
PERFORM MULTIPLICATION
SET RESULT'S DATA TYPE AS APPROPRIATE

:DIVIDE:
PERFORM DIVISION
SET RESULT'S DATA TYPE AS APPROPRIATE

:INDEX:
IF RESULT STACK IS NOT EMPTY, THEN
GET I-DIM FOR TOP OPERAND IN RESULT STACK
SET # WORDS PER ELEMENT BASED ON TYPE OF TOP ENTRY OF RESULT STACK
ELSE EQUATE SUBSCRIPT FOR OBJECT IS BEING EVALUATED
GET I-DIM FOR FIRST ENTRY IN SYMBOL TABLE
SET # WORDS PER ELEMENT BASED ON TYPE OF FIRST ENTRY IN SYMBOL TABLE
ENDIF
CALCULATE RESULT (I.E., INDEX = IDIM*(SECOND OPERAND-1)+FIRST OPERAND)

:SUBSCR:
IF RESULT STACK IS NOT EMPTY, THEN
GET # WORDS/ELEMENT BASED ON TOP ENTRY OF RESULT STACK ENTRY
ELSE OBJECT IS BEING SUBSCRIPTED
GET # WORDS/ELEMENT BASED ON TYPE OF FIRST ENTRY IN SYMBOL TABLE
ENDIF
CALCULATE RESULT (I.E., DISPLACEMENT = (OPERAND-1)*#WORDS PER ENTRY)
IF RESULT STACK IS NOT EMPTY AND TOP ENTRY IS NOT FREE OR CHARACTER, THEN
CALL XPREP TO RETRIEVE A LOGICAL VALUE FOR TOP OPERAND AT DISPLACEMENT
DECREMENT RESULT STACK POINTER TO DISCARD ARRAY NAME
ELSE OBJECT IS BEING SUBSCRIPTED
ENDIF
SET TYPE = -2

:EXPO:
PERFORM EXPONETIATION
SET RESULT'S DATA TYPE AS APPROPRIATE

:MINUS:
CHANGE SIGN OF OPERAND FOR RESULT

ENDCASE
EXPEDIT IF OVERFLOW OR UNDERFLOWH IS INDICATED TO :OVER:
PUSH RESULT AND TYPE ONTO RESULT STACK
EXIT XOPR

SET MESSAGE TO BE OUTPUT TO "OVERFLOW OR UNDERFLOW DETECTED"
1 BEGIN XIPCS
2 POP OPERAND AND DATA TYPE FROM TOP ENTRY OF RESULT STACK
3 IF TARGET TYPE AND DATA TYPE ARE NOT EQUAL, THEN
4 CASE (TARGET TYPE) :
5 INTS: CONVERT OPERAND TO INTEGER
6 REAL: CONVERT OPERAND TO SINGLE PRECISION REAL
7 DBLE: CONVERT OPERAND TO DOUBLE PRECISION REAL
8 ENDCASE
9 ENDIF
10 EXIT XIPCS
11 END OVER:
12 SET MESSAGE TO BE OUTPUT TO "OVERFLOW OR UNDERFLOW DETECTED"
13 CALL XMSG TO OUTPUT MESSAGE TO USER
14 CALL XLSIT TO LIST SYMBOLIC STRING
15 CALL XPIIT TO EXIT PROCESSOR
16 END XIPCS
2222 1 CD0 FORTRAN CALLING PROCEDURE:
2223 1 CD0 CALL XIPS1
2224 1 CD0
2225 1 CD0
2226 1 CD0
2227 1 CD0
2228 1 CD0 XIPS1 IS USED BY ASSGN TO TRANSLATE THE SYMBOLIC STRING EXPRESSION
2229 1 CD0 TO A POST-FIX NOTATION STRING
2230 1 CD0
2231 1 CD0
2232 1 CD0
2233 1 CD0 INPUTS FROM ASGCOM
2234 1 CD0 LU, TOKENS, STRING, EXPTR, SYNTAX, FNCTBL
2235 1 CD0
2236 1 CD0
2237 1 CD0
2238 1 CD0
2239 1 CD0
2240 1 CD0 OUTPUTS TO ASGCOM
2241 1 CD0 EXPTR, RNGSTK, POLISH, SYNTAX
2242 1 CD0
2243 1 CD0
2244 1 CD0
2245 1 CD0
2246 1 CD0 INTERNAL VARIABLES
2247 1 CD0 GRSSTK - 2x120 ARRAY USED TO TRACK FUNCTIONS, SUBSCRIPTS, AND
2248 1 CD0 PARENTHEITICAL GROUPINGS
2249 1 CD0 OPSTK - 2x120 ARRAY; EACH ENTRY CONTAINS OPERATOR TOKEN AND ITS OUTPUT
2250 1 CD0 PRIORITY
2251 1 CD0 PRCD - PRECEDENT TYPE FOR PRECEDING CHARACTER; USED FOR SYNTAX CHECK
2252 1 CD0 TKPTR - POINTER TO TOKEN BEING PROCESSED IN SYMBOLIC STRING
2253 1 CD0 TOKEN - TOKEN CURRENTLY BEING PROCESSED
2254 1 CD0
2255 1 CD0
2256 1 CD0 EXTERNAL REFERENCES
2257 1 CD0 RTE - IAMO, MIND
2258 1 CD0
2259 1 CD0
2260 1 CD0
2261 1 CD0 FDS - XPS1, XILLS, XIMSG, XISYM
2262 1 CD0
2263 1 CD0
1 BEGIN SETUP
4 CASE TOKEN
7 SYMBOL: / = + - * / \ scrutinize NAME CHARACTER
10 SYMBOL: < > + - * / \ augment COUNT = BINARY/UNARY FLAG AND LIMIT = 0 onto GROUPING STACK
13 ENDLOOP
16 EXITIF TOKEN NAME MATCHES FUNCTION NAME
19 INCRTCTO FUNCTION'S INDEX IN TABLE + 128
22 PUSH COUNT = BINARY/UNARY FLAG AND LIMIT = 0 onto GROUPING STACK
25 ENDLOOP
28 EXITIF TOKEN IS AN OPERAND
31 IF Token IS A NAME AND NEXT Token IS (, THEN
34 JET SUBSCRIPT FLAG
37 EXITIF
40 CALL XSYM TO STORE SYMBOL AND SUBSCRIPT FLAG
43 PUSH SYMBOL INDEX onto EXPRESSION STACK
46 ENDIF
49 :CHAR:
52 PUSH NEGATIVE POINTER TO CHARACTER STRING onto EXPRESSION STACK
55 :PLUS:
58 IF PRECEDING Token WAS LEFT BRACKET OR ( OR , OR =, THEN
61 INCRTCTO NEXT Token and EXIT TO :ENDO:
64 ENDIF
67 :MINUS:
70 IF PRECEDING Token WAS LEFT BRACKET OR ( OR , OR =, THEN
73 CHANGE Token TO UNARY MINUS
76 ENDIF
79 :ASTERISK:
82 IF NEXT Token IS *, THEN
85 CHANGE Token TO EXPONENTIATION
88 ENDIF
91 :LPAREN:
94 IF PRECEDING Token WAS A VARIABLE (DATA ELEMENT NAME), THEN
97 PUSH COUNT AND LIMIT ALLOWING DOUBLE SUBSCRIPTING onto GROUPING STACK
100 ELSE EXPRESSION IS BEING PROCESSED
103 PUSH COUNT AND LIMIT ALLOWING NO SUBSCRIPTING onto GROUPING STACK
106 ENDIF
109 END CASE
112 END
BEGIN STRING
SET INDEX INTO SYNTAX TABLE TO MIN(TOKEN,40)
ERREXIT TO :SYNTAX1: IF THIS TOKEN IS NOT VALID ACCORDING TO SYNTAX TABLE
IF TOKEN IS NOT AN OPERAND, THEN
DO UNTIL TOKEN IS PUSHED ONTO OPERATOR STACK OR DISCARDED
IF INPUT PRIORITY OF THIS TOKEN > OUTPUT PRIORITY OF TOP ENTRY IN OPERATOR
STACK, THEN
IF TOKEN IS , THEN
INCREMENT COUNT FOR TOP ENTRY IN GROUPING STACK.
ERREXIT TO :SYNTAX1: IF COUNT > COMMA LIMIT FOR (TOP ENTRY IN GROUPING STACK
IF COMMA LIMIT SHOWS SUBSCRIPTING IN FUNCTION LIST, THEN
PUSH TOKEN AND OUTPUT PRIORITY ONTO OPERATOR STACK
ELSE
DISCARD FUNCTION LIST IS BEING PROCESSED
ENDIF
ELSE OPERATOR IS NOT ;
PUSH TOKEN AND ITS OUTPUT PRIORITY ONTO OPERATOR STACK
ENDIF
ELSE INPUT PRIORITY IS < OR = OUTPUT PRIORITY
IF INPUT PRIORITY < OUTPUT PRIORITY OF TOP ENTRY IN OPERATOR STACK OR
INPUT PRIORITY = OUTPUT PRIORITY NOT = 2, THEN
ERREXIT TO :SYNTAX1: IF INPUT PRIORITY = 0 AND OPERATOR STACK IS EMPTY
IF TOP ENTRY OF OPERATOR STACK IS -, THEN
ERREXIT IF THIS IS NOT LAST ENTRY ON OPERATOR STACK TO :BADEQL:
ERREXIT IF GROUPING STACK IS NOT EMPTY TO ;SYNTAX2:
DISCARD TOKEN (; OR ESS)
ENDIF
POP OPERATOR STACK
POP OPERATOR ONTO EXPRESSION STACK
ELSE BRACKETS OR PARENTHESES HAVE BEEN MATCHED
IF TOP ENTRY OF OPERATOR STACK IS ( THEN
ERREXIT TO :SYNTAX2:
IF TOP ENTRY OF GROUPING STACK INDICATES SUBSCRIPTING(LIMIT>0), THEN
PUSH SUBSCRIPTING OPERATOR ONTO EXPRESSION ARRAY
ENDIF
ELSE TOP OPERATOR ENTRY IS OPEN BRACKET (FUNCTION LIST)
ERREXIT IF CURRENT TOKEN IS NOT CLOSE BRACKET TO ;SYNTAX2:
ERREXIT TO :FLSTER: IF FUNCTION LIST IS NOT COMPLETE (TOP OF GROUPING STACK COUNT=0)
ENDIF
POP OPERATOR STACK
POP GROUPING STACK
DISCARD CURRENT TOKEN
ENDIF
ENDIF
END
END DO
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2400</td>
<td>1</td>
<td>BEGIN RANGE</td>
</tr>
<tr>
<td>2401</td>
<td>2</td>
<td>INITIALIZE RANGE TABLE FOR 4 RANGES STARTING AT 1, ENDING AT 1, INCREMENT = 1</td>
</tr>
<tr>
<td>2402</td>
<td>3</td>
<td>IF LAST TOKEN INPUT WAS 3, THEN</td>
</tr>
<tr>
<td>2403</td>
<td>4</td>
<td>DO UNTIL ESS IS REACHED OR FOUR RANGES ARE PROCESSED</td>
</tr>
<tr>
<td>2404</td>
<td>5</td>
<td>ERREXIT IF NEXT TOKEN IS NOT NAME TO :RNGSYM:</td>
</tr>
<tr>
<td>2405</td>
<td>6</td>
<td>CALL XISYM TO INSERT NAME IN SYMBOL TABLE, IF NECESSARY, FLAG ENTRY AS AN INDEX, AND RETURN SYMBOL TABLE INDEX</td>
</tr>
<tr>
<td>2406</td>
<td>7</td>
<td>ERREXIT IF NEXT TOKENS ARE NOT &quot;-INTEGER,INTEGER&quot; TO :RNGSYM:</td>
</tr>
<tr>
<td>2407</td>
<td>8</td>
<td>IF SECOND INTEGER VALUE &lt; FIRST INTEGER VALUE, THEN</td>
</tr>
<tr>
<td>2408</td>
<td>9</td>
<td>SET INCREMENT TC = -1</td>
</tr>
<tr>
<td>2409</td>
<td>10</td>
<td>ENDF</td>
</tr>
<tr>
<td>2410</td>
<td>11</td>
<td>PUSH START, END, AND INCREMENT VALUES AND SYMBOL TABLE INDEX ONTO STACK</td>
</tr>
<tr>
<td>2411</td>
<td>12</td>
<td>ENDO</td>
</tr>
<tr>
<td>2412</td>
<td>13</td>
<td>ERREXIT TO :RNGSYM: IF ESS HAS NOT BEEN REACHED</td>
</tr>
<tr>
<td>2413</td>
<td>14</td>
<td>ENDF</td>
</tr>
<tr>
<td>2414</td>
<td>15</td>
<td>EXIT RANGE</td>
</tr>
<tr>
<td>2415</td>
<td>16</td>
<td>:RNGSYM:</td>
</tr>
<tr>
<td>2416</td>
<td>17</td>
<td>SET MESSAGE TO BE OUTPUT TO &quot;RANGE SPECIFICATION DOES NOT FOLLOW: &quot;;NAME=</td>
</tr>
<tr>
<td>2417</td>
<td>18</td>
<td>&quot;INTEGER,INTEGER&quot;</td>
</tr>
<tr>
<td>2418</td>
<td>19</td>
<td>CALL XMSG TO OUTPUT MESSAGE TO USER</td>
</tr>
<tr>
<td>2419</td>
<td>20</td>
<td>CALL XLIST TO LIST SYMBOLIC STRING</td>
</tr>
<tr>
<td>2420</td>
<td>21</td>
<td>CALL XPIUT TO EXIT PROCESSOR</td>
</tr>
<tr>
<td>2421</td>
<td>22</td>
<td>END RANGE</td>
</tr>
</tbody>
</table>
FORTRAN CALLING PROCEDURE:

CALL XIPS2

XIPS2 IS USED BY ASSGN TO EVALUATE THE POST-FIX STRING GENERATED BY XIPS1 AND TO STORE THE RESULTS INTO THE OBJECT DATA ELEMENT

INPUTS FROM ASGCOM

LU, TOKENS, LASTY, SYMTAB, SSTRING, RESULT, OPRNDS

OPINFO, NUMWDS, DATTPS, RSLTPY, EXTR, CLSRQ, CLSTRN, DECLS,

RNGTRX, POLISH, SYNTAX, FNCTBL

OUTPUTS TO ASGCOM

SYMTAB, RESULT, RERST, RSLTPY, OPINFO

INTERNAL VARIABLES

ENTRY - ENTRY FROM POST-FIX STRING WHICH IS CURRENTLY BEING PROCESSED

EXTERNAL REFERENCES

FDS - XPRTE, XPXIT, XRMOW, XICHR, XZFMN, XZFRME, XZFXD, XZLSS, XZMSG,

ZIOPB, ZOPS, ZIRET

RTE - IABS, IAND, MAXD

RTE - IABS, IAND, MAXD
2465 1 BEGIN XIPS2
2466 2 PERFORM TOC TO RETRIEVE TOC ENTRIES FOR ALL DATA ELEMENTS IN SYMBOL TABLE
2467 3 PERFORM DATA1 TO RETRIEVE DATA FOR NON-SUBSCRIPTED FIXED-TYPE DATA ELEMENTS
2468 4 INITIALIZE RANGE VALUES FOR ITERATION
2469 5 DO UNTIL ALL RANGES ARE FINISHED
2470 6 DO UNTIL POLISH STRING IS EVALUATED (STARTING WITH SECOND ENTRY OF STRING)
2471 7 POP ENTRY FROM POLISH STRING
2472 8 IF ENTRY IS AN OPERAND, THEN
2473 9 PERFORM RPUH TO RETRIEVE DATA IF AVAILABLE AND PUSH ONTO RESULT STACK
2474 10 ELSE ENTRY IS AN OPERATOR
2475 11 IF OPERATOR IS NOT =, THEN
2476 12 PERFORM EXEVAL TO EVALUATE POLISH STRING
2477 13 ELSE OPERATOR IS =
2478 14 PERFORM REPLAC TO STORE VALUE INTO OBJECT DATA ELEMENT
2479 15 ENDEF
2480 16 ENDF
2481 17 END DO
2482 18 PERFORM RNGSET TO DETERMINE CORRECT RANGE VALUES
2483 19 END DO
2484 20 END XIPS2
2486 1 BEGIN TOC
2487 2 DO UNTIL ALL ENTRIES IN SYMBOL TABLE ARE PROCESSED
2488 3 IF SYMBOL NOT = 0 (Symbol IS DE OR INDEX), THEN
2489 4 IF ENTRY IS A DATA ELEMENT (Symbol'S FLAG WORD NOT= 1), THEN
2490 5 CALL XPREQ TO RETRIEVE TOC ENTRY
2491 4 ELSE SYMBOL IS A RANGE INDEX
2492 5 SET DATA TYPE TO INTEGER
2493 4 ENDIF
2494 3 ENDF
2495 2 END DO
2496 2 CALL XPREQ WITH A CLOSE BUFFER REQUEST
2497 1 END TOC
2499 1 BEGIN DATA1
2500 2 DO UNTIL ALL ENTRIES IN SYMBOL TABLE ARE PROCESSED
2501 3 IF SYMBOL IS A NON-SUBSCRIPTED FIXED-TYPE DATA ELEMENT, THEN
2502 4 CALL XPREF TO QUEUE REQUEST FOR DATA RETRIEVAL
2503 5 ENDIF
2504 6 END DO
2505 7 CALL XPREF WITH A CLOSE BUFFER REQUEST
2506 8 END DATA1
2508 1 BEGIN RPUSH
2509 2 IF OPERAND IS A CHARACTER STRING POINTER (OPERAND < 0), THEN
2510 3 PUSH ABSOLUTE VALUE OF OPERAND AND -1 DATA TYPE ONTO RESULT STACK
2511 4 ELSE OPERAND IS A SYMBOL TABLE INDEX
2512 5 CALL SIZE TO RETRIEVE DATA AND DATA TYPE FOR OPERAND
2513 6 PUSH RETRIEVED VALUE AND DATA TYPE ONTO RESULT STACK
2514 7 ENDRIF
2515 8 END RPUSH
BEGIN EVAL
IF OPERATOR IS A FUNCTION, THEN
DETERMINE # OPERANDS FROM FUNCTION TABLE
ELSE
DETERMINE # OPERANDS FROM SYNTAX TABLE
ENDIF
DO FOR EACH OPERAND
IF RESULT STACK CONTAINS A SYMBOL INDEX, THEN
ERREXIT IF DATA TYPE IS FREE OR CHARACTER TO "MIXERR;"
CALL XPED TO RETRIEVE DATA AND TYPE FOR FIRST ELEMENT OF ARRAY
STORE DATA AND TYPE IN RESULT STACK AT THIS ENTRY
ELSE
ERREXIT IF DATA TYPE IS FREE OR CHARACTER TO "MIXERR;"
ENDIF
END DO
IF THE OPERAND IS A FUNCTION, THEN
DO FOR EACH FUNCTION OPERAND
SET TARGET TYPE FROM FUNCTION TABLE
CALL XIPCS TO POP OPERAND, CONVERT IF NECESSARY, AND SET UP
ENDIF
CALL XIFMC TO PERFORM FUNCTION FOR RESULTS
SET DATA TYPE FOR RESULTS FROM FUNCTION TABLE
ELSE, THIS IS AN OPERATOR OTHER THAN A FUNCTION OR "="
DETERMINE TYPE REQUIREMENTS FOR THIS OPERATOR FROM SYNTAX TABLE
CASE (TYPE REQUIREMENT+1) :SAME:, :INTEGER:
SAME:
IF # OPERANDS > 1, THEN
SET TARGET TYPE TO MAX OF TWO DATA TYPES
ELSE, # OPERANDS = 1
SET TARGET TYPE TO OPERAND'S DATA TYPE
ENDIF
:INTEGER:
SET TARGET TYPE TO INTEGER
ENDIF
DO FOR EACH OPERAND
CALL XIPCS TO POP OPERAND, CONVERT IT TO TARGET TYPE, AND SET UP
END DO
CALL XIOPR TO PERFORM ARITHMETIC OPERATION AND PUSH RESULT AND TYPE ONTO STACK
ENDIF
EXIT EVAL
ENDIF
MIXERR:
SET MESSAGE TO BE OUTPUT TO "FREE OR CHARACTER DATA ELEMENT FOUND IN AN "
EXPRESSION"
CALL XOSS TO OUTPUT MESSAGE TO USER
CALL XLS2 TO LIST SYMBOLIC STRING
CALL XPRP TO EXIT PROCESSOR
EXIT EVAL
BEGIN REPLAC
IF THE TOP ENTRY OF THE RESULT STACK IS A SYMBOL TABLE INDEX, THEN
IF THE DATA TYPE IN THE SYMBOL TABLE IS FIXED, THEN
CALL XPREQ TO RETRIEVE FIRST ELEMENT OF THE ARRAY
STORE DATA AND TYPE IN TOP RESULT STACK ENTRY
ENDIF
CASE (OBJECT DATA TYPE )
FREE:, FIXED:, FIXED:, FIXED:, CHAR:, CHAR:
CHAR:, CHAR:, CHAR:
FEE:
CALL XPREQ TO RETRIEVE DATA AND SET UP FOR STORE
FIXED:
CALL FIXOBJ TO RETRIEVE DATA, CONVET IF NECESSARY, AND SET UP FOR STORE
CHAR:
CHAR:
CHAR:
END CASE
IF TOP ENTRY OF RESULT STACK IS A DISPLACEMENT, THEN
POP OBJECT'S DISPLACEMENT FROM RESULT STACK
ELSE OBJECT HAS NOT BEEN SUBSCRIPTED
SET OBJECT'S DISPLACEMENT TO 0
ENDIF
BUILD XPREQ REQUEST TO STORE # WORDS CALCULATED INTO OBJECT AT OBJECT'S DISPLACEMENT
CALL XPREQ TO STORE DATA IN OBJECT
1 END REPLAC
FORTRAN CALLING PROCEDURE:

CALL XIRET(ENTRY)

XIRET IS USED BY THE ASGN ROUTINE XIPS2 TO RETRIEVE THE DATA VALUE
FOR A GIVEN SYMBOL INDEX FROM THE SYMBOL TABLE AND PUSH IT ONTO THE
RESULT STACK ALONG WITH ITS DATA TYPE.
BEGIN XIRET

IF SYMBOL ENTRY IS FOR NON-SUBSCRIPTED FIXED-TYPE DATA (DATA ELEMENT OR
RANGE INDEX), THEN
SET RETURN VALUE = VALUE IN SYMBOL TABLE FOR THIS ENTRY
SET RETURN DATA TYPE = DATA TYPE IN SYMBOL TABLE FOR THIS ENTRY
ELSE SYMBOL IS CHARACTER OR FREE DATA OR SUBSCRIPTED FIXED DATA
SET RETURN VALUE = SYMBOL TABLE INDEX
SET RETURN DATA TYPE = -1
ENDIF

END XIRET
**FORTRAN CALLING PROCEDURE**

1. **CD0**
   - CALL ZXSTM (TKMPTR, FLAG, SYMIND)

2. **CD1**
   - ZXSTM PROVIDES SYMBOL TABLE BUILDING AND ACCESS FOR ASSTM. IT
   - USES ZXSTM TO PERFORM GENERAL SYMBOL TABLE FUNCTIONS THEN
   - ACCOMPLISHES RANGE AND SUBSCRIPT PROCESSING DIRECTLY

3. **CD****

4. **CD2**
   - INPUT

5. **CD2**
   - TKMPTR - INDEX INTO INPUT SYMBOLIC STRING (SSTRM) OF TOKEN TO BE

6. **CD2**
   - ENTERED OR LOCATED IN SYMBOL TABLE (SYMTAB)

7. **CD2**
   - FLAG - SPECIAL PROCESSING FLAG

8. **CD2**
   - 0, NORMAL PROCESSING

9. **CD2**
   - 1, SYMBOL IS A RANGE

10. **CD2**
    - 2, SYMBOL IS SUBSCRIPED

11. **CD2**
    - ASCOM COMMON - LU, SSTRM, STWIDE, SYMTAB, TOKENS,

12. **CD2**
13. **CD3**
14. **CD3**
   - SYMIND - INDEX INTO SYMBOL TABLE (SYMTAB) WHERE SYMBOL IS LOCATED.
   - VALUE OF SYMIND INDICATES ENTRY NUMBER, I.E., 1, 2, ...

15. **CD3**
16. **CD3**
    - ASCOM COMMON - SYMTAB

17. **CD4****

18. **CD4**
19. **CD4**
    - INTEJIAL

20. **CD4**
    - SYMTPR - INDEX TO FLAG WORD OF ENTRY IN SYMTAB. VALUE IS EQUAL TO
    - SYMIND * STWIDE

21. **CD4**
22. **CD5**
    - EXTERNAL REFERENCES

23. **CD5**
24. **CD5**
    - XPIIT, XILSS, XIMSG, ZXSTM
2687 1 BEGIN XSYM
2688 2 CALL XSYM TO ENTER TOKEN INTO TABLE OR RETURN INDEX TO EXISTING ENTRY
2689 2 IF TOKEN IS A NAME, THEN
2690 3 IF SPECIAL PROCESSING FLAG IS SET (1= RANGE INDEX, 2= SUBSCRIPTED), THEN
2691 4 IF FLAG INDICATES RANGE INDEX, THEN
2692 5 EXIT TO :BADREG: WITH ERROR AS01 IF OBJECT (FIRST ENTRY IN TABLE)
2693 5 EXIT TO :BADREG: WITH ERROR AS02 ENTRY IS ALREADY SUBSCRIPTED
2694 5 EXIT TO :BADREG: WITH ERROR AS03 ENTRY IS ALREADY A DEFINED RANGE
2695 4 ENDIF
2696 4 SET ENTRY FLAG WORD TO FLAG VALUE
2697 3 ENDIF
2698 2 ENDIF
2699 2 ADD BIAS OF 256 TO SYMBOL INDEX
2700 1 EXIT XSYM

2701 2 BADREG:
2702 2 CALL XISMG TO OUTPUT ERROR DESCRIPTION
2703 2 CALL XISLS TO DISPLAY SYMBOLIC STRING AND POINT TO ERROR
2704 2 CALL XPRT TO TERMINATE PROCESSOR
2705 1 END XSYM
FORTRAN CALLING PROCEDURE

CALL XYSY (TKPTR, SYMIND)

PROVIDED KEY VALUES ARE SUITABLY LOCATED IN COMMON, XYSY PROVIDES
A GENERAL CAPABILITY FOR SYMBOL TABLE ACCESS FOR TOKENS INPUT VIA
A SYMBOLIC STRING

INPUT

TKPTR - INDEX INTO INPUT SYMBOLIC STRING (SSTRNG) OF TOKEN TO BE
ENTERED OR LOCATED IN SYMBOL TABLE (SYMAB)

ASCOM COMMON - LASTSY, SSTRNG, SSTRIDE, SYMAB, TOKENS

SYMIND - INDEX INTO SYMBOL TABLE (SYMAB) WHERE SYMBOL IS LOCATED.
VALUE OF SYMIND INDICATES ENTRY NUMBER, I.E., 1, 2, ...

ASCOM COMMON - LASTSY, SYMAB

INTERNAL

DISP - DISPLACEMENT INTO SYMBOL TABLE ENTRY FOR FIELD TO BE USED
AS KEY. NAMES USE A VALUE OF 1, CONSTANTS A VALUE OF 8.

EDOOP - INDEX OF LAST ALLOCATED WORD IN SYMAB. VALUE IS EQUAL
TO LASTSY*STRIDE

IBUF - INTERNAL BUFFER FOR SYMBOL VALUE USED TO ASSURE LAST
WORDS OF INTEGER AND REAL CONSTANTS ARE ZERO.

EXTERNAL REFERENCES

XRCPR, XRMOW

SPECIAL REMARKS

THE REQUIRED FORMAT OF EACH SYMBOL TABLE ENTRY IS

I TYPE (1 WORD) I NAME (4 WORDS) I VALUE I OTHER I

I = TOKEN CODE I (3 WORDS) I RESERVED FOR TOC ENTRY I (3 WORDS) I (4 WORDS)
BEGIN

SET BUFFER TO BE USED IN MOVING TOKEN
IF TOKEN IS A NAME
THEN
SET COMPARISON DISPLACEMENT IN TABLE TO 1 (NAME FIELD)
ELSE
SET COMPARISON DISPLACEMENT IN TABLE TO 8 (VALUE FIELD)
ENDIF
EXIT IF ENTRY MATCHES BUFFER CONTENTS AND TYPE FIELD MATCHES TOKEN CODE
STORE TOKEN CODE IN TYPE FIELD OF NEXT ENTRY
STORE BUFFER CONTENTS INTO APPROPRIATE ENTRIES
END SET STANDING TO ENTRY NUMBER

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR.
DO - CONDITIONAL ITERATION (LOOPING) PROCESSOR

SCHEDULED BY FBS

THE DO AND ENDDO UTILITY PROCESSOR PAIR PROVIDE FBS USERS WITH THE CAPABILITY OF LOOPING THROUGH A BLOCK OF SEQUENCE TABLE ENTRIES EITHER UNTIL A GIVEN CONDITION IS SATISFIED OR WHILE THE CONDITION IS TRUE.

INPUT FROM MANAGER VIA SCHEDULING PARAMETERS

LU - LOGICAL UNIT OF USER'S TERMINAL
FLAGS - DEBUG FLAGS FROM USER SIGN-ON
SENDSP - INDEX OF CURRENTLY EXECUTING SEQUENCE TABLE ENTRY

INPUT FROM INTERFACE TABLE

COTYPE - 4CH CHARACTER STRING SPECIFYING LOOP CONDITION TYPE
UNTIL - LOOP UNTIL RELATION BETWEEN OPRND1 AND OPRND2 IS TRUE. TEST IS MADE AT THE END OF EACH LOOP EXECUTION.
WHILE - LOOP WHILE RELATION BETWEEN OPRND1 AND OPRND2 IS TRUE. TEST IS MADE PRIOR TO STARTING EACH LOOP EXECUTION.

OPRND1 - FIRST REAL QUANTITY TO COMPARE (VALUE NOT RETRIEVED, CHARACTERISTICS PASS TO ENDDO)
OPRND2 - SECOND REAL QUANTITY TO COMPARE (SEE OPRND1)

INPUT FROM AWA ON REQUEST TO MANAGER

INTBUF - INTERFACE TABLE (LESS LITERAL AREA)
LITERAL - NINE WORD BUFFER FOR HOLDING ORIGINAL INTERFACE TABLE LITERAL AREA
SOSTK - SEE OUTPUT DEFINITION
BNRTB - EXEC COPY OF CURRENTLY EXECUTING SEQUENCE TABLE

OUTPUT TO THE MANAGER VIA RETURN PARAMETERS

RETURN - RETURN CODE TO MANAGER
0 - NORMAL TERMINATION. CONTINUE SEQUENTIAL EXECUTION
3 - NORMAL TERMINATION. SKIP TO SPECIFIED SEQUENCE NUMBER FOR CONTINUED EXECUTION
8 - ABNORMAL TERMINATION. ABORT SEQUENCE EXECUTION

RESET - $FABNCE RESET NUMBER OF ENDDO IF RETURN = 3

INPUT/OUTPUT FROM TO AWA

AOSTK - CONTROL INFORMATION FOR DO/ENDDO PROCESSORS. DIMENSIONED
(27,n) where n is the current number of active loops.
Each entry has the following form:
Words 1-7 - first seven words of do interface table
with the following changes (see.fds 390,
Volume IV, figure 1.2-17)
Words 8-14 - interface table entry for oprrn1 (same as
in original table except for possible new
values pointing into literal area)
Words 15-21 - interface table entry for oprrn2 (see
above)
Words 22-25 - literal area for value(s)/subscript(s) of
oprrn1 & 2
Word 26 - relation id code as follows
0 = \\
1 = =
2 = or \\
3 = =
4 = or \\
5 = <
Word 27 - sequence number of top of loop

80 1 CD3
81 1 CD4
82 1 CD5
83 1 CD6
84 1 CD7
85 1 CD8
86 1 CD9
87 1 CD10
88 1 CD11
89 1 CD12
90 1 CD13
91 1 CD14
92 1 CD15
93 1 CD16
94 1 CD17
95 1 CD18
96 1 CD19
97 1 CD20
98 1 CD21
99 1 CD22
100 1 CD23
101 1 CD24
102 1 CD25
103 1 CD26
104 1 CD27
105 1 CD28
106 1 CD29
107 1 CD30
108 1 CD31
109 1 CD32
110 1 CD33
111 1 CD34
112 1 CD35
113 1 CD36
114 1 CD37
115 1 CD38
116 1 CD39
117 1 CD40
118 1 CD41
119 1 CD42

Internal variables

code - array of eight acceptable relation mnemonics and
corresponding internal codes
doent - index into gosstx for new 27 word entry
pruff - 64 word manager communications buffer. eight word
entries are of the form*
rest - ama management request code
class - class and type of data
name - six character data name
size - size of data
disp - displacement into data of transaction origin
clsn - class i/o number through which data is
transmitted
top - index into 56thb of top of loop
xpcls - class i/o number for manager communications (set by
xpget)

Referenced routines
exec, iand, mod, xnpal, xnpal, xpreq(xpget), xpset, xcrpr, xrmov,
xset, xudg, xwpam, xirmg

Notes

Dosstx must be used in pairs
Dosstx is allocated in the ama for interprocessor communications
zescn cleans up any residual gosstx after execution controller
termination

The maximum number of nested loops is 4
All nesting (structure) errors are left for execution time

Detection
124 1 BEGIN DO
125 2 CALL XGROUP TO INITIALIZE ACCESS TO XPREP AND TO RETRIEVE DOTYPE AND RELATE
126 3 CALL XVPAM TO RETRIEVE BOOST INTO BUFFER
127 4 IF RETRIEVAL FAILED, THEN
128 5 INITIALIZE BUFFER FOR BUILDING FIRST BOOST ENTRY
129 6 ENDIF
130 7 IF BOOST IS NOT FULL (NOT MAXIMUM NUMBER OF NESTS), THEN
131 8 IF XGROUP INTERFACE TABLE BUFFER INDICATES LITERAL DATA EXIST, THEN
132 9 CALL XGROUP TO RETRIEVE LITERALS
133 10 ENDIF
134 11 IF RELATE IS A VALID RELATION OPERATOR, THEN
135 12 SET RELATION CODE IN NEW ENTRY IN BUFFER
136 13 SET INTERFACE TABLE HEADER WITH NAME OF BINTAB AND NUMBER OF PARAMETER: OF 2
137 14 DO FOR EACH OPRAND
138 15 MOVE OPRAND ENTRY INTO NEW INTERFACE TABLE BUFFER
139 16 IF OPRAND HAS LITERAL VALUE OR DOUBLE SUBSCRIPTS, THEN
140 17 MOVE LITERAL DATA
141 18 ADJUST LITERAL POINTERS
142 19 ENDFIX
143 20 ENDFIX
144 21 CALL XGROUP TO RETRIEVE XGROUP (EXECUTING SEQUENCE TABLE)
145 22 IF XGROUP DISPLACEMENT (XGROUP) > 0, THEN
146 23 SET TOP OF LOOP TO NEXT SEQUENCE NUMBER IN TABLE (OF IF END OF TABLE)
147 24 ELSE INSERTED COMMAND
148 25 EXIT TO :ERRO2: IF SEQUENCE NUMBER IS ZERO (MANUAL)
149 26 LOCATE ORIGINAL SEQUENCE ENTRY
150 27 IF ORIGINAL ENTRY WAS ALSO A DO (OVERLAP CONDITION), THEN
151 28 SET TOP OF LOOP TO NEXT SEQUENCE NUMBER IN TABLE (OR ZERO)
152 29 ELSE (INSERT)
153 30 SET TOP OF LOOP TO CURRENT NUMBER
154 31 ENDFIX
155 32 ENDFIX
156 33 CASE (:WHILE:, :UNTIL:, :OTHER:) DOTYPE
157 34 :WHILE:
158 35 INVERT RELATION CODE
159 36 INITIALIZE NEST COUNTER TO 1
160 37 START SEARCH FROM TOP OF LOOP ENTRY UNTIL ALL ENTRIES HAVE BEEN EXAMINED
161 38 IF COMMAND IS ENDOO, THEN
162 39 DECREMENT NEST COUNTER
163 40 ELSE
164 41 IF COMMAND IS ANOTHER DO, THEN
165 42 INCREMENT NEST COUNTER
166 43 ENDFIX
167 44 ENDFIX
168 45 EXIT IF NEST COUNTER IS ZERO
169 46 SET RESET NUMBER TO CURRENT SEQUENCE NUMBER (ENDDO JUST FOUND)
170 47 END LOOP
171 48 EXIT TO :ERRO4: FOR NO MATCHING ENDOO
172 49 END SEARCH
173 50 :UNTIL:
174 51 CLEAR RESET NUMBER (CONTINUE SEQUENTIAL EXECUTION)
175 52 :OTHER:
176 53 TERMINATE WITH ERRO7 FOR UNRECOGNIZED DOTYPE
177 54 END CASE
178 55 CALL XGROUP TO OUTPUT NEW EXPANDED BOOST
179 56 ELSE INVALID RELATION
TERMINATE WITH ERR05 FOR INVALID RELATION
ENDIF
ELSE BDSTK OVERFLOW
TERMINATE WITH ERR01 FOR BDSTK FULL
ENDIF
1 EXIT DO
2 :ERR02: TERMINATE FOR EXECUTING IN MANUAL MODE
2 :ERR04: TERMINATE FOR NO MATCHING ENDDO FOUND DURING WHILE PROCESSING
1 END DO
REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR
237 1 CD-------
238 1 CD0  ENDDO - LOOP TERMINATION PROCESSOR
239 1 CD0  SCHEDULED BY FD
240 1 CD0  -------
241 1 CD1  THE DO AND ENDDO UTILITY PROCESSOR PAIR PROVIDE FDs USERS WITH THE
242 1 CD1  CAPABILITY OF LOOPING THROUGH A BLOCK OF SEQUENCE TABLE ENTRIES
243 1 CD1  EITHER UNTIL A GIVEN CONDITION IS SATISFIED OR WHILE THE CONDITION
244 1 CD1  IS TRUE.
245 1 CD0  -------
246 1 CD1  INPUT FROM MANAGER VIA SCHEDULING PARAMETERS
247 1 CD1  -------
248 1 CD2  LU - LOGICAL UNIT OF USER'S TERMINAL
249 1 CD2  --- DEBUG FLAGS FROM USER SIGN-ON
250 1 CD2  --- INPUT/OUTPUT FROM/TO AWA
251 1 CD2  BDOTK - CONTROL INFORMATION FOR DO/ENDDO PROCESSORS. DIMENSIONED
252 1 CD2  (27,N) WHERE N IS THE CURRENT NUMBER OF ACTIVE LOOPS.
253 1 CD2  SEE DO PROCESSOR FOR CONTENTS
254 1 CD2  THE FDS SYSTEM STATUS TABLE, XVSTH, IS ACCESSED FOR THE MANAGER/
255 1 CD2  PROCESSOR CLASS I/O COMMUNICATIONS NUMBER
256 1 CD2  -------
257 1 CD3  OUTPUT TO THE MANAGER VIA RETURN PARAMETERS
258 1 CD3  RETURN - RETURN CODE TO MANAGER
259 1 CD3  0 - NORMAL TERMINATION. CONTINUE SEQUENTIAL
260 1 CD3  EXECUTION
261 1 CD3  3 - NORMAL TERMINATION. SKIP TO SPECIFIED SEQUENCE
262 1 CD3  NUMBER FOR CONTINUED EXECUTION
263 1 CD3  8 - ABNORMAL TERMINATION. ABORT SEQUENCE EXECUTION
264 1 CD3  RESET - SEQUENCE RESET NUMBER OF TOP OF LOOP IF RETURN = 3
265 1 CD3  -------
266 1 CD4  --- INTERNAL VARIABLES
267 1 CD4  DOENT - INDEX INTO BDOTK FOR LAST 27 WORD ENTRY
268 1 CD4  MRBUF - 64 WORD MANAGER COMMUNICATIONS BUFFER. EIGHT WORD
269 1 CD4  ENTITIES ARE OF THE FORM
270 1 CD4  REGST - AWA MANAGEMENT REQUEST CODE
271 1 CD4  NAME - SIX CHARACTER DATA NAME
272 1 CD4  SIZE - SIZE OF DATA
273 1 CD4  DISP - DISPLACEMENT INTO DATA OF TRANSACTION ORIGIN
274 1 CD4  CLASS - CLASS I/O NUMBER THROUGH WHICH DATA IS
275 1 CD4  TRANSMITTED
276 1 CD4  XPCI - CLASS I/O NUMBER FOR MANAGER COMMUNICATIONS
277 1 CD4  -------
278 1 CD5  --- REFERENCES TO ROUTINES
279 1 CD5  EXEC, IAND, XRPAR, XORGET, XPREQ(XPGET), XPXIT, XRLOC, XRMOV,
280 1 CD5  XUDBG, XVPW, XMMSG
298 1 BEGIN ENDDO
299 2 FIND CLASS I/O NUMBER FOR MANAGER COMMUNICATIONS IN XVSTB
300 2 CALL XVPAW TO RETRIEVE BDOSTK
301 2 IF RETRIEVAL WAS SUCCESSFUL, THEN
302 3 WRITE GINTAB TO CLASS I/O FROM LAST BDOSTK ENTRY
303 3 WRITE NON-LITERAL PORTION OF GINTAB TO CLASS I/O (LEAVE FOR XPGET)
304 3 CALL XVPAW TO RESTORE GINTAB FROM FIRST CLASS BUFFER INTO AWA
305 3 IF RESTORE SUCCESSFUL, THEN
306 4 CALL XPGET TO RETRIEVE OPRNDS
307 4 CALL XSEQL TO EVALUATE RELATION
308 4 IF RELATION IS TRUE, THEN
309 5 CLEAR RESET NUMBER (CONTINUE SEQUENTIAL EXECUTION)
310 5 IF THIS IS LAST BDOSTK ENTRY, THEN
311 6 DELETE BDOSTK FROM AWA
312 6 ELSE
313 7 CALL XPSEQ TO REPLACE BDOSTK LESS LAST ENTRY
314 7 ENDIF
315 4 ELSE
316 5 SET RESET NUMBER TO TOP OF LOOP
317 5 ENDIF
318 3 ELSE
319 4 TERMINATE WITH ERRO6 FOR AWA OVERFLOW
320 4 ENDIF
321 2 ELSE
322 3 TERMINATE WITH ERRO3 FOR ENDDO WITH OUT DO
323 2 ENDIF
324 1 END ENDDO
1 CDO******
2 CDO FORTRAN MAIN PROGRAM - STANDARD FDS SCHEDULING PARAMETERS
3 CDO********
4 CD1 THE ENDF้าง UTILITY PROCESSOR RETURNS TO THE FDS MANAGER (VIA XPXIT) SO
5 CD1 THAT NORMAL PROCESSOR EXECUTION SEQUENCE MAY CONTINUE
6 CDO********
7 CD2 INPUTS
8 CD2 SCHEDULING PARAMETERS
9 CD2 LU = LOGICAL UNIT NO. OF FDS USER
10 CD2 FLAGS = DEBUG FLAGS -- BIT 11 ON WILL CAUSE DEBUG PRINT
11 CDO********
12 CD3 OUTPUTS
13 CD3 RMPAICS - RETURN PARAMETERS FOR FDS MANAGER VIA XPXIT
14 CD3 (1) = 0 => CONTINUE NORMAL PROCESSOR EXECUTION SEQUENCE
15 CDO********
16 CD5 ROUTINES USED
17 CD5 RMPAIR
18 CD5 XPXIT
19 CD5 XUDBG
20 CDO********
21 CDO
FORTTRAN MAIN PROGRAM - STANDARD FDS SCHEDULING PARAMETERS

THE IF UTILITY PROCESSOR COMPARES TWO INPUT QUANTITIES ACCORDING TO A
SPECIFIED RELATION. WHEN THE COMPARISON IS FALSE, THE SEQUENCE RESET
NUMBER IS SET (VIA XPXIT) TO THE PROCESSOR FOLLOWING THE CORRESPONDING
ELSE (OR ENDIF). IF THE CONDITION IS TRUE THE NORMAL PROCESSOR
EXECUTION SEQUENCE IS FOLLOWED.

INPUT

SCHEDULING PARAMETERS

LU - LOGICAL UNIT NO. OF THIS FDS USER
FLAGS - DEBUG FLAGS -- BIT 11 ON WILL CAUSE DEBUG PRINT

INTERFACE TABLE PARAMETERS

OPRD1 - FIRST REAL VALUE TO COMPARE
RELAT - RELATION IDENTIFIER (>), (=), <=, =, <=, <, <=, >=, =, <=, <
OPRD2 - SECOND REAL VALUE TO COMPARE

OUTPUT

RPARMS - RETURN PARAMETERS SENT TO FDS MANAGER VIA XPXIT
(1) = 0 => CONTINUE WITH NORMAL PROCESSOR EXECUTION SEQUENCE
= 3 => PROCESSOR EXECUTION SEQUENCE TO CONTINUE AT THE SEQUENCE
= 1 => SEQUENCE NO. GIVEN IN RPARMS(2) NO. GIVEN IN RPARMS(2)
= 8 => ABNORMAL TERMINATION OF THE PROCESSOR EXECUTION SEQUENCE
(2) = SEQUENCE NO. TO BE EXECUTED NEXT WHEN RPARMS(1) = 3

REFERENCE ROUTINES

RMPAR
XGET
XPXIT
XIEVL
XISCM
XEXIT
XJUGA

1 BEGIN IF
2   CALL IMPAR TO GET INPUT (SCHEDULING) PARAMETERS
3   CALL XPGET TO RETRIEVE VALUES FOR INTERFACE TABLE INPUTS
4   CALL RXGET TO RECEIVE REAL-TIME INPUTS
5   IF TRUE, THEN
6     VERIFY RELATIONAL OPERATOR INPUT AS VALID AND TRANSLATE IT TO A CODE
7   ELSE IF RELATIONAL OPERATOR INVALID :ERROR:
8     CALL XIXML TO EVALUATE THE RELATIONAL EXPRESSION
9     IF THE EXPRESSION IS FALSE, THEN
10    CALL IXINCH TO LOCATE THE ELSE OR ENL: CORRESPONDING TO THIS IF
11    AND SET SEQUENCE RESET NUMBER
12   ELSE
13    CLEAR SEQUENCE RESET NUMBER FOR NORMAL CONTINUATION OF THE SEQUENCE
14  ENDIF
15  IF
16  :ERROR: CALL XIMSG 'INVALID RELATIONAL OPERATOR - MUST BE #,>=,>,=,<,<=,OR =>
17  END IF
INPUT
OPRD1 - FIRST REAL VALUE TO COMPARE
OPRD2 - SECOND REAL VALUE TO COMPARE
RELAT - RELATION CODE AS FOLLOWS
0 = NOT EQUAL
1 = GREATER THAN
2 = GREATER THAN OR EQUAL
3 = EQUAL
4 = LESS THAN OR EQUAL
5 = LESS THAN

INTERNAL VARIABLES
TABLE - BIT MASK REPRESENTING TRUTH TABLE VALUES DERIVED AS
FOLLOWS
OPRD1 - OPRD2

<table>
<thead>
<tr>
<th>CODE RELATION</th>
<th>OPRD1 - OPRD2</th>
<th>OPRD1 - OPRD2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>&gt;</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>&gt;=</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>=</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>&lt;=</td>
<td>F</td>
</tr>
</tbody>
</table>

(By complementing code > 2)

Therefore for OPRD1 - OPRD2 positive (bits 0-2) 1, 2, 1
For OPRD1 - OPRD2 zero (bits 3-5) 0, 0, 1
And for OPRD1 - OPRD2 negative (bits 6-8) 1, 0, 0
(Bits 9-15 are not used) if RELAT is 0, 1, 2, or 3
Complement if RELAT is 3, 4, or 5

REFERENCES
Routines
XEXIT
No checks for valid relation code; or overflow/under flow are made
1 BEGIN X:EV.
2 CASE (4:1, 4:0, 4:0): DIFFERENCE OF OPFRD1 AND OPFRD2
3 1: SET FIELD OFFSET TO ZERO (BITS 0-2 OF TTABLE)
4 2: SET FIELD OFFSET TO THREE (BITS 3-5 OF TTABLE)
5 3: SET FIELD OFFSET TO SIX (BITS 6-8 OF TTABLE)
6 4: END CASE
7 5: IF RELATH > 2 (BOTTOM OF TRUTH TABLE), THEN
8 6: COMPLEMENT TTABLE
9 7: DECREMENT RELATH BY 3
10 8: ENDIF
11 9: ADD RELATH TO FIELD OFFSET (INDEXES TO CORRECT 9TH FOR RELATH AND DIFFERENCE)
12 10: SET FUNCTION VALUE TO INDEXED BIT OF TTABLE
13 11: END X:EV.
CALL ZISCH (SRCFG, RPAMS)

ZISCH IS CALLED TO LOCATE THE CORRESPONDING ELSE/ENDIF IN THE CURRENT
SEQUENCE TABLE. THE RETURN PARAMETERS FOR THE FBS MANAGER ARE OUTPUT

INPUTS

CALLING SEQUENCE

SRCFG - FLAG INDICATING ORIGINATING OF THIS CALL
= 0 => CALLED BY IF TO FIND MATCHING ELSE OR ENDIF
= 1 => CALLED BY ELSE TO FIND MATCHING ENDIF

COMMON

DEBUG - FLAG INDICATING WHETHER ONLINE DEBUG TO BE OUTPUT
= 0 => NO DEBUG
> 0 => DEBUG

RPAMS - SCHEDULE PARAMETERS FROM THE FBS MANAGER
(1) - LOGICAL UNIT NO. OF THE FBS USER
(5) - SEQSP, INDEX INTO SEQUENCE TABLE (&SERQTAB) OF THE CURRENT
COMMAND

OUTPUTS

CALLING SEQUENCE

RPAMS - PARAMETERS TO BE RETURNED TO FBS MANAGER VIA XPIT?

(1) - = 0 => CONTINUE NORMAL PROCESSOR EXECUTION SEQUENCE
= 3 => EXECUTE SEQUENCE NO. GIVEN BY RPAMS(2) NEXT
= 6 => ABNORMALLY TERMINATE PROCESSOR EXECUTION SEQUENCE
(2) - SEQUENCE NO. TO BE EXECUTED NEXT IF RPAMS(1) = 3

Routines Used

EXEC

RMPAR

XPAN

XUDS

XRCPR

XIRC

EXEC

Routines Used
BEGIN XISCH
RETRIEVE &SESTA3 FROM THE AVA USING XVPAM
STARTSEARCH UNTIL ALL COMMANDS IN &SESTAB
EXIT IF CURRENT COMMAND IS FOUND
SET NUMBER OF IF NESTS TO 1
STARTSEARCH FROM NEXT COMMAND IN &SESTAB UNTIL ALL FOLLOWING COMMANDS
IF COMMAND IS FOR ENDIF PROCESSOR, THEN
DECREMENT NUMBER OF IF NESTS BY 1
ELSE
IF COMMAND IS FOR IF PROCESSOR, THEN
INCREMENT NUMBER OF IF NESTS BY 1
ELSE
IF CALLED BY IF PROCESSOR, AND
COMMAND IS FOR ELSE PROCESSOR, THEN
ERREXIT IF THIS IS THE END OF &SESTAB :ERROR1:
IF NUMBER OF IF NESTS IS 1, THEN
DECREMENT NUMBER OF IF NESTS TO 0
ENDIF
ENDIF
EXIT IF NUMBER OF IF NESTS IS 0
SET SEQUENCE RESET NUMBER (RPARMS(2)) TO BE SEQUENCE NUMBER OF THE
NEXT COMMAND IN THE TABLE
ENDLOOP
ERREXIT :ERROR1:
ENDSEARCH
ENDLCP
ERREXIT :ERROR4:
ENDSEARCH
EXIT XISCH
:ERROR1: CALL X2MSG - 'IF CANNOT "E EXECUTED WITHOUT MATCHING ENDIF' 
:ERROR4: CALL X.MSG - 'SYSTEM ERROR - NO $SESTAB'
END XISCH
SYMBOL DEFINITION TABLE

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO</td>
<td>124</td>
</tr>
<tr>
<td>ELSE</td>
<td>230</td>
</tr>
<tr>
<td>ENDDO</td>
<td>298</td>
</tr>
<tr>
<td>ENDF</td>
<td>361</td>
</tr>
<tr>
<td>ERRO1</td>
<td>591</td>
</tr>
<tr>
<td>ERRO2</td>
<td>186</td>
</tr>
<tr>
<td>ERRO3</td>
<td>428</td>
</tr>
<tr>
<td>ERRO4</td>
<td>187</td>
</tr>
<tr>
<td>ERRD4</td>
<td>592</td>
</tr>
<tr>
<td>IF</td>
<td>415</td>
</tr>
<tr>
<td>OTHER</td>
<td>175</td>
</tr>
<tr>
<td>UNTIL</td>
<td>173</td>
</tr>
<tr>
<td>WHILE</td>
<td>157</td>
</tr>
<tr>
<td>XIEVL</td>
<td>490</td>
</tr>
<tr>
<td>YISCH</td>
<td>560</td>
</tr>
<tr>
<td>:=</td>
<td>494</td>
</tr>
<tr>
<td>:=</td>
<td>492</td>
</tr>
<tr>
<td>:D</td>
<td>493</td>
</tr>
</tbody>
</table>

FIN
6.0 DETAILED LOGIC FLOW LISTING - PROGRAM EXECUTION

The initial pages and tailsheet of the program execution that produced this volume is presented.
<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>TYPE</th>
<th>DATE</th>
<th>TIME</th>
<th>SEQ #</th>
<th>SIZE-PRE, TEXT (CYCLE, WORD)</th>
<th>PSRMODE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORT2</td>
<td></td>
<td>ELT SYMB</td>
<td>15 APR 77</td>
<td>12:55:40</td>
<td>1</td>
<td>55 s 0 1</td>
<td>1972</td>
<td></td>
</tr>
<tr>
<td>FSTURD</td>
<td></td>
<td>ELT SYMB</td>
<td>15 APR 77</td>
<td>12:56:07</td>
<td>2</td>
<td>20 s 0 1</td>
<td>1843</td>
<td></td>
</tr>
<tr>
<td>SORT2</td>
<td></td>
<td>ELT SYMB</td>
<td>15 APR 77</td>
<td>12:56:32</td>
<td>3</td>
<td>55 s 0 1</td>
<td>1975</td>
<td></td>
</tr>
<tr>
<td>XUPDL</td>
<td></td>
<td>ELT SYMB</td>
<td>15 APR 77</td>
<td>12:56:32</td>
<td>2</td>
<td>55 s 0 1</td>
<td>1932</td>
<td></td>
</tr>
<tr>
<td>POLIST</td>
<td></td>
<td>ELT SYMB</td>
<td>08 AUG 77</td>
<td>09:10:28</td>
<td>4</td>
<td>87 s 0 1</td>
<td>2016</td>
<td></td>
</tr>
<tr>
<td>RAP</td>
<td></td>
<td>ELT SYMB</td>
<td>09 AUG 77</td>
<td>09:10:34</td>
<td>6</td>
<td>2 s 1 1</td>
<td>2103</td>
<td></td>
</tr>
<tr>
<td>POLIST</td>
<td></td>
<td>ABSOLUTE</td>
<td>08 AUG 77</td>
<td>09:10:43</td>
<td>7</td>
<td>280 s 1 1</td>
<td>2105</td>
<td></td>
</tr>
<tr>
<td>XC</td>
<td></td>
<td>ELT SYMB</td>
<td>08 AUG 77</td>
<td>09:10:43</td>
<td>8</td>
<td>100 s 0 1</td>
<td>2345</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td>ELT SYMB</td>
<td>10 NOV 77</td>
<td>09:19:22</td>
<td>9</td>
<td>394 s 2 3</td>
<td>2494</td>
<td></td>
</tr>
<tr>
<td>MESSGE</td>
<td></td>
<td>ELT SYMB</td>
<td>11 JAN 78</td>
<td>23:22:28</td>
<td>10</td>
<td>55 s 0 1</td>
<td>2888</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td>ELT SYMB</td>
<td>12 JAN 78</td>
<td>00:40:28</td>
<td>11</td>
<td>373 s 2 3</td>
<td>2923</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td>ELT SYMB</td>
<td>13 FEB 78</td>
<td>19:17:22</td>
<td>12</td>
<td>824 s 9 5</td>
<td>3267</td>
<td></td>
</tr>
<tr>
<td>XI</td>
<td></td>
<td>ELT SYMB</td>
<td>17 FEB 78</td>
<td>12:48:36</td>
<td>13</td>
<td>627 s 1 5</td>
<td>4120</td>
<td></td>
</tr>
<tr>
<td>XV</td>
<td></td>
<td>ELT SYMB</td>
<td>18 FEB 78</td>
<td>23:50:36</td>
<td>14</td>
<td>320 s 2 5</td>
<td>4747</td>
<td></td>
</tr>
<tr>
<td>COMM C</td>
<td></td>
<td>ELT SYMB</td>
<td>03 MAR 78</td>
<td>22:20:04</td>
<td>15</td>
<td>200 s 2 5</td>
<td>4657</td>
<td></td>
</tr>
<tr>
<td>POLIST</td>
<td></td>
<td>RELOCATABLE</td>
<td>03 MAR 78</td>
<td>05:36:51</td>
<td>16</td>
<td>2 s 1 1</td>
<td>5067</td>
<td></td>
</tr>
<tr>
<td>FSTURD</td>
<td></td>
<td>RELOCATABLE</td>
<td>03:17:36</td>
<td>17</td>
<td>1</td>
<td>3 s 0 1</td>
<td>5068</td>
<td></td>
</tr>
<tr>
<td>XA</td>
<td></td>
<td>ELT SYMB</td>
<td>17 APR 78</td>
<td>10:54:29</td>
<td>18</td>
<td>101 s 10 5</td>
<td>5094</td>
<td></td>
</tr>
<tr>
<td>XP</td>
<td></td>
<td>ELT SYMB</td>
<td>17 APR 78</td>
<td>10:54:52</td>
<td>19</td>
<td>534 s 2 5</td>
<td>5193</td>
<td></td>
</tr>
<tr>
<td>XT</td>
<td></td>
<td>ELT SYMB</td>
<td>17 APR 78</td>
<td>11:43:48</td>
<td>20</td>
<td>384 s 6 5</td>
<td>5729</td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td></td>
<td>ELT SYMB</td>
<td>18 APR 78</td>
<td>09:12:58</td>
<td>21</td>
<td>1548 s 4 5</td>
<td>6113</td>
<td></td>
</tr>
<tr>
<td>XE</td>
<td></td>
<td>ELT SYMB</td>
<td>20 AUG 78</td>
<td>12:52:31</td>
<td>22</td>
<td>403 s 2 5</td>
<td>7661</td>
<td></td>
</tr>
<tr>
<td>XR</td>
<td></td>
<td>ELT SYMB</td>
<td>20 AUG 78</td>
<td>12:52:39</td>
<td>23</td>
<td>508 s 2 5</td>
<td>8094</td>
<td></td>
</tr>
<tr>
<td>XU</td>
<td></td>
<td>ELT SYMB</td>
<td>20 AUE 78</td>
<td>12:52:42</td>
<td>24</td>
<td>151 s 5 5</td>
<td>8572</td>
<td></td>
</tr>
<tr>
<td>XP</td>
<td></td>
<td>ELT SYMB</td>
<td>25 SEP 78</td>
<td>09:46:47</td>
<td>25</td>
<td>118 s 3 5</td>
<td>8723</td>
<td></td>
</tr>
<tr>
<td>XE</td>
<td></td>
<td>ELT SYMB</td>
<td>25 SEP 78</td>
<td>09:47:13</td>
<td>26</td>
<td>1201 s 21 5</td>
<td>8841</td>
<td></td>
</tr>
<tr>
<td>XK</td>
<td></td>
<td>ELT SYMB</td>
<td>25 SEP 78</td>
<td>09:47:13</td>
<td>27</td>
<td>600 s 29 5</td>
<td>10042</td>
<td></td>
</tr>
<tr>
<td>XI</td>
<td></td>
<td>ELT SYMB</td>
<td>25 SEP 78</td>
<td>09:47:39</td>
<td>28</td>
<td>326 s 2 3</td>
<td>10642</td>
<td></td>
</tr>
</tbody>
</table>

Next available location:
- ASSEMBLER PROCEDURE TABLE EMPTY
- COBOL PROCEDURE TABLE EMPTY
- FORTRAN PROCEDURE TABLE EMPTY
- ENTRY POINT TABLE EMPTY

DXQT F.POLIST