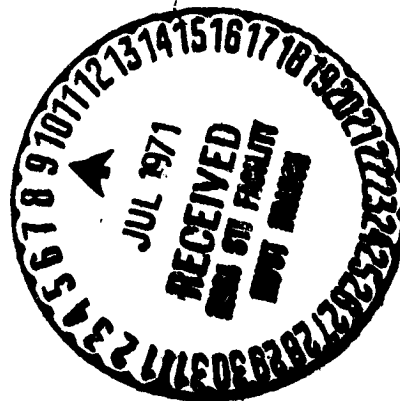
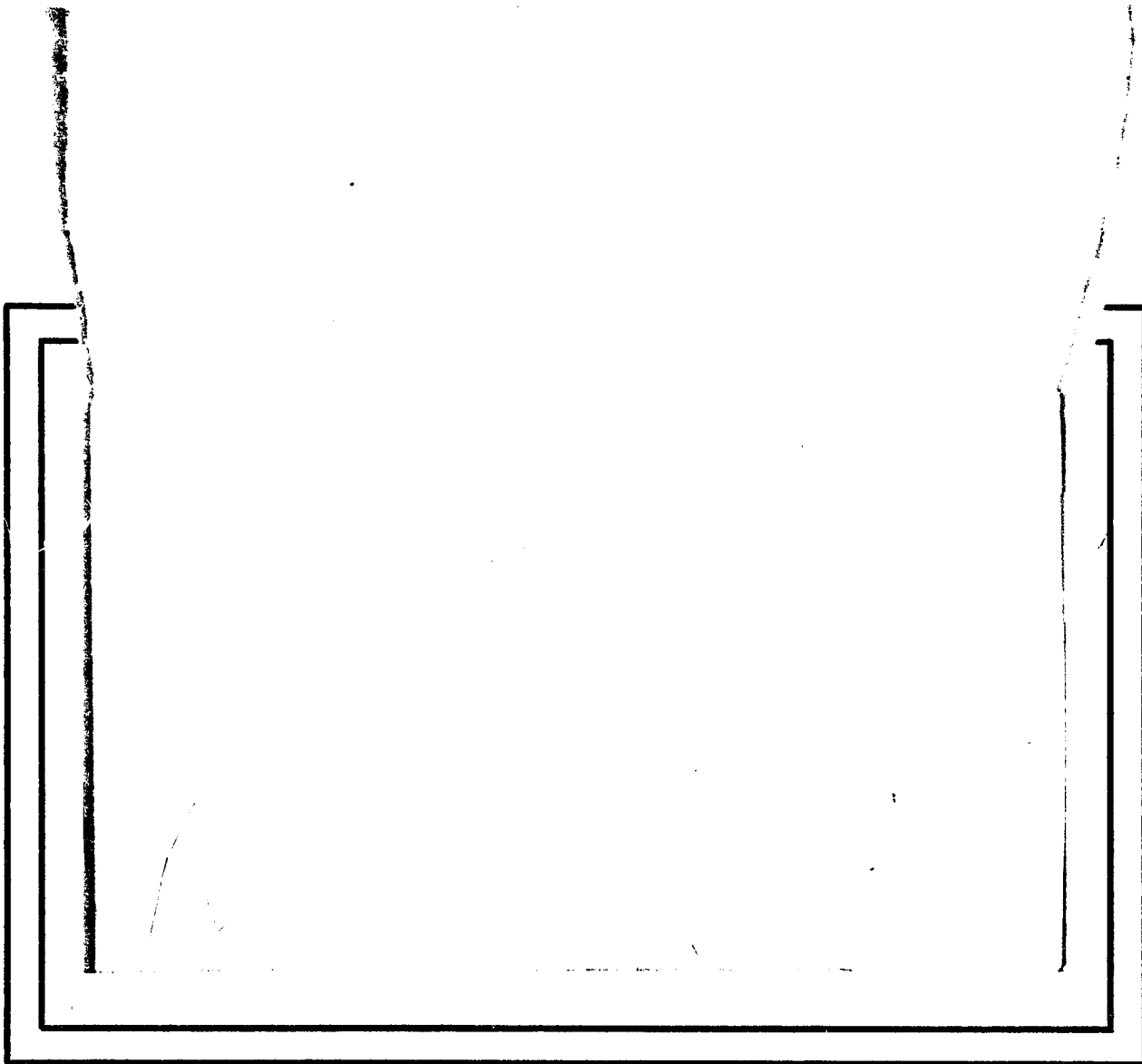


General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.



UNIVERSITY OF MARYLAND COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND

N71-30109

FACILITY FORM 602

(ACCESSION NUMBER)

468

(PAGES)

OL-119188

(NASA CR OR TMX OR AD NUMBER)

(THRU)

G3

(CODE)

08

(CATEGORY)

TECHNICAL REPORT 70-134
JPL CONTRACT NO. 952695
NGL-21-002-008

SEPTEMBER 1970

FORMAL SYSTEM:
TECHNICAL DOCUMENTATION
BY
CHARLES K. MESZTENYI

THIS RESEARCH WAS SUPPORTED BY CONTRACT 952695 FROM THE JET
PROPULSION LABORATORY OF THE CALIFORNIA INSTITUTE OF TECHNOLOGY
AND BY GRANT NGL-21-002-008 FROM THE NATIONAL AERONAUTICS AND
SPACE ADMINISTRATION TO THE COMPUTER SCIENCE CENTER OF THE
UNIVERSITY OF MARYLAND.

ABSTRACT

FORMAL IS AN EXTENSION OF FORTRAN V, WHICH PROVIDES THE USER WITH THE CAPABILITY OF PERFORMING SYMBOLIC ALGEBRAIC MANIPULATION. IT IS ALSO AVAILABLE AS AN INTERACTIVE SYMBOLIC MANIPULATOR. THIS DOCUMENT CONSISTS OF THE TECHNICAL DOCUMENTATION, INCLUDING PROGRAM LISTINGS, OF FORMAL FOR THE UNIVAC 1108.

ACKNOWLEDGMENT

THE AUTHOR OF THIS REPORT GRATEFULLY ACKNOWLEDGES THE CONTRIBUTIONS OF HIS COLLEAGUES:

MR. HANS BREITENLOHNER ANALYZED AND WROTE THE PREPROCESSOR FOR THE FORMAL SYSTEM AND HELPED IN MANY OTHER PARTS OF THE PROJECT WHICH WOULD BE TOO NUMEROUS TO LIST.

MR. RAY PARDO, MR. ROBERT NUNN AND MR. JOSEPH YEH HELPED THE AUTHOR TO DEVELOP THE OBJECT TIME ROUTINES AND THE DOCUMENTATION.

LAST, BUT NOT LEAST, THE AUTHOR IS VERY GRATEFUL TO MR. JOHN MENARD, ASSOCIATE DIRECTOR, WHOSE ENCOURAGEMENT AND ADMINISTRATIVE HELP MADE IT POSSIBLE TO FINISH THE PROJECT IN A RELATIVELY SHORT TIME.

TABLE OF CONTENTS

	ABSTRACT		
	ACKNOWLEDGEMENT		
1.	INTRODUCTION	1-1	
2.	PREPROCESSOR	2-1	
2.1.	PREPROCESSOR MAIN PROGRAM	2-1	
2.2.	FORTRAN PROCEDURES	2-20	
2.2.1.	PARAM	2-20	
2.2.2.	BUF	2-20	
2.2.3.	DEFINE	2-20	
2.3.	UTILITY SUBROUTINES	2-23	
2.3.1.	PACK	2-23	
2.3.2.	CHAR	2-23	
2.3.3.	TYPE	2-24	
2.4.	SCANNING SUBROUTINES	2-33	
2.4.1.	GET	2-33	
2.4.2.	GIT	2-33	
2.4.3.	SKIP	2-33	
2.5.	STATEMENT TRANSLATION	2-37	
2.5.1.	STDET	2-37	
2.5.2.	NONARITH	2-37	
2.5.3.	FUNTST	2-38	
2.5.4.	LIST	2-38	
2.5.5.	STACK	2-38	
3.	OBJECT TIME ROUTINES	3-1	
3.1.	DATA STRUCTURE AND BASIC UTILITY ROUTINES	3-2	
3.1.1.	CORE STORAGE	3-2	
3.1.2.	DRUM STORAGE	3-19	
3.1.3.	FILE STORAGE	3-34	
3.1.4.	INITIALIZATION ROUTINES	3-35	
3.1.5.	SCANNER ROUTINE	3-43	
3.1.6.	ERROR ROUTINE	3-59	
3.1.7.	SYMBOL TABLE ROUTINES	3-62	
3.1.8.	STRING AND ARRAY UTILITY ROUTINES	3-75	
3.1.9.	CONSTANT ARITHMETIC OPERATION ROUTINES	3-78	
3.1.10.	VARIABLE LIST HANDLING ROUTINES	3-87	
3.2.	EXPRESSION TRANSLATION FROM INFIX TO PREFIX	3-90	
3.2.1.	EXPRES	3-101	
3.2.2.	STOUT0	3-111	
3.2.3.	STOUT1	3-131	
3.2.4.	STOUT2	3-161	
3.2.5.	STOUT3	3-169	
3.2.6.	STOUT4	3-172	
3.2.7.	STOUT5	3-184	
3.3.	EXPRESSION TRANSLATION FROM PREFIX TO INFIX	3-186	
3.3.1.	FMLWR AND FMLPKS	3-186	
3.3.2.	FMLPCK	3-195	
3.4.	STATEMENT ROUTINES	3-200	
3.4.1.	I/O STATEMENT	3-200	

3.4.2.	ROLOUT STATEMENT	3-214
3.4.3.	SAVE AND RESET STATEMENTS	3-216
3.4.4.	DUMP STATEMENT	3-229
3.4.5.	ASSIGN STATEMENT	3-234
3.4.6.	OPTION STATEMENT	3-236
3.4.7.	ERASE STATEMENT	3-240
3.4.8.	DO AND END STATEMENTS	3-242
3.4.9.	FORTTRAN VALUED FUNCTIONS AND NCOUNT	3-248
3.4.10.	FMLEQU ROUTINE	3-254
3.5.	INTERACTIVE MAIN PROGRAM	3-258
4.	INSTALLATION	4-1
4.1.	TAPE FORMAT	4-1
4.2.	MANIPULATION OF THE TAPE	4-2
4.2.1.	INSTALLATION	4-2
4.2.2.	PRINT THE USER'S MANUAL	4-2
4.2.3.	LISTING THE PRINT FILE	4-3
5.	UPDATING PROCEDURE	5-1
5.1.	UPDATING PHILOSOPHY -- SSG	5-1
5.2.	STANDARD UPDATING PROCEDURE	5-1
5.2.1.	STREAM GENERATION STATEMENTS	5-3
5.2.2.	USER IMPLEMENTED UPDATES	5-8
5.3.	EXAMPLE	5-8
6.	CROSS-REFERENCE TABLES	6-1
7.	INDEX FOR PROGRAM ELEMENTS	7-1

1. INTRODUCTION

FORMAL IS A PROGRAMMING SYSTEM WHICH IS AN EXTENSION OF FORTRAN. IT CONSISTS OF A LANGUAGE, A PREPROCESSOR, THE FORTRAN COMPILER, AND OBJECT TIME ROUTINES. THE PURPOSE OF THE SYSTEM IS TO PROVIDE A CAPABILITY FOR PERFORMING SYMBOLIC ALGEBRAIC MANIPULATIONS IN ADDITION TO THE NUMERICAL CALCULATION CAPABILITY OF FORTRAN. THE SYSTEM IS ALSO AVAILABLE AS AN INTERACTIVE SYSTEM WITHOUT FORTRAN. THE SYSTEM IS IMPLEMENTED FOR THE UNIVAC 1108 UNDER EXEC 8. THE DESCRIPTION OF THE LANGUAGE AND ITS USE IS CONTAINED IN TECHNICAL REPORT 70-133, OR AS A USER'S MANUAL IN COMPUTER NOTE CN-1. THIS REPORT CONTAINS THE COMPLETE TECHNICAL DOCUMENTATION OF THE SYSTEM.

THE EXECUTION OF A FORMAL PROGRAM IS DONE IN THREE STEPS. FIRST, THE USER'S SOURCE PROGRAM IS PROCESSED BY A PREPROCESSOR. THE PREPROCESSOR TRANSFORMS THE FORMAL TYPE OF STATEMENTS INTO FORTRAN SUBROUTINE CALLS. THESE SUBROUTINES CORRESPOND TO THE OBJECT TIME ROUTINES. FOLLOWING THE PREPROCESSING, THE FORTRAN COMPILER TRANSLATES THE SOURCE PROGRAM INTO A MACHINE LANGUAGE PROGRAM. LAST, THE MACHINE LANGUAGE PROGRAM IS EXECUTED WITH THE HELP OF THE PROVIDED OBJECT TIME ROUTINES. CHAPTERS 2 AND 3 CONTAIN THE DETAILED DOCUMENTATION OF THE PREPROCESSOR AND OBJECT TIME ROUTINES.

THE INTERACTIVE FORMAL SYSTEM CONSISTS ONLY OF THE OBJECT TIME ROUTINES AND OF A MAIN PROGRAM DESCRIBED IN CHAPTER 3 SECTION 5.

CHAPTERS 4 AND 5 CONTAIN THE INSTALLATION AND UPDATING PROCEDURES, RESPECTIVELY. AT THE TIME OF THE IMPLEMENTATION OF THE SYSTEM, UNIVAC'S FORTRAN V COMPILER HAD ERRORS WHICH MADE THE DEBUGGING ALMOST IMPOSSIBLE. BECAUSE OF THIS, THE UNIVERSITY OF MARYLAND'S RALPH COMPILER HAS BEEN USED. A COPY OF THIS COMPILER IS INCLUDED ON THE PROGRAM TAPE.

CHAPTERS 6 AND 7 CONSIST OF CROSS REFERENCE AND INDEX TABLES FOR THE PROGRAMS DESCRIBED IN CHAPTERS 2 AND 3. THE CROSS REFERENCE TABLES WERE GENERATED AUTOMATICALLY FROM THE FORMAL SYSTEM PROGRAM FILE.

THE DESCRIPTION OF INDIVIDUAL PROGRAMS IN CHAPTERS 2 AND 3 IS PROVIDED IN THREE DIFFERENT FORMS. EACH SECTION STARTS WITH TEXT SUMMARIZING THE PROGRAMS RELEVANT TO THE SECTION. THIS IS FOLLOWED BY A DEPENDENCE TREE OF THESE PROGRAMS. THIS IS NOT A FLOW CHART, BUT IT INDICATES WHICH PROGRAMS CALL OTHER PROGRAMS AS SUBROUTINES. THE LAST FORM OF PROGRAM DOCUMENTATION ARE THE LISTS OF THE SOURCE PROGRAMS. THE SOURCE PROGRAMS WERE COMMENTED SUCH THAT A SPECIAL LIST PROGRAM SEPARATED THE COMMENTS FROM THE OTHER

1-2

STATEMENTS AND PROVIDED A LOGICAL SEQUENCE CHART FOR THE PROGRAM; AGAIN THIS IS NOT A FLOW CHART BUT IT IS ARRANGED SUCH THAT THE MAIN COMMENTS ARE THOSE WHICH START ON THE LEFT SIDE, AND COMMENTS DESCRIBING DETAILS ARE INDENTED.

THIS DOCUMENT WAS PROCESSED BY UNIVAC'S DOCUMENT PROCESSOR WHICH HAD BEEN MODIFIED TO ALLOW LARGER PAGE SIZES FOR PROGRAM LISTINGS.

2. PREPROCESSOR

THE PREPROCESSOR TRANSLATES FORMAL SOURCE PROGRAMS TO FORTRAN SOURCE PROGRAMS BY CONVERTING THE FORMAL STATEMENTS TO FORTRAN SUBROUTINE CALLS.

THE TRANSLATION IS DONE IN TWO PASSES. IN THE FIRST PASS SOURCE INPUT AND POSSIBLY CORRECTIONS ARE COLLECTED (BY SIR). THEY ARE LISTED TOGETHER WITH DIAGNOSTICS, TRANSLATED, AND WRITTEN INTO A SCRATCH-FILE. IN THE SECOND PASS THE SCRATCH FILE IS COPIED TO THE SOURCE OUTPUT FILE, AND THE GENERATED PROGRAM IS LISTED. IT IS NECESSARY TO USE TWO PASSES, BECAUSE UPDATING OF SOURCE INPUT AND WRITING OF SOURCE OUTPUT CANNOT BE DONE SIMULTANEOUSLY, IF THEY HAPPEN TO GO TO THE SAME PROGRAM FILE.

2.1. PREPROCESSOR MAIN PROGRAM

THE MAIN PROGRAM IS FORMAL. IT INITIALIZES H\$MONITOR (WHICH CATCHES CONTINGENCIES) AND PRINTS A SIGN-ON LINE (STANDARD PROCESSOR FORMAT). THE VERSION IS TAKEN FROM THE ELEMENT F\$MLVER. FORMAL THEN CALLS PREPRO TO DO THE FILE-ASSIGNMENTS, ASSIGNS A SCRATCH FILE, OPENS SOURCE INPUT BY CALLING OPNSRC, AND OPENS SCRATCH OUTPUT BY CALLING SDFOO. THEN IT CALLS SUBROUTINE STDET TO DO THE ACTUAL TRANSLATION OF THE STATEMENTS.

ON RETURN FROM STDET (ENTIRE PROGRAM TRANSLATED), FORMAL CLOSES THE SCRATCH FILE, AND COPIES IT INTO THE SOURCE OUTPUT ELEMENT, USING SOR. IT THEN CALLS POSTPR TO FREE THE FILES, AND PRINTS THE NUMBER OF ERRORS FOUND, IF ANY, AND A SIGN-OFF LINE.

THE ELEMENT FORMAL ALSO CONTAINS SUBROUTINES USED BY OTHER PROGRAMS:

CRDIN GETS AN IMAGE THROUGH GETSRC

IT IS CALLED WITH THREE ARGUMENTS, A BUFFER ADDRESS, THE VARIABLE WHICH IS TO BE SET TO THE NUMBER OF WORDS IN THE IMAGE, AND AN EOF RETURN.

THE ROUTINE WILL LIST THE PREVIOUS IMAGE, IF THERE WAS ONE, AND IF LISTFL IS SET. THEN IT WILL CALL GETSRC FOR A NEW IMAGE, TRANSFER IT TO THE USER, AND RETURN. IF GETSRC RETURNS AT THE EOF RETURN, CLOSRC IS CALLED TO CLOSE THE SOURCE INPUT, AND THE EOF RETURN TO THE CALLER IS TAKEN.

CRDOUT WRITES AN IMAGE TO THE SCRATCH FILE

IT IS CALLED WITH TWO ARGUMENTS, THE IMAGE AREA AND THE IMAGE

2-2

LENGTH. IT CALLS SDFO TO WRITE THE IMAGE TO THE SCRATCH FILE.

ERR AND WARN PRINT DIAGNOSTICS

THEY ARE BOTH CALLED WITH TWO ARGUMENTS, THE ADDRESS OF A DIAGNOSTIC MESSAGE, AND ITS LENGTH IN WORDS. ERR SETS THE FLAG THAT THERE WERE ERRORS, AND INCREMENTS THE ERROR COUNT. BOTH ROUTINES PRINT THE CURRENT SOURCE-LINE, UNLESS IT HAS ALREADY BEEN PRINTED, AND THE DIAGNOSTIC MESSAGE, WHICH IS PREFIXED BY '<<<< ERROR: ' OR '<<<< WARNING '. THEN THEY RETURN TO THE CALLER.

LISTOP SETS THE LISTING-OPTION

THIS ROUTINE SETS LISTFL TO THE VALUE OF THE ARGUMENT IT IS CALLED WITH.

160

FORMAL,FMLVER
VERSION FOR THE PREPROCESSOR

FORMAL,FORMAL

PREPROCESSOR MAIN PROGRAM
STANDARD PROCESSOR SIGNON LINE

DEFINE BUFFER LENGTH FOR SDF

SIGNOFF LINES

ASG AND FREE IMAGES FOR SCRATCH FILE

ERROR MESSAGES

FMLVER*	EQU	' 0902A'	
END			
	AXRS		
SIGNON	'FORMAL'		
+	FMLVER		
	'-MM/DD-HH:MM-(00)		
SIGNXX			
CHAR	'%',077		
BUFLEN	EQU	4	LENGTH OF THE SDF BUFFERS IN S
BUFWDS	EQU	28*BUFLEN	SAME IN WORDS
PRINT*	PROC		PRINT WOULD DO THE SAME
	L	A0,PRINT(1,1)	
	ER	PRINTS	
	END		
P	PRJC		
SNZ*	NAME		STORE NON-ZERO, THIS IS REALLY
	SR	R15,P(1,1),P(1,2),P(1,3)+P(0,1)	
	END		
PF	FORM	12,6,18	
S(2)			
PRO	+	0101,S+1	TTY SIGNOFF
	'**%'		
PROO	+	0204,S+1	BATCH SIGNOFF
	'PREPROCESSOR TERMINATED'		
FREE	'@FREE'		SO WE CAN FREE THE FILE
ASG	'@ASG,T FORMAL\$TEMPS,F , '		ASSIGN IMAGE FOR THE SCRATCH=F
PRIO	+	0104,S+1	
	'I/O XX FILE '		
PRPFI	+	0104,S+1	
	'PFIS ERR XX '		
PRUND	+	0105,S+1	
	'PF NOT DEFINED: '		
PRCOR	+	0110,S+1	
	'CORRECTION CARD ERROR: '		
MONTHS	' JAN FEB MAR APR MAY JUN'		..THE MONTHS OF THE YE
	' JUL AUG SEP OCT NOV DEC'		IN CASE YOU DIDN'T N
MSGERR	'<<<< ERROR: '		
MSGWRN	'<<<< WARNING: '		
ERRPRB	+	0114,MONTHS	

HEADING FOR SOURCE OUTPUT LISTING

WALKBACK PACKET FOR ERROR TRACE

SCRATCHFILE SDF PACKET

ERRPRA	+	0103,S+1	.
	'	X ERRORS FOUND	.
SIGNPR	PF	1,SIGNXX-SIGNON,SIGNON	.
HEAD2	+	07704,S+1	.
	'	PREPROCESSOR OUTPUT:	.
WB	'	FORMAL	.
SAVX11	+	0	.
	+	'FOR',0	.
CRDWD	+	14,CBUF	.
PRWD	+	0100,COUNT	.
PRWD1	+	0101,COUNT	.
K10	+	1,0	.
ERRBT	+	1,MONTHS+2	.
FLAG	+	0	.
ERRFLG	EQUF	FLAG,,S1	.
EFLG	EQUF	FLAG,,S1	.
WOPT	EQUF	FLAG,,S2	.
OHOG	EQUF	FLAG,,S3	.
ERRCNT	EQUF	FLAG,,H2	.
BTWO	+	1,CBUF	.
SDFPKT			.
SDFOPK	'	FORMALSTEPS	.
	+	0	.
	+	WS,0,0	.
	+	BUFWDS,0	.
	+	0	.
	+	BU1,BU2	.
	+	BUFLN,14	.
	+	1,000	.
	+	1,0	.
	+	0	.
LISTFL	EQUF	SDFPKT+10,,S5	.
COUNT	'		.
CBUF	RES	14	.
BOPTN			.
ADDING	+	4,S+1	.
	'	QADD,E FORMALSADDs.	.
BU1			.
		ELTS	.
BOPT1	L,U	A0,ADDASG	.
	ER	CSFs	.
	JN	A0,0,X1	.

THIS SUBROUTINE PUTS A CONTROL CARD INTO THE ADD-FILE FORMALSADDs, WHICH WILL COMPILE THE GENERATED OUTPUT ELEMENT, AND CREATE A RELOCATABLE ELEMENT AS GIVEN BY SPEC4.
ASSIGN THE SCRATCH FILE

GET SPEC 2

PUT IT INTO THE IMAGE

GET SPEC 4

PACK IT TOO

SET SDF END OF FILE

WRITE THE BUFFER TO DRUM

READY TO BE ADDED

I/O ERROR, DON'T ADD

NO SPEC 2 GIVEN, USE SPEC1

SPEC1 MISSING TOO, USE NAMES

PACK ,<SPEC>

	L,U	A0,PREINF
	S,H2	A0,INFORS
	L,U	A0,2
	LMJ	X11,SELTS
B0PT2	J	B0PT10
	SZ	CFN
	LMJ	X11,B0PT8
	L,U	A0,4
	LMJ	X11,SELTS
	J	B0PT3
	L	A0,CFN
	TG,U	A0,3
	SZ	CFN
	LMJ	X11,B0PT7
B0PT3	L	A0,ADDEOF+2
	L,U	A1,0
	SN	A1,14,A0
	AN,U	A0,ADDSEC-15
	S,H1	A0,ADDIOR+4
	L,U	A0,ADDIOR
	ER	IOWs
	TNZ,S1	3,A0
	J	1,X1
	J	ERRIO
	L,U	A0,1
B0PT10	LMJ	X11,SELTS
	J	B0PT9
	J	B0PT2
	L	A0,ADDEOF+1
B0PT9	S	A0,ENAME
	SZ	ELTs
	SZ	ELTs+1
	L,U	A0,5
	S	A0,ENL
	J	B0PT2
	DL	A0,ADDEOF+4
B0PT7	L	A3,ADDEOF+6
	L,U	A5,,,'
	EX	0,*A1

PACK <SPEC>

MAKE SURE COMPLETE FILE DESCRIPTION FITS INTO IMAGE

PACK FILE SPECIFICATIONS

PACK . AFTER FILENAME

PACK ELEMENT NAME

	J	BOPT81
BOPT8	DL	A0,ADDEOF+4
	L	A3,ADDEOF+6
BOPT81	TNZ	CFN
	TNZ	ELT\$
	J	BOPT85
	L	A6,FNL
	A	A6,FCL
	A	A6,RKL
	A	A6,WKL
	A,U	A6,6
	AN	A3,A6
	L	R1,FGL
	DL	A4,FQUAL
	LMJ	A2,BOPTPK
	A	A3,A6
	L,U	A5,'*'
	TNZ	FGL
	TZ	IQF
	EX	0,*A1
	L	R1,FNL
	DL	A4,FNAME
	LMJ	A2,BOPTPK
	L	R1,FCL
	TNZ	R1
	J	BOPT82
	L,U	A5,'('
	EX	0,*A1
	AN,U	A3,1
	L	A4,FCYC
	LMJ	A2,BOPTPK
	L,U	A5,'))'
	EX	0,*A1
BOPT82	TNZ	RKL
	TZ	WKL
	J	BOPT84
BOPT83	L,U	A5,'.,'
	EX	0,*A1
BOPT86	L	R1,ENL
	DL	A4,ENAME
	LMJ	A2,BOPTPK
	L,U	A5,'/'
	TZ	EVL
	EX	0,*A1
	L	R1,EVL

SAVE POINTERS

PACK KEYS

ELEMENT ONLY, CHECK IF LEADING PERIOD IS NECESSARY

CHECK IF CURRENT IMAGE IS FULL

YES, MAKE A CONTINUATION CARD

	DL	A4,EVEN
	LMJ	A2,BOPTPK
	TNZ	ECL
	J	BOPT88
	AN,U	A3,1
	L,U	A5,('
	EX	0,*A1
	L	R1,ECL
	L	A4,ECYC
	LMJ	A2,BOPTPK
	L,U	A5,('
	EX	0,*A1
BOPT88	DS	A0,ADDEOF+4
	S	A3,ADDEOF+6
	J	0,X11
BOPT84	L,U	A5,('
	EX	0,*A1
	L	R1,RKL
	L	A4,RKEY
	LMJ	A2,BOPTPK
	L	R1,WKL
	TNZ	R1
	J	BOPT83
	L,U	A5,('
	EX	0,*A1
	L	A4,WKEY
	LMJ	A2,BOPTPK
	J	BOPT83
BOPT85	TZ	CFN
	J	BOPT83
	J	BOPT86
BOPTPK	AN	A3,R1
	JGD	A3,BOPTP1
	L	A3,A5
	L,U	A5,('
	EX	0,*A1
	L	A5,A3
	DL	A0,ADDEOF+2
	A,U	A0,15
	S	A0,ADDEOF+2
	L,U	A3,78
	J	BOPTPK
BOPTP0	LDSC	A4,6
	EX	0,*A1
BOPTP1	JGD	R1,BOPTP0

PRE-FABRICATED SDF-IMAGES FOR QADU-ING

I/O PACKET FOR ADD FILE

THIS LETS US USE THE SPACE TWICE

START, INITIALIZE HSMONITOR

PUT DATE/TIME INTO SIGNON LINE

```
J      0,A2
S,S6   A5,0,*A0
LMJ     A1,0,A1
BOPTX   S,S1   A5,0,A0
        S,S2   A5,0,A0
        S,S3   A5,0,A0
        S,S4   A5,0,A0
        S,S5   A5,0,A0
        LMJ     A1,BOPTX-2

ADDSEC  '[00000*SDF*
+        14,0,0
'GRALPH,S

ADDSC1  DO 12  , , '
+        14,0,0
DO 14    , , '
+        14,0,0
DO 14    , , '
ADDEOF  -      0
        'NAMES '
+        1,ADDSEC+3
+        1,BOPTX
+        1,ADDSC1
+        1,BOPTX
+        66

ADDAS6  'BASG,T
ADDIOP  'FORMALSADD
+        0
+        W3,0,0
+        48,ADDSEC
+        0

RES      BU1-S+BUFWD
RES      BUFWD
PARTBL*  RES    40
/
S(1)

FORMAL  LMJ     X11,HCONTS
+        0,WB
L        R15,COUNT
DS       A4,A14

L        A1,R1
L,U      A0,'-'
LDL      A0,12
LSSL     A0,6
A,U      A0,'/'
```

- EVERYONE USES THIS
-
-
- INITIALIZE HSMONITOR
-
- CONVENIENT WAY TO STORE BLANKS

CALL PREPRO FOR FILE ASSIGNMENT

INITIALIZE OPTION FLAGS

CALL BOPT1 TO PROCESS AUTOMATIC COMPILER CALL

```
LDL  A0,12
S    A0,SIGNON+2
L    A0,R2
LSSL A0,18
DSL  A0,54
DI,U A0,60
DSL  A0,36
DI,U A0,600
DSL  A1,36
DI,U A1,60
DSL  A2,36
DI,U A2,10
A,U  A0,'-0'
LSSL A0,6
A,U  A0,'0',A1
LSSL A0,12
A,U  A0,':0',A2
LSSL A0,6
A,U  A0,'0',A3
S    A0,SIGNON+3
```

```
LMJ  X11,PREPRO
ER    EXITS
L,S6  A0,PARTBL+13
DSL  A0,36
DI,U A0,10
LSSL A0,6
A,U  A0,'00',A1
S,T2 A0,SIGNON+4
PRINT SIGNPR
PRINT PRWD
```

```
L,U  A2,1
L    A5,A15
TNE,U A14,4
SZ    A2
TEP,U A5,1*/('Z'-'S')
L,U  A2,1
TEP,U A5,1*/('Z'-'L')
L,U  A2,2
TEP,U A5,1*/('Z'-'N')
SZ    A2
S    A2,LISTFL
S    A2,OHDS
TEP,U A5,1*/('Z'-'W')
SNZ   WOPT
SSL   A5,'Z'-'B'
TOP,U A5,1
```

```
LMJ  X1,BOPT1
```

. IGNORE SECONDS

```
. WE NEED PREPRO
. FOR THE CYCLE NUMBER
.
. CONVERT IT TO FIELDATA
.
.
. AND PRINT IT
. A BLANK LINE FOR PRETTYNESS
```

```
. NOW INITIALIZE LISTFL
. RECOVER OPTIONS
```

```
. AND N IF DEMAND
. THIS IS AN EXPLICIT S
.
. L OVERRIDES S
.
. N OVERRIDES THEM ALL
.
. DO WE HAVE W OPTION?
. YES, SET FLAG
```

ASSIGN SCRATCH FILE

OPEN OUTPUT

OPEN SIR

CALL STDET TO DO THE ACTUAL TRANSLATION

CLOSE SCRATCH FILE OUTPUT

PRINT OUTPUT HEADING

OPEN SOR

COPY IMAGES FOR SCRATCH TO SOURCE OUTPUT

SLOOP
END

SOLOOP

```

SZ      BOPIN
L,H2    AO, PARTBL
AND,U   AO, 0777767
S,H2    A1, PARTBL

L,U     AO, ASG
ER      CSFs

L,U     AO, SDFOPK
LMJ     X11, SDF00
L,U     AO, 4
S,S1    AO, PARTBL+36

LMJ     X11, OPNSRC
J       ERR1

LMJ     X11, STDET
+       S-FORMAL, WB

L,U     AO, SDFOPK
LMJ     X11, SDFOC
J       ERR2
SNZ     COUNT
SNZ     COUNT+1
L,U     A15, 2
L,U     AO, RS
S,T1    AO, SDFPKT+3
L,U     AO, BUFWDs
S,H1    AO, SDFPKT+4
SZ      SDFPKT+5
L,U     AO, 14
S,H2    AO, SDFPKT+7
L       AO, BTWD
S       AO, SDFPKT+8
L       AO, HEAU2
TEP     A15, OHDO

ER      PRINTs
SZ,S3   PRWD
PRINT   PRWD
SNZ,S3  PRWD

LMJ     X11, SSOR
J       ERR3
L,U     AO, SDFPKT
LMJ     X11, SDF10
J       ERR4

```

```

. ERASE W OPTION HERE
. SO WE DO NOT GET ALL THE
. CORRECTION CARDS PRINTED

. INITIALIZE THE SCRATCH FILE
. I DON'T CARE IF WE GOT IT

. OPEN SDF0
.
. SET THE LANGUAGE BITS
.

. OPEN SIR
. I/O ERROR FROM SUFI

. CALL THE TRANSLATOR
. WALK BACK WORD
. NORMAL RETURN: TRANSLATION IS

. CLOSE SDF0
.
. I/O ERROR
. BLANK OUT THE LINE NUMBER
. AND CORRECTION INDICATOR
. NOW WE CAN CHECK FOR L-OPTION
. SWITCH SDFPKT TO READ
.
. RESET THE NUMBER OF WORDS IN T
. ACCESS WORD
. AND ZERO DRUM ADDRESS
. RESTORE THE RIGHT NUMBER OF WO
. FOR MAXIMUM IMAGE LENGTH
.
. PRINT A NICE HEADING FOR THE N
. IF THE FLAG IS SET

. AND ANOTHER BLANK LINE
.

. OPEN SOURCE OUTPUT ROUTINE
. I/O (A2=2) OR PF UNDEFINED (A2
.
. OPEN SDFI
. I/O ERROR

```


END, CLOSE FILES

FREE SCRATCH FILE

CALL POSTPR TO FREE SOURCE INPUT, SOURCE OUTPUT, ETC.

PRINT ERROR SUMMARY

AND PRINT SIGNOFF

EOF2

ERREX

NOERR

L,U A0,SDFPKT
LMJ X11,SDFI
J ERR5
J EOF2
TP SDFPKT+10
J SOL00P
L,S2 A1,SDFPKT+10
S,H1 A1,CRDWD
TE A15,LISTFL
SZ LISTFL
LMJ A3,PRTING
L A0,CRDWD
LMJ X11,SOR
J ERR6
J SOL00P

L,U A0,SDFPKT
LMJ X11,SDFIC
LMJ X11,ESOR
J ERR7
SNZ ASG

L,U A0,FREE
ER CSF\$

LMJ X11,POSTPR
SNZ EFLG
TNZ ERRFLG

J NOERR
L A1,ERRCNT
SZ A0
TLE,U A1,100
J \$+6
DI,U A0,100
A,U A0,'0'
S,S1 A0,ERRPRA+1
SZ A0
J \$+3
TLE,U A1,10
J \$+4
DI,U A0,10
A,U A0,'0'
S,S2 A0,ERRPRA+1
A,U A1,'0'
S,S3 A1,ERRPRA+1
PRINT ERRPRA

.
. GET A CARD
. I/O ERROR
. END OF FILE, WE HAVE ALL OF IT
. GO BACK IF THIS IS A CONTROL I
. GET THE WORD-COUNT INTO A1
. AND INTO THE CONTROL WORD FOR
. TURN LISTFLAG OFF
. UNLESS L-OPTION WAS IN EFFECT
. PRINT THE IMAGE IF OPTION IS R
. OUTPUT THE IMAGE THROUGH SOR
. I/O ERROR
. BACK FOR MORE

.
. CLOSE SDFI
. CLOSE SOR
. I/O OR PFIS ERROR
. INCLUDING THE SCRATCH FILE

. FREE ALL THE FILES WE TOOK
. DID WE HAVE ANY ERRORS ?
. NO
. YES, PRINT OUT THE COUNT
. I HOPE WE DIDN'T HAVE THAT MAN
. GOOD
. NOT SO GOOD
. CONVERT TO FD
. AND PUT IN APPROPRIATE PLACE
. CLEAR AGAIN FOR DIVISION
. PRINT THE TENS IF THERE WERE H
. DO NOT PRINT TENS IF NOT NECES
. SKIP THIS

. ALWAYS PRINT ONES
. LET THE GUY KNOW HOW BAD HE WA
. GOODIE, WE DIDN'T HAVE ANY

ADD GRAPH CARD IF POSSIBLE

CALL SIR TO READ A CARD, PRINT IT, AND
TRANSFER IT TO THE CALLER

ON EOF CLOSE SIR

WRITE CARD TO SCRATCH FILE

L A0,PRO
L A1,0136
TG,U A1,0500
L A0,PRO0
ER PRINTS
L A0,ADDIMG
TZ A0

ER CSFs
ER EXITS

CRDIN* S X11,SAVX11
L,U A1,14
LMJ A3,PRTIMG
GET L A0,CROWD
LMJ X11,GETSRC
J ERR8
J EOF
JN A1,GET
TZ WOPT
S,H1 A3,COUNT+1
L,U R1,14
L X11,SAVX11
L A3,K10
S R1,*1,X11
A,U A3,*0,X11
L A2,BTWD
BT A3,*A2
J 4,X11

EOF LMJ X11,CLOSRC
J ERR9
L X11,SAVX11
J 2,X11

CRDOUT* SZ A3
S X11,SAVX11
L,U A0,*0,X11
S,H2 A0,SDFOPK+8
L A0,*1,X11
S,S2 A0,SDFPKT+10
L,U A0,SDFOPK
LMJ X11,SDF0
J ERR10
L X11,SAVX11
J 3,X11

. AND EXIT, PRINTING
. A SHORT OR LONG MESSAGE,
. DEPENDING ON WHETHER THIS IS
. DEMAND OR NOT
.

. HEAD A CARD FROM SIR
. IMAGE LENGTH FOR PRINT
. PRINT PREVIOUS IMAGE
. GET ONE IMAGE
. FROM SIR
. I/O OR CORRECTION CARD ERR
. END OF FILE RETURN
. GO BACK IF THIS WAS CONTROL IN
. ONLY IF WE HAD W OPTION
. NEW, -XX, OR BLANKS
. IMAGE LENGTH
. RESTORE X11
. CLEAR A3 FOR INDIRECTS
. GIVE WORD COUNT TO CALLING ROU
. GET THE BUFFER AREA INTO A3
. LOAD A2 WITH 1,CBUF
. TRANSFER THE IMAGE TO USER, N1
. EXIT

. CLOSE SOURCE IF THIS WAS AN EO
. I/O OR PFIS ERROR
. AND EXIT
.

. FOR INDIRECTS
. SAVE X11
. PUT BUFFER LOCATION INTO PKT
. MAKE AN IMAGE CONTROL WORD
.

. SDF0 WILL DO THE REST
. I/O ERROR
. EXIT SUCCESSFULLY
.

PRINT AN IMAGE WITH LINE NUMBER

INCREMENT LINE NUMBER

PRINT AN ERROR MESSAGE

PRINT A WARNING

PRINT LINE NUMBER IF LISTING IS OFF

PRTIMG

```

L,S3      A0,PRWD
A,U       A1,0102
S,H1      A1,PRWD
JZ        A0,0,A3
L         A0,COUNT
L,U       A2,12
SSC       A0,12
A,U       A0,1

```

CTLOOP

```

AND,U     A0,077
TNE,U     A1,'A'
A,U       A0,'1'-'A'
TE,U      A1,'9'+1
J         ENDL
A,U       A0,066
SSC       A0,6
A,U       A2,6
J         CTLOOP
LSSC      A0,A2
S         A0,COUNT
L         A0,PRWD
TZ        LISTFL
ER        PRINTS
J         0,A3

```

ENDL

ERR*

```

SNZ       ERR,FLG
L         A0,ERRCNT
A,U       A0,1
S         A0,ERRCNT
DL        A0,MSGERR
J         S+2

```

WARN*

```

DL        A0,MSGWRN
DS        A0,MONTHS
L         A0,PRWD1
TNZ       LISTFL

```

```

ER        PRINTS
L         A3,K10
L         R1,+1,X11
A,U       A3,+0,X11
L         A2,ERRBT
L         A1,R1
A,U       A1,2
S,S3      A1,ERRPRB

```

. SUBROUTINE TO INCREMENT LINE N

```

. PRINT THE LINE IF LISTFL.NE.0
. CONSTRUCT A PRINT CONTROL WORD
. RETURN THE FIRST TIME
. GET THE OLD SEQUENCE NUMBER
. INITIAL SHIFT COUNT
. AND SHIFT IT INTO PLACE
. INCREASE IT BY ONE

```

```

. AND EXAMINE THE LAST CHARACTER
. IF IT IS NOW AN A (WAS BLANK?)
. MAKE A ONE OUT OF IT
. IF IT IS NOT GREATER THAN 9
. WE ARE DONE
. ELSE WE MAKE A ZERO OUT OF IT
. INCREASE THE NEXT HIGHER DIGIT
. ONE, SHIFT AND INCREASE SHIFT
. AND RETURN TO THE START OF THE
. SHIFT BACK INTO PLACE
. STORE IT

```

```

. AND UNLESS LISTFLG IS ZERO
. PRINT THE IMAGE, WITH LINE NUM
. RETURN

```

```

. ENTRY TO PRINT ERROR MESSAGES
. SET FLAG THAT WE HAD ERRORS
. AND COUNT

```

```

. TELL THAT THIS IS AN ERROR

```

```

. ENTRY TO PRINT WARNINGS
. THIS IS NOT SO BAD
. WE DON'T NEED THE MONTHS ANY M

```

```

. TWO BIRDS WITH ONE STRIKE

```

```

. PICK UP REGISTER FOR BT
. THIS ONE IS PRE-FABRICATED

```

I/O ERRORS, PRINT FILE NAME AND STATUS

ERROR FROM ESOR, I/O OF PFIS

ERROR FROM CLOSRC, I/O OF PFIS

	BT	A2,,*A3	. MOVE THE NASTY MESSAGE
	PRINT	ERRPRB	. AND PRINT IT
	J	3,X11	.
			.
LISTOP*	SZ	A3	. SUBROUTINE TO ADJUST LISTING 0
	L	A0,*0,X11	. FOR INDIRECTS,
	S	A0,LISTFL	.
	OR	A0,OHDS	. PUT IT INTO THE FLAG
	S	A1,OHDS	. UPDATE HEADING FLAG
	J	2,X11	.
			. AND EXIT
			.
ERN1			. OPNSRC
ERN2			. SDFOC
ERN4			. SDF10
ERR5			. SDF1
ERR6			. SOR
ERN10			. SDF0
ERR10			. AND ALL OTHERS IF THEY ARE I/O
	DL	A1,,A0	. PUT THE FILENAME (HOPEFULLY)
	DS	A1,PRIO+3	. INTO THE MESSAGE
	LDSL	A4,33	. CONVERT THE STATUS
	LSSL	A4,3	. TO FD
	LDSL	A4,3	.
	A,U	A4,' 00'	.
	S,H2	A4,PRIO+1	.
	L	A0,PRIO	.
ERRPR	ER	PRINTS	. PRINT MESSAGE
	SNZ	EFLG	. LET THEM KNOW THAT WE HAD AN E
	J	ERREX	. AND RETURN TO ERROR EXIT
			.
ERR3			. SSOR
	TNE,U	A2,2	. IS THIS AN I/O ERROR?
	J	ERR10	. YES
	DL	A1,PARTBL+27	. FILE IS NOT PF
	DS	A1,PRUND+4	. FILENAME TO MSG
	L	A0,PRUND	.
	J	ERRPR	. AND PRINT IT
			.
ERR7			. ESOR
	DL	A3,PARTBL+27	.
	J	S+2	.
			.
ERR9			. CLOSRC
	DL	A3,PARTBL+14	. GET THE FILE NAME
	JNZ	A5,ERR10	. JUMP IF I/O ERROR
	DS	A3,PRPF1+3	. PUT FILENAME INTO MESSAGE
	LDSL	A1,33	. CONVERT STATUS TO FD

GETSRC ERROR, I/O ON CORRECTION CARD ERROR

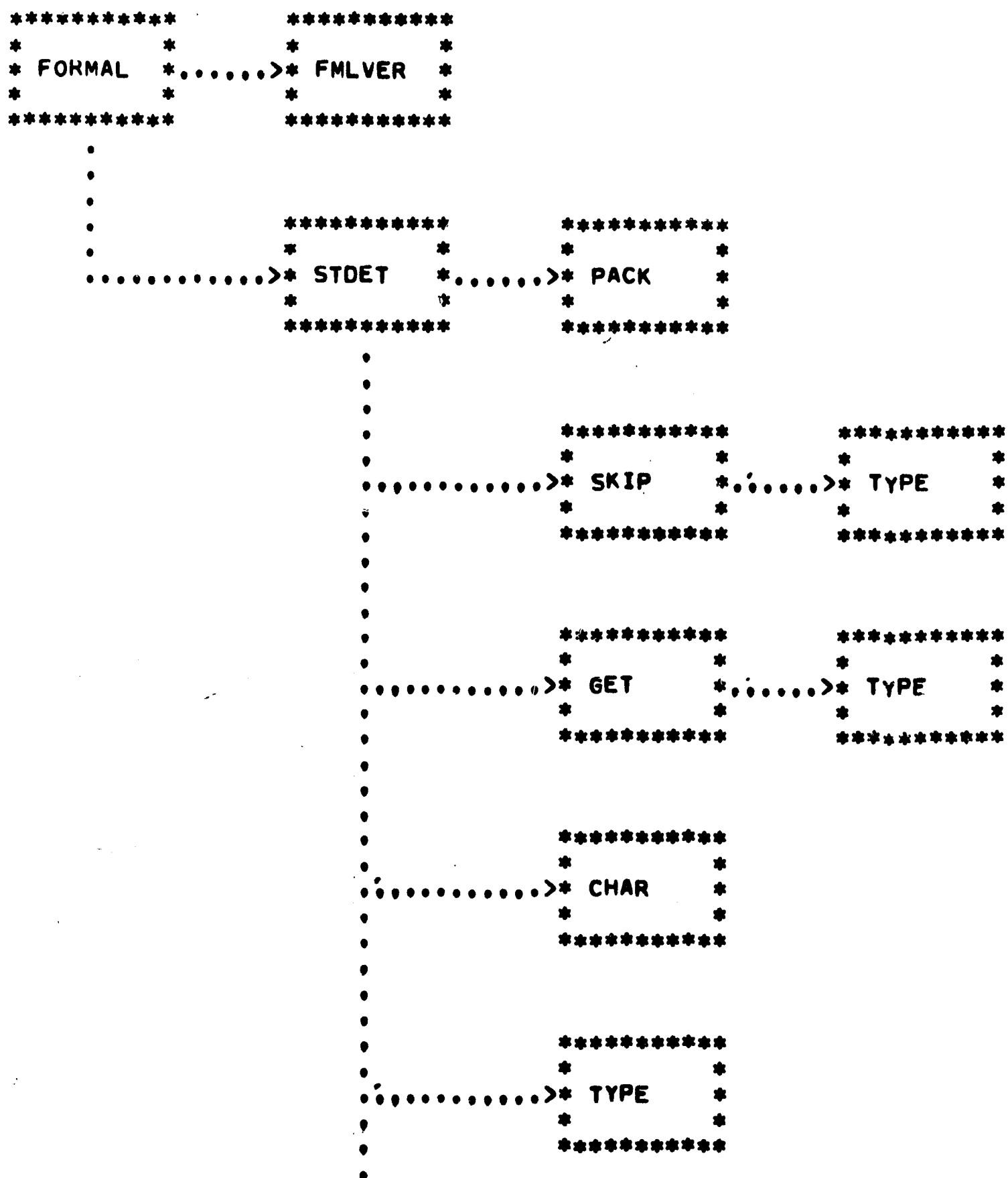
ERRA

LSSL A1,3
 LDSL A1,9
 A,U A1,00 '
 S,H2 A1,PRPFI+1
 L A0,PRPFI
 J ERRPR

JNZ A5,ERRIO
 DL A1,,A0
 DS A1,PRCOR+5
 DL A1,2,A0
 DS A1,PRCOR+7
 PRINT PRCOR
 J GET
 END FORMAL

.
 .
 .
 .
 . GO TO COMMON ERROR PRINT ROUTI
 .
 . GETSRC ERROR
 . GO IF IT IS AN I/O ERROR
 . MOVE THE BAD CORRECTION CARD
 . TO THE ERROR MESSAGE
 .
 .
 . PRINT THE MESSAGE
 . AND GO BACK FOR MORE

THE SUBROUTINE DEPENDENCE FOR THE PREPROCESSOR IS SHOWN IN THE FOLLOWING TABLES:



*
.....>* FORMAL *
*

*
.....>* GIT *.....>* GET *
*

*
.....>* TYPE *
*

*
.....>* FORMAL *
*

*
.....>* NONARI- *.....>* PACK *
* TH *

*
.....>* SKIP *
*

```

*****
* LIST *.....>* STACK *.....>* FORMAL *
* * * * *
*****

```

```

.
.
.
.....>* CHAR *
* *
*****
.
*****
.
.....>* TYPE *
* *
*****
.
*****
.
.....>* FORMAL *
* *
*****

```

```

*****
* TYPE *.....>* CHAR *.....>* PACK *
* * * * *
*****

```

```

.
.
.
.....>* FORMAL *
* *
*****
.
*****
.
.....>* FORMAL *
* *
*****

```


2.2. FORTRAN PROCEDURES

THE ELEMENT HPROCS CONTAINS THREE PROCEDURES:

2.2,1. PARAM

CONTAINS PARAMETER STATEMENTS ASSOCIATING NAMES WITH THE OCTAL VALUES FOR THE CHARACTERS. THESE START WITH CHAR, AND END WITH THE SYMBOL (IN THE CASE OF ALPHABETIC OR NUMERIC CHARACTERS) OR A TWO-LETTER ABBREVIATION FOR SPECIAL CHARACTERS.

2.2,2. BUF

DEFINES THE COMMON BLOCK BUF, WHICH CONTAINS FLAGS AND VARIABLES USED BY MORE THAN ONE PROGRAM.

2.2,3. DEFINE

CONTAINS DEFINE STATEMENTS FOR HALFWORDS AND SIXTHWORDS.

FORMAL,SPROCS

FORTRAN PROCEDURES FOR THE PREPROCESSOR
PARAMETER STATEMENTS FOR:

LETTERS

DIGITS

SPECIAL CHARACTERS

PARAM* PROC

IMPLICIT INTEGER (A-Z)

PARAMETER		CHARA = 06K,	CHARB = 07K,	Q A B
\$ CHARC = 10K,		CHARD = 11K,	CHARE = 12K,	Q C U E
\$ CHARF = 13K,		CHARG = 14K,	CHARH = 15K,	Q F G H
\$ CHARI = 16K,		CHARJ = 17K,	CHARK = 20K,	Q I J K
\$ CHARL = 21K,		CHARM = 22K,	CHARN = 23K,	Q L M N
\$ CHARO = 24K,		CHARP = 25K,	CHARQ = 26K,	Q O P Q
\$ CHARR = 27K,		CHARS = 30K,	CHART = 31K,	Q R S T
\$ CHARU = 32K,		CHARV = 33K,	CHARW = 34K,	Q U V W
\$ CHARX = 35K,		CHARY = 36K,	CHARZ = 37K,	Q X Y Z

PARAMETER		CHAR0 = 60K,	CHAR1 = 61K,	Q 0 1
\$ CHAR2 = 62K,		CHAR3 = 63K,	CHAR4 = 64K,	Q 2 3 4
\$ CHAR5 = 65K,		CHAR6 = 66K,	CHAR7 = 67K,	Q 5 6 7
\$ CHAR8 = 70K,		CHAR9 = 71K,		Q 8 9

PARAMETER		CHARLB = 01K,	CHARRB = 02K,	Q []
\$ CHARNM = 03K,		CHARDE = 04K,	CHARNP = 40K,	Q # A)
\$ CHARM = 41K,		CHARPL = 42K,	CHARLT = 43K,	Q + <
\$ CHAREO = 44K,		CHARGT = 45K,	CHARAM = 46K,	Q > @
\$ CHARD0 = 47K,		CHARAS = 50K,	CHARLP = 51K,	Q \$ * (
\$ CHARPC = 52K,		CHARCL = 53K,	CHAROM = 54K,	Q % : ?
\$ CHAREM = 55K,		CHARCM = 56K,	CHARBS = 57K,	Q ! , \
\$ CHARAP = 72K,		CHARSC = 73K,	CHARSL = 74K,	Q ' /
\$ CHARPD = 75K,		CHARSP = 76K,	CHARST = 77K,	Q ' " #
\$ CHARBL = 05K,		CHARS = 100K,	CHARMS = 00K,	Q

REFERENCES ON

END
BUF* PROC

COMMON BLOCK DEFINITION

NWD TWO WORDS FOR PACKING IDENTIFIERS
 NCHAR CURRENT CHARACTER
 TERMCH CHARACTER TO TERMINATE LIST SUBROUTINE
 STTYPE (COUNT) SET TO TYPE OF NON ARITHMETIC
 STATEMENT OR HOLLERITH COUNT
 FMLFLG SET IF CURRENT STATEMENT IS FORMAL
 HOLFLG SET WHEN SCANNING HOLLERITH FIELD
 DIGIT SET BY GET IF IT FOUND A DIGIT
 EOS SET IF WE REACHED END OF STATEMENT
 ENDPRO SET IF WE REACHED END OF PROGRAM
 SKIPCT SET TO NUMBER OF CHARACTERS TO SKIP ON
 SPECIAL FORTRAN STATEMENTS
 BLANK AS THE NAME SAYS
 MASK MASKS FOR THE FIRST 1,2,3,4,5,6
 CHARACTERS OF A WORD

```
LOGICAL HOLFLG, FMLFLG
LOGICAL DIGIT
LOGICAL EOS, ENDPRO
COMMON /BUF / NWD(2), NCHAR, TERMCH, STTYPE,
+ FMLFLG, HOLFLG, DIGIT, EOS, ENDPRO,
+ SKIPCT, BLANK, MASK(6)
```

```
END
DEFINE* PROC
```

DEFINITIONS OF PARTWORDS

```
DEFINE S (I,J) = FLD (6*(I-1), 6, J) @ SIXTH WORDS = CHARACTERS
DEFINE H1 (I) = FLD (0, 18, I) @ LEFT HALF
DEFINE H2 (I) = FLD (18, 18, I) @ RIGHT HALF
DEFINE S1 (I) = FLD (0,6,I)
DEFINE S2 (I) = FLD (6,6,I)
DEFINE S3 (I) = FLD(12,6,I)
DEFINE S4 (I) = FLD(18,6,I)
DEFINE S5 (I) = FLD(24,6,I)
DEFINE S6 (I) = FLD(30,6,I)
END
```

2.3. UTILITY SUBROUTINES

2.3.1. PACK

ENTRIES TO PACK ARE PACK, UNPACK, MOVE, STRCON, AND ISRCH. FOR A COMPLETE DESCRIPTION SEE 3.1.8

2.3.2. CHAR

THIS ROUTINE IS QUITE COMPLICATED, AS IT COMBINES MANY FUNCTIONS. IT'S PRIMARY PROPOSE IS TO RETRIEVE ONE CHARACTER AT A TIME FROM THE INPUT-IMAGES AND TO COPY IT TO THE OUTPUT IMAGE, GIVING AN ABNORMAL RETURN AT THE LOGICAL END OF EACH STATEMENT. IT AUTOMATICALLY JOINS CONTINUATION-CARDS, AND THROWS OUT IMBEDDED COMMENTS AND COMMENT-CARDS. AS EACH INPUT IMAGE IS EXHAUSTED, THE PROGRAM CALLS CRDIN TO GET A NEW IMAGE, AND AS EACH OUTPUT-IMAGE IS FILLED, IT CALLS CRDOUT TO WRITE IT TO THE SCRATCH FILE.

THE FOLLOWING ENTRIES FULFILL SPECIAL FUNCTIONS:

SAVE

SETS A FLAG TO INHIBIT COPYING INPUT TO OUTPUT, AND AT THE SAME TIME SAVES THE CURRENT LOCATION IN THE STRING.

RESTOR

RESETS THE ABOVE FLAG, AND RESUMES SCANNING WHERE SAVE WAS LAST CALLED.

RESET

ALSO RESETS THE FLAG, BUT DISCARDS THE SAVED INFORMATION, AND RESUMES SCANNING AT THE CURRENT POSITION.

FTRSTK

SWITCHES OUTPUT TO AN INTERNAL BUFFER, WHERE IT WILL BE SAVED TO BE WRITTEN OUT LATER. THIS MAY BE DONE IN LEVELS, WITH THE MAXIMUM GIVEN BY THE PARAMETER LEVMAX (CURRENTLY 20).

FTRUNS

SWITCHES OUTPUT BACK TO THE NEXT LOWER LEVEL, AND KEEPS A POINTER TO THE SAVED STRING.

FTROUT

ADDS THE SEQUENCE ' ', TO THE CURRENT OUTPUT, PUTS OUT THE NUMBER OF ENTRIES SAVED AT THE NEXT HIGHER LEVEL, AND IF THERE ARE ANY, CHAINS THEM TOGETHER, AND ADDS THEM TO THE CURRENT LEVEL. IF THE CURRENT LEVEL IS NOT LEVEL ZERO, THIS SIMPLY MEANS RE-CHAINING, OTHERWISE THE RESULT IS PUT INTO OUTPUT IMAGES.

2-24

WHENEVER THE SAVE-FLAG IS SET, CALLS TO FTRSTK, FTRUNS, AND FTROUT ARE IGNORED.

CHARIN
INITIALIZES VARIABLES IN THE SUBROUTINE FOR EACH STATEMENT

NEW
MOVES THE NUMBER OF CHARACTERS GIVEN BY THE FIRST ARGUMENT FROM THE SECOND ARGUMENT TO THE CURRENT OUTPUT BUFFER.

BACKSP
REMOVES THE LAST CHARACTER FROM THE CURRENT OUTPUT BUFFER

ICRD
CAUSES THE CURRENT CARD-IMAGE TO BE WRITTEN OUT.

IDEC
CONVERTS THE ARGUMENT TO DECIMAL (MAXIMUM 3 DIGITS), AND PACKS IT INTO THE CURRENT OUTPUT BUFFER.

2.3.3. TYPE

THIS ROUTINE IS CALLED TO RETRIEVE A CHARACTER AND BRANCH ON ITS TYPE. IF EOS IS SET, IT SETS NCHAR TO THE END-OF-STATEMENT CHARACTER, AND RETURNS AT THE SPECIAL CHARACTER RETURN. THE ROUTINE THEN CHECKS IF HOLFLG IS SET, INDICATING THAT IT IS TO SCAN A HOLLERITH FIELD. IF HOLFLG IS SET, COUNT INDICATES THE TYPE OF HOLLERITH FIELD SCANNED. FOR A COUNTED HOLLERITH FIELD (NH...) COUNT CONTAINS THE NUMBER OF CHARACTERS, AND FOR AN UNCOUNTED HOLLERITH FIELD (DELIMITED BY QUOTES) IT IS NEGATIVE. THE HOLLERITH FIELD IS SCANNED COMPLETELY, THEN HOLFLG IS SET TO FALSE. IF AN END OF STATEMENT IS ENCOUNTERED BEFORE THE END OF THE HOLLERITH FIELD, AN ERROR MESSAGE IS PRINTED. THEN, CHAR IS CALLED TO RETRIEVE THE NEXT CHARACTER. BLANKS ARE IGNORED, AND THE PROGRAM RETURNS ACCORDING TO THE TYPE OF THE CHARACTER, RETURN 1 FOR ALPHABETIC, RETURN 2 FOR NUMERIC, AND RETURN 3 FOR ANYTHING ELSE. WHEN THE END OF STATEMENT IS REACHED, NCHAR IS SET TO OCTAL 0100, EOS IS SET, AND THE SPECIAL-CHARACTER RETURN IS TAKEN.

FORMAL.CHAR
ROUTINE TO INPUT SOURCE A CHARACTER AT A TIME

... SIZE OF INTERMEDIATE ARRAYS

... MAXIMUM NUMBER OF NESTING BETWEEN FOR AND FML

ENTRY TO GET NEXT CHARACTER
*... BEGIN, CHECK FOR INITIAL CALL
*
*... CHECK IF WE NEED NEW CARD
*
*... RESUME ON CONTINUATION CARD
*
*... INCREMENT POINTER, CHECK IF STILL ON CARD
*
*... GET CHARACTER
*
*... TAKE ANYTHING IF HOLLERITH
*
*... OTHERWISE CHECK FOR IMBEDDED COMMENT
*
*... AND BLANK
*
*... PUT CHARACTER INTO OUTPUT OR SAVE AREA

FUNCTION CHAR (*)
INCLUDE PARAM
INCLUDE BUF
INCLUDE DEFINE

PARAMETER JMAX = 1320
PARAMETER JMAX1 = JMAX + 1

PARAMETER LEVMAX = 20
DIMENSION CARDI (JMAX)
DIMENSION CARD0 (84)
DIMENSION CARDU (JMAX)
EQUIVALENCE (CARDI(JMAX1), CARD0(1))
EQUIVALENCE (CARD0(85), CARDU(1))
DIMENSION CARD(14)
DIMENSION ITAB(LEVMAX), IARG(LEVMAX)
LOGICAL NEWCRD, FULL, INIT
LOGICAL SAVEFL, RESTFL
LOGICAL BLANKF
DEFINE ILEV = ITAB (LEVEL)
DEFINE IFULL = IL .GT. JMAX
DEFINE OFULL = IO .GE. 73
DEFINE UFULL = IU .GT. JMAX
DEFINE SAVE2 = SAVEFL .AND. RESTFL
DATA INIT /.TRUE./
DATA BLANKF /.TRUE./

IF (INIT) GO TO 31

IF (NEWCRD .AND. .NOT. RESTFL) GO TO 33
GO TO 9

8 IF (FULL) GO TO 31
GO TO 10

9 I = I + 1
IF (I .GT. IMAX) GO TO 30

10 CHAR = CARDU (I)

IF (HOLFLG) GO TO 12

IF (CHAR .EQ. 0 .AND. .NOT. RESTFL) GO TO 31

IF (CHAR .EQ. CHARBL) GO TO 13
BLANKF = .FALSE.

```

*
*... WHEN SAVING FORTRAN ARGUMENTS. PUT INTO CARDI
*
*... WRITE OUT CARD WHEN FULL
*
*... AND RETURN
*
*... ONLY RETURN ONE OF ANY GROUP OF BLANKS
*
*... SAVE IN CARDU IF SAVE IS IN EFFECT
*
*... CHECK FOR OVERFLOW OF STACK
*
*... SAVE FORTRAN ARGUMENTS IN STACK
*
*... END OF CARD, GET ANOTHER
*
*... WE ALREADY HAD SOMETHING
*
*... READ NEXT CARD
*
*... IGNORE BLANK CARD
*
*... OR COMMENT CARD
*
*... UNPACK IT
*
*... SET STARTING COLUMN
*
*... GO BACK IF THIS IS CONTINUATION
*

```

```

12  IF (SAVEFL) GO TO 20
    IF (LEVEL .NE. 0) GO TO 22
    IO = IO + 1
    CARDO(IO) = CHAR
11  IF (OFULL) CALL CRD
    CONTINUE
    RETURN
13  IF (FMLFLG .OR. BLANKF) GO TO 9
    BLANKF = .TRUE.
    GO TO 12
20  IF (RESTFL) GO TO 11
    IU = IU + 1
    IF (UFULL) GO TO 899
    CARDU (IU) = CHAR
    GO TO 11
22  IL = IL + 1
    IF (IFULL) GO TO 899
    CARDI(IL) = CHAR
    GO TO 11
30  IF (.NOT. RESTFL) GO TO 31
    IF (.NOT. SAVEFL) IU = 72
    I = ISAV
    IMAX = IMXSAV
    RESTFL = .FALSE.
    IF (NEWCRD) GO TO 35
    GO TO 9
31  CALL CRDIN (CARD, NWDS, 540)
    IF (NWDS .EQ. 0) GO TO 31
    IF (S1(CARD(1)) .EQ. CHARC) GO TO 31
    IMAX = MIN(6*NWDS, 72)
    CALL UNPACK (IMAX, CARD, CARDU)
    I = 7
    IF (INIT) GO TO 34
32  IF (CARDU(6) .NE. CHARO .AND. CARDU(6) .NE. CHARBL) GO TO 8
    FULL = .FALSE.

```



```

*
*
* .... RETURN 1 MEANS END OF STATEMENT
*
* .... WRITE OUT OLD OUTPUT IMAGE
*
* .... RETURN 1 FOR END OF PROGRAM
*
* .... MOVE LABEL TO OUTPUT
*
* .... INITIALIZE FLAGS
*
*
*
*
* .... AND GET CHARACTER
*
* .... EOF RETURN FROM CRDIN MEANS END OF PROG
*
*
* ENTRY TO INITIALIZE THE STACKS FOR EACH STATEMENT
*
* .... ZERO OUT ITAB, IARG, AND LEVEL
*
*
*
*
* ENTRY TO STACK FORTRAN ARGUMENTS IN INTERMEDIATE BUFFER
*
* .... IGNORE IF SAVING
*
* .... CHECK MAXIMUM LEVEL
*
*
*
* .... PUT OUT SEQUENCE NUMBER FOR FORTRAN ARGUMENT
*
*
* .... LINK UP WITH OTHER ARGUMENTS
*
*
* .... CHECK FOR FULL OUTPUT BUFFER
*

```

```

NEWCRD = .TRUE.
35 CONTINUE
RETURN 1
33 CALL CRD
IF (ENDPRO) RETURN 1
34 CALL MOVE (6, CARDU, CARD0)
IO = 6
I = 7
LEVEL = 0
NEWCRD = .FALSE.
IL = 72
IU = 72
INIT = .FALSE.
BLANKF = .FALSE.
GO TO 10
40 ENDPRO = .TRUE.
GO TO 32
ENTRY CHARIN
DO 50 LEVEL = 1, LEVMAX
ITAB(LEVEL) = 0
50 IARG(LEVEL) = 0
LEVEL = 0
RETURN
ENTRY FTRSTK
IF (SAVEFL) RETURN
IF (LEVEL .GE. LEVMAX) GO TO 64
IARG(LEVEL+1) = IARG(LEVEL+1) + 1
II = IO + JMAX
IF (LEVEL .NE. 0) II = IL
CALL DEC (IARG(LEVEL+1))
IF (LEVEL .EQ. 0) GO TO 62
IL = II
H2(ILEV) = IL
GO TO 63
62 IO = II - JMAX

```



```

*
*
*
*... LINK WITH PREVIOUS ARGUMENT, IF ANY
*
*
*
*... WRITE A , AND RETURN
*
*
*... ERROR
*
ENTRY TO RETURN FROM A FORTRAN ARGUMENT TO NEXT LOWER LEVEL
*
*
*... CHECK IF NEXT HIGHER LEVEL IS WRITTEN OUT
*
*
*
ENTRY TO LINK STACKED FORTRAN ARGUMENTS WITH THE
* REGULAR OUTPUT. IF THE CURRENT LEVEL IS ZERO,
* EVERYTHING WILL PUT OUT, OTHERWISE IT WILL BE
* RECHAINED.
*
*
*... TERMINATE WITH '
*
*
*... CONVERT NUMBER OF ARGUMENTS
*
*
*... IS THIS LEVEL ZERO
*
*... NO, RETURN

```

```

63 IF (OFULL) CALL CRD
CONTINUE
LEVEL = LEVEL + 1
IL = IL + 1

IF (ILEV .EQ. 0) GO TO 60
II = H2(ILEV)
H1(CARDI(II)) = IL
GO TO 61
60 H1(ILEV) = IL
61 CONTINUE

CARDI(IL) = CHARCM
IF (IFULL) GO TO 899
RETURN

64 CALL ERR ('NESTING BETWEEN FORMAL AND FORTRAN TOO DEEP',8)
GO TO 61

ENTRY FTRUNS
IF (SAVEFL) RETURN

IF (ITAB(LEVEL+1) .NE. 0) CALL ERR ('ILLEGAL FTRUNS', 3)
H2(ILEV) = IL
LEVEL = LEVEL - 1
IF (LEVEL .EQ. 0) GO TO 65
II = H2(ILEV)
H1(CARDI(II)) = IL + 1
65 CONTINUE
RETURN

ENTRY FTROUT
IF (SAVEFL) RETURN
II = IO + JMAX
IF (LEVEL .NE. 0) II = IL

CARDI (II) = CHARSC
CARDI (II+1) = CHARAP
CARDI (II+2) = CHARCM
II = II + 2

CALL DEC (IARG(LEVEL+1))
IF (IARG(LEVEL+1) .NE. 0) II = II + 1
IARG(LEVEL+1) = 0

IF (LEVEL .EQ. 0) GO TO 70

```

```

*
*
*
*
*... LEVEL ZERO, WRITE OUT THE CHAIN
*
*
*
*... FILL ONE IMAGE AT A TIME
*
*... CHECK IF END OF CHAIN
*... CHECK FOR FORWARD LINK
*
*
*... FINISHED, RESET POINTERS AND RETURN
*
*
* ENTRY TO SAVE IMAGE FOR RE-SCANNING
*
*
* ENTRY TO IGNORE SAVED INFORMATION AND CONTINUE SCAN
*
*
* ENTRY TO RE-SCAN SAVED STRING
*
*... RESET POINTERS
*
*

```

```

IL = II
IF (IFULL) GO TO 899
H1(CARDI(IL)) = H1(ITAB(LEVEL+1))
II = H2(ITAB(LEVEL+1))
H1(CARDI(II)) = IL + 1
ITAB(LEVEL+1) = 0
RETURN

70  IJ = H1(ITAB(1))
    IO = II - JMAX
    IF (OFULL) CALL CRD
    II = IO
    IF (IJ .EQ. 0) GO TO 73

71  DO 72 II = IO+1,73
    CARDO(II) = CARDI(IJ)

    IF (IJ .EQ. H2(ITAB(1))) GO TO 73

    JJ = H1(CARDI(IJ))
    IJ = IJ + 1
    IF (JJ .NE. 0) IJ = JJ

72  CONTINUE
    IO = 73
    CALL CRD
    GO TO 71

73  IO = II
    IL = 72
    ITAB(1) = 0
    IF (OFULL) CALL CRD
    RETURN

ENTRY SAVE
SAVEFL = .TRUE.
IF (RESTFL) ISAV2 = 1
RETURN

ENTRY RESET
SAVEFL = .FALSE.
IF (.NOT. RESTFL) IO = 72
ISAV2 = 0
RETURN

ENTRY RESTOR

IF (SAVE2) GO TO 790
ISAV = I
IMXSAV = IMAX
IMAX = IO

```

[illegible]

```

790  SAVEFL = .FALSE.
      I = 72
      IF (ISAV2 .NE. 0) I = ISAV2
      ISAV2 = 0

      RESTFL = .TRUE.
      EOS = .FALSE.
      RETURN

      ENTRY NEW (N,C)
      IF (LEVEL .NE. 0) GO TO 800
      CALL UNPACK (N, C, CARD0(IO+1))
      IO = IO + N
      IF (OFULL) CALL CRD
      GO TO 801
800   CALL UNPACK (N, C, CARDI(IL+1))
      IL = IL + N
      IF (IFULL) GO TO 899
801   CONTINUE
      RETURN

      ENTRY BACKSP
      IF (LEVEL .NE. 0) IL = IL - 1
      IF (LEVEL .EQ. 0) IO = IO - 1
      RETURN

899   FULL = .TRUE.
      CALL ERR ('INTERMEDIATE STACK FULL',4)
      GO TO 31

      ENTRY ICRD
      CALL CRD
      IO = 6
      RETURN

      ENTRY IDEC (NNNN)
      II = IO + JMAX
      CALL DEC (NNNN)
      IO = II - JMAX
      RETURN

      SUBROUTINE CRD

      IF (IO .LT. 72) CALL STRCON (12, BLANK, CARU)

      CALL PACK (IO, CARD0, CARD)
      CALL CRDOUT (CARD, 12)
      IF (IO .LE. 72) GO TO 900

      CALL STRCON (5, BLANK, CARD0)

```

*... AND STORE CONTINUATION CHARACTER

*... MOVE ANY LEFT OVER TEXT

*... AND RESTORE POINTER

SUBROUTINE TO PUT DECIMAL NUMBER INTO OUTPUT IMAGE

*... CONVERT

*... IS FIRST DIGIT NEEDED

*... IS SECOND DIGIT NEEDED

*... ALWAYS PUT ONE DIGIT

*... END

CARD0(6) = CHARPL

CALL MOVE (IO-72, CARD0(73), CARD0(7))

900 IO = IO - 66
CONTINUE
RETURN
END

SUBROUTINE DEC (N1)

N2 = N1
N30 = N2/100
N2 = N2 - 100*N30
N31 = N2/10
N2 = N2 - 10*N31

IF (N30 .EQ. 0) GO TO 921
II = II + 1
CARDI(II) = N30 + CHAR0

921 IF (N30 + N31 .EQ. 0) GO TO 922
II = II + 1
CARDI(II) = N31 + CHAR0

922 II = II + 1
CARDI(II) = N2 + CHAR0
RETURN
END

END

FORMAL TYPE

GETS ONE CHARACTER FROM CHAR AND RETURNS ACCORDING TO ITS TYPE

DO WE HAVE END OF STATEMENT

.... NO, TEST FOR HOLLERITH FIELD

.... YES, COUNTED HOLLERITH ?

.... NO, SEARCH FOR A QUOTE NOT FOLLOWED
BY ANOTHER ONE.

.... END OF STATEMENT WITHIN HOLLERITH

.... END OF STATEMENT AFTER QUOTE

.... YES, SKIP ACCORDING TO COUNT

FINISHED WITH HOLLERITH FIELD, OR
NO HOLLERITH

.... GET ONE CHARACTER

.... IGNORE BLANKS

.... RETURN ACCORDING TO TYPE
--- RETURN 1 FOR ALPHABETIC

--- RETURN 2 FOR NUMERIC

--- RETURN 3 FOR ANYTHING ELSE

.... END OF STATEMENT, SET FLAG

.... SET END OF STATEMENT CHARACTER

AND RETURN

SUBROUTINE TYPE (*, *, *)
INCLUDE PARAM
INCLUDE BUF
INCLUDE DEFINE
EQUIVALENCE (COUNT, SITYPE)

IF (EOS) GO TO 900

IF (.NOT. HOLFLG) GO TO 200

IF (COUNT .GE. 0) GO TO 150

100 NCHAR = CHAR (\$101)
IF (NCHAR .NE. CHARAP) GO TO 100
NCHAR = CHAR (\$102)
IF (NCHAR .EQ. CHARAP) GO TO 100
HOLFLG = .FALSE.
GO TO 800

101 CALL ERR ('ILLEGAL END OF HOLLERITH', 4)

102 HOLFLG = .FALSE.
GO TO 899150 DO 151 I = 1, COUNT
151 NCHAR = CHAR (\$101)
HOLFLG = .FALSE.

200 NCHAR = CHAR (\$899)

IF (NCHAR .EQ. CHARBL) GO TO 200

800 IF (NCHAR .GE. CHARA .AND. NCHAR .LE. CHARZ) RETURN 1
IF (NCHAR .GE. CHARO .AND. NCHAR .LE. CHAR9) RETURN 2

801 RETURN 3

899 EOS = .TRUE.

900 NCHAR = CHARES

GO TO 801
END

Q GET NEXT CHAR, RETURN 1 FOR

Q DO WE HAVE AN END OF STATEMENT?

Q GO IF WE DON'T HAVE HOLLERITH

Q GO FOR COUNTED HOLLERITH

Q UNCOUNTED HOLLERITH

Q SKIP UNTIL QUOTE

Q GET ONE MORE

Q GO BACK IF IT'S DOUBLE QUOTE

Q CLEAR HOLLERITH FLAG

Q AND BRANCH ON TYPE OF NEXT

Q CLEAR HOLLERITH FLAG

Q GET NUMBER OF CHARACTERS

Q CLEAR HOLLERITH FLAG

Q GET NEXT CHARACTER

Q REJECT BLANKS

Q LETTER

Q DIGIT

Q SPECIAL

Q SET END OF STATEMENT FLAG

Q SET END OF STATEMENT CHARACTER

Q AND RETURN

2.4. SCANNING SUBROUTINES

2.4.1. GET

GETS N CHARACTERS (FIRST ARGUMENT, UP TO 12), AND PACKS THEM INTO NWD(1), NWD(2), LEFT JUSTIFIED, SPACE FILLED. RETURN 2 IF FIRST CHARACTER IS NOT ALPHABETIC. IF A DIGIT IS ENCOUNTERED IN THE SCAN, THE LOGICAL VARIABLE DIGIT IS SET, RETURN 3 IF A SPECIAL CHARACTER IS FOUND BEFORE THE COUNT IS EXHAUSTED.

2.4.2. GIT

GIT SCANS AN ITEM OF THE FORM '<VARIABLE>,' , '<VARIABLE>)' , '<VARIABLE><END OF STATEMENT>' , OR '<VARIABLE>(<SUBSCRIPTS>),' . IT PRINTS ERROR MESSAGES IF THE FIRST CHARACTER IS NOT ALPHABETIC, IF AN ILLEGAL DELIMITER IS FOUND, OR IF THE END OF STATEMENT IS REACHED WITHIN A NEST OF PARENTHESES.

GIT1 DOES THE SAME AS GIT, EXCEPT IS WILL ONLY SCAN '<VARIABLE>(' AND NOT THE SUBSCRIPT EXPRESSION.

GIT2 DOES THE SAME AS GIT, BUT WILL ALSO SCAN AN INTEGER CONSTANT.

2.4.3. SKIP

SCANS N CHARACTERS (AS GIVEN BY FIRST ARGUMENT) IF THEY ARE ALPHANUMERIC, AND RETURNS. IT WILL RETURN 2 IF IT GETS A SPECIAL CHARACTER.

FORMAL GET ROUTINE TO BUILD IDENTIFIER

THIS ROUTINE GETS N CHARACTER (UP TO 12), AND PACKS THEM INTO NWD(1), NWD(2), LEFT JUSTIFIED, SPACE FILLED, IF THE FIRST CHARACTER IS NOT ALPHABETIC, RETURN 2 IF ANY CHARACTER IS NUMERIC, DIGIT IS SET TO TRUE, IF A SPECIAL CHARACTER IS ENCOUNTERED, THE SCAN IS TERMINATED, AND RETURN 3 IS TAKEN

BLANK OUT THE WORDS AND INITIALIZE DIGIT

CHECK IF FIRST CHAR IS ALPHABETIC

NO, RETURN 2

YES, GET MORE

PACK PREVIOUS CHARACTER

AND GET ANOTHER

CHARACTER IS NUMERIC

CHARACTER IS ALPHABETIC

COUNT EXHAUSTED, NORMAL RETURN

SPECIAL CHARACTER RETURN

SUBROUTINE GET (N, **, *)

Q GET N CHARACTERS, RETURN 2 IF

INCLUDE BUF
INCLUDE DEFINE
IMPLICIT INTEGER (A-Z)

NWD (1) = BLANK
NWD (2) = NWD (1)
DIGIT = .FALSE.
J = N

CALL TYPE (S1, S2, S2)

2 RETURN 2

1 DO 4 K = 1,2
DO 4 I = 1,6

S(I, NWD(K)) = NCHAR
J = J - 1
IF (J .EQ. 0) GO TO 7

CALL TYPE (S4, S3, S8)

3 DIGIT = .TRUE.

4 CONTINUE

7 RETURN

8 RETURN 3
END

FORMAL.GIT
ROUTINE TO OBTAIN ITEMS FROM A LIST

GET NAME WITH UP TO 6 CHARACTERS

IGNORE ANY MORE ALPHANUMERICS

TEST SPECIAL CHARACTER

.... , RETURN

.... EOS RETURN

....) RETURN

.... NOT (: ILLEGAL CHARACTER

RETURN FOR GIT1
SKIP (...) FOR GIT

COUNT PARENTHESES

SEARCH FOR NEXT DELIMITER

.... EOS, SHOULD NOT OCCUR

.... (INCREMENT COUNTER

.... NOT), IGNORE

DECREMENT COUNTER, GO BACK IF NON-ZERO

IF ZERO, GET NEXT CHARACTER,

SHOULD NOT BE ALPHANUMERIC

SAME AS GET, BUT STOP SCAN AT (

SCAN ANYTHING ALPHANUMERIC, THEN CHECK
DELIMITERS AS FOR GIT

FIRST CHARACTER IS BAD

SUBROUTINE GIT
INCLUDE PARAM
INCLUDE BUF
INCLUDE DEFINE
LOGICAL SWITCH
SWITCH = ,FALSE.

Q GET ONE ITEM FROM LIST

Q SWITCH FOR PASSING BY PARENTHESES

11 CALL GET (6, \$2, \$7)

6 CALL TYPE (\$6,\$6,\$7)

7 IF (NCHAR ,EQ. CHARCM) GO TO 10 Q RETURN

IF (NCHAR ,EQ. CHARES) GO TO 10 Q RETURN

IF (NCHAR ,EQ. CHARRP) GO TO 10 Q RETURN

IF (NCHAR ,NE. CHARLP) GO TO 3

IF (SWITCH) GO TO 10

Q RETURN IF NOT TO PASS (...)

PARS = 1

8 CALL TYPE (\$8,\$8,\$9)

9 IF (NCHAR ,EQ. CHARES) GO TO 4

IF (NCHAR ,EQ. CHARLP) PARS = PARS + 1

IF (NCHAR ,NE. CHARRP) GO TO 8

PARS = PARS - 1

IF (PARS) 8,,8 .

CALL TYPE (\$3, \$3, \$10)

ENTRY GIT1
SWITCH = ,TRUE.
GO TO 11

Q GET ONE ITEM, DO NOT PASS (

ENTRY GIT2
SWITCH = ,FALSE.
CALL TYPE (\$12, \$12, \$7)

Q GET NUMERIC ITEM ALSO

RETURN

BAD DELIMITER, CHECK FOR ' IF FORMAL-STATEMENT

EOS IN PARENTHESES NEST.

```

2 CALL ERR ('LIST STARTS BADLY', 3)
10 RETURN
3 IF (.NOT.FMLFLG.OR.NCHAR.NE.CHARAP) CALL ERR ('BAD CONSTRUCTION',3)
  GO TO 10
4 CALL ERR ('STATEMENT ENDS BADLY', 4)
  GO TO 10
  END

```

FORMAL.SKIP :

SKIPS CHARACTERS IN THE SCAN

SKIP N CHARACTERS IF THEY ARE ALPHANUMERIC

ALL WERE ALPHANUMERIC, RETURN

SPECIAL CHARACTERS FOUND, RETURN 2

SUBROUTINE SKIP (N,S)

```

IF (N .EQ. 0) GO TO 3
DO 1 I=1,N
  CALL TYPE (S1,S1,S2)
1 CONTINUE

```

3 RETURN

```

2 RETURN 2
  END

```

0 SKIP N CHARACTERS IF THEY ARE

0 NONE TO SKIP

2.5. STATEMENT TRANSLATION

2.5.1. STDET

THIS PROGRAM ANALYZES ALL STATEMENTS AND DETERMINES THEIR TYPE. FORTRAN AS WELL AS FORMAL STATEMENTS ARE DIVIDED INTO TWO GROUPS, LONG ONES (WITH MORE THAN SIX CHARACTERS IN THE NAME) AND SHORT ONES. THIS SEPARATION FACILITATES THE STATEMENT DETERMINATION: STATEMENTS STARTING WITH SEVEN OR MORE ALPHABETIC CHARACTERS BEFORE THE FIRST DELIMITER COULD BE LONG OR SHORT STATEMENTS, STATEMENTS STARTING WITH SEVEN OR MORE CHARACTERS, BUT INCLUDING DIGITS, MUST BE SHORT STATEMENTS, AND STATEMENTS STARTING WITH SIX OR LESS ALPHANUMERIC CHARACTERS MUST BE SHORT STATEMENTS OR ASSIGN STATEMENTS.

FOR FORTRAN STATEMENTS THE SUBROUTINE NONARITH RETURNS AN INDEX TO INDICATE WHAT ACTION STDET SHOULD TAKE. IT CAN

1. COMPLETELY IGNORE A STATEMENT (THAT IS TRANSFER IT TO THE OUTPUT EXACTLY AS IT IS)
2. SCAN A FORTRAN LIST (WHICH MAY INCLUDE FORMAL FUNCTIONS)
3. CHECK TYPE DECLARATIONS WHETHER THEY ARE OF THE FORM <TYPE> FUNCTION, IN WHICH CASE THEY ARE TREATED AS 4.
4. SCAN ENTRY, SUBROUTINE, OR FUNCTION STATEMENTS TO FIND ALL FORMAL ARGUMENTS, REPLACE THEM BY AN ASTERISK, AND PUT OUT SEPARATE CALLS TO FMLEQU FOR EACH ONE.
5. SCAN LOGICAL IF STATEMENTS. AT THE END OF THE LOGICAL EXPRESSION THE PROGRAM RETURNS TO THE BEGINNING TO SCAN FOR THE EXPRESSION FOLLOWING THE IF, WHICH COULD BE ANYTHING, INCLUDING A FORMAL STATEMENT. THIS EXPRESSION IS THEN TRANSLATED SEPARATELY.
6. SCAN SPECIAL STATEMENTS. AT THE TIME THESE INCLUDE START EDIT SOURCE AND STOP EDIT SOURCE STATEMENTS.

FOR FORMAL STATEMENTS THE APPROPRIATE SUBROUTINE CALL IS SUBSTITUTED, THEN LIST PROCESSES THE FORMAL EXPRESSION. FORMAL EXPRESSIONS MAY INCLUDE FORTRAN ARGUMENTS, WHICH WILL BE COLLECTED AND PUT OUT AT THE END OF THE CALLING SEQUENCE.

2.5.2. NONARITH

NONARITH HAS TWO ENTRIES, STLONG AND STSHRT. STLONG IS CALLED FROM STDET TO DETERMINE THE TYPE OF LONG FORTRAN STATEMENTS (7 OR MORE CHARACTERS IN THE NAME). IF THE STATEMENT (AS GIVEN IN NWD1) IS NOT FOUND IN THE TABLE LONG1, THE PROGRAM GOES ON TO STSHRT. IF IT FINDS THE STATEMENT IN THE LIST, IT SKIPS CHARACTERS ACCORDING TO THE TABLE LONG2, AND TESTS THE LAST ONE FOR A MATCH. IT THEN SETS SKIPCT AND STTYPE, AND RETURNS IF SKIPCT IS NOT ZERO. IF SKIPCT IS ZERO, INDICATING A STATEMENT WHICH IS AMBIGUOUS WITHIN THE FIRST 7 CHARACTERS, IT GETS 5 MORE CHARACTERS INTO NWD2

AND SEARCHES THE TABLE FOLLOWING THE FIRST MATCH. IF THE STATEMENT IS FOUND, THE REMAINING TESTS ARE REPEATED. IF ANY OF THE ABOVE TEST FAILS, THE PROGRAM TRANSFERS TO STSHRT. AT STSHRT A TABLE OF SHORT FORTRAN STATEMENTS (LESS THAN 7 CHARACTERS) IS SEARCHED. NWD1 IS MASKED AND COMPARED WITH THE TABLE. IF A MATCH IS FOUND, STTYPE IS SET, AND THE NORMAL RETURN IS TAKEN. IF NO MATCH IS FOUND, RETURN 1 IS TAKEN.

2.5.3. FUNTST

THIS SUBROUTINE IS CALLED BY STDET AFTER A TYPE DECLARATION TO TEST FOR A STATEMENT OF THE FORM <TYPE> FUNCTION IF THAT IS SO, RETURN 1 IS TAKEN, OTHERWISE THE PROGRAM RETURNS NORMALLY.

2.5.4. LIST

THIS PROGRAM SCANS STANDARD FORTRAN AND FORMAL EXPRESSIONS. THE INITIAL MODE IS GIVEN BY THE ENTRY POINT: LIST SCANS FORTRAN EXPRESSIONS, FMLST SCANS FORMAL EXPRESSIONS, AND FMLST1 SCANS A FORMAL EXPRESSION UP TO THE FIRST UNMATCHED RIGHT PARENTHESIS. FOR THE ENTRY LIST THE COMMON VARIABLE TERMCH DETERMINES WHICH CHARACTER WILL TERMINATE THE SCAN, AND FOR THE ENTRY FMLST THE SCAN WILL BE TERMINATED AT THE END OF STATEMENT.

THIS PROGRAM TAKES CARE OF REFORMATTING FORTRAN ARGUMENTS IN FORMAL EXPRESSIONS AND DOES A PRELIMINARY SYNTAX CHECK TO SEE IF PARENTHESES AND BRACKETS ARE MATCHED, IF FORTRAN ARGUMENTS ARE FORMED PROPERLY, AND IF FORMAL STATEMENTS TERMINATE CORRECTLY. THE PROGRAM USES THE SUBROUTINES STACK AND CHAR TO STACK OPERATORS AND TO MANIPULATE FORTRAN ARGUMENTS.

2.5.5. STACK

THIS ROUTINE KEEPS A PUSHDOWN STACK OF OPERATORS, FOR THE PURPOSE OF CHECKING BALANCING BRACKETS AND PARENTHESES, AND NESTING BETWEEN FORTRAN AND FORMAL. THE SIZE OF THE PUSHDOWN STACK IS GIVEN BY THE PARAMETER STDIM (CURRENTLY 200). THE STACK ENTRIES CONSIST OF A CHARACTER IN THE LOWER HALF, AND A FLAG IN THE UPPER HALF OF EACH WORD. THE FLAG CAN BE ZERO FOR NORMAL ENTRIES, OR NEGATIVE FOR ENTRIES WHICH WILL CAUSE A SPECIAL RETURN WHEN THEY ARE UNSTACKED. THE POINTER IND POINTS TO THE FIRST FREE WORD IN THE STACK. THERE ARE THREE ENTRIES:

STCK PUTS A CHARACTER INTO THE STACK AS NORMAL ENTRY. STACKING A ZERO WILL CAUSE THE STACK TO BE CLEARED, AND A WARNING WILL BE PRINTED IF THE STACK WAS NOT EMPTY.

STCKTM PUTS A CHARACTER INTO THE STACK AS SPECIAL ENTRY.

UNSTCK REMOVES AN ENTRY FROM THE STACK. THE FOLLOWING RETURNS ARE POSSIBLE:

RETURN 3 IF LAST ENTRY DOES NOT MATCH.

RETURN 2 IF LAST ENTRY MATCHES, AND HAS SPECIAL FLAG

NORMAL RETURN IF LAST ENTRY MATCHES, AND HAS FLAG = 0

FORMAL,STDET

SCANS ALL FORTRAN AND FORMAL STATEMENTS, DETERMINES THEIR
TYPE, AND DOES THE APPROPRIATE PROCESSING

LIST OF SPECIAL FORMAL STATEMENTS

MAXIMUM NUMBER OF FORMAL ARGUMENTS ON SUBROUTINE ETC.

START PROCESSING EACH STATEMENT

RESUME HERE AFTER FIRST PART OF IF

GET STATEMENT NAME

.... POSSIBLE LONG FORTHAN STATEMENT

.... SPECIAL FORTRAN STATEMENT
SKIPCT IS LENGTH IN CHARACTERS
STTYPE IS INDEX FOR TYPE OF STATEMENT

.... SKIP OVER STATEMENT

.... BRANCH ON STTYPE, GO TO
.... 9999 FOR COMPLETELY IGNORABLE
.... 401 FOR SCANNING FORTRAN EXPRESSIONS
.... 306 FOR TYPE STATEMENTS
.... 309 FOR ENTRY, SUBROUTINE, FUNCTION
.... 316 FOR IF STATEMENTS
.... 323 FOR START EUIT SOURCE

SUBROUTINE STDET

INCLUDE PARAM

INCLUDE BUF

INCLUDE DEFINE

EQUIVALENCE (NWD(1),NWD1), (NWD(2),NWD2)

PARAMETER FMSTLL = 3

Q NUMBER OF LONG FML STATEMENTS

PARAMETER FMSTSL = 9

Q NUMBER OF SHORT FML STMTS

DIMENSION FMSTL(FMSTLL), FMSTS(FMSTSL)

DATA FMSTL /'DUMPALDUMPSYDUMPEX'/

DATA FMSTS /'ERASQREADQWRITQPRINWWPUNCQOROLOGQSAVEQRESEQQ',

+ 'OPTIQQ' /

DIMENSION SKIPS (FMSTSL)

DATA SKIPS / 5, 4, 5, 5, 6, 4, 5, 6 /

DATA BLANK, MASK / ' ', 0770000000000, 0777700000000,

+ 0777777000000, 0777777770000, 0777777777700, 0777777777777/

DIMENSION TAB(8)

DATA TAB /48HOPT(' IO1(' IO2(' IO4(' ERS(' OUT(' SAV(' RES(' /

LOGICAL TWOFLG

PARAMETER EQU MAX = 50

DIMENSION EQUTAB(EQU MAX), EQUTBB(EQU MAX)

1 TWOFLG = .FALSE.

EQU PTR = 0

CALL CHARIN

10 CALL SAVE
FMLFLG = .FALSE.

CALL GET (7, \$2, \$6)

IF (DIGIT) GO TO 200

CALL STLONG (\$9000)

300 CALL RESTOR

CALL SKIP (SKIPCT, 0)

```

      .... 324 FOR STOP EDIT SOURCE
      .... SHORT FORTRAN STATEMENTS
      .... FIRST CHARACTER IS DELIMITER
      .... IF ' THIS IS FORMAL STATEMENT
      .... ERROR, ILLEGAL CHARACTER
      .... RETURN FOR END OF PROGRAM
      .... OTHERWISE GO WRAP UP STATEMENT
      .... DELIMITER FOUND, COULD BE SHORT OR ARITHMETIC
      .... = MEANS ARITHMETIC OR DO STATEMENT
      .... PRELIMINARY CHECK FOR POSSIBLE SHORT STATEMENT
      .... SCAN (...)
      .... CHECK NEXT CHARACTER
      .... = MEANS ARITHMETIC
      .... OTHERWISE MUST BE SHORT

FORTRAN STATEMENTS
*... TYPE STATEMENTS
*
* .... IS IT <TYPE> FUNCTION
*
* .... NO, IGNORE IT
*
*... FUNCTION, SUBROUTINE, ENTRY
*
* .... FIND START OF ARGUMENT LIST
*
* .... COUNT ARGUMENTS
*
* .... GET NEXT CHARACTER
*
* .... IF IT IS NOT ' SKIP THIS ARGUMENT

```

GO TO (9999, 401, 306, 309, 316, 323, 324), STTYPE

200 CALL STSHRT (\$9000)
GO TO 300

2 IF (NCHAR .EQ. CHARAP) GO TO 500
IF (.NOT. (EOS.OR.TWOFLG)) CALL ERR ('ILLEGAL START OF STATEMENT',5)
IF (ENDPRO) RETURN
CALL RESTOR
GO TO 9999

6 IF (NCHAR .EQ. CHAREQ) GO TO 400
IF (NCHAR .NE. CHARLP) GO TO 200
CALL STSHRT (\$400)
TERMCH = CHARRP
CALL LIST
CALL TYPE (\$200, \$200, \$7)

7 IF (NCHAR .EQ. CHAREQ) GO TO 400
GO TO 200

306 CONTINUE
CALL FUNTST (\$309)
GO TO 9999
Q LOGICAL
Q RETURN TO 309 IF FUNCTION

309 CALL TYPE (\$309,\$309,\$3090)
3090 IF (EOS) GO TO 9999
FMLFLG = .TRUE.
I = 0
Q PASS BY FUNCTION NAME
Q SUBROUTINE, ENTRY, FUNCTION

3091 I = I + 1
CALL SAVE
CALL TYPE (\$3094, \$3094, \$3092)

```
* . . . . GET ITEM FROM LIST
* . . . .
* . . . . AND PUT IT INTO THE TABLE
* . . . .
* . . . . GET NEXT DELIMITER
* . . . .
* . . . . PUT OUT * FOR ARGUMENT
* . . . . IF TERMINATED BY , GO BACK FOR NEXT
* . . . .
* . . . . COMPLAIN IF NOT END OF STATEMENT
* . . . .
* . . . . PUT OUT ) AND FINISH UP STATEMENT
* . . . .
* . . . . SKIP OVER ARGUMENT
* . . . .
* . . . . ERROR, TOO MANY ARGUMENTS
* . . . .
*... IF STATEMENT
* .
* .
* . . . . IS IT TERMINATED BY (
* .
* . . . . SCAN (...)
* .
* . . . . SET FLAG FOR PART 2 OF STATEMENT
* .
* . . . . AND GO BACK TO GET IT
*... START EDIT SOURCE, SET OPTION
*
*... STOP EDIT SOURCE, SET OPTION
*
*... ARITHMETIC OR DO STATEMENT
* .
* . . . . SCAN UNTIL EOS
```

```

3092 IF (NCHAR ,NE. CHARAP) GO TO 3094
      CALL GIT
      EQUPTX = EQUPTX + 1
      IF (EQUPTX ,GT. EQUMAX) GO TO 3099

      EQUTAB (EQUPTX) = NWD1
      EQUTBB (EQUPTX) = I

30920 CALL TYPE ($30920, $30920, $3093)
3093  CALL RESET

      CALL NEW (2, 2H*, )

3095  IF (NCHAR ,EQ. CHARCM) GO TO 3091

      IF (NCHAR ,NE. CHARRP) CALL WARN ('STATEMENT SHOULD END WITH )',5)
      CALL BACKSP

      CALL NEW (1, 1H) )
      GO TO 9999

3094  CALL RESTOR
      CALL GIT
      GO TO 3095

3099  CALL ERR ('TOO MANY FORMAL PARAMETERS', 5)
      GO TO 9999

316   CONTINUE                                @ IF
      CALL TYPE ($401,$401, $3161)

3161  IF (NCHAR ,NE. CHARLP) GO TO 401

      TERMCH = CHARRP
      CALL LIST

      TWOFLG = ,TRUE.

      GO TO 10

323   CALL LISTOP (2)                          @ START EDIT SOURCE
      GO TO 9999                               @ RETURN

324   CALL LISTOP (3)                          @ STOP EDIT SOURCE
      GO TO 9999                               @ RETURN

400   CALL RESTOR                             @ RIGHT SIDE OF ARITHMETIC OR UO

401   TERMCH = CHARES

```

FORMAL STATEMENTS

```

*
* WE ARE POSITIONED AFTER THE
* INITIALIZE FLAGS AND PUT OUT FIRST PART OF CALL
*
*... CHECK FOR LONG NAME
*
*... YES, SEARCH TABLE
*
*... CHECK TABLE OF SHORT STATEMENTS
*
*.... FOUND, SKIP NUMBER OF CHARACTERS
*
*
*... BRANCH ON STATEMENT
*
*... FIRST CHARACTER BAD
*
*... DID NOT GET 7 CHARACTERS, CHECK FOR
* SHORT AR ASSIGN STATEMENT
*
*.... = MEANS ASSIGN
*
*.... NOT ( MEANS SHORT
*
*.... IF THIS IS SECOND TRY USE SPECIAL SCAN
*
*.... SCAN OVER (...), AND COUNT
* COMMAS IN LEVEL 1
*
*
*.... IF THIS CONTAINS FORTRAN ARGUMENTS, USE LIST
* FOR SCAN
*
*

```

```

CALL LIST
GO TO 9999                                @ RETURN

500  FMLFLG = .TRUE.
     TWOFLG = .FALSE.
     CALL NEW (8,8HCALL FML)

     CALL RESET
     CALL SAVE

501  CALL GET (7, $507, $504)

     I = ISRCH (FMSTLL,FMSTL,NWD1,$506)
     GO TO 505

506  I = FMSTLL + ISRCH (FMSTSL,FMSTS,AND(NWD1,MASK(4)),$507)

     CALL RESTOR
     CALL SAVE
     SKIPCT = SKIPS(I-FMSTLL)
     CALL SKIP (SKIPCT, 0)

505  GO TO (602, 603, 604, 606, 607, 608, 609, 610, 611,612,613,601), I

507  CALL ERR ('FORMAT STATEMENT CANNOT BE RECOGNIZED', 7)
     GO TO 9999

504  S1(NWD2) = NCHAR

     IF (NCHAR .EQ. CHAREQ) GO TO 510
     IF (NCHAR .NE. CHARLP) GO TO 506
     IF (TWOFLG) GO TO 5045

     FPARS = 1
     COMMAS = 0
5040 CALL TYPE ($5040, $5040, $5041)
5041 IF (NCHAR .EQ. CHARLP) FPARS = FPARS + 1
     IF (NCHAR .EQ. CHARRP) FPARS = FPARS - 1
     IF (NCHAR .EQ. CHARCM .AND. FPARS .EQ. 1) COMMAS = COMMAS + 1
     IF (NCHAR .EQ. CHARES) GO TO 801

     IF (NCHAR .EQ. CHARNM) GO TO 5044
     IF (NCHAR .EQ. CHARAP) GO TO 506
     IF (FPARS) 5040,,5040

```



```

*.... FINISHED, GET NEXT CHARACTER
*   IF NOT DELIMITER, SPECIAL STATEMENT
*.... RESTORE THINGS IF WE USED SPECIAL SCAN
*.... = MEANS ASSIGN
*.... INITIALIZE SPECIAL SCAN, SAVE NAME
*.... CALL FMLST1 TO SCAN (...)
*.... THEN CHECK WHAT FOLLOWS
*.... ASSIGN STATEMENT
*   SCAN ALL OF IT
*.... OPTION STATEMENT
*.... DUMP ALL ... STATEMENT, CHECK SPEC
*.... SYMBOLS: 21
*.... EXPRESSIONS: 22
*.... PUT IT INTO OUTPUT IMAGE
*.... DUMP SYMBOLS
*..... DUMP EXPRESSIONS
*.... ERASE STATEMENT

```

```

5046 CALL TYPE ($506, $506, $5043)
5043 IF (TWOFLG) NWD1 = SAVNW1
    IF (TWOFLG) NWD2 = SAVNW2
    IF (NCHAR .NE. CHAREQ) GO TO 506
    GO TO 510
5044 TWOFLG = .TRUE.
    SAVNW1 = NWD1
    SAVNW2 = NWD2
    CALL RESTOR
    CALL SAVE
    GO TO 501
5045 CALL FMLST1
    GO TO 5046
510 CALL NEW (6,6HASE ())
    CALL RESTOR
    GO TO 700
601 I = 1
    GO TO 690
602 CALL TYPE ($6020, $6021, $6021) @ DUMP ALL ...
6021 CALL ERR ('BAD CONSTRUCTION', 3)
    GO TO 9999
6020 I = '21' @ SYMBOLS
    IF (NCHAR .EQ. CHARE) I = '22' @ EXPRESSIONS
6022 CALL NEW (5,5HOMP ())
    CALL NEW (2,I)
    CALL NEW (1,1H))
    GO TO 9999
603 I = '11' @ DUMP SYMBOL TABLE
    GO TO 6022
604 I = '12' @ DUMP EXPRESSION TABLE
    GO TO 6022
606 CONTINUE @ ERASE
    I = 5

```


GO TO 690

```

*... I/O STATEMENTS
*   I01  READ
*   I02  PRINT
*   I03  PRINT (LEFT MARGIN, LINE LENGTH)
*   I04  PUNCH
*   I05  READ (UNIT)
*   I06  WRITE (UNIT)
*   I07  WRITE (UNIT, LM, LL)

```

```

*... READ
*
*... CHECK FOR READ (UNIT)
*
*... SCAN UNIT
*
*... THEN SCAN ARGUMENT LIST

```

```

*... WRITE
*
*... CHECK IF ANY SPECIFICATIONS
*... YES, I07
*... SCAN ARGUMENTS

```

```

*... PRINT
*
*... CHECK FOR PRINT (LM, LL)
*
*... SCAN 2 PARAMETERS

```

```

607  CONTINUE                                @ READ
      I = 2

      IF (S5(NWD1) .NE. CHARBL .OR. S1(NWD2) .NE. CHARLP) GO TO 690
      I = 'I05('
      CALL SKIP1

6076  CALL NEW (6,I)
      CALL RESET
      CALL GIT2
      IF (NCHAR .NE. CHARRP) CALL ERR ('BAD I/O LIST', 2)
      CALL BACKSP

      CALL NEW (2,2H,'')
      GO TO 700

608  CONTINUE                                @ WHITE
      CALL SKIP1
      CALL RESET
      I = 'I06('

      IF (COMMAS .EQ. 0) GO TO 6076

      CALL NEW (5,5H'I07( )
      CALL GIT2

      GO TO 6090

609  CONTINUE                                @ PRINT (LM,LL,CC)
      I = 3

      IF (S6(NWD1) .NE. CHARBL .OR. S1(NWD2) .NE. CHARLP) GO TO 690
      CALL NEW (5,5H'I03 ( )
      CALL SKIP1
      CALL RESET

6090  DO 6091 I = 1,2
      CALL GIT2
      IF (NCHAR .NE. CHARCM) GO TO 6092

6091  CONTINUE

```

```

* .... CHECK END OF LIST
*
*
*
* .... AND SCAN LIST OF ARGUMENTS
*... PUNCH STATEMENT
*
*
*... ROLOUT STATEMENT
*
*
*... SAVE STATEMENT
*
*
*... RESET STATEMENT
*
* .... PUT OUT STATEMENT NAME
*
*
* .... THEN SCAN SPECIFICATION AS HOLLERITH FIELD
*
*
*... PUT OUT NAME OF SUBROUTINE CALL
*
*
*... SCAN LIST
*
*... TERMINATE IT BY )
*
*... SCAN UNTIL EOS
*
*
*... WERE THERE ANY FORMAL ARGUMENTS
*
*
* .... PUT OUT PREVIOUS
*

```

```

6092 IF (NCHAR .NE. CHARRP .OR. I .NE. 2)
+ CALL ERR ('BAD LIST FOR WRITE OR PRINT', 5)
CALL BACKSP
CALL NEW (3,3H,' )
GO TO 700
610 CONTINUE                                @ PUNCH
SKIPCT = 5
I = 4
GO TO 690
611 CONTINUE                                @ ROLOUT
I = 6
GO TO 690
612 CONTINUE                                @ SAVE
I = 7
GO TO 680
613 CONTINUE                                @ RESET
I = 8
680 CALL RESET
CALL NEW (5, TAB(I))
STTYPE = -1
HOLFLG = .TRUE.
681 CALL SKIP1
CALL BACKSP
CALL NEW (3, 3H'') )
GO TO 9999
690 CONTINUE
CALL RESET
CALL NEW (5, TAB(I))
700 CALL FMLST
701 CALL NEW (1, '') )
9999 CALL TYPE ($9999, $9999, $99990)
99990 IF (.NOT. EOS) GO TO 9999
EOS = .FALSE.
IF (EQUPTTR .EQ. 0) GO TO 1
DO 99991 I = 1, EQUPTTR
CALL ICRO

```

```

* .... PUT OUT CALL TO FMLEQU
*
*
*
*
*
*
* .... RETURN FOR NEXT STATEMENT
*
* .... CANNOT RECOGNIZE STATEMENT
*
*
* .... SYNTAX ERROR

```

SUBROUTINE TO SKIP ONE CHARACTER,
REGARDLESS OF ITS TYPE

```

CALL NEW (12, 'CALL FMLEQU(' )
CALL IDEC (EQUTBB(I))
CALL NEW (2, 2H, ' )
CALL NEW (6, EQUTAB(I))
CALL NEW (2, 2H, ' )
99991 CONTINUE

GO TO 1

9000 CALL WARN ('UNRECOGNIZABLE STATEMENT', 4)
CALL RESTOR
GO TO 9999

801 CALL WARN ('FORMAL EXPRESSION SHOULD END WITH ''' , 6)
GO TO 506

SUBROUTINE SKIP1
CALL TYPE ($9898, $9898, $9898)
9898 RETURN
END
END

```

8 SKIP ONE CHAR, REGARDLESS

FORMAL, NONARITH
CHECKS FOR SPECIFIC FORTRAN STATEMENTS

NUMBER OF STATEMENTS IN EACH GROUP

LIST OF LONG FORTRAN STATEMENTS

LONG1 CONTAINS THE NAMES

LONG2 CONTAINS:

S3 = LENGTH OF STATEMENT IN CHARACTERS,
ZERO FOR THOSE AMBIGUOUS WITHIN THE
FIRST SEVEN CHARACTERS.

S4 = SKIP COUNT

S5 = LAST CHARACTER OF STATEMENT

S6 = INDEX FOR STATEMENT TYPE

LIST OF SHORT FORTRAN STATEMENTS

SHORT1 CONTAINS THE NAMES

SHORT2 CONTAINS

SUBROUTINE STLONG (*)

INCLUDE PARAM

INCLUDE BUF

INCLUDE DEFINE

Q FIND LONG OR SHORT STATEMENTS

PARAMETER LONGST = 27

PARAMETER SHORTS = 23

DIMENSION LONG1(LONGST), LONG2(LONGST)

DIMENSION SHORT1(SHORTS), SHORT2(SHORTS)

DATA LONG1(1), LONG2(1) /	'ABNORM', 010012101 /
DATA LONG1(2), LONG2(2) /	'INCLUD', 007001201 /
DATA LONG1(3), LONG2(3) /	'DIMENS', 011022301 /
DATA LONG1(4), LONG2(4) /	'INTEGE', 007002703 /
DATA LONG1(5), LONG2(5) /	'COMPLE', 007003503 /
DATA LONG1(6), LONG2(6) /	'DOUBLE', 017102303 /
DATA LONG1(7), LONG2(7) /	'LOGICA', 007002103 /
DATA LONG1(8), LONG2(8) /	'EQUIVA', 013041201 /
DATA LONG1(9), LONG2(9) /	'IMPLIC', 010013101 /
DATA LONG1(10), LONG2(10) /	'NAMELI', 010013101 /
DATA LONG1(11), LONG2(11) /	'PARAME', 011022701 /
DATA LONG1(12), LONG2(12) /	'EXTERN', 010012101 /
DATA LONG1(13), LONG2(13) /	'FUNCTI', 010012304 /
DATA LONG1(14), LONG2(14) /	'SUBROU', 012031204 /
DATA LONG1(15), LONG2(15) /	'BLOCKU', 011020601 /
DATA LONG1(16), LONG2(16) /	'CONTIN', 010011201 /
DATA LONG1(17), LONG2(17) /	'ENDFIL', 007001202 /
DATA LONG1(18), LONG2(18) /	'BACKSP', 011021202 /
DATA LONG1(19), LONG2(19) /	'DRUMFI', 010011202 /
DATA LONG1(20), LONG2(20) /	'STARTE', 000001100 /
DATA LONG1(21), LONG2(21) /	'ITSOU ', 017031206 /
DATA LONG1(22), LONG2(22) /	'ITCOD ', 015011201 /
DATA LONG1(23), LONG2(23) /	'STOPED', 000001600 /
DATA LONG1(24), LONG2(24) /	'TSOUR ', 016021207 /
DATA LONG1(25), LONG2(25) /	'TCODE ', 014001201 /
DATA LONG1(26), LONG2(26) /	'REQUES', 007003102 /
DATA LONG1(27), LONG2(27) /	'REFERE', 012033001 /

S5 = LENGTH IN CHARACTERS
S6 = INDEX FOR STATEMENT TYPE

SEARCH LONG TABLE

..... FOUND, SKIP UNTIL END OF STATEMENT
..... CHECK LAST CHARACTER
..... SET SKIPCT (TOTAL LENGTH)
..... AND RETURN IF IT IS NOT ZERO
..... IF IT IS AMBIGUOUS, GET 5 MORE CHARACTERS
..... AND SEARCH AGAIN
..... IF FOUND, REPEAT ALL TESTS

ENTRY TO SEARCH TABLE OF SHORT FORTRAN
STATEMENTS (UNDER 7 CHARACTERS). ALSO
COMES HERE IF SEARCH FOR LONG
STATEMENT WAS UNSUCCESSFUL

DO A MASKED SEARCH ON THE SHORT LIST

DATA SHORT1(1), SHORT2(1) / 'ASSIGN', 00602 /
DATA SHORT1(2), SHORT2(2) / 'COMMON', 00601 /
DATA SHORT1(3), SHORT2(3) / 'DECODE', 00602 /
DATA SHORT1(4), SHORT2(4) / 'DEFINE', 00602 /
DATA SHORT1(5), SHORT2(5) / 'DELETE', 00601 /
DATA SHORT1(6), SHORT2(6) / 'ENCODE', 00602 /
DATA SHORT1(7), SHORT2(7) / 'FORMAT', 00601 /
DATA SHORT1(8), SHORT2(8) / 'RETURN', 00602 /
DATA SHORT1(9), SHORT2(9) / 'REWIND', 00602 /
DATA SHORT1(10), SHORT2(10) / 'ENTRYW', 00504 /
DATA SHORT1(11), SHORT2(11) / 'PAUSEW', 00501 /
DATA SHORT1(12), SHORT2(12) / 'PRINTW', 00502 /
DATA SHORT1(13), SHORT2(13) / 'PUNCHW', 00502 /
DATA SHORT1(14), SHORT2(14) / 'WRITEW', 00502 /
DATA SHORT1(15), SHORT2(15) / 'CALLW', 00402 /
DATA SHORT1(16), SHORT2(16) / 'DATAW', 00401 /
DATA SHORT1(17), SHORT2(17) / 'GOTOW', 00402 /
DATA SHORT1(18), SHORT2(18) / 'READW', 00402 /
DATA SHORT1(19), SHORT2(19) / 'REALW', 00403 /
DATA SHORT1(20), SHORT2(20) / 'STOPW', 00401 /
DATA SHORT1(21), SHORT2(21) / 'ENDW', 00301 /
DATA SHORT1(22), SHORT2(22) / 'DOBW', 00202 /
DATA SHORT1(23), SHORT2(23) / 'IFBW', 00205 /

I = ISRCH (LONGST, LONG1, NWD(1), S30)

10 CALL SKIP (S4(LONG2(I)), S30)

IF (NCHAR .NE. S5(LONG2(I))) GO TO 30

SKIPCT = S3(LONG2(I))

IF (SKIPCT .NE. 0) GO TO 33

DO 20 L = 1,5
CALL TYPE (S20, S30, S30)
S(L,NWD(2)) = NCHAR

20

I = I + ISRCH (3, LONG1(I+1), NWD(2), S30)

GO TO 10

ENTRY STSHRT (*)

8 RECOGNIZE SHORT FORTRAN STATEME

30 DO 31 I = 1,SHORTS

•
•
•
.... NOT FOUND, RETURN 1
•
.... FOUND, SET STTYPE AND RETURN

SKIPCT = S5(SHORT2(I))
IF (AND(MASK(SKIPCT), NWD(1)) ,EQ, SHORT1(I)) GO TO 33
31 CONTINUE

RETURN 1

33 STTYPE = S6(SHORT2(I))
RETURN
END

FORMAL.FUNTST FUNCTION TEST ROUTINE

THIS SUBROUTINE IS CALLED AFTER A TYPE DECLARATION TO CHECK
IF WE ARE LOOKING AT A STATEMENT OF THE FORM
<TYPE> FUNCTION ...
IF THIS IS SO, THE CHARACTER POINTER IS LEFT AFTER 'FUNCTION'
AND RETURN 1 IS TAKEN.
OTHERWISE, THE POINTER IS LEFT WHERE IT WAS, AND THE
NORMAL RETURN IS TAKEN

SAVE CHARACTER POINTER

IS FIRST CHARACTER = 'F'

SKIP THE NEXT SEVEN IF THEY ARE ALPHABETIC

AND CHECK THE LAST ONE FOR 'N'

IT LOOKS LIKE 'FUNCTION', RESTORE CHARACTER POINTER

AND SKIP OVER THE WORD

NOT 'FUNCTION', RESTORE CHARACTER POINTER AND RETURN

SUBROUTINE FUNTST (*)

0 RETURN 1 IF FUNCTION AFTER TYPE

INCLUDE PARAM
INCLUDE BUF

CALL SAVE
CALL TYPE (\$1,\$2,\$2)

1 IF (NCHAR .NE. CHARF) GO TO 2 0 FIRST CHARACTER = 'F'

CALL SKIP (7,\$2)

IF (NCHAR .NE. CHARN) GO TO 2 0 LAST CHARACTER = 'N'

CALL RESTOR

CALL SKIP (8,0)
RETURN 1

2 CALL RESTOR
RETURN
END

.... NO MORE THAN 6 CHARACTERS
 .
 CHECK IF (FOLLOWS
 .
 STACK IT
 .
 NEXT SHOULD BE '
 .
 SEARCH TABLE
 .
 .
 NOT FOUND, RETURN
 .
 FOUND, STACK / FOR FORTRAN TO
 FORMAL
 .
 AND GO SCAN FORMAL

QUOTE, SET COUNT TO NEGATIVE

.
 .
 COUNTED HOLLERITH, SET HOLFLG
 RESUME SCAN

ERROR

RETURN FROM FORTRAN TO FORMAL

NO FORMAL, THIS SHOULD BE END OF STACK

ERROR, STATEMENT WAS FORMED BADLY

ENTRY TO SKIP OVER (...) IN FORMAL STATEMENTS
TO DETERMINE STATEMENT TYPE.

STACK) AS TERMINATOR

FORMAL STATEMENT SCANNER,

INITIALIZE

CALL TYPE (\$8,\$8,\$101)
 101 IF (NCHAR .NE. CHARLP) GO TO 15
 CALL STCK (CHARRP)
 CALL TYPE (\$10,\$11,\$102)
 102 IF (NCHAR .NE. CHARAP) GO TO 15
 DO 1 I = 1,FMFUNL @ CHECK IF FORMAL FUNCTION
 IF (NWD(2) .EQ. FMFUNT(I)) GO TO 103
 1 CONTINUE
 GO TO 15
 103 CALL STCK (CHARSL)
 GO TO 700
 25 CONTINUE
 COUNT = -1 @ UNCOUNTED HOLLERITH
 20 HOLFLG = .TRUE. @ COUNTED HOLLERITH
 GO TO 9
 300 CALL ERR ('STATEMENT ENDS BADLY', 4)
 400 CALL UNSTCK (CHARBS, 0, \$401)
 CALL FTRUNS @ GO BACK TO HIGHER LEVEL
 GO TO 700
 401 CALL UNSTCK (0, 0, \$500)
 GO TO 501
 500 CALL ERR('BRACKETS/PARENS IN THIS STATEMENT DON'T BALANCE',8)
 501 RETURN
 ENTRY FMLST1 @ SPECIAL ENTRY FOR
 CALL STCK (0)
 CALL STCKTM (CHARRP)
 GO TO 700
 ENTRY FMLST @ PROCESS A FORMAL STATEMENT
 CALL STCK (0)

700 CALL TYPE (\$700, \$700, \$701)

```

701  IF (NCHAR .EQ. CHARLP) CALL STCK (CHARRP)
      IF (NCHAR .EQ. CHARLB) CALL STCK (CHARRB)

```

```

IF (NCHAR .EQ. CHARNM) GO TO 706
IF (NCHAR .EQ. CHARRP) CALL UNSTCK (NCHAR, $8000, $500)
IF (NCHAR .EQ. CHARRB) CALL UNSTCK (NCHAR, 0, $500)

IF (NCHAR .EQ. CHARAP) GO TO 800

IF (NCHAR .EQ. CHARES) GO TO 801
GO TO 700

```

706 CONTINUE @ FORTRAN EXPRESSION IN FORMAL

CALL TYPE (\$7601, \$7609, \$7609)
CALL FIRSTK

```

7601  CALL TYPE ($7601, $7609, $7609)
      CALL FTRSTK

      CALL STCK (CHARBS)
      CALL TYPE ($7609, $7609, $7602)
7602  IF (NCHAR .NE. CHARLP) GO TO 7609

```

TERMCH = CHARRP

GO TO 2

```

7609 CALL ERR ('FORTRAN ARGUMENT FORMED BADLY', 5)
      GO TO 400

```

```

800  CONTINUE                                Q END OF FORMAL EXPRESSION
      CALL FTROUT                            Q OUTPUT : ,N,LIST)

```

CALL UNSTCK (CHARSL, 0, \$8000)

CALL TYPE (\$8003, \$8003, \$8002) @ THE NEXT CHARACTER SHOULD BE 7
8002 IF (NCHAR, NE, CHARRP) GO TO 8003
CALL UNSTCK (NCHAR, 0, \$500)

GO TO 9

8003 CALL WARN ('THIS FORMAL EXPRESSION SHOULD END WITH A '), 7)

CALL NEW (1, 1H))
GO TO 9

```

8000  CALL UNSTCK (0, 0, $500)
      RETURN

```

TRY TO CORRECT IT

```
801 CALL WARN ('THIS FORMAL EXPRESSION SHOULD END WITH A '''' 7)
    CALL NEW (1, 1H' )
    GO TO 800
    END
```

FORMAL STACK
STACKS FORTRAN AND FORMAL PARENTHESES, BRACKETS, ETC.
FOR PRELIMINARY SYNTAX SCAN

SET TOP FOR NO TERMINATING CHARACTER

CHAR=0 MEANS INITIALIZE

IF NOT EMPTY PRINT WARNING AND CLEAR STACK

CHECK FOR STACK OVERFLOW

STACK CHARACTER AND INDICATOR

INCREMENT STACK POINTER

AND RETURN

STACK A TERMINATING CHARACTER
SET TOP TO NEGATIVE, THEN TREAT LIKE STACK.

```
SUBROUTINE STCK (CHR)
  REFERENCES ON
  IMPLICIT INTEGER (A-Z)
  PARAMETER STDIM = 200
  DIMENSION ISTACK (STDIM)
  INCLUDE DEFINE
  DATA IND / 1 /
```

TOP = 0

IF (CHR .NE. 0 .OR. IND .EQ. 1) GO TO 10

CALL WARN ('PREVIOUS STATEMENT BADLY FORMED', 6)

IND = 1

10 CONTINUE

IF (IND .GT. STDIM) GO TO 99

ISTACK(IND) = CHR
H1(ISTACK(IND)) = TOP

IND = IND + 1

RETURN

ENTRY STCKTM (CHR)

ERROR, STACK IS FULL

UNSTACK A SYMBOL
 RETURN 2 IF SYMBOL WAS TERMINATING CHARACTER
 RETURN 3 IF SYMBOL IS NOT ON TOP OF STACK

DECREMENT STACK POINTER

TEST SYMBOL

RIGHT SYMBOL, CHECK FOR END OF STACK

END OF STACK, RESET IND

RETURN 3 IF STACK WAS NOT CLEAR

NO, RETURN 2 FOR TERMINATING CHARACTER

RETURN

TOP = -1
 GO TO 10

99 CALL ERR ('OPERATOR STACK FULL', 4)
 RETURN

ENTRY UNSTCK (CHR, *, *)

I = IND - 1

IF (H2(ISTACK(I)) .NE. CHR) RETURN 3
 IND = I

IF (CHR .NE. 0) GO TO 31

IND = 1

IF (I .NE. 1) RETURN 3

31 IF (ISTACK(IND) .LT. 0) RETURN 2

RETURN
 END

3. OBJECT TIME ROUTINES

THE OBJECT TIME ROUTINES OF THE FORMAL SYSTEM WERE DESIGNED SUCH THAT THEY PERFORM THE NECESSARY OPERATIONS TO EXECUTE THE FORMAL STATEMENTS IN A FORTRAN PROGRAM, AND AT THE SAME TIME, THEY CAN BE USED INDEPENDENTLY IN THE INTERACTIVE FORMAL WITH THE HELP OF THE INTERACTIVE MAIN PROGRAM, THIS CHAPTER DESCRIBES ALL OBJECT TIME ROUTINES IN DETAIL.

THE SUBDIVISION OF THIS CHAPTER FOLLOWS THE LOGICAL STRUCTURE OF THE OBJECT TIME ROUTINES, THE SYNTAX AND SEMANTICS OF THE FORMAL STATEMENTS WERE DESCRIBED IN TECHNICAL REPORT 70-133. ONE MAY REGARD THE SYMBOLIC EXPRESSIONS IN THE FORMAL STATEMENTS AS EXTERNAL, INFIX FORM, INTERNALLY, THESE EXPRESSIONS ARE REPRESENTED IN PREFIX FORM. THE DIFFERENCES OF THESE TWO FORMS DEFINES ROUGHLY THE LAYOUT OF THIS CHAPTER.

SECTION 1 DESCRIBES THE INTERNAL STORAGE OF THE OBJECT TIME FORMAL SYSTEM TOGETHER WITH ITS UTILITY ROUTINES. THE MOST IMPORTANT SUBSECTIONS ARE 1 AND 7. SUBSECTION 1 DESCRIBES THE CORE STORAGE IN WHICH THE PAIR OF ARRAYS, C AND D, HOLD THE OBJECT TIME SYMBOL TABLE AND THE SYMBOLIC EXPRESSIONS IN THE INTERNAL PREFIX FORM. THE TREE STRUCTURE OF THE SYMBOL TABLE IS DESCRIBED IN SUBSECTION 7 TOGETHER WITH ITS UTILITY ROUTINES. THE INTERNAL PREFIX FORM OF THE SYMBOLIC EXPRESSIONS WAS DESCRIBED IN TECHNICAL REPORT 70-133. IN THAT REPORT, TABLE 1, IN APPENDIX A SUMMARIZED THE INTERNAL CODES USED FOR THE LINEARLY STORED EXPRESSIONS IN PREFIX FORM. THIS TABLE IS REPRINTED IN SECTION 2 OF THIS CHAPTER.

SECTION 2 DESCRIBES THE ROUTINES ASSOCIATED WITH THE TRANSLATION OF THE SYMBOLIC EXPRESSIONS FROM THEIR EXTERNAL, INFIX FORM TO THEIR INTERNAL, PREFIX FORM. THIS TRANSLATION PROCESS IS A BASIC INFIX TO POSTFIX TRANSLATION USING AN OPERATOR AND OUTPUT STACK. THE TRANSLATION PROCESS IS EXTENDED TO PROVIDE AUTOMATIC SIMPLIFICATION AND THE EXECUTION OF SPECIAL FORMAL FUNCTIONS DURING THE TIME OF THE TRANSLATION. BECAUSE OF THE RECURSIVE NATURE OF THE SIMPLIFICATION, THE PROCESS CAN BE IMPLEMENTED BY A PUSH DOWN STACK. THE PUSH DOWN STACK FOR THE SIMPLIFICATION COINCIDES WITH THE OPERATOR STACK OF THE TRANSLATION. SUBSECTION 1 DESCRIBES THE BASIC TRANSLATION PROCESS, AND SUBSECTIONS 2 TO 7 THE SIMPLIFICATION PROCESS AND THE EXECUTION OF SPECIAL FORMAL FUNCTIONS. SUBSECTION 2, STOUTO, ALSO PERFORMS THE POSTFIX TO PREFIX TRANSFORMATION.

SECTION 3 DESCRIBES THE ROUTINES ASSOCIATED WITH THE TRANSLATION OF THE SYMBOLIC EXPRESSIONS FROM THEIR INTERNAL, PREFIX FORM TO THEIR EXTERNAL, INFIX FORM. THESE ROUTINES ARE CALLED BY FORMAL OUTPUT STATEMENTS.

SECTION 4 DESCRIBES THE ROUTINES WHICH HANDLE THE ACTUAL FORMAL STATEMENTS. THESE ROUTINES ARE CALLED DIRECTLY BY THE CALL STATEMENTS GENERATED BY THE PREPROCESSOR, OR BY THE INTERACTIVE MAIN PROGRAM.

SECTION 5 DESCRIBES THE INTERACTIVE MAIN PROGRAM WHICH RECOGNIZES THE TYPE OF STATEMENT, AND CALLS THE APPROPRIATE STATEMENT ROUTINE.

3.1. DATA STRUCTURE AND BASIC UTILITY ROUTINES

THE MAIN STORAGE AREA OF THE OBJECT TIME ROUTINES ARE THE LABELLED COMMONS IN CORE. IN ADDITION, $2*35*PLNSIZ$ SECTORS OF DRUM STORAGE ARE RESERVED AS AN AUXILIARY STORAGE. THE DRUM STORAGE IS USED ONLY FOR SYMBOLIC EXPRESSIONS WHEN THE CORE STORAGE IS FULL OR WHEN THE USER DIRECTLY REQUESTS IT THROUGH THE ROLOUT STATEMENT; BOTH THE CORE AND THE DRUM STORAGE AREAS CAN BE SAVED IN A USER DEFINED FILE AND RELOADED AGAIN BY USING THE SAVE AND RESET STATEMENTS.

3.1.1. CORE STORAGE

THERE ARE 3 LABELLED COMMON AREAS RESERVED: FMLCM1, FMLCM2, AND FMLCM3. FMLCM1 CONTAINS VARIOUS POINTERS, FMLCM2 AND FMLCM3 ARE RESERVED FOR THE SYMBOL TABLE AND SYMBOLIC EXPRESSIONS. BOTH, FMLCM2 AND FMLCM3, CONTAIN ONE ARRAY (C AND D, RESPECTIVELY) OF THE SAME LENGTH DEFINED BY THE PARAMETER CDIM (PRESENTLY DEFINED AS 2048). THE LAST 15 BITS OF EACH WORD IN THE C-ARRAY (FMLCM2) ARE USED FOR LINKAGE TO ACHIEVE DYNAMIC STORAGE ALLOCATION.

THE FOLLOWING FORTRAN PROCEDURES DESCRIBE THE LAYOUT OF THE COMMON BLOCKS TOGETHER WITH FIELD DEFINITIONS USED IN THE C-D ARRAYS. (THE ASSEMBLY COMMON BLOCK ELEMENT IS LISTED ALSO.)

FORMAL, APROCS

THIS PROCEDURE DEFINES THE COMMON BLOCK AREAS TO BE USED
BY EACH FORMAL OBJECT TIME SUBROUTINES.

CMMN PROC

IMPLICIT INTEGER(A-Z)
PARAMETER CDIM = 2048
PARAMETER PLNSIZ = 100

'CDIM' DEFINES THE SIZE OF THE C-D STORAGE AREA
'PLNSIZ' DEFINES THE SIZE OF THE ROLLOUT MAP

THERE ARE THREE COMMON BLOCKS DEFINED.
THEY ARE AS FOLLOWS ::

*--- COMMON BLOCK 'FMLCM1'

* IT DEFINES THE FREE CHAIN POINTERS, SYMBOL TABLE
* POINTER AND THE LOGICAL SWITCHES
* NXNXNX NEXT LOCATION IN THE FREE CHAIN
* ILILIL LAST LOCATION IN THE FREE CHAIN
* :
* NS FIRST ENTRY IN SUBPROGRAM TABLE
* : (EMPTY=0) I.E.
* : THE BEGINING OF THE SYMBOL TABLE
* XQTOPT OPTION WORD FROM THE QXQT STATEMENT
* PRODEX SWITCH FOR EXPANSION OF POWERS
* :
* INTGSW EVALUATE INTEGER VALUED FUNCTIONS
* MATHSW EVALUATE MATHEMATICAL FUNCTIONS
* EXPDSW USE DISTRIBUTIVE LAW
* POWER EXPAND SUMS RAISED TO POSITIVE
* : INTEGERS
* BASE 0, 1, 2, 3 . . . BASE0, BASE2,
* : BASE10, BASEE
* :
* SIMPSW SWITCH USED BY STOUT ROUTINES
* BITSW SWITCH USED BY STOUT ROUTINES
* IOUNIT UNIT FOR FORMAL I/O STATEMENTS
* :
* NSB POINTER TO SUBROUTINE NAME
* NSY POINTER TO N1
* NSY1 POINTER PRECEDING N1
* NSU POINTER TO N2-N3
* NSU1 POINTER PRECEDING N2-N3
* N1 VARIABLE NAME
* N2 NUMBER OF SUBSCRIPTS
* N3 SUBSCRIPT WORD
* IY 1 FOR NONSUBSCRIPTED, 3 FOR
* : SUBSCRIPTED
* :
* FTRARG NUMBER OF FORTHAN ARGUMENTS
* DEFARG NUMBER OF ARGUMENTS IN A DEFINED
* : FUNCTION
* DEFFUN 1 IF EXPRESSION IS A DEFINED

COMMON/ FMLCM1/ NXNXNX, ILILIL,

+ NS, XQTOPT, PRODEX,

+ INTGSW, MATHSW, EXPDSW, POWER, BASE,

+ SIMPSW, BITSW, IOUNIT,

+ NSB, NSY, NSY1, NSU, NSU1, N1, N2, N3, IY,


```

*      .      FUNCTION , 0 ELSE
*      .... NK      START OF ARGUMENT CHAIN FOR READ,
*      .      WRITE, ERASE, ..
*      .... CBUF      I/O BUFFER
*      .      :
*      .... NP      PUSHDOWN STACK POINTER
*      .... EPTR      POINTER TO START OF EXPRESSION
*      .      :
*      .... TEMPORARY FLAGS
*      .      XFLG1 AND XFLG2 ARE USED BY OUTPUT ROUTINES
*      .      :
*      .... ROLOUT AND BITPLN ARE USED BY THE ROLLOUT
*      .      ROUTINE
*      .      :
*----- COMMON BLOCK 'FMLCM2'
*      IT DEFINES THE C STORAGE AREA
*      .
*----- COMMON BLOCK 'FMLCM3'
*      IT DEFINES THE D STORAGE AREA

```

THIS PROCEDURE DEFINES THE PARTIAL WORD OPERATIONS
IN THE C-D STORAGE AREA

```

+ FTRARG, DEFARG, DEFFUN, NK, CBUF(22),
+ NP, EPTR
+ , XFLG1, XFLG2, XFLG3, XFLG4, XFLG5, XFLG6,
+ XFLG7, XFLG8, XFLG9, XFLG10, XFLG11, XFLG12,
+ ROLFLG(4), BITPLN(1)

COMMON /FMLCM2/ C(1)

COMMON /FMLCM3/ D(1)
LOGICAL INTGSW, MATHSW, POWER
LOGICAL SIMPSW, BITSW
LOGICAL PRODEX
REFERENCES ON

END
PWORD PROC

DEFINE IBIT(I)=FLO(0,1,C(I))
DEFINE ITYP(I)=FLO(1,5,C(I))
DEFINE ITYPB(I)=FLO(0,6,C(I))
DEFINE LAST(I)=FLO(6,15,C(I))
DEFINE NEXT(I)=FLO(21,15,C(I))
DEFINE H1(I)=FLO(0,18,D(I))
DEFINE H2(I)=FLO(18,18,D(I))

END

```


FORMAL,ACOMMON
COMMON BLOCKS FOR ASSEMBLY PROGRAMS

	DEF	
CCDIM	EQU	2048
PLNSIZ	EQU	100
NXNXNX	EQU	FMLCM1
ILILIL	EQU	NXNXNX+1
NS	EQU	ILILIL+1
XQTOPT	EQU	NS+1
PRODEX	EQU	XQTOPT+1
INTGSW	EQU	PRODEX+1
MATHSW	EQU	INTGSW+1
EXPDSW	EQU	MATHSW+1
POWER	EQU	EXPDSW+1
BASE	EQU	POWER+1
SIMPSW	EQU	BASE+1
HITSW	EQU	SIMPSW+1
IOUNIT	EQU	HITSW+1
NSB	EQU	IOUNIT+1
NSY	EQU	NSB+1
NSY1	EQU	NSY+1
NSU	EQU	NSY1+1
NSU1	EQU	NSU+1
N1	EQU	NSU1+1
N2	EQU	N1+1
N3	EQU	N2+1
IY	EQU	N3+1
FTRARG	EQU	IY+1
DEFARG	EQU	FTRARG+1
DEFFUN	EQU	DEFARG+1
NK	EQU	DEFFUN+1
CBUF	EQU	NK+1
NP	EQU	CBUF+22
EPTR	EQU	NP+1
XFLG1	EQU	EPTR+1
XFLG2	EQU	XFLG1+1
XFLG3	EQU	XFLG2+1
XFLG4	EQU	XFLG3+1
XFLG5	EQU	XFLG4+1
XFLG6	EQU	XFLG5+1
XFLG7	EQU	XFLG6+1
XFLG8	EQU	XFLG7+1
XFLG9	EQU	XFLG8+1
XFLG10	EQU	XFLG9+1
XFLG11	EQU	XFLG10+1
XFLG12	EQU	XFLG11+1
ROLFLG	EQU	XFLG12+1
PLANE	EQU	ROLFLG+4
PTRLGG	EQU	PLANE-FMLCM1+PLNSIZ
COMMON*	PROC	

```
IND*      END  
A          PROC  
           EQU*  
           +  
           END  
1,1  
*IND(1,1),A3  
A
```

[illegible]

FORMAL,FMLCHAIN

TEMPORARY CHAIN ROUTINE

THIS ROUTINE IS A TEMPORARY MEASURE TO CONVERT THE
OLD CALLS TO FMLCHA ROUTINES INTO NEW CALLS, UNTIL
ALL THE CALLING SEQUENCES CAN BE ADJUSTED.

```

FUNCTION IGETF1 (*)
  IMPLICIT INTEGER (A-Z)
  IGETF1 = FMLCH1 (S1)
  RETURN
  ENTRY ILINK1 (J, IT, ID)
  IGETF1 = FMLCH2 (S2, J, IT, ID)
  RETURN
  ENTRY ILINKN (J,I,IK)
  IGETF1 = FMLCH3 (S2, J, I, IK)
  RETURN
  ENTRY ILINK2 (J, I)
  IGETF1 = FMLCH4 (S2, J, I)
  RETURN
  ENTRY IADDDN (JJ, JJ1, JJ2)
  IGETF1 = FMLCH5 (S2, JJ, JJ1, JJ2)
  RETURN
  ENTRY ICOPY0 (*, J)
  IGETF1 = FMLCH6 (S1, J)
  RETURN
  ENTRY EXCHXX (I1, I2, J1, J2)
  IGETF1 = FMLCH7 (S2, I1, I2, J1, J2)
  RETURN
  RETURN 1
  PRINT ' ( ( ( ERROR IN CHAIN 0 ( ( ( '
  STOP
  END

```

1

2

FORMAL,FMLCHAIN1

TEMPORARY CHAIN ROUTINE

THIS ROUTINE IS A TEMPORARY MEASURE TO CONVERT THE
OLD CALLS TO FMLCHB ROUTINES INTO NEW CALLS, UNTIL
ALL THE CALLING SEQUENCES CAN BE ADJUSTED.

```
FUNCTION IFREEN (I,IK)
  IMPLICIT INTEGER (A-Z)
  IFREEN = FMLCHA (S1, I, IK)
  RETURN
ENTRY IFREE1 (I)
  IFREEN = FMLCHB (S1, I)
  RETURN
ENTRY IFREE2 (I)
  IFREEN = FMLCHC (S1, I)
  RETURN
ENTRY IFREE0 (I)
  IFREEN = FMLCHD (S1, I)
  RETURN
ENTRY IFREE3 (I)
  IFREEN = FMLCHM (S1, I)
  RETURN
ENTRY REMOVN (J,JK)
  IFREEN = FMLCHE (S1, J, JK)
  RETURN
ENTRY REMOV1 (I)
  IFREEN = FMLCHF (S1, I)
  RETURN
ENTRY REMOV2 (J)
  IFREEN = FMLCHG (S1, J)
  RETURN
ENTRY RMOVFN (J, JK)
  IFREEN = FMLCHH (S1, J, JK)
  RETURN
ENTRY RMOVF1 (J)
  IFREEN = FMLCHI (S1, J)
  RETURN
ENTRY RMOVF2 (J)
  IFREEN = FMLCHK (S1, J)
  RETURN
ENTRY MOVEXX (I1, I2, J1, J2)
  IFREEN = FMLCHL (S1, I1, I2, J1, J2)
  RETURN
1 PRINT '('' ERROR IN CHAIN 1 '')'
  STOP
END
```

FORMAL,FMLCHA

 GENERAL CHAIN ROUTINES

FUNCTION FMLCH1(*)
 INCLUDE CMMN
 INCLUDE PWORD

IN THIS PROGRAM ELEMENT THERE ARE SEVEN ROUTINES DEFINED.
 THEY ARE AS FOLLOWS :

1. FMLCH1 (*) (IGETF1)
 THIS ROUTINE OBTAINS A STORAGE SPACE FROM THE FREE STORAGE CHAIN, AND RETURNS THE LOCATION OF THIS CELL AS THE FUNCTION VALUE.
2. FMLCH2 (*,J,IT,IO) (ILINK1)
 THIS ROUTINE FIRST OBTAINS A CELL, C(I)-D(I), FROM THE FREE STORAGE CHAIN. THEN IT STORES 'IT' INTO FLD(0,6,C(I)) AND 'IO' INTO D(I). AND FINALLY IT INSERTS THIS CELL C(I)-D(I) INTO A CHAIN FOLLOWING THE CELL AT THE LOCATION J, C(J)-D(J).
 THE VALUE OF THE NEW CELL, I.E. I, IS RETURNED AS THE FUNCTION VALUE.
3. FMLCH3 (*,J,I,IK) (ILINKN)
 THIS ROUTINE INSERTS THE CHAIN STARTING AT C(I)-D(I), ENDING AT C(IK)-D(IK) INTO A CHAIN FOLLOWING THE PAIR C(J)-D(J). THE VALUE OF IK IS RETURNED AS FUNCTION VALUE.
4. FMLCH4 (*,J,I) (ILINK2)
 THIS ROUTINE IS EQUIVALENT TO ILINKN (*,J,I,IK) SUCH THAT IK = NEXT(I) IF ITYP(I) = 1 OR 7, IK=I OTHERWISE.
5. FMLCH5 (*,JJ,JJ1,JJ2) (IADDNN)
 THIS ROUTINE COPIES A CHAIN STARTING AT C(JJ1)-D(JJ1), ENDING AT C(JJ2)-D(JJ2) SUCH THAT COPIED CHAIN STARTS AT C(I)-D(I) ENDS AT C(IK)-D(IK). THEN IT INSERTS THE COPIED CHAIN INTO A CHAIN FOLLOWING THE PAIR, C(JJ)-D(JJ). THE VALUE OF IK IS RETURNED AS FUNCTION VALUE.
6. FMLCH6 (*,J) (ICOPY0)
 THIS ROUTINE COPIES A CHAIN STARTING AT C(J)-D(J) ENDING WITH A ZERO LINK, SUCH THAT THE COPIED CHAIN STARTS AT C(I)-D(I) ENDS WITH A ZERO LINK. THE VALUE OF I IS RETURNED AS FUNCTION VALUE.
7. FMLCH7 (*,I1,I2,J1,J2) (EXCHXX)
 THIS ROUTINE EXCHANGES THE POSITIONS OF TWO SUBCHAINS WHICH STARTS AT C(I1)-D(I1) AND C(J1)-D(J1) AND ENDS AT C(I2)-D(I2) AND C(J2)-D(J2) RESPECTIVELY. THESE TWO SUBCHAINS MAY NOT HAVE COMMON PAIRS OF C-D. THE VALUES OF I1,I2 AND J1,J2 ARE INCHANGED UPON RETURN FROM THIS ROUTINE.

FMLCH1 (*) (IGETF1)
 GET A LOCATION 'K' FROM THE FREE CHAIN

JUMP=1

```

:
CHECK FOR EMPTY FREE CHAIN
..... NOT EMPTY
:
:
GO BACK TO THE CALLING PLACES
:
RETURN WITH VALUE K
:
FMLCH2 (*, J, IT, ID)      (ILINK1)
INSERT DATA FOLLOWING LOCATION 'J'
:
:
:
:
:
:
:
:
:
:
FMLCH3 (*, J, I, IK)      (ILINKN)
LINK IN A CHAIN, I-IK
:
:
:
:
:
:
FMLCH4 (*, J, I)          (ILINK2)
INSERT SINGLE OR DOUBLE WORD CHAIN, I
:
:
:
:
:
:
FMLCH5 (*, JJ, JJ1, JJ2)  (IADDNN)
COPY AND LINK CHAIN, JJ1-JJ2, AFTER JJ
:
:
COMMON PART FOR IADDNN AND ICOPYO
:

```

```

10  CONTINUE
    IF (NXNXNX .EQ. 0) CALL FMLSV1
    K=NXNXNX
    NXNXNX=NEXT(K)
    C(K)=0
    D(K) = 0
    GO TO (40,140,170),JUMP
40  FMLCH1=K
    RETURN
    ENTRY FMLCH2 (*,J,IT,ID)
    JUMP=2
    GO TO 10
140  D(K)=ID
    ITYPB(K)=IT
    IF (J .EQ. 0) GO TO 40
    KK=K
150  IN=NEXT(J)
    LAST(K)=J
    IF (J .NE. 0) NEXT(J)=K
    NEXT(KK)=IN
    IF (IN .NE. 0) LAST(IN)=KK
    FMLCH1=KK
    RETURN
    ENTRY FMLCH3 (*,J,I,IK)
    K = I
    KK=IK
    GO TO 150
    ENTRY FMLCH4 (*,J,I)
    K=I
    KK=K
    IF ((ITYP(KK) .EQ. 1) .OR. (ITYP(KK) .EQ. 7)) KK=NEXT(KK)
    GO TO 150
    ENTRY FMLCH5 (*,JJ,JJ1,JJ2)
    KK=JJ2
    JN=NEXT(JJ)
    K1=JJ
160  K2=JJ1

```


END OF COPY, RETURN WITH FIRST POINTER

SAVE FIRST POINTER FOR COPY

END OF IADDNN, RETURN WITH LAST POINTER

FMLCH6 (*, J) (ICOPY0)
COPY A CHAIN STARTING AT 'J' ENDING WITH A ZERO LINK

FMLCH7 (*, I1, I2, J1, J2) (EXCHXX)
EXCHANGE CHAINS I1-I2 AND J1-J2

```

JUMP=3
GO TO 10
170 D(K)=D(K2)
    C(K)=C(K2)
    LAST(K)=K1
    IF (K1 .EQ. 0) GO TO 190
    NEXT(K1)=K
    IF (K2 .EQ. KK) GO TO 200
180 K1=K
    K2=NEXT(K2)
    IF (K2 .NE. 0) GO TO 10

    FMLCH1=IIK
    RETURN

190 IIK=K
    GO TO 180

200 NEXT(K)=JN
    IF (JN .NE. 0) LAST(JN)=K
    GO TO 40

ENTRY FMLCH6 (*, JJ1)
JN=0
K1=0
KK=0
204 K2 = H1(JJ1)
    IF (K2 .EQ. 0) GO TO 205
401 FMLCH1 = ROLLIN ($300, $400, -K2, 0)
    RETURN
205 IF (NXNXNX .EQ. 0) GO TO 206
    IF (WCOUNT(NXNXNX) .GE. WCOUNT(JJ1)) GO TO 160
206 CALL FMLSV1
    GO TO 204

ENTRY FMLCH7 (*, I1, I2, J1, J2)
JUMP=0
IF (NEXT(I2) .NE. J1) GO TO 220
JUMP=1
210 IN=I1
    I1=J1
    J1=IN
    IN=I2
    I2=J2
    J2=IN
    IF (JUMP .EQ. 0) RETURN
220 IL=LAST(I1)
    IN=NEXT(I2)

```


END

```
JL=LAST(J1)
JN=NEXT(J2)
NEXT(J2)=IN
LAST(I1)=JL
IF (IN .NE. 0) LAST(IN)=J2
IF (JL .NE. 0) NEXT(JL)=I1
IF (JN .NE. I1) GO TO 230
LAST(J1)=I2
NEXT(I2)=J1
IF (JUMP .EQ. 0) GO TO 210
RETURN
230 LAST(J1)=IL
NEXT(I2)=JN
IF (IL .NE. 0) NEXT(IL)=J1
IF (JN .NE. 0) LAST(JN)=I2
GO TO 210
300 RETURN 1
400 CALL FMHSV1
GO TO 401

END
```

FORMAL,FMLCHB

FUNCTION FMLCHA (*, I, IK)

CONTINUATION OF GENERAL CHAIN ROUTINES

THE FOLLOWING ROUTINES ARE THE CONTINUATION OF THE
GENERAL CHAIN ROUTINES LISTED IN THE LAST DECK.

THE ROUTINES IN THIS DECK ARE :

1. FMLCHA (*, I, IK) (IFREEN)
THIS ROUTINE RETURNS A CHAIN, I-IK, TO THE FREE
STORAGE CHAIN.
THE VALUE OF NEXT(IK) IS RETURNED.
2. FMLCHB (*, I) (IFREE1)
THIS ROUTINE RETURNS A SINGLE-WORD CHAIN TO THE
FREE STORAGE CHAIN.
THE VALUE NEXT(I) IS RETURNED.
3. FMLCHC (*, I) (IFREE2)
THIS ROUTINE RETURNS A CELL AT I, AND ALSO NEXT(I)
TO THE FREE CHAIN IF THIS IS A DOUBLE WORD .
THE VALUE NEXT(I) IS RETURNED.
4. FMLCHD (I, I) (IFREE0)
THIS ROUTINE RETURNS A CHAIN STARTING AT I, ENDING
WITH ZERO LINK TO THE FREE CHAIN.
THE VALUE NEXT(I) IS RETURNED.
5. FMLCHE (*, J, JK) (REMOVN)
THIS ROUTINE REMOVES A SUBCHAIN, J-JK, FROM A
CHAIN CONTAINING THIS SUBCHAIN.
THE VALUE OF NEXT(JK) AND LAST(J) ARE BOTH SET TO
ZERO, THE ORIGINAL VALUE OF NEXT(JK) IS RETURNED.
6. FMLCHF (*, I) (REMOV1)
THIS ROUTINE REMOVES A SINGLE-WORD CHAIN, I.
THE VALUE OF NEXT(I) IS RETURNED.
7. FMLCHG (*, J) (REMOV2)
THIS ROUTINE REMOVES A SINGLE-WORD CHAIN, J, AND
ALSO NEXT(J) IF IT IS A DOUBLE WORD.
THE VALUE OF NEXT(J) IS RETURNED.
8. FMLCHH (*, J, JK) (RMOVFN)
THIS ROUTINE IS EQUIVALENT TO THE SEQUENCE OF
REMOVN(J,JK) AND IFREEN(J,JK) I.E. IT RETURNS AND
FREES A SUBCHAIN , J-JK.
THE VALUE OF NEXT(IK) IS RETURNED.
9. FMLCHI (*, J) (RMOVF1)
THIS ROUTINE IS EQUIVALENT TO THE SEQUENCE OF
REMOV1(J) AND IFREE1(J) I.E. IT REMOVES AND FREES
A SINGLE-WORD SUBCHAIN.
THE VALUE NEXT(J) IS RETURNED.
10. FMLCHK (*, J) (RMOVF2)
THIS ROUTINE IS EQUIVALENT TO THE SEQUENCE OF
REMOV2(J) AND IFREE2(J), I.E. IT REMOVES AND FREES
A SINGLE-WORD OR DOUBLE-WORD SUBCHAIN.

THE VALUE NEXT(J) IS RETURNED.

11. FMLCHL (*, I1, I2, J1, J2) (MOVEXX)

THIS ROUTINE REMOVES A SUBCHAIN, I1-I2, FROM THE CHAIN CONTAINING IT. AND AT THE SAME IT REPLACES SUBCHAIN, I1-I2, BY SUBCHAIN J1-J2.

THE VALUE OF NEXT(I2) IS RETURNED.

12. FMLCHM (*, I) (IFREE9)

THIS ROUTINE DOES THE SAME THING AS FMLCHD, EXCEPT THAT IT CHECKS H1(I), AND INTERPHETS IT AS A POINTER TO A ROLLED OUT EXPRESSION IF IT IS NON-ZERO

FMLCHA (*, I, IK) (IFREEN)

FMLCHB (*, I) (IFREE1)

FMLCHC (*, I) (IFREE2)

FMLCHD (*, I) (IFREE0)

FMLCHM (*, I) (IFREE9)

FMLCHE (*, J, JK) (REMOVN)

INCLUDE CMMN
INCLUDE PWORD

45 KK=IK
K=I
IF (NXNXNX .NE. 0) GO TO 50
NXNXNX = K
GO TO 51
50 NEXT(ILILIL)=K
51 L=NEXT(KK)
ILILIL=KK
NEXT(ILILIL)=0
FMLCHA = L
RETURN
40 FMLCHA = IN
RETURN

ENTRY FMLCHB (*, I)
KK=I
GO TO 45

ENTRY FMLCHC (*, I)
KK=I
IF ((ITYP(I) .EQ. 1) .OR. (ITYP(I) .EQ. 7)) KK=NEXT(KK)
GO TO 45

ENTRY FMLCHD (*, I)
KK=I
80 DO 90 L = 1,32768
IF (NEXT(KK) .EQ. 0) GO TO 45
90 KK=NEXT(KK)
PRINT 100
100 FORMAT ('0ERROR IN RELEASING A LOOP CHAIN')
STOP

ENTRY FMLCHM (*, I)
KK = I
IF (H1(KK) .NE. 0) CALL ROLLIN (\$300, 0, -H1(KK), 2)
GO TO 80

```

ENTRY FMLCHE (*, J, JK)
JUMP=1
KK=JK
K=J
IL=LAST(K)
IN=NEXT(KK)
LAST(K)=0
IF (JUMP .EQ. 1) NEXT(KK) = 0
IF (IL .NE. 0) NEXT(IL)=IN
IF (IN .NE. 0) LAST(IN)=IL
GO TO (40,50),JUMP

ENTRY FMLCHF (*, J)
JUMP=1
KK=J
GO TO 115

ENTRY FMLCHG (*, J)
KK=J
JUMP=1
IF ((ITYP(KK) .EQ. 1) .OR. (ITYP(KK) .EQ. 77)) KK=NEXT(KK)
GO TO 115

ENTRY FMLCHH (*, J, JK)
JUMP=2
GO TO 112

ENTRY FMLCHI (*, J)
KK=J
JUMP=2
GO TO 115

ENTRY FMLCHK (*, J)
KK=J
IF ((ITYP(KK) .EQ. 1) .OR. (ITYP(KK) .EQ. 77)) KK=NEXT(KK)
GO TO 130

ENTRY FMLCHL (*, I1, I2, J1, J2)
K=I1
KK=I2
I1=J1
I2=J2
JUMP=2

IF ((J1 .EQ. NEXT(KK)) .OR. (K .EQ. NEXT(J2))) GO TO 120

IL=LAST(K)
IN=NEXT(KK)
JL=LAST(J1)
JN=NEXT(J2)

```

```

LAST(J1)=IL
NEXT(J2)=IN
IF (IL .NE. 0) NEXT(IL)=J1
IF (IN .NE. 0) LAST(IN)=J2
IF (JL .NE. 0) NEXT(JL)=JN
IF (JN .NE. 0) LAST(JN)=JL
GO TO 50
300 RETURN 1
END

```

```

FORMAL,FMLSV1
*****
ROLLOUT ROUTINE FOR ALL EXPRESSIONS
*****

```

SUBROUTINE LEVEL

SYMBOL LEVEL

INDIRECT

UNDEFINED VARIABLE

NONSUBSCRIPTED, DEFINED VARIABLE

```

SUBROUTINE FMLSV1
INCLUDE CMMN
INCLUDE PWORD
LOGICAL FLAG
FLAG = .FALSE.
PRINT 111
111  FORMAT (' SYMBOL TABLE FULL, ALL EXPRESSIONS WILL BE ROLLED OUT')

      IF (NS .EQ. 0) GO TO 10
      SR = NS
1      CONTINUE
      IF (LAST(SR) .EQ. 0) GO TO 1000
      EX = LAST(SR)

2      CONTINUE'
      IF(C(EX).LT.0) GO TO 21
      I = ITP(EX)
      IF (AND(I,24)-8) 22,23,24

21     CONTINUE
      GO TO 2000

22     CONTINUE
      GO TO 2000

23     CONTINUE
      IF (LAST(EX) .EQ. 0) GO TO 2000
      IT = LAST(EX)

```

```

:
:
SUBSCRIPTED VARIABLE
:
:
SUBSCRIPT LEVEL
:
:
ASSIGNED SUBSCRIPTED VARIABLE
:
:
EXPRESSION LEVEL
:
:
UNASSIGNED SUBSCRIPTED VARIABLE
:
:
END OF SUBSCRIPTS
:
:

```

```

ASSIGN 2000 TO JUMP
GO TO 4

24  CONTINUE
    IF (LAST(EX) .EQ. 0) GO TO 2000
    SS = LAST(EX)
    ASSIGN 3000 TO JUMP

3   CONTINUE
    I = FLD (3,3,C(SS))
    IF (FLD(0,3,C(SS)) - 6) 31, 32, 33

31  CONTINUE
    IF (LAST(SS) .EQ. 0) GO TO 3000
    IT = LAST(SS)

4   CONTINUE
    IF (H1(IT) .EQ. 0) FLAG = .TRUE.
    CALL ROLOUT ($99, IT)
    GO TO JUMP

32  CONTINUE
    GO TO 3000

33  CONTINUE
3000 IF (NEXT(SS) .EQ. 0) GO TO 2000
    SS = NEXT(SS)
    GO TO 3
2000 IF (NEXT(EX) .EQ. 0) GO TO 1000
    EX = NEXT(EX)
    GO TO 2
1000 IF (NEXT(SR) .EQ. 0) GO TO 10
    SR = NEXT(SR)
    GO TO 1
10  CONTINUE
    IF (FLAG) RETURN
    PRINT 88
88  FORMAT (' SYMBOL TABLE FULL, NO SYMBOLS COULD BE ROLLED OUT')
    CALL DUMP(0)
    STOP
99  CALL ERRRET ('FMLSv1')
    RETURN
    END

```

3.1.2. DRUM STORAGE

THE DRUM IS USED IN PAIRS OF SECTORS CORRESPONDING TO 56 WORDS. THERE ARE 35*PLNSIZ PAIRS OF SECTORS RESERVED ON DRUM. PLNSIZ IS PRESENTLY SET TO 100 BY A PARAMETER STATEMENT IN FMLCM1. AN ARRAY PLANE, OF SIZE PLNSIZ, IS RESERVED IN CORE. EACH WORD CORRESPONDS TO 35 PAIRS OF CONSECUTIVE SECTORS, SUCH THAT THE CORRESPONDING BIT POSITION IS USED TO INDICATE WHETHER THE PAIR OF SECTORS IS FREE TO USE (0) OR IF IT IS ALREADY USED (1). THE FIRST BIT OF EACH WORD OF THE PLANE IS ALWAYS SET. THE 56 WORDS OF A PAIR OF SECTORS IS USED AS FOLLOWS:

FIRST WORD: UPPER HALF: NWDS, THE NUMBER OF C-D PAIRS IN THE BLOCK (UP TO 47)

LOWER HALF: LINKAGE TO THE NEXT PAIR OF SECTORS
(ZERO IF THIS IS THE LAST PAIR FOR AN EXPRESSION)

WORDS 2-9: THE FIRST 6 BITS OF EACH OF THE NWDS C-WORDS IN PACKED FORM, (THE LAST 30 BITS OF THE C-WORDS ARE USED FOR CORE LINKAGE, THUS IT IS NOT TRANSFERRED TO DRUM.)

WORDS 10-56: THE NWDS D-WORDS.

THE FOLLOWING PAGES GIVE THE DEPENDENCE OF THE DRUM STORAGE HANDLING PROGRAM ELEMENTS AND THEIR LISTINGS.

```

*****
*          *          *          *
* ESDRUM  *.....>* FMLCHB *.....>* ESDRUM  *
*          *          *          *
*****
.
.
.
.....>* IOINTF *.....>* SEC2IO *
*          *          *          *
*****
*          *          *          *
*          *          *          *
*****

```


FORMAL,ESDRUM

DRIVER FOR SAVING/RESTORING EXPRESSIONS ON DRUM

DATA STRUCTURE: CMMN AND PWORD

ARGUMENTS: * ERROR RETURN
 INDEX POINTER TO EXPRESSION TO BE ROLLED OUT
 THIS ROUTINE SAVES ONE EXPRESSION, STARTING AT INDEX. IT
 MARKS THE EXPRESSION AS BEING ON DRUM (LEFT HALF OF FIRST
 D-WORD NON-ZERO, = DRUM ADDRESS), AND RELEASES THE REST
 OF THE CHAIN

```
MAKE SURE FILE IS ASSIGNED
:
IF EXPRESSION IS ALREADY OUT, RETURN
:
COUNT LENGTH OF EXPRESSION
:
CALL EXPOUT TO ROLL OUT
:
SET H1(INDEX) TO DRUM ADDRESS
:
RELEASE REST OF EXPRESSION
:
AND CLEAR FORWARD LINK
```

DRUM ROLL IN ROUTINE

ARGUMENTS: * ERROR RETURN
 * NO ROOM RETURN
 IDENT > 0 POINTER TO D-WORD WITH DRUM ADDRESS
 < 0 DRUM ADDRESS
 FLAG = 0 READ
 = 1 READ AND PURGE
 = 2 PURGE

THIS ROUTINE ROLLS EXPRESSIONS IN FROM DRUM, AND/OR PURGES
 THEM. IF IDENT > 0 THE EXPRESSION WILL START AT IDENT, OTHERWISE
 IT WILL BE PUT INTO AN INDEPENDENT CHAIN.

```
RETURN 2 IF THERE IS NO ROOM
:
IDENT < 0 MEANS DRUM ADDRESS
:
... IF THIS IS PURGE, FREE FIRST LOCATION
:
:
```

FUNCTION ROLOUT(\$,INDEX)

```
INCLUDE CMMN
INCLUDE PWORD
```

EQUIVALENCE (A0,ROLOUT),(NWDS,LSTNUX)

CALL S2INIT ('FORMAL\$SAVES', ERR, \$900)

IF(H1(INDEX).GT.0) RETURN W GO AWAY IF ALREADY DONE

NWDS = WCOUNT(INDEX)

A0=EXPOUT(NWDS,INDEX,ERR,\$900)

H1(INDEX)=A0 B PUT IDENT IN PROPER PLACE

CALL FMLCHD (\$800, NEXT(INDEX))

```
NEXT(INDEX)=0
RETURN
```

ENTRY ROLLIN(\$,\$,IDENT,FLAG)

SW=FLAG+1

W BIAS OUT ZERO FOR INDEXING

IF(FLAG.LT.2.AND,NXNXNX.EQ.0) RETURN 2

IF(IDENT.LT.0) GO TO 35

W PURE IDENT OR INDEX?

IF(FLAG.LT.2) GO TO 31

W PURGE-ONLY FREES CHAIN CELL

ID=H1(IDENT)

W IDENT IS CHAIN INDEX HERE

CALL FMLCHB (\$800, IDENT)

GO TO 36

END

FORMAL JOINT
EXPRESSION PACKING AND UNPACKING FOR ROLLIN/ROLOUT

NWDS = WDOUNT (INDEX)

BPADRS = EXPDUT (NWDS , INDEX , ERR , S)

	AXRS, COMMON, LIT.	
S(2)		
S(1).		
WCOUNT*.		
	L,U	A0,077777.
	L	A2,(1,1).
	L	A1,*0,11.
WCT1.		
	AND	A0,FMLCM2-1,A1.
	TZ	A1.
	JMGI	A2,WCT1.
	L,U	A0,,A2.
	J	2,11.
/.		
EXPDUT*.		
	TNZ	(!ITFLAG!);
	J	EXP01.
	L,U	A0,13.
EXRTNA	S	A0,*2,11.
	J	*3,11.
EXP01.		
	L	A0,*0,11.
	DSL	A0,36.
	S	A1,A2.
	TG,U	A1,47.
	DI,U	A0,47.
	MSI,U	A0,8.
	A	A0,A2.
	JZ	A1,EXP01A.
	A,U	A0,8.
EXP01A.		
	LXI,U	11,,11.
	LMJ	11,IRCHSN.
	J	EXRTN.
	LR,U	R2,4.
	J	EXP02.
EXRTN.		
	LXM,U	11,0.
	NOP	0,0,*11.
	J	EXRTNA.
EXP02.		
	LMJ	11,GETUSF.
	J	EXRTN.
	JZ	A0,EXP07.
	TZ	A1.
	S	A1,A5.
	LXI,U	A0,1.
	S	A0,A1.

IS SEC2IO INITIALIZED?
YES
NO - NEED FILE - ERROR 13
ERROR CODE => ERR
TAKE S-RETURN.

GET NWDS

DIVISION AVOIDABLE?
NO - FOR EVERY 47 EXPRESSION WDS
THERE ARE 8 C-TABLE (CTAB) WDS

- AND 8 CTAB WDS FOR REMAINDER

MAKE X11 SAVE ITSELF
INITIALIZE SLAB CHAIN
WHUUPS!! - ERROR
SET CTABS FOR 1ST CTAB WORD
- AND GO BUILD OUR SLAB

RECOVER OLD X11

GO TAKE ERR RETURN

GET US A BUFFER
RETURN ANY ERROR CODE
NO BUF MEANS END-OF-CHAIN
IDENT RTN FOR 1ST BUFFER?
YES - SAVE
FORM BUFFER POINTERS
(A1)=(1,CTAB)

INDEX = EXPRIN (NWDS, BPADRS, ERR, S)

	A,U	A0,8.	(A0)=(1,D-AREA)
	LR,U	R3,7.	INDEX FOR 8 C-TABLE WORDS
	LN	A3,R2.	1ST C-TAB WORD??
	TNE,U	A3,1.	IF SO, 1ST BUFFER??
	J	EXP04.	NOT 1ST, GET A2 FROM 'SAVEA2'
	SZ	A4.	1ST BUFFER, GET A2 FROM CALLER
	L	A2,11.	GET ORIG INDEX FROM ORIG X11
	SSL	A2,18.	
	L	A2,*1,A2.	
	J	EXP05.	
EXP04.			
	LR,U	R2,4.	1ST CTAB WD FROM BELOW EXP02
	L	A2,('SAVEA2').	RECOVER CHAIN POINTER
	J	S+2.	
EXP04A	L,U	R2,5.	SUBSEQUENT CTAB WDS HAVE 6 CTABS
	SZ	A4.	CLEAR CTAB WD PACK REGISTER
EXP05.			
	L	A3,FMLCM3-1,A2.	D-WORD TO D-AREA
	S	A3,*A0.	GET LINK
	L	A3,FMLCM2-1,A2.	PACK IDENT
	LDSC	A3,6.	
	DSC	A2,21.	15-BIT FWD INDEX TO A2
	SSL	A2,21.	END OF CHAIN?
	JZ	A2,EXP06.	NO - SAVE CURRENT ITEM INDEX
	S	A2,('SAVEA2').	C-WORD FULL?
	J6D	R2,EXP05.	YEP - OFF TO CTAB
	S	A4,*A1.	C-TABLE SATURATED?
	J6D	R3,EXP04A.	YES - WRITE OUT & GET NEW BUFFER
	J	EXP02.	
EXP06.			
	A,U	A4,040.	LAST CWORD - LAST IITYP IS MODULO
	L	A2,R2.	LEFT JUSTIFY IT
	MSI,U	A2,6.	
	LSSL	A4,A2.	
	S	A4,A1.	
	J	EXP02.	WRITE OUT LAST BUFFER
EXP07.			
	L	A0,A5.	RECOVER DRUM EXPRESSION IVENT
	LXM,U	11,0.	RECOVER OLD X11
	NOP	0,0,*11.	
	L	A2,('SAVEA2').	RECOVER LAST ITEM INDEX
	S	A2,*0,11.	RETURN TO CALLER
	J	5,11.	
/%			
EXPRIN*.			
	SZ	A3.	INDIRECT CAN BE TOUCHY WITH RALPH
	L	A4,*1,11.	GET BPADRS
	LXI,U	11,11.	MAKE X11 SAVE ITSELF
	LMJ	11,WCOUNT.	SO WE CAN GO COUNT
	+	NXNXNX.	THE FREE CHAIN, HOW MUCH ROOM?

	+	0.	
	S	A0,A5.	- THAT MUCH
	L	A0,A4.	RECOVER BPAURS
	LMJ	11,READSF.	INITIATE READ , NWDS =>(A1)
	J	EXRTN.	
	DSL	A1,36.	
	S	A2,A3.	
	TG,U	A2,56.	DIVISION AVOIDABLE?
	DI,U	A1,56.	NO - FOR EVERY 56 SLAB WORDS
	MSI,U	A1,8.	- THERE ARE 8 CTAB WORDS
	AN	A3,A1.	THIS GETS NWDS OF TRUE EXPRESSION
	TZ	A2.	UNLESS THERE IS INCOMPLETE END SL
	AN,U	A3,8.	IT TOO HAS A-WD C-TABLE
	TG	A5,A3.	NOW! - W I L L IT FIT??
	J	EXPR1.	YES. GO PLAY WITH IT
	L,U	A0,14.	NO, IT WILL NOT - ERROR 14
	J	EXRTN.	TELL HIM SAD NEWS
EXPR1.			
	L	A1,11.	UN-QUEER OUR RETURN
	SSL	A1,18.	SO WE CAN GIVE BACK NWDS
	S	A3,*0,A1.	
	L	A2,NXNXNX.	GET NEXT FREE LOCATION
	S,H2	A2,EXPR7.	TELL HIM THAT'S WHERE IT'S AT
	SZ	A5.	SET BACK LINK = 0
EXPR2.			
	LXI,U	A0,1.	SET UP BUFFER POINTERS
	S	A0,A1.	- (A1)=(1,CTAB)
	A,U	A0,8.	- (A0)=(1,DTAB)
	L	A2,NXNXNX.	PICK UP CURRENT NFL HERE
	LR,U	R2,4.	5 CTABS IN WORD 1
	LR,U	R3,7.	HOPE TO FIND 8 C-TABLE WORDS
	L	A4,*A1.	GET 1ST CTAB WORD
	LSSL	A4,6.	SHIFT OUT THE GAPING HOLE
	J	EXPR5.	
EXPR4.			
	LR,U	R2,5.	6 CTABS IN SUBSEQUENT C-WORDS
	L	A4,*A1.	GET THEM IN STAGING AREA
EXPR5.			
	L	A3,*A0.	D-WORD TO D-REGION LUNTO CAESAR
	S	A3,FMLCM3-1,A2.	GET CHAIN WORD
	L	A3,FMLCM2-1,A2.	GET RID OF BACKLINK GARBAGE
	LSSL	A3,3.	SPLICE IN OUR BACK LINK
	LXI	A3,A5.	CURRENT INDEX => NEW B.C.
	S	A2,A5.	LEFT JUSTIFY
	LSSL	A3,3.	SET UP CURRENT ITYP
	LSSC	A4,6.	IF NEGATIVE,WE'RE DONE
	TEP,U	A4,040.	GO UN-MODULO-32 IT
	J	EXPR6.	IF OK, SHIFT INTO OUR C-WORD
	DSC	A3,6.	STONE COMPLETED WORD
	S	A3,FMLCM2-1,A2.	

CALL PURGE (BPADRS,ERR,S)

CALL S2INIT ('FILENAME',ERR,S) [MUST BE > 6 CHARACTERS]

```

LSSL      A4,6.
DSC        A2,15.
SSL        A2,21.
JGD        R2,EXPR5.
JGD        R3,EXPR4.
S          A2,NXNXNX.
SZ         A0.
LMJ        11,READSF.
J          EXRTN.
J          EXPR2.

EXPR6.    AN,U      A4,040.
          DSC        A3,6.
          AND,XU     A3,7*/15.
          S          A4,FMLCM2-1,A2.
          DSC        A2,15.
          SSL        A2,21.
          S          A2,NXNXNX.

EXPR7.    L,U      A0,S-S.
          LXM,U     X11,0.
          NOP       0,0,*11.
          J         5,11.

/.
PURGE*.   L         A0,*0,11.
          S,H2      11,PURG1.
          LMJ       11,PURGSF.
          J         PURG2.

PURG1.    LX,U     X11,S-S.
          J         4,11.

PURG2.    LX,H2    X11,PURG1.
          S        A0,*1,11.
          J        *2,11.

/.
S2INIT*.  TNZ      ('ITFLAG');
          J        4,11.
          S,H2     11,PURG1.
          L        A0,*11.
          DL       A0,A0.
          LMJ      11,INITSO.
          J        PURG2.
          SZ       ('ITFLAG');
          J        PURG1.
          END.

```

CRANK UP NEXT ITYPB
SHIFT OVER CURRENT INDEX
NOW BECOMES NEXT CHAIN INDEX
CTAB WORD EMPTY?
YES - C-TABLE EXHAUSTED?
YES - SAVE NEXT WRITE LOC
REQUEST CONTINUATION FROM
YOU-KNOW WHO

- AND SEE WHAT ELSE WE'VE UNEARTH

UNPACK DOCTORED ITYP

ZERO THE FORWARD LINK

GET NWL

RECOVER INDEX TO EXPRESSION

GET RETURN BACK

FORMAL,SEC2IO
ROLLOUT ROUTINE

TRUNCATE PLNSIZ TO 936 IF NECESSARY

```

COMMON
AXRS,
LIT.
(2)
DO PLNSIZ>936 ,1
EQU 936.
PLNSIZ1* EQU PLNSIZ
MAXSLB EQU PLNSIZ*35.
SLBMAX EQU MAXSLB+1,,U.
MAXTRK EQU MAXSLB//32;
MAX SLAB ADDRESS
USED FOR TG'S ONLY!!
HIGHEST TRACK NEEDED TO CONTAIN

```

MAXSLB DUAL-SECTORS (SLABS)

LBCD CONVERTS AN ARGUMENT TO FD, LJSF

```

LBCD* FUNC
K(1) EQU
B(1) EQU LBCD(1)
RM* FUNC
RMDR EQU B(1)-RM(1)*(B(1)/RM(1))+0'
B*(1) EQU B(1)/RM(1)
END RMDR*/30
DO 6 , DO 1-(B(1)=0) ,K(1) EQU (07777777777**K(1)*/-6)+RM(10)
DO K(1)= ' ,K(1) EQU '0 '
END K(1)
ASG 'QASG,T '
FILNAM RES 2.
'F/1//'.
+ LBCD(MAXTRK)
+ S-ASG,ASG.
CSASG
/.
CSFS FILE ASSIGNMENT IMAGE
CSFS PACKET

```

THE BIT PLANE CONSISTS OF 35*PLNSIZ ENTRIES. THE SIGN BIT OF EACH WORD IS ALWAYS ON TO PREVENT LSC AMBIGUITY.

```

BIT + 1*/34
FULL - 0.
PPOS EQU ROLFLG
RPLANE EQU ROLFLG+1
HIGH EQU ROLFLG+2
LOW EQU ROLFLG+3
MASK TO FIND HOLES IN PLANE

```

BOTH 'HIGH' AND 'LOW' ARE IN PLANE ADDRESS, NOT DRMAO

```

/.
S(1).
HFIND J S-S.

```

FINDS THE LOWEST HOLE IN PLANE, FILLS IT AND CONVERTS BIT ADDR TO DRUM SECTOR ADDRESS. RETURNS (A0)=SECTOR ADR, (A1)=PLANE A A2 VOLATILE - A3 AND BEYOND NOT DISTURBED

```

LR R1,RPLANE
L A2,PPOS.
L A0,FULL.
SNE A0,PLANE,*A2.
J ERR(1).
INDEX TO LOWEST DISCONTINUOUS ADR
MASK TO FIND HOLES
SEARCH FOR AT LEAST 1 HOLE
PLANE SATURATED

```

FED WITH (A1)= PLANE ADDRESS, DRILLS HOLE IN PLANE AT THAT ADDRESS. RETURNS (A0)= DRUM SECTOR ADDRESS, (A1)= PLANE ADDRESS
A2 VOLATILE - A3 AND BEYOND NOT DISTURBED

	LSC	A0, PLANE-1, A2.	FIND HIT COORD OF HOLE
	A	A0, BIT.	FILL HOLE
	SSC	A0, A1.	RESTORE WORD
	TNZ	A0.	IF WORD IS FULL, IT'LL BE ZERO
	L	A0, FULL.	- SO MAKE IT A FULL WORD OF 1'S
	S	A0, PLANE-1, A2.	UPDATE PLANE
	AN, U	A2, 1.	CONVERT TO LAST-HOLE-FOUND ADDRESS
	S	A2, PPOS.	SAVE FOR FUTURE SEARCHES
	L	A0, R1.	UPDATE SNE INDEX
	A, U	A0, 1.	
	S	A0, RPLANE.	
	L, U	A0, A2.	
	TNZ	A0.	IS X COORD = 0?
	TZ	A1.	IFSO, IS Y-COORD ALSO = 0 ??
	J	HFOUND.	
	L, U	A1, ZROADR.	RETURN SENTINEL FOR ZERO PLANE AD
	J	*HFOUND.	
HFOUND1.			
	MSI, U	A0, 35.	
	A	A0, A1.	VOILA! - PLANE ADDRESS
	S	A0, A1.	
	AU, U	A1, 1	
	TG	A2, HIGH	
	S	A2, HIGH	
	LSSL	A0, 1.	X 2 = FASTRAND SECTOR ADDRESS
	J	*HFOUND.	
	HSET	J	S-S.
	TNE, U	A1, ZROADR.	CONVERT 0 BPADRS
	SZ	A1.	
	TN	A1.	IS ADDRESS OUT OF BOUNDS?
	TG	A1, SLBMAX.	
E(2)	J	ERR(2).	
	SZ	A0.	
	LXI	A2, A1.	SAVE PLANE ADDRESS
	TG, U	A1, 35.	AVOID DIVISION IF POSSIBLE
	DI, U	A0, 35.	CONVERT TO PLANE COORDINATES
	LXM	A2, A0.	
	L	A0, BIT	
	SSC	A0, 0, A1	
	AND	A0, PLANE, A2.	SEE IF HIT ALREADY DRILLED
E(3)	JZ	A1, ERR(3).	YES - ERROR 3
HSET1.			
	XOR	A0, PLANE, A2.	TURN BIT OFF
	S	A1, PLANE, A2.	
	DSL	A0, 72.	
	LDSL	A1, 18.	(A1)=PLANE ADDRESS

I/O PACKET FOR LINK-SEARCH IN PURGE FUNCTION

STANDARD I/O EQUIVALENCES

BASIC I/O PRIMITIVE

	L	A0,A1.	SECTOR ADDRESS
	LSSL	A0,1.	ARE WE BELOW PREVIOUS LOW-HOLE?
	TG	A1,LOW.	NOPE - LOW IS LOWEST
	J	*HSET.	
HSET2.	S	A1,LOW.	UPDATE ADDRESS OF LOWEST HOLE
	SSL	A2,18.	RECOVER PLANE WORD ADDRESS
	S,H2	A2,PP0S.	UPDATE PLANE POINTEK
	AN,U	A2,PLNSIZ.	UPDATE SNE INDEX FOR HFINU
	SNA	A2,RPLANE.	
	J	*HSET.	
/.			
S(2).	D-BANK AND EQUF'S FOR MAIN I/O ROUTINES		
BUFF1	RES	56.	EACH BUFF = 2 FASTRAND SECTORS
BUFF2	RES	56.	
PKT1	RES	3.	WDS 1 AND 2 ARE FILE NAME
	+	037*/30.	STATUS CODE = 037 NO FILE ASG
	+	56,BUFF1.	ACCESS WORD
	RES	3.	DRMAD AND PAD OUT OF UNOSLO PACKE
PKT2	RES	4.	
	+	56,BUFF2	
	RES	3.	
PKSW	+	PKT2,PKT1.	H2=CURRENT PACKET IN USE
ZROADR	EQU	077777.	ZERO ADDRESS SENTENAL (PLANE)
PGPK	RES	4.	
	+	1, TOP.	REASUS ONLY 1 WORD (ONUM FWD LINK?)
	RES	3.	
	+	0.	
TOP			DRUM LINK READ BUFFER
NWDS	EQUF	TOP,,H1.	
FWD	EQUF	TOP,,H2.	
STATUS	EQUF	3,A3,S1.	STATUS OF LAST FUNC PERFORMED
STATWD	EQUF	3,A3.	STATUS WORD FOR TP BEFORE WAITS
FUNC	EQUF	3,A3,S2.	I/O FUNCTION TO BE PERFORMED
INPCNT	EQUF	3,A3,H2.	INPUT WORD COUNT (READ ONLY)
BUFLOC	EQUF	4,A3,H2.	LOC OF LOC OF CURRENT BUFFER
DRMAD	EQUF	5,A3.	FASTRAND SECTOR ADDRESS
/.			
S(1).			
IOPRIM.			
	J	S-S.	
	L	A3,PKSW.	CURRENT PKT IN H2
	SSC	A3,18.	GET PREVIOUS PKT
	TP	STATWD.	LAST FUNCTION COMPLETE?
	ER	WAITS.	WAIT TILL IT IS
	TZ	STATUS.	COAST CLEAR?
	J	FIOERR.	NO - FIND OUT WHY
	SSC	A3,18.	OK - RECOVER CURRENT PACKET

BUFFER CHAIN INITIALIZATION.
MUST BE CALLED PRIOR TO STARTING NEW BUFFER CHAIN,
FED WITH (A0) = TOTAL WORD COUNT OF CHAIN

```

S      A0,DRMAD.
L      A0,A3.
ER     IOW$.
TZ     STATUS.
J      FIOERR.
L      A0,TOP.
JN     A0,PURG2.
AND,U  A0,077777.
J      PURG1.

PURG2. SZ      NWDS.
J      1,11.

/.
IBCH$N*.

```

(A0) RETURNED WITH SECTOR ADDRESS
GET PACKET
WAIT ON THIS I/O - NOTHING TO DO
IT'S COMPLETE, ALL OK?
NO - WHY?
GET INFO.
END SENTENAL
PLANE ADDRESS STRIPPED TO A1
FOLLOW LINKAGES

KEEP WRITE-COMplete FLAG OFF

```

TZ     NWDS.
J      ERR(6).
L,U    A0,,A0.
E(8)   JZ     A0,ERR(8).
S      A0,NWDS.
SZ     ('1STBUF').
J      1,11.

/.
GETB$F*.

```

IS LAST CHAIN FINISHED?
NO - SAME ERROR AS BEFORE
NWDS MAX IS 262143
HUM???
THIS, AS A FLAG, ALLOWS OUTPUT
FLAG TO SQUELCH 1ST BUFFER WRITE

BASIC OUTPUT ROUTINE.
MUST BE CALLED TO ACQUIRE A BUFFER AND CLEAR A FILLED BUFFER
BY OUTPUTTING TO FASTHAND. BUFFER WRITE IS AUTOMATIC ON ANY
CALL OTHER THAN FIRST UNTIL ORIGINAL NWDS SPECIFICATION FROM
IBCH\$N INITIALIZATION HAS BEEN EXHAUSTED BY 55 WORD DECREMENT
EACH TIME A WRITE OCCURS. . ANY ATTEMPT TO WRITE AN EMPTY BUFF
RESULTS IN ERROR RETURN.
RETURNS (A0)= BUFFER ADDRESS (55 WORDS)

(A1) = INITIAL WRITE LOCATION (PLANE) ON INITIAL CALL

```

E(9)   TNZ
J      NWDS.
L      ERR(9).
TNZ    A3,PKSW.
J      ('1STBUF');
L,U    GETB1.
S      A0,W$.
SLJ    A0,FUNC.
SSC    IOFRIM.
L      A3,18.
SSC    A0,BUFLOC.
TN      A3,18.
J      0,A0.
DSL    GETB2.
SZ     A0,72.
NWDS.

```

ARE WE IN PROGRESS?
NO - NEED INITIALIZATION
GET CURRENT PACKET
IS THIS 1ST CALL?
YES - ALLOW THIS BUFFER TO FILL
NO - WRITE OUT USED BUFFER
SET WRITE FUNCTION
ACTUATE PRIMITIVE
PKTS ARE SWITCHED - GET PREVIOUS
GET THE BUFFER WE'RE WRITING.
RESTORE THE %#!*A* PACKET POINTER
WAS LINK END SENTINEL?
NO - FEED HIM NEXT BUFFER
YES - KILL ERRONEOUS RETURN INFO
FLAG OURSELVES WE'RE DONE

	J	1,11.	
GETB1.	SLJ	HFINO.	1ST CALL - GET A FREE SEAM
	S	A1,FWD.	SET UP FOR GETB2
	S,H2	A1,GETB6.	SAVE 1ST LINK FOR RETURN
GETB2.	L	A0,FWD.	CONVERT FWD LINK TO CURRENT ORMD
	TNE,U	A0,2ROADR.	CONVERT IF END SENTINEL
	SZ	A0,	
	LSSL	A0,1.	BPAURS X 2 = SECTOR ADDRESS
	S	A0,ORMAD.	
	L	A0,NWDS.	DECREMENT WORD COUNT
	AN,U	A0,55.	
	TLE,U	A0,1.	UNDERFLOW OR ZERO?
	J	GETB4.	YES - NO LINK
	S,H2	A0,GETB3.	NO - NEED FWD LINK
	SLJ	HFINO.	GET SAME
	LXI	A1,NWDS.	MAKE A LINK WORD
	L	A2,BUFLOC.	INSTALL AT BUFFER HEAD
	S	A1,,A2.	
GETB3	LXI,U	A1,S-S.	NEXT LINK REFLECTS NWDS=55
	S	A1,TOP.	
	J	GETB5.	
GETB4.	SZ	FWD.	CURRENT BUFFER IS LAST & 0 FWD CI
	LN	A0,NWDS.	KEEP NWDS IF 1ST IS LAST, BUT
	S	A0,NWDS.	SET END SENTINEL
	LSSL	A0,18.	MAKE A SLAM LINK WORD
	L	A2,BUFLOC.	FLAG OURSELVES *****
	S	A0,,A2.	
GETB5.	L	A0,A2.	RECOVER CURRENT BUFLOC
	A,U	A0,1.	BIAS PAST LINK WORD
	USL	A1,72.	
	TNZ	('1STBUF');	1ST BUFFER?
GETB6	L,U	A1,S-S.	YES - RECOVER BPAURS
	S	A0,('1STBUF').	RESET FLAG
	J	1,11.	

/.
READSF*.

ROUTINE TO READ A CHAIN OF SLABS FROM FASTRAND.
 -- SELF INITIALIZING --
 IF ENTERED WITH (A0) NON-ZERO, CONTENTS ARE TAKEN
 TO BE THE BIT-PLANE ADDRESS (BPADRS) OF THE START
 OF A BUFFER CHAIN. INITIALIZATION ENSUES:
 ONE READ REQUESTED FROM IOPRIM
 FOLLOWED BY WAIT. NWDS FROM LINK WORD
 IS SAVED FOR RETURN IN A1.
 THEREAFTER, ROUTINE PROCEEDS AS DESCRIBED BELOW.
 IF ENTERED WITH (A0)=0, THIS IS TAKEN AS A CONTIN-

UATION REQUEST:

THE LINK WORD FROM THE PREVIOUS READ
IS CONSULTED FOR FWD LINK. THIS IS USED
AS THE DRMAU FOR NEXT READ. ROUTINE
EXITS DURING SECOND READ WITH
(A0) = ADDRESS OF BUFFER CONTAINING PREVIOUS
SLAB.

	TZ	NWDS.	IS LAST WRITE COMPLETED?
	J	ERR(6).	NO
	L	A3,PKSW.	ALL CLEAR - GET CURRENT PACKET
	JNZ	A0,READ2.	CONTINUATION REQUESTED?
	TZ	('CONTNU');	YES - CONTINUATION POSSIBLE?
	J	ERR(10).	NO - NO WAY OUT
E(10)			
READ4.			
	SSC	A3,18.	OK - GET PREVIOUS PACKET
	TP	STATWD.	DONE?
	ER	WAITR.	NO - LET IT GO TO COMPLETION
	L	A0,BUFLOC.	SO WE WILL RETRIEVE CORRECT
	L	A1,A0.	LINK WORD
	JN	A1,READ3.	END SENTINEL? - QUIT
	USC	A0,18.	NO - LINK ->(A0),
	SSL	A0,18.	NWDS ->(A1)
	TZ	('CONTNU');	CONTINUATION?
	S,H2	A1,('CONTNU');	INITIALIZATION - SAVE NWDS
	SSC	A3,18.	BACK TO CURRENT PACKET
	SLJ	READ1.	START NEXT SLAB READ
	DSL	A1,72.	CLEAR ERRONEOUS RETURN 'INFO'
	L	A0,BUFLOC.	GIVE HIM A BUFFER
	A,U	A0,1.	-BIASED PAST LINK WORD
	TZ	('CONTNU');	
	L,H2	A1,('CONTNU');	INITIALIZATION - RECOVER NWDS
	SZ	('CONTNU');	SET FOR CONTINUATION
	J	1,11.	
READ1.			
	J	S-S.	
	DSL	A0,36.	
	TNE,U	A1,ZROADR.	CONVERT IF ZERO SENTINEL
	SZ	A1.	
	S	A1,A2.	SAVE HPADRS
	TG,U	A1,34.	AVOID DIVISION IF POSSIBLE
	DI,U	A0,35.	NOT - SEPARATE PLANE COORDINATES
	L	A0,PLANE,A0.	HAS THIS SLAB BEEN PURGED?
	LSSL	A0,1,A1	
E(11)	JP	A0,ERR(11)	
	LSSL	A2,1.	SLAB OK - HPADRS X 2 = SECTOR ADD
	S	A2,DRMAD.	SET SAME
	L,U	A0,RS.	SET READ FUNCTION
	S	A0,FUNC.	
	SLJ	IOPRIM.	REQUEST I/O
	J	*READ1.	

- UNLESS

ROUTINE TO HANDLE I,O ERRORS

READ2.

TP
J
TG
TNE,U
J
J
S
SLJ
J

AO,
ERR(7).
AO,SLBMAX.
AO,ZROADR.
\$+2.
ERR(2).
AO,('CONTNU').
READ1.
READ4.

INITIAL READ, NEW CHAIN
DON'T ACCEPT GARBAGE

FLAG CHANGES ROLES IN THIS CONTEX
BRING IN 1ST SLAB
THIS WILL WAIT AND EXIT READING N

READ3.

S
A,U
LN
SSL
J

AO,('CONTNU').
AO,1.
A1,A1.
A1,18.
1,11.

END SENTINEL - NO CONTINUATION
BIAS BUFFER ADDRESS PAST LINK
RECOVER POSSIBLE NON-ZERO NWDS
- IF 1ST BUFFER IS ALSO LAST

/.
ERGEN
ERR(ERGEN)
ERRX.

DO

E ,;
LMJ

AO,ERRX.

DSL
AN,U
LXI,U
J

A1,72.
AO,ERR(1).
AO,0.
0,11.

ROUTINE TO RETURN ERROR INDEX
THUS THE ERROR INDEX
CLEAR OUT INC, JUST IN CASE.....

/.
FIOERR.

LSSNAP
L,U
J
END.

'FIOERR',2,17,PKT1.
AO,E+1.
0,11.

3.1.3. FILE STORAGE

THE LIMITATIONS ON STORAGE AREA AS WELL AS THE WISH TO SAVE INTERMEDIATE RESULTS, ESPECIALLY FROM DEMAND TERMINALS, MAKE IT DESIRABLE TO BE ABLE TO SAVE THE WORKING STORAGE OF ANY FORMAL PROGRAM IN A SIMPLE MANNER. THIS IS DONE BY TRANSFERING THE THREE COMMON BLOCKS (3.1.1) AND THE EXPRESSIONS WHICH ARE ROLLED OUT TO THE TEMPORARY DRUM-FILE (3.1.2) ONTO A MASS-STORAGE FILE. THESE AREAS CONTAIN ALL THE INFORMATION ABOUT THE STATE OF THE FORMAL PROGRAM.

THE SAVED DATA MAY BE EITHER IN THE FORM OF A BINARY DATA FILE OR IN THE FORM OF AN ABSOLUTE ELEMENT IN A PROGRAM FILE. THE FORMAT OF AN ABSOLUTE ELEMENT WAS CHOSEN BECAUSE IT WILL ALLOW THE ELEMENT TO BE COPIED, PUNCHED, ETC., BY FURPUR, AND IS LEAST LIKELY TO CAUSE COMPLICATIONS.

THE DATA ARE STORED IN THE FOLLOWING FORM, STARTING AT SECTOR 0 IN CASE OF A DATA FILE OR AT THE NEXT FREE LOCATION IN CASE OF A PROGRAM FILE:

LENGTH (SECTORS)	DESCRIPTION
1	LABEL BLOCK
PTRLG//28	FMLCM1
CDIM//28	FMLCM2
CDIM//28	FMLCM3
N*2	DRUM-AREA

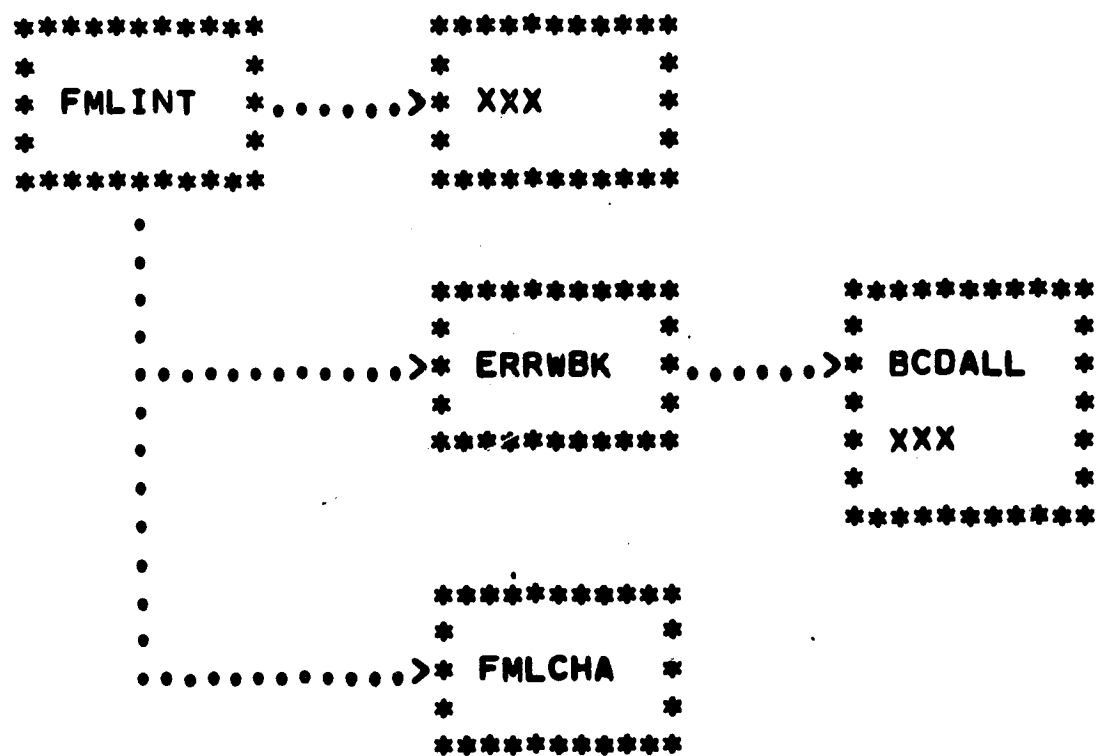
WHERE PTRLG AND CDIM ARE THE LENGTHS OF THE COMMON BLOCKS AND N IS THE NUMBER OF 56 WORD BLOCKS ON DRUM. THIS NUMBER MAY BE ZERO. THE LABEL BLOCK LOOKS AS FOLLOWS:

LABEL+0	NWDS
+1	0
.	.
.	.
+13	0
+14	'FORMAL'
+15	PTRLG,CDIM
+16	N
+17	0
.	.
.	.
+27	0

WHERE NWDS SPECIFIES THE NUMBER OF WORDS FOR THE ENTIRE ELEMENT (OR FILE). FOR A DESCRIPTION OF THE SAVE-RESET STATEMENTS, ERROR MESSAGES, AND PROGRAM LISTINGS, SEE 3.4.3 .

3.1.4. INITIALIZATION ROUTINES

THERE ARE TWO TYPES OF INITIALIZATION HANDLED THROUGH PROGRAM ELEMENT FMLINT. AN OVERALL INITIALIZATION OCCURS BEFORE THE FIRST FORMAL STATEMENT OF THE USERS PROGRAM IS PROCESSED. IN THIS CASE, THE POINTERS IN THE COMMON BLOCK FMLCM1 ARE INITIALIZED, THE DYNAMIC CORE STORAGE, FMLCM2, IS LINKED TOGETHER, THE BIT-LANE FOR ROLL-OUT IS INITIALIZED, AND THE CURRENT VERSION OF FORMAL IS PRINTED OUT BY PROGRAM ELEMENT XXX. AN OTHER INITIALIZATION OCCURS BEFORE EACH FORMAL STATEMENT. THIS INCLUDES THE RETRIEVAL OF FORTRAN TYPE OF ARGUMENTS, AND THE NAME OF THE CALLING ROUTINE FOR THE SETTING OF THE SYMBOL TABLE ENTRY NS IN COMMON BLOCK FMLCM1. THE FOLLOWING PAGES CONTAIN THE ELEMENT DEPENDENCE DIAGRAM AND THE LISTINGS OF THE RELEVANT PROGRAM ELEMENTS: FMLINT, XXX, ERRWBK, AND BCDALL.



FORMAL,FMLINT
OVERALL INITIALIZATION ROUTINE AND INITIALIZATION FOR
FORMAL STATEMENTS

IF THIS IS THE FIRST TIME, OR THE FIRST CALL AFTER
 A SAVE, INITIALIZE THE CHAIN, BITPLANE, AND OPTIONS.

CHAIN THE C-AREA

AND ZERO OUT THE BIT-PLANE

```

COMMON
AXRS.
S(1)
ASSUME(1) EQU 0
ASSUME(2) EQU 1
ASSUME(3) EQU 1
ASSUME(4) EQU 1
ASSUME(5) EQU 0
ASSUME(6) EQU 0
ASSUME(7) EQU 0
FMLIN1* S X11,FLAG.
      . THE FOLLOWING INSTRUCTION IS USED BY SAVE/RESET
FMLINT* T2,H2 WB+1 . INITIALIZED ?
      J INIT0. YES

ER OPTS. GET THE OPTIONS
S A0,XGTOPT. AND DEPOSIT IN COMMON

SET* PROC
II EQU SET(1,2)
DO ASSUME(II)=0 , SZ SET(1,1)
DO ASSUME(II)=1 , SA A1,SET(1,1)
DO ASSUME(II)>1 , L,U A2,ASSUME(II)
DO ASSUME(II)>1 , SA A2,SET(1,1)
END
L,U A1,1
S A1,NXNXNX
SZ NS
SET PRODEX,1
SET INTG5W,2
SET MATHSW,3
SET EXPDSW,4
SET POWER,5
SET BASE,6
SET XFLG2,7
L A0,VERSPR
ER PRINTS
L,U A0,CDIM ILILIL = CDIM
S A0,ILILIL
L A1,INIWRD
S,H2 A1,FMLCM2-2,A1
TLEM,U A1,CDIM
J S-2
SZ FMLCM2+CDIM-1
L,U A1,PLNSIZ-1
L A0,INIWRD+1
S A0,PLANE,A1
JGD A1,S-1
L A0,K10
S A0,RQLFLG
  
```


GET PROPER NUMBER OF FORTRAN ARGUMENTS

GET NAME OF PROGRAM WHICH CALLED THE CALLING ROUTINE,
AND INSERT IT INTO THE SYMBOL TABLE, IF IT WAS NOT
ALREADY THERE. ALSO, SET NS, AND IF NECESSARY ALSO NSB

	L,U	A0,PLNSIZ
	S	A0,ROLFLG+1
	SZ	ROLFLG+2
	SZ	ROLFLG+3
INIT0	L,H2	A0,2,X11.
	L,H2	A1,1,A0.
	L	A2,*1,X11.
	SZ	FTRARG
	JZ	A2,INIT01
	A	A1,A2.
	S,H2	A1,ARG1.
	AN,U	A1,1.
	L	A1,*0,A1.
	S	A1,FTRARG.
	A	A1,A2.
	S,H1	A1,1,A0.
	A,H2	A1,1,A0.
INIT02	L	A1,0,A1.
	L	A0,MAINPR
	TZ,U	0,A1
	L	A0,0,A1.
	SZ	A2.
	L	A3,NS.
INIT1	JZ	A3,INIT3.
	T6	A0,FMLCM3-1,A3
	J	INIT2.
	L	A1,FMLCM2-1,A3
	AND,U	A1,077777.
	DSC	A2,36.
	J	INIT1.
INIT01	A,H1	A1,1,A0
	TNZ	FLAG
	J	INIT02
	L	A0,,A1
	L	A1,1,A0
	JMGI	A1,INIT02
INIT2	TNE	A0,FMLCM3-1,A3
	J	INIT6.
INIT3	S,H2	X11,WB+1.
	S	A0,INITR.
	DS	A2,INITR+1.
	LMJ	X11,FMLCH1

WALKBACK WORD
RETURN WORD
CALL PARAMETER

CALL WITH 0 MEANS NO FORTRAN ARGU
BASE OF FORTRAN ARGUMENTS
INITIALIZE ARG

GET NUMBER OF FORTRAN ARGUMENTS
AND STORE IN COMMON
CONNECT THE NUMBER OF ARGUMENTS
IN THE WALKBACK PACKET
GET RETURN ADDRESS

WALKBACK WORD OF CALLING PROGRAM
. MAIN IF IT'S BFOR MAIN PROG

PROGRAM NAME IS NOW IN A0
SET FOR NO PREVIOUS PROGRAM
START OF SUBROUTINE CHAIN.

END OF CHAIN

D(A3) .LE. NAME
A2 = NEXT(A3)

INTERCHANGE
REPEAT SEARCH

FOUND
NOT FOUND
SAVE REGISTERS.

. NSB = IGETF1 (SERM)

SUBROUTINE TO GET A FONTRAN ARGUMENT

```

J      ERR
+      WB.
L      X11,WB+1.
S      A0,NSB
L      A1,INITR.
S      A1,FMLCM3-1,A0%
DL     A2,INITR+1.
S      A3,FMLCM2-1,A0%
JZ     A2,INIT7.
L      A3,FMLCM2-1,A2%
AND,XU A3,0700000.
A      A4,A0.
S      A4,FMLCM2-1,A2%
INIT5  SZ      FLAG
J      3,X11
INIT6  S      A3,NSB,
J      INIT5.
INIT7  S      A0,NS.
J      INIT5.

ARG+   L      A0,*1,X11.
        AN,U   A0,1.
        JN     A0,ARGERR
        TG     A0,FTRARG,
J      ARGERR.
ARG1   L      A0,S-S,A0.
        L      A0,0,A0.
        J      3,X11.
ARGERR S,M2    X11,WB+1.
        LMJ    X11,ERRERR
J      ERR
+      S+2.
+      S-FMLINT,WB
+      0
ERR     L      X11,WB+1.
J      0,X11.
S(2),WB 'INIT'
+      2,0
+      'FOR',0
INITR   +      0
        +      0
        +      0
FLAG    +      0
MAINPR  'MAIN'
K10+    +      1,0
INIWRD  +      1,2
        +      1*/35
END

```

D(NSB) = NAME
 NEXT(NSB)=A3, LAST&ITYP = 0
 IF(A2 .NE. 0) NEXT(A2)=NSB
 . ALWAYS CLEAR SPECIAL FLAG
 . RETURN
 IT IS ALREADY THERE
 UPDATE NS
 GET ARGUMENT INDEX
 TEST IF WITHIN RANGE
 ERROR RETURN
 . 0 MEANS NORMAL, NON-0 FMLIN1

FORMAL,XXX
SIGN-ON LINE AND COMMON DECLARATION FOR SUBROUTINE

```

$ (1)
VERSN 'FORMALLIB VERSION '
      '0902C'
VERLG EQU $-VERSN
VERSPR* + 0100+VERLG,VERSN
FMLCM1 INFO 2 2
FMLCM2 INFO 2 3
FMLCM3 INFO 2 4
$ (2)
FMLCM1*
      COMMON
CDIM* EQU CCDIM
PTRLG* EQU PTRLG6
      RES PTRLG
$ (3)
FMLCM2*
      RES CDIM
$ (4)
FMLCM3*
      RES CDIM
      END

```

FORMAL,ERRWBK

COMMON
AXRS.

.....
 ERROR AND DEBUGGING ROUTINE FOR OBJECT-TIME F O R M A L
 CALL SNPERR (S , <STATEMENT #>)
 CALL ERRERR (S , <STATEMENT #>)
 SNPERR PRECEDES ERROR BLURBS WITH SNAP DUMP OF X, A, R.
 ERRERR PRINTS STATEMENT # AND FINDS AND PRINTS LINE# FROM WHICH
 IT WAS CALLED TOGETHER WITH NAME OF CALLING PROGRAM. IT THEN
 DOES A FORTRAN-TYPE WALKBACK PRINTING CALLING LINE AND PROGRAM
 UP LEVEL-BY-LEVEL TILL IT REACHES MAIN PROGRAM AT WHICH TIME
 IT INDICATES THIS CONDITION AND RETURNS TO THE S-RETURN.
 IF 0 IS USED INSTEAD OF S-RETURN, CONTROL RESUMES AT THE
 NORMAL RETURN TO THE CALL,

S(2)	LIT		
BL1	'FORMAL OBJECT-TIME ERROR CONDITION AT STAT',		
	'EMENT',		
STATMT	RES	2.	
PF	FORM	12,6,18.	
BLURB1	PF	2,S-BL1,BL1.	
BL2	'IN FORMAL LIBRARY SUBROUTINE',		
SUBRTN	'		
BLURB2	PF	1,S-BL2,BL2.	
S(1).			
SNPERR*	LSSNAP	'SNPERR',7,0,0.	COBOL LIB SNAP ROUTINE
	L	A4,('SNPERR'),	IDENT OF ENTRY
	J	ERRERR+1.	
/.			
ERRERR*	L	A4,('ERRERR'),	
	S	A4,WB.	TELL TRACE WHO WE ARE
	S,H2	X11,WB+1.	AND WHO CALLED US
	L	A0,11.	GET RETURN DESIG
	YNZ	0,A0.	NORMAL REQUESTED?
	L,U	A0,3,11.	YEP - LOAD NORMAL
	S,H2	A0,RETURN.	SAVE
	L	A0,1,11.	GET STATEMENT #
	SLJ	FLDEC1.	CONVERT TO FIELDATA
	DL	A0,FWORD.	RECOVER CONVERSION
	DS	A0,STATMT.	
	L	A0,2,11.	GET CALLERS WB WORD
	DSL	A0,18.	LINE# => (A0)
	DSL	A1,54.	WBPKT LOC => (A2)
	L	A0,A2.	GET PROGRAM NAME
	S	A0,SUBRTN.	PREP INITIAL BLURBS - INSERT HIS
	PSRINT	BLURB1.	
	PSRINT	BLURB2.	
	LMJ	X11,TRACE	
	+	S-SNPERR-1,WB	

S(2).

WB

+

1.

+

2,S-S.

+

'FOR',0.

S(1).

RETURN

J

S-S.

END.

OUR NAME GOES HERE

HARKS WE WERE CALLED WITH - (WE MO

LET'S MAKE US LOOK LIKE WE'RE RAL

AND GO BACK TO WHERE WE SHOULD

FORMAL,BCDALL

BINARY TO FIELDATA CONVERSION ROUTINE

U O M 1108 F O R M A L ASSEMBLER LIBRARY

AXRS.

ANYTHING COMING IN IN (A0) GETS CONVERTED ACCORDING TO
POINT OF ENTRY:

SLJ	FLDECI	YIELDS DECIMAL BCD
SLJ	FLDOCT	YIELDS OCTAL BCD
SLJ	FLDBIN	YIELDS BINARY BCD

IN EACH CASE
THE RESULTS ARE STORED LJSF IN (FWORD,...,FWORD+5)

ALL VOLATILE REGISTERS ARE SAVED AND RESTORED.

S(2)

FWORD*

BSET

RG

LIT.

RES

6.

RES

5.

DCINS

DSL

A0,36.

DI,U

A0,10.

OCINS

DSL

A0,3.

SSL

A1,33.

DECIMAL CONVERSION

INSERT.

OCTAL CONVERSION

INSERT.

S(1).

OUT

DL

A0,RG.

DL

A2,RG+2.

L

A4,RG+4.

J

*IN.

RET

SLJ

IN.

FLDECI*

J

S-S.

J

S-2.

SLJ

IN.

FLDOCT*

J

S-S.

	J	S-2.	
	SLJ	IN.	
FLDBIN*	J	S-3.	
	J	S-2.	
IN	J	S-3.	
	DS	A0,R6.	
	DS	A2,R6+2.	
	S	A4,R6+4.	
	L,H2	A3,IN.	
	J	S-RET,A3.	
	DL	A1,DCINS.	
	DS	A1,FLD1.	
	J	FDOC.	
	DL	A1,OCINS.	
	DS	A1,FLD1.	
	J	FDOC.	
	.		
BIN.		BINARY CONVERSION TO BCD.	
	L,U	A4,5.	RESULT OCCUPIES 6 WORDS.
	L	A3,(-1,5).	STORING BACK TO FRONT.
BIN1.			
	L,U	A2,5.	6 CHAR'S PER WORD.
BIN2.			
	DSL	A0,1.	GRAB A BIT.
	SSL	A1,5.	MAKE ROOM FOR ZONE BITS.
	JGD	A2,BIN2.	GO BACK FOR MORE.
	A	A1,('000000').	ADD ZONE BITS.
	S	A1,FWORD,*A3.	INTO FWORD BACK TO FRONT.
	JGD	A4,BIN1.	AND DO THE NEXT WORD.
	J	OUT.	
	.		
FDOC.		DECIMAL OR OCTAL CONVERSION, DEPENDING ON INSERTS.	
	DL	A2,BSET.	BLANK-FILL ACCUMULATOR
FLD1.			
	S-S.	INSERT A.	
	S-S.	INSERT B.	
	A,U	A1,'0'.	CONVERT TO BCD
	LSSL	A2,6.	SHIFT
	DSL	A1,6.	INTO ACCUMULATOR SET
	DSC	A2,6.	AND MAKE ROOM FOR ONE MORE.
	JNZ	A0,FLD1.	GO BACK FOR MORE.
	LDSC	A2,6.	LJSF SET.
	DS	A2,FWORD.	RESULTS IN LOCATABLE LOCATION.
	DL	A0,BSET.	
	DS	A0,FWORD+2.	
	DS	A0,FWORD+4.	
	J	OUT.	
	END.		

3.1.5. SCANNER ROUTINE

SCANNING OF INPUT FORMAL EXPRESSIONS IS IMPLEMENTED BY THE GENERAL SCANNING ROUTINE GSCAN. THE STRING IS INPUT TO GSCAN BY

CALL ISCAN(<ARRAY>,<DIMENSION>)

WHICH DEFINES THE STRING AS RESIDING IN THE <DIMENSION> WORDS OF ARRAY <ARRAY>. THEREAFTER EACH CALL TO GSCAN OF THE FOLLOWING FORM

CALL GSCANR(\$<A>,<INDEX>,<ITEM>,<ITC>,<ICC>)

RETURNS INFORMATION ABOUT THE NEXT ITEM IN THE FORMAL EXPRESSION:

<INDEX> INDICATES THAT THE NEXT ITEM IS AN
 =1 IDENTIFIER (SIX CHARACTERS),
 =2 FLOATING POINT NUMBER (MACHINE FORMAT),
 =3 INTEGER (MACHINE FORMAT),
 =4 SPECIAL CHARACTER (SEE SPECIAL CHARACTER CODE BELOW).

<ITEM> IS THE IDENTIFIER, FLOATING POINT NUMBER, INTEGER, OR SPECIAL CHARACTER INDICATED BY <INDEX>.

<ITC> IS SET IF <ITEM> WAS AN IDENTIFIER (<INDEX>=1). IF THE IDENTIFIER WAS TERMINATED BY A '(' THEN <ITC>=1, BY A '[' THEN <ITC>=2, OTHERWISE <ITC>=0.

<ICC> IS THE CHARACTER COUNT OF THE IDENTIFIER ABOVE IF IT WAS OVER SIX CHARACTERS IN LENGTH.

THE FOLLOWING TABLE LISTS THE SPECIAL CHARACTERS AND THEIR CORRESPONDING CODES.

@	1	<	8	(	15	'	22
[	2	=	9	%	16	:	23
]	3	>	10	:	17	/	24
#	4	&	11	?	18	.	25
)	5	\$	12	!	19	"	26
-	6	*	13	,	20		27
+	7	**	14	\	21		


```
*****
INITIALIZATION ENTRY
*****
```

```

:
:
:
IDENTIFIER PROCESSING SWITCH
:
DECIMAL MODE INDICATOR
:
STRING EXHAUSTION SWITCH
:
FORTRAN ARGUMENT FLAG
:
:
:
:
:
:
:
LOADER INITIALIZATION
:
TEMPORARY CHARACTER HOLD .
:

```

 MAIN SCANNING ENTRY

```

:
CHECK FOR FAILURE TO INITIALIZE
:
CLEAR EXTERNAL DECIMAL MODE
:
LETTER COUNT (IDENTIFIERS)
:
ASSUME INTEGER MODE AT FIRST
:
IDENTIFIER TERMINATOR CODE
:
IDENTIFIER LENGTH (>0)
:
IDENTIFIER ACCUMULATOR
:

```

SUBROUTINE ISCANGL (LOC, SIZE, SUB)
IMPLICIT INTEGER (A-Z)

```

PARAMETER CHARAS='00000+', CHARBL='00000 ', CHARPD='00000.'
PARAMETER CHARI='00000I', CHARR='00000R'
LOGICAL NUMRL

```

LOGICAL IDENT

LOGICAL DPT

LOGICAL EOS

LOGICAL ARG

```

DIMENSION SPCHAR(27)/0,1,2,3,040,041,042,043,044,045,046,047,050,
      ' * ',051,052,053,054,055,056,057,072,073,074,
      075,076,077/

```

```

DIMENSION ITCODE(2)/051.1/
EQS=.FALSE.

```

CALL LOAD(Q,LOC,SIZE)

```

THOLD=0
RETURN

```

ENTRY GSCANR(S,IND,DATA,IT,ICT)

DATA=0

```
IF (EOS) RETURN 1
```

CALL BCDNPT

AC=0

IND=1

IT=0

ICT=0

ACCUM= 9

IDENTIFIER PROCESSING SWITCH

IDENTIFIER PACKING SWITCH

HANGINGOVER DATA CAUGHT

PROCESS ACCORDING TO TYPES
 GO TO LABEL 21 FOR ALPHABETIC CHARACTER
 GO TO LABEL 50 FOR NUMERALS
 GO TO LABEL 100 FOR NUMERAL DELIMITERS
 GO TO LABEL 150 FOR NON-'*' SPECIAL CHARACTERS
 GO TO LABEL 170 FOR HUNG SC

... LOAD A CHARACTER

IGNORE ALL BLANKS

PROCESS ALPHABETIC CHARACTER

NUMERAL, IDENT FRAGMENT OR DECIMAL POINT

DON'T PROCESS IDENTIFIER

FEED CHARACTER TO EXTERNAL ACCU

FETCH ANOTHER

DELIMITER ENCOUNTERED

ARE WE PROCESSING IDENTIFIER?

CONVERT ACCUMULATED NUMBER

... ALPHABETIC CHARACTER (IDENT FRAGMENT)

CHECK FOR NUMERAL DELIMITER

... IF SO, SAVE THE GOODIES

REMEMBER WHAT THEY ARE

ARG = .FALSE.

IDENT=.FALSE.

NUMRL=.FALSE.

DPT=.FALSE.

ASSIGN 100 TO ALPHA

IF (THOLD) ,20,

DATA=THOLD

THOLD=0

GO TO TYPE

20 CALL LOAD(\$900,DATA,\$50,\$150,\$21)

21 IF (DATA.EQ.CHARBL) GO TO 20
 IF (ARG) GO TO 400

GO TO ALPHA

50 CONTINUE

IF (IDENT) GO TO 102
 NUMRL=.TRUE.

CALL BCDSTR(DATA)

GO TO 20

51 CONTINUE

IF (IDENT) GO TO 103

CALL BCDBIN(DATA)
 IF (ARG) GO TO 310
 RETURN

100 CONTINUE

IF (.NOT.NUMRL) GO TO 101

THOLD=DATA

ASSIGN 100 TO TYPE

```

: AND GO TAKE CARE OF NUMBER
:
: ... ELSE, SET SWITCH TO AVOID FMT'D CONVE
: SIGNAL OTHER ROUTINES TO COME
: SET ALPHABETIC INDEX
: HERE
: COUNT CHARACTER
: PACK NO MORE THAN 6
: LJZF CHARACTER
: PACK CHARACTER RJZF
: GO AND FETCH ANOTHER CHARACTER
: DELIMITER ENCOUNTERED
: RECOVER PACKED IDENTIFIER
: LJZF IF NECESSARY
: RETURN IDENT LGTH 36
:
: ... NON- SPECIAL CHARACTERS
: IT-CODE FOR IDENT DELIMITER
:
: ELSE SAVE FOR NEXT SCAN
: NUMBER BEING PROCESSED?
:
: FIRST DECIMAL IN CURRENT CONTEX
: SET EXTERNAL DECIMAL SWITCH
: FLAG CALLER
:
: GO TO 51
: ASSIGN 102 TO ALPHA
: IDENT=.TRUE;
: IND=3
: 102 CONTINUE
: AC=AC+1
: IF(6-AC) 20,,
: DATA=LSSL(DATA,30)
: CALL LDSL(ACCUM,DATA,6)
: GO TO 20
: 103 CONTINUE
: DATA=ACCUM
: IF(AC,LT,6) DATA=LSSC(ACCUM,36-6*AC)
: IF(AC,GT,6) ICT=AC
: RETURN
:
: 150 IF(AC) ,155,
: DO 151 I=1,2
: IF(DATA,NE,ITCODE(I)) GO TO 151
: IT=I
: GO TO 103
: 151 CONTINUE
: GO TO 160
: 155 IF(.NOT,NUMRL) GO TO 170
: IF(DATA,NE,CHARPD,OR,DPT) GO TO 160
: DPT=.TRUE,
: CALL BCDPNT
: IND=2
: GO TO 20
: 160 THOLD=DATA

```

```

:
: NEXT SCAN PROCESSES HUNG SC
: MEANWHILE, FINISH THIS SCAN
:
: ... DATA IS LONE SC NOT BELONGING TO IDENT OR
: NUMBER
: FORM SC-CODE
:
: FLAG CALLER FOR LONE SC
:
: ASTERISK?
:
: PERIOD? NO, RETURN
:
: CHECK TO SEE IF NEXT CHARACTER IS A
: NUMERAL
:
: ... YES, WE'VE FOUND A DECIMAL NO.
:
: FORGET PERIOD NOW & PROCESS N
:
: DATA IS A LONE '*'
: CHECK FOR POSSIBLE '**'
:
: ... NEXT WAS A NUMERAL
:
: ... '**' = CODE 14
:
: ... NEXT WAS NON-'*' SC
:
: ... NEXT WAS ALPHABETIC
:
: SAVE FORWARD CHARACTER
:
: AND GIVE BACK WHAT WE HAVE
:

```

```

ASSIGN 170 TO TYPE
GO TO 51

170 DO 175 I=1,27
    IF(DATA,NE,SPCHAR(I)) GO TO 175
    IT = OR (DATA,' ')
    DATA=I
    IND=4
    IF(DATA,EQ,13) GO TO 180
    IF (DATA ,EQ. 4) GO TO 300
    IF(DATA,NE,25) RETURN

CALL LOAD($900,ASTR,$171,$186,$190)

171 DPT=.TRUE.
    CALL BCPNT
    IND=2

DATA=ASTR
GO TO 50
175 CONTINUE
GO TO 20

180 CALL LOAD($900,ASTR,$181,$185,$190)
181 ASSIGN 50 TO TYPE
GO TO 200
185 IF(ASTR,NE,CHARAS) GO 1

DATA=14
IT = ' **'
RETURN

186 ASSIGN 150 TO TYPE
GO TO 200

190 ASSIGN 21 TO TYPE

200 THOLD=ASTR

RETURN

```


GSCAN USES SEVERAL SUBROUTINES TO PERFORM ITS FUNCTIONS. THESE ARE GROUPED IN THE PROGRAM ELEMENTS LOAD, BCD-BIN, AND SPL-LINK, DESCRIBED BELOW.

LOAD IS THE SUBPROGRAM THAT MAINTAINS THE STRING BEING SCANNED AND RETURNS EACH CHARACTER ENCOUNTERED. THE ROUTINE CALL

CALL LOAD (0,<LOCATION>,<SIZE>)

INITIALIZES THE PROGRAM WITH A STRING OF CHARACTERS STORED IN THE ARRAY <LOCATION> (AND HAS LENGTH <SIZE> WORDS). THEREAFTER EACH ROUTINE CALL

CALL LOAD(\$<A>,<DATA>,\$,\$<C>,\$<D>)

RETURNS THE NEXT CHARACTER IN THE STRING RIGHT-JUSTIFIED IN <DATA> AND BRANCHES TO

- <A> IF THE STRING HAS BEEN EXHAUSTED,
- IF <DATA> IS NUMERIC,
- <C> IF <DATA> IS A SPECIAL CHARACTER,
- <D> IF <DATA> IS ALPHABETIC.

FORMAL,LOAD

AXRS.

1108 - PROJECT FORMAL.
GENERAL OBJECT-TIME SCAN ROUTINE
MUST BE INITIALIZED ORIGINALLY AND AFTER STRING EXHAUSTION BY
LOAD(0,ARRAY LOC,ARRAY SIZE)
AND THEREAFTER:
CALL LOAD(S , DATA , S , S , S)
THRU WITH STRING..
DATA IS NUMERIC...
..DATA IS ALPHABETIC
..DATA IS SPECIAL CHARACTER
- UNTIL STRING EXHAUSTION
- R.A.N. - JULY '69

S(2) LIT .
PF FORM 12,6,18.
WD + 1,S-S.
+ 1,LTAB.
NUMB + 1,1.
+ 1,9.
ALPHA + 1,1.
+ 1,2.
FLAG + 7.
S(1).
LTABR LMJ A2,,A2.
LTAB L,S1 A0,,A1.
L,S2 A0,,A1.
L,S3 A0,,A1.
L,S4 A0,,A1.
L,S5 A0,,A1.
LMJ A2,S+1.
L,S6 A0,,A1.
TBENTY TLEM,U A1,S-S.
J LTABR.
J XAUST.
/.
LOAD* TNZ *0,11.
J INITL.
TZ FLAG.
J *0,11.
DL A1,WD.
EX 0,*A2.
DS A1,WD.
QUAL.
S A0,*1,11.
DL A1,NUMB.
TW A1,A0.
J QUALA.

WORD POSITION IN STRING
CHR POS LOAD INDEX
GREATEST LOWER BOUND-1, NUMERAL S
LEAST UPPER BOUND , NUMERALS
GLB-1, ALPHA CHARACTERS
LUB , ALPHA
INITIALIZATION SWITCH

CAPTURE LOCAL P; RETURN EXTERNAL

CAPTURE EXTERNAL P

INCREMENT & TEST FOR STRING EXHAU
STRING OK
THAT'S ALL FOLKS

IS THIS AN INITIALIZATION?

IF NOT, ARE WE INITIALIZEUP
TAKE EXHAUSTION RETURN
OK - LOAD INDICES
GRAB NEXT CHARACTER
RESTORE INUICES

RETURN THE GOODIES
TEST FOR NUMBER

	J	*2,11.	TAKE NUMBER RETURN
QUALA.	DL	A1,ALPHA.	TEST FOR ALPHABETIC
	TW	A1,A0.	
	J	*3,11.	SPECIAL CHARACTER
	J	*4,11.	ALPHABETIC
INITL.	INITIALIZATION ENTRY:		SET FLAG FOR INITIALIZATION
	SZ	FLAG.	
	L,U	A2,LTAB.	
	S,H2	A2,WD+1.	GET STRING ADDRESS
	L,H2	A0,1,11.	COMPUTE LAST WORD ADDRESS
	AU	A0,*2,11.	
	AN,U	A1,1.	STORE WORD INDEX
	S,H2	A0,WD.	STORE LAST WORD ADDRESS
	S,H2	A1,TBEMTY.	
	J	4,11.	
XAUST.	STRING EXHAUSTION, DE-INITIALIZE.		
	S	A1,FLAG.	RESTORE FLAG
	J	QUAL.	RETURN THIS TIME, BUT NOT NEXT
	END		

BCD-BIN ALLOWS A USER TO CONVERT A FIELD-DATA NUMBER TO AN INTEGER OR FLOATING POINT NUMBER. THE ROUTINE IS USED BY SENDING IT THE INTEGER PART OF A NUMBER, ONE FIELDATA DIGIT AT A TIME BEGINNING WITH THE MOST SIGNIFICANT DIGIT, AND THEN FOLLOWING IT WITH THE FRACTIONAL PART OF THE NUMBER IN LIKE MANNER. ONCE THE COMPLETE NUMBER HAS BEEN TRANSFERRED EITHER A SINGLE-PRECISION INTEGER OR A SINGLE-PRECISION FLOATING POINT NUMBER IS RETURNED.

TWO SUBPROGRAMS SET THE MODE OF INPUT:

CALL BCDNPT

SETS THE INTEGER MODE OF INPUT, WHILE

CALL BCDPNT

SETS THE FRACTIONAL MODE OF INPUT. THE DIGITS ARE TRANSFERRED, ONE AT A TIME, BY THE ROUTINE CALL

CALL BCDSTR(<DIGIT>)

WHERE <DIGIT> IS A SINGLE FIELDATA NUMBER RIGHT-JUSTIFIED. THE RESULT IS RETURNED IN <RESULT> BY THE ROUTINE CALL

CALL BCDBIN(<RESULT>)

WHERE <RESULT> IS INTEGER IF THE MODE WAS LAST SET BY BCDNPT AND FLOATING POINT IF THE MODE WAS LAST SET BY BCDPNT.

A MORE DETAILED DESCRIPTION IS GIVEN IN THE COMMENTS OF THE FOLLOWING LISTING.

	DL	A0,INTGR,X1.	LOAD ACCUM & SCALE BY 10
	LDSL	A0,2.	
	DA	A0,INTGR,X1.	
	LDSL	A0,1.	
	DA	A0,A2.	ADD NEW ENTRY
	JN	A0,OVRFLW.	IF UP CAN'T HACK IT, WE YIELD - V
	DS	A0,INTGR,X1.	
BODST1.			
	A	X1,FCT.	INCREMENT DECIMAL COUNT - IF ANY
	S	X1,FCT.	
	L	X1,('SAVEX1').	
	J	2,11.	
BODPNT*	L,U	A0,2.	SET FLOATING POINT MODE
	S	A0,SW.	
	J	1,11.	
BODNPT*	SZ	SW.	RESTORE INTEGER MODE
	J	1,11.	
BODBIN*	TNZ	SW.	
	J	FIXED.	INTEGER MODE
	L,U	A0,1084.	
	DSL	A4,72.	CLEAR DP FRAC ACCUM
	TNZ	FRAC.	HAVE WE A FRACTION?
	TZ	FRAC+1.	
	J	S+2.	
	J	DINT.	NO
	DFP	A0,FRAC.	YES - DFLOAT FRAC
	LN	A3,FCT.	LOAD SCALING FACTOR
	DFM	A1,LSPWF8,A3.	GOOD OL' COMOL TABLES SAVE THE DA
	DS	A1,A4.	
DINT.			
	DFP	A0,INTGR.	DFLOAT INTEGRAL
	DFA	A1,A4.	ADD FRAC
	DS	A1,DFETCH.	SAVE GOODIES FOR POSSIBLE FUTURE
	FCL	A0,A1.	MAKE SOW'S EAR FROM OUR SILK PUNKS
	JO	CPOVFL.	AND DIE IF THE PUT-DOWN WAS UNBREA
	J	RSTR.	OTHERWISE WE'RE DONE
FIXED.			
	TZ	INTGR.	DP-INTEGERS WE CAN'T HANDLE
	J	INOVFL.	
	L	A0,INTGR+1.	
/.			
RSTR.			
	S	A0,*0,11.	RETURN SOW'S EAR
	SZ	SW.	
	SZ	INTGR.	
	SZ	INTGR+1.	
	SZ	FRAC.	
	SZ	FRAC+1.	
	SZ	FCT.	
	J	2,11.	

GARBGE	ER NOP. NOP.	ERRs.
OVRFLW	ER NOP. NOP.	ERRs.
CPOVFL	ER NOP. NOP.	ERRs.
INOVFL	ER NOP. NOP.	ERRs.
DFETCH*	RES END.	2.

SPL-LINK PROVIDES FORTRAN LINKAGES TO THE MACHINE LANGUAGE SHIFT AND PARTIAL WORD INSTRUCTIONS. THE TABLE BELOW SUMMARIZES.

THE FUNCTION BELOW LOADS X WITH THE RESULTS OF A

X=ISSL(Y,K)	K-BIT SINGLE-SHIFT LOGICAL(RIGHT)
X=ISSA(Y,K)	K-BIT SINGLE-SHIFT ALGEBRAIC(RIGHT)
X=LSSL(Y,K)	K-BIT SINGLE-SHIFT LOGICAL(LEFT)
X=ISSC(Y,K)	K-BIT SINGLE-SHIFT CIRCULAR(RIGHT)
X=LSSC(Y,K)	K-BIT SINGLE-SHIFT CIRCULAR(LEFT)

OF Y.

THE ROUTINE CALL BELOW CAUSES THE CONTENTS OF X(1),X(2) TO BE DOUBLE SHIFTED K-BITS

CALL IDSL(X(1),X(2),K)	LOGICAL(RIGHT)
CALL IDSA(X(1),X(2),K)	ALGEBRAIC(RIGHT)
CALL LDSL(X(1),X(2),K)	LOGICAL(LEFT)
CALL IDSC(X(1),X(2),K)	CIRCULAR(RIGHT)
CALL LDSC(X(1),X(2),K)	CIRCULAR(LEFT)

IN THE FUNCTION CALLS BELOW X IS LOADED WITH THE

X=IRH(Y)	RIGHT HALF
X=LH(Y)	LEFT HALF
X=IRHX(Y)	RIGHT HALF, SIGN-EXTENDED
X=LHX(Y)	LEFT HALF, SIGN-EXTENDED
X=IT1X(Y)	BITS 0-11
X=IT2X(Y)	BITS 12-23
X=IT3X(Y)	BITS 24-35
X=IS1(Y)	BITS 0-5
X=IS2(Y)	BITS 6-11
X=IS3(Y)	BITS 12-17
X=IS4(Y)	BITS 18-23
X=IS5(Y)	BITS 24-29
X=IS6(Y)	BITS 30-35

OF Y.

THE LISTING FOR SPL-LINK IS SHOWN BELOW.

FORMAL.SPL-LINK

AXRS.

• • •
FORTRAN LINKAGES TO ASSEMBLER SHIFT INSTRUCTIONS
AND PARTIAL WORD LOADS.

\$ (1).		
COMMON	EQUF	INS-ISSA, A3.
EX	L	A0, *0, 11.
	L	A2, *1, 11.
	EX	COMMON.
	J	3, 11.
EX1	L	A0, *0, 11.
	L	A1, *1, 11.
	L	A2, *2, 11.
	EX	COMMON.
	S	A0, *0, 11.
	S	A1, *1, 11.
	J	4, 11.
ISSL*	LMJ	A3, EX.
ISSA*	LMJ	A3, EX.
LSSL*	LMJ	A3, EX.
ISSC*	LMJ	A3, EX.
LSSC*	LMJ	A3, EX.
IDSL*	LMJ	A3, EX1.
IDSA*	LMJ	A3, EX1.
LDL*	LMJ	A3, EX1.
IDSC*	LMJ	A3, EX1.
LDSC*	LMJ	A3, EX1.
INS	SSL	A0, , A2.
	SSA	A0, , A2.
	LSSL	A0, , A2.
	SSC	A0, , A2.
	LSSC	A0, , A2.
	DSL	A0, , A2.
	DSA	A0, , A2.
	LDL	A0, , A2.
	DSC	A0, , A2.
	LDSC	A0, , A2.

AND - FOR AN EXTRA ADDED ATTRACTION

IRH*	LMJ	A1,MNE.	RIGHT HALF
	L,H2	A0,*0,11.	
LH*	LMJ	A1,MNE.	LEFT HALF
	L,H1	A0,*0,11.	
IRHX*	LMJ	A1,MNE.	RIGHT HALF, EXTENDED
	L,XH2	A0,*0,11.	
LHX*	LMJ	A1,MNE.	LEFT HALF, EXTENDED
	L,XH1	A0,*0,11.	
IT3X*	LMJ	A1,MNE..	THIRD THIRD, EXTENDED

	L,T3	A0,*0,11.	
IT2X*	LMJ	A1,MNE.	SECOND THIRD, EXTENDED
	L,T2	A0,*0,11.	
IT1X*	LMJ	A1,MNE.	FIRST THIRD, EXTENDED
	L,T1	A0,*0,11.	
IS6*	LMJ	A1,MNE.	SIXTH SIXTH
	L,S6	A0,*0,11.	
IS5*	LMJ	A1,MNE.	FIFTH ''
	L,S5	A0,*0,11.	
IS4*	LMJ	A1,MNE.	FOURTH ''
	L,S4	A0,*0,11.	
IS3*	LMJ	A1,MNE.	THIRD ''
	L,S3	A0,*0,11.	
IS2*	LMJ	A1,MNE.	SECOND ''
	L,S2	A0,*0,11.	
IS1*	LMJ	A1,MNE.	FIRST ''
	L,S1	A0,*0,11.	
MNE	SZ	A3.	
	EX	0,A1.	
	J	2,11.	
	END.		

3.1.6. ERROR ROUTINE

THERE ARE TWO ROUTINES FOR WRITING ERROR MESSAGES, FMLERR AND ERRRET. FMLERR WRITES A MESSAGE ACCORDING TO A SPECIFIED ERROR INDEX

CALL FMLERR(\$<A>,<DATA>,<I>,<J>)

PRINTS MESSAGE <J>. IN ADDITION, THE WORD <DATA> IS PRINTED IN EITHER

I10 (<I>=1) OR
A6 (<I>=2) OR
O12 (<I>=3) OR
G12.5 (<I>=4) FORMAT.

IN ALL CASES THE ERROR PROGRAM RETURNS TO LOCATION <A>. THE OTHER ERROR MESSAGE ROUTINE, ERRRET, SIMPLY RETURNS THE ERROR MESSAGE

<<<<<ERROR RETURN TO FORMAL SUBROUTINE <NAME>

WHEN CALLED WITH THE SEQUENCE

CALL ERRRET(<NAME>).

THE TWO ERROR ROUTINES ARE LISTED BELOW.

FORMAL,FMLERR

FORMAL ERROR MESSAGE ROUTINE

THIS ROUTINE PROCESSES THE FORMAL ERROR DIAGNOSTIC
MESSAGES.'DATA' : IS THE INFORMATION TO BE INSERTED AT THE
END OF A DIAGNOSTIC MESSAGE.

'I' : INDICATES THE TYPE OF DATA.

0 FOR INTEGER

1 FOR ALPHA-NUMERIC

2 FOR OCTAL

3 FOR REAL

'J' : INDICATES THE PARTICULAR MESSAGE TO BE
PRINTED OUT.

INSERT THE INFORMATION AT THE END

PRINT IT OUT

RETURN

SUBROUTINE FMLERR(\$,DATA,I,J)

PARAMETER MSGCT = 29

INTEGER MSG(8,MSGCT),FMT(4)

DEFINE EMSG(N) = (MSG(I,N),I=1,7)

DATA EMSG(1)/(2X'ILLEGAL USE OF CHARACTER ',

DATA EMSG(2)/(2X'THE VARIABLE NAME IS TOO LONG ',

DATA EMSG(3)/(2X'ILLEGAL NUMBER OF ARGUMENTS ',

DATA EMSG(4)/(2X'DOES NOT MATCH TO THE LEFT SIDE ',

DATA EMSG(5)/(2X'TOO MANY SUBSCRIPTS ',

DATA EMSG(6)/(2X'ILLEGAL BASE TRANSFORMATION ON ',

DATA EMSG(7)/(2X'ILLEGAL REFERENCE OF VARIABLE ',

DATA EMSG(8)/(2X'DIMENSIONING IS REDEFINED IN ',

DATA EMSG(9)/(2X'ILLEGAL VARIABLE NAME ',

DATA EMSG(10)/(2X'ILLEGAL DELIMETER AS IS USED ',

DATA EMSG(11)/(2X'WARNING : OPTION NOT DEFINED ',

DATA EMSG(12)/(2X'ILLEGAL SPECIFICATION FOR ',

DATA EMSG(13)/(2X'ARGUMENT MUST BE POSITIVE INTEGER ',

DATA EMSG(14)/(2X'ARGUMENT INDEX IS OUT OF RANGE ',

DATA EMSG(15)/(2X'SUBSCRIPT VALUE IS OUT OF RANGE ',

DATA EMSG(16)/(2X'EXPONENTIATION OF NEGATIVE NUMBER ',

DATA EMSG(17)/(2X'ROLLIN/ROLLOUT ERROR ',

DATA EMSG(18)/(2X'IDENTIFIER IS ',

DATA EMSG(28)/(2X'SYNTAX ERROR IN FORTRAN ARGUMENT ',

DATA EMSG(29)/(2X'SUBSCRIPT MUST BE INTEGER '

DATA FMT /'I10) A6) 012) G12,5)'/

IF (J .GT. MSGCT) GO TO 10

MSG(8,J) = FMT(I+1)

PRINT MSG(1,J),DATA

RETURN 1

PRINT 20

10
20 FORMAT (1H0,5X,'ILLEGAL ERROR-MESSAGE INDEX')


```
RETURN 1  
END
```

```
FORMAL,ERRRET
```

```
*****  
ERROR ROUTINE TO PRINT SUBROUTINE NAME  
*****
```

```
SUBROUTINE ERRRET (NAME)
```

```
PRINT 10, NAME  
RETURN  
10  FORMAT ('1<<<<<< ERROR RETURN TO FORMAL SUBROUTINE ',A6)  
END
```

3.1.7. SYMBOL TABLE ROUTINES

THE SYMBOL TABLE ROUTINES MAINTAIN A TABLE OF SYMBOLIC NAMES AS A THREE LEVEL TREE: SUBPROGRAM NAME, VARIABLE NAME, AND SUBSCRIPT. THREE SUBPROGRAMS ARE USED TO ACCESS THIS TABLE: SYMBOL, SYMBOLFIND, AND SYMBOLSTOR. THE LISTINGS FOR THESE SUBPROGRAMS APPEAR BELOW. SUBPROGRAM SYMBOL ALLOWS A SYMBOL TO BE RETRIEVED, STORED, OR ERASED FROM THE TREE THROUGH TWO ENTRY POINTS, SYMBOL AND SYMBOL1. SUBPROGRAM SYMBOLFIND SEARCHES THE TABLE FOR A VARIABLE NAME (ENTRY FINDP1) OR SUBSCRIPT (ENTRY FINDP2). IF THE SYMBOL IS NOT FOUND, IT RETURNS A POINTER TO THE LOCATION AT WHICH IT SHOULD BE ENTERED. SUBPROGRAM SYMBOLSTOR STORES A SYMBOL AND/OR A SUBSCRIPT INTO THE TABLE THROUGH ENTRY STSYEN. FOLLOWING THE LISTINGS ARE TABLES DESCRIBING THE VARIABLE NAME AND SUBSCRIPT ENTRY FORMATS.

1378

FORMAL SYMBOL

SYMBOL TABLE ROUTINE

SYMBOL TREE TABLE

THE SYMBOL TREE TABLE OCCUPIES PART OF THE CHAINED C-D STORAGE.
THE TREE IS ARRANGED IN 3 LEVELS:

1. SUBPROGRAM
2. SYMBOL
3. SUBSCRIPT LEVELS.

POINTER 'NS' POINTS TO THE FIRST ENTRY ON THE SUBPROGRAM LEVEL.
NS=0 INDICATES AN EMPTY TREE TABLE.

1. THE SUBPROGRAM LEVEL CONSISTS OF ONE FORWARD CHAINED TABLE, WITH FIRST ENTRY IN NS, AND LAST FORWARD CHAIN IS ZERO. THE D-WORD OF EACH ENTRY CONTAINS THE ALPHANUMERIC NAME OF A USER'S SUBPROGRAM. BITS 0-5 OF THE C-WORDS ARE ZERO, BITS 6-20 OF THE C-WORDS ARE THE POINTERS TO THE FIRST ENTRY OF THE SYMBOL TABLE OF THE CORRESPONDING SUBPROGRAM. THE SUBPROGRAM TABLE IS CHAINED ALPHANUMERICALLY CORRESPONDING TO THE D-WORDS.
2. THE SYMBOL LEVEL CONSISTS OF A SET OF INDEPENDENT SYMBOL TABLES EACH CORRESPONDING TO A SUBPROGRAM. EACH SYMBOL TABLE IS ALPHANUMERICALLY CHAINED BY THE D-WORDS, AND ZERO LINK IS USED FOR THE END OF EACH SYMBOL TABLE. THE D-WORDS CONTAIN THE NAMES OF VARIABLES OR DEFINED FUNCTIONS. THE BITS IN C-WORD:

BITS 0-2 = 000 UNDEFINED LOCAL VARIABLE, MUST BE REFERENCED INDIRECTLY (BIT 4 ON)

= 001 NON-SUBSCRIPTED LOCAL VARIABLE

= 011 SUBSCRIPTED LOCAL VARIABLE

= 100 INDIRECT, BITS 6-20 POINT TO SYMBOL ENTRY

= 101 INDIRECT, BITS 6-20 POINT TO SUBSCRIPT ENTRY

BITS 4 = 1 IF THIS IS USED AS INDIRECTLY REFERENCED ENTRY

BITS 6-20 = POINTER TO

AN EXPRESSION IF BITS 0-2 = 001,

A SUBSCRIPT TABLE IF BITS 0-2 = 011,

AN ENTRY IN SYMBOL TABLE (POSSIBLY ON SUBSCRIPT LEVEL) IF BIT 4 = 1.

3. THE SUBSCRIPT LEVEL CONSISTS OF A SET OF INDEPENDENT SUBSCRIPT TABLES EACH CORRESPONDING TO A SYMBOL. EACH TABLE IS SORTED ACCORDING TO BITS 3-5 OF C-WORDS AND THE D-WORDS. THE LAST ENTRY OF A SUBSCRIPT LEVEL IS USED ONLY TO POINT BACK TO ITS SYMBOL. THE ENTRIES (C-D) ARE AS FOLLOWS:

BITS 0-2 OF C = 010 NORMAL SUBSCRIPTED VARIABLE WITH BITS 6-20 POINTING TO AN EXPRESSION

= 110 UNASSIGNED SUBSCRIPTED VARIABLE.

BITS 6-20 POINTS TO THE SYMBOL ENTRY.

= 111 END OF SUBSCRIPT TABLE, BITS 6-20 POINT TO THE SYMBOLIC NAME (LOOP CHAIN)

BITS 3-5 OF C = NUMBER OF SUBSCRIPTS

BITS 6-20 OF C = POINTER TO AN EXPRESSION OR SYMBOL.
THE D-WORD.

BITS 0-8 = VALUE OF THE FIRST SUBSCRIPT

9-17 = SECOND
18-26 = THIRD
27-35 = FOURTH

BITS 21-35 OF EVERY C-WORD IS USED FOR THE FORWARD CHAIN
OF EVERY TABLE, WITH ZERO INDICATING THE END.

SUBROUTINE SYMBOL (*,KK)
INCLUDE CMMN
INCLUDE PWORD
LOGICAL WTSBS

THIS IS A COMMON ENTRY FOR RETRIEVE, STORE AND ERASE
THE VALUE OF A VARIABLE OR FUNCTION.

K = 1 RETRIEVE VALUE
K = 2 STORE VALUE
K = 3 ERASE VALUE

N1 IS THE NAME OF THE VARIABLE
N2 IS THE NUMBER OF SUBSCRIPTS
N3 IS THE SUBSCRIPT-WORD

EPTR IS USED AS A POINTER TO THE EXPRESSION VALUE OF THE SYMBOL
TO BE RETRIEVED OR STORED. IT IS SET BY THIS ROUTINE WHEN
K = 1, AND IT IS ASSUMED TO BE SET BY THE CALLING ROUTINE
WHEN K = 2. WHEN THE SYMBOL IS NOT IN THE TABLE AND K = 1,
THEN EPTR IS SET TO ZERO.

WTSBS = .TRUE.
GO TO 11

SYMBL1 IS AN EQUIVALENT ENTRY TO SYMBOL, EXCEPT WHEN IT IS CALLED
TO RETRIEVE THE VALUE OF A VARIABLE WITHOUT SPECIFIED SUBSCRIPT,
AND THE VARIABLE IS IN THE SYMBOL TABLE WITH SUBSCRIPT. IT DOES
NOT MAKE AN ERROR RETURN, BUT IT SETS THE POINTER TO THE VALUE,
IE, TO ZERO AND A NORMAL RETURN OCCURS. THIS ENTRY IS USED WHEN
A FULL ARRAY MUST BE RETRIEVED AS IN THE CASE OF WRITE STATEMENT.

11 ENTRY SYMBL1 (*, KK)
WTSBS = .FALSE.
CONTINUE
IF (KK .EQ. 1) EPTR = 0
N1S = N1
IY=1
IF (N2.NE.0) IY=3

CALL FINDP1(\$500)

GO TO (100,200,300),KK

FIND SYMBOL

*
*--- SYMBOL FOUND, PROCESS BY K
* :
* :... RETRIEVE VALUE
* : RETURN EPTR AS THE POINTER TO THE EXPRESSION
* : VALUE , 0 IF IT IS NOT IN TABLE

```

* :
* : BRANCH ACCORDING TO TYPE
* :
* : ... RETRIEVE UNDEFINED
* : MUST BE TERMINAL, RETURN WITH ZERO
* :
* : ... RETRIEVE NONDIMENSIONED
* :
* : ... RETRIEVE DIMENSIONED, SEARCH FOR
* : SUBSCRIPT
* :
* : RETURN WITH ZERO IF THE FULL ARRAY
* : SHOULD BE RETURNED
* :
* : ... RETRIEVE - INDIRECT, REDEFINE SYMBOL
* : AND GO BACK
* :
* : ... RETRIEVE - INDIRECT, POINTS TO SUBSCRIPT
* :
* :
* : ... STORE VALUE
* :
* : BRANCH ACCORDINT TO TYPE
* :
* : ... STORE UNDEFINED
* :
* : ... STORE - NONDIMENSIONED
* :
* : ... STORE - DIMENSIONED
* : FIND SUBSCRIPTS

```

```

100 J=FLD(0,3,C(NSY))+1
    GO TO (110, 120, 600, 150, 160, 170, 600, 600), J

110 IF (FLD(4,1,C(NSY)) .EQ. 0) GO TO 31
    GO TO 10

120 IF (N2 .NE. 0) GO TO 30
    EPTR = LAST(NSY)
    RETURN

150 IF (N2 .EQ. 0 .AND. WTSBS) GO TO 30
    IF (N2 .EQ. 0) RETURN
    CALL FINDP2($10)
    IF (FLD(0,3,C(NSU)) .NE. 2) GO TO 10
155 EPTR = LAST(NSU)
    RETURN

160 NSY=LAST(NSY)
    N1=D(NSY)
    GO TO 100

170 NSU = LAST(NSY)
    IF (N2 .NE. 0) GO TO 31
    IF (FLD(0,3,C(NSU)) .EQ. 2) GO TO 155
    N1 = D(LAST(NSU))
    N2 = FLD(3,3,C(NSU))
    N3 = D(NSU)
    GO TO 10

200 J=FLD(0,3,C(NSY))+1
    GO TO (210, 230, 600, 250, 270, 280, 600, 600), J

210 IF (FLD(4,1,C(NSY)) .EQ. 0) GO TO 31
    FLD(0,3,C(NSY)) = IY
    IF (N2 .NE. 0) GO TO 520
    GO TO 528

230 IF (N2 .NE. 0) GO TO 30
    CALL FMLCHM ($20, LAST(NSY))
    GO TO 528

250 IF (N2 .EQ. 0) GO TO 30

```


[illegible]

```

380      I=NEXT(NSU)
        CALL FMLCHB ($20, NSU)
        IF (NSU1.EQ.0) GO TO 390
        NEXT(NSU1)=I
        RETURN
390      LAST(NSY)=I
        IF (FLD(0,3,C(I)).NE.7) RETURN
        CALL FMLCHB ($20, I)
        GO TO 325

400      NSU1=0
        NSU=LAST(NSY)
        IF (NSU .EQ. 0) GO TO 600
410      II=FLD(0,3,C(NSU))
        IF (II.EQ.7) GO TO 440
        IF (II .NE. 2 .AND. II .NE. 6) CALL ERRERR ($20, 410)
        IF (II.EQ.6) GO TO 430
        I=LAST(NSU)
        IF (I .NE. 0) CALL FMLCHM ($20, 2)
        IF (ITRM.EQ.1) GO TO 420
        I=NEXT(NSU)
        CALL FMLCHB ($20, NSU)
        LAST(NSY)=I
        NSU=I
        GO TO 410
420      FLD(0,3,C(NSU)) = 6
        LAST(NSU)=NSY
430      NSU=NEXT(NSU)
        GO TO 410

440      IF (ITRM.EQ.1) RETURN
        CALL FMLCHB ($20, NSU)
        GO TO 325

450      NSY=LAST(NSY)
        GO TO 300

460      ITRM = 1
        IF (N2 .NE. 0) GO TO 31
        NSU = LAST (NSY)
        IF (FLD(0,3,C(NSU)) .EQ. 6) GO TO 530
        NSU1 = NSU
465      NSU1 = NEXT (NSU1)
        IF (NSU1 .EQ. 0) GO TO 600
        IF (FLD(0,3,C(NSU1)) .NE. 7) GO TO 465
        NSY = LAST (NSU1)
        GO TO 371

500      GO TO (510,520,530),KK

```



```

*      :... RETRIEVE - VALUE : SET EPTR TO ZERO
*      :
*      :... STORE - STORE SYMBOL
*      :
*      :
*      :
*      :
*      :... ERASE - RETURN
*      :
RETURN WITH EPTR = 0
*
ERROR RETURN
*
*
*
*
END

```

```

510 GO TO 10
520 CALL STSYEN($20)
    IF (N2 .NE. 0) GO TO 529
528 LAST (NSY) = EPTR
    GO TO 530
527 FLD (0,3,C(NSU)) = 2
529 LAST (NSU) = EPTR

530 RETURN

10 RETURN

600 CALL ERRERR ($20,600)
30 CALL FMLERR ($20, N1S, 1, 8)
31 CALL ERRERR ($20, NSY)
20 RETURN 1

END

```


FORMAL,SYMBFIND

SYMBOL TABLE SEARCHING ROUTINE

FIND POINTERS IN ALL LEVELS

=====

FINDP1

LEVEL 2 SEARCH: IT IS ASSUMED THAT 'N1' CONTAINS THE BCD
NAME OF THE VARIABLE TO BE SEARCHED FOR. TWO DIFFERENT
EXITS MAY OCCUR FROM THE SEARCH:

1. 'N1' IS NOT IN THE TABEL: THEN
NSY1 IS THE POINTER TO THE SYMBOL ENTRY
WHICH SHOULD BE FOLLOWED BY 'N1'
2. 'N1' IS IN THE TABLE
NSY IS THE POINTER TO N1
NSY1 IS THE POINTER TO THE PRECEDING SYMBOL ENTRY

FINDP2

LEVEL 3 SEARCH: IT IS ASSUMED THAT 'NSY' IS THE POINTER TO THE
SYMBOL WHICH IS SUBSCRIPTED, 'N2' CONTAINS THE NUMBER OF
SUBSCRIPT, 'N3' CONTAINS THE SUBSCRIPT WORD TO BE SEARCHED
FOR. THERE ARE 2 EXITS:

1. SUBSCRIPT WORD WAS NOT FOUND, THEN
NSU1 IS THE POINTER WHICH SHOULD FOLLOWED BY N2-N3
2. SUBSCRIPT WORD WAS FOUND, THEN
NSU IS THE POINTER TO N2-N3
NSU1 IS THE POINTER PRECEDING NSU
NSU1=0 INDICATES NO PRECEDING ENTRY IN LEVEL 3.

IN ANY CASE, IT IS ASSUMED THAT FMLINT WAS CALLED
(NSB POINTS TO THE PROGRAM ENTRY IN THE SYMBOL TABLE)
RETURN 1 OCCURS WHEN THE SYMBOL WAS NOT FOUND IN THE TREE TABLE

FINOP1(\$)

SUBROUTINE FINDP1(\$)

INCLUDE CMMN

INCLUDE PWORD

NSY1=0

NSU = 0

NSU1 = 0

LEVEL 2 SEARCH

·:
·:
·:
LOOP TO FIND 'N1' IN THE TABLE

*
*--- NOT FOUND, CONTINUE TO SEARCH

*
·:
*
·:
*

*--- NOT IN THE TABLE, EXIT

50 NSY=LAST(NSB)
IF (NSY.EQ.0) GO TO 170

IF (N1-D(NSY)) 41,160,60

60 NSY1=NSY
NSY=NEXT(NSY)
GO TO 50

```

41      NSY = 0
        GO TO 170

        ENTRY FINDP2($)
        NSU1=0
        NSU=LAST(NSY)
        IF (NSU.EQ.0) GO TO 170

80      IF (N2-FLD(3,3,C(NSU))) 105,90,100

90      IF (N3-D(NSU)) 105,160,100

100     IF (FLD(0,3,C(NSU)),EQ,7) GO TO 105
        NSU1=NSU
        NSU=NEXT(NSU)
        GO TO 80

105     NSU=0
        GO TO 170

160     RETURN

170     RETURN 1
        END

```

FORMAL.SYMBSTOR

 ROUTINE TO STORE SYMBOL ENTRIES

 STORE ENTRIES IN SYMBOL TABLE

NSB = POINTER TO SUBROUTINE NAME
 N1 = NAME OF THE SYMBOL
 NSY = ENTRY OF N1, IF ZERO, SYMBOL SHOULD BE STORED AFTER
 NSY1
 IY = TYPE OF SYMBOL
 IY1 = VARIABLE OR FUNCTION TYPE
 N2=NUMBER OF SUBSCRIPTS, ZERO=NON-SUBSCRIPTED WITH 010 TYPE
 N3=SUBSCRIPT WORD WHEN N2 IS NOT ZERO

IS THE SYMBOL STORED ALREADY IN LEVEL 2

... NO

... STORE N1 IN LEVEL 2 AS FIRST ENTRY

... STORE N1 IN LEVEL 2 FOLLOWING NSY1

... YES

LEVEL 2 IS FINISHED, CHECK LEVEL 3 IF ANY
 IF NOT, RETURN

YES, THERE ARE LEVEL 3, THUS STORE N2-N3

... AS FIRST ENTRY

SUBROUTINE STSYEN(\$)
 INCLUDE CMMN
 INCLUDE PWORD

IF (NSY,NE,0) GO TO 300

NSY = FMLCH1 (\$170)
 IF (NSY1,NE,0) GO TO 290

NEXT(NSY)=LAST(NSB)
 LAST(NSB)=NSY
 GO TO 280

290 CONTINUE
 NEXT(NSY)=NEXT(NSY1)
 NEXT(NSY1)=NSY
 280 D(NSY) = N1
 ITPB(NSY) = IY,LS,3
 NSU = 0
 NSU1 = 0

300 IF ((N2,EQ,0).OR,(NSU,NE,0)) RETURN

NSU = FMLCH1 (\$170)
 IF (NSU1,NE,0) GO TO 310

NEXT(NSU)=LAST(NSY)
 LAST(NSY)=NSU
 IF (NEXT(NSU),NE,0) GO TO 320
 KT = FMLCH1 (\$170)
 NEXT(NSU)=KT
 C(KT) = 70K10
 D(KT) = 0

RETURN
ERROR RETURN
... AS ENTRY FOLLOWING NSU1

LAST(KT)=NSY
GO TO 320
310 CONTINUE
NEXT(NSU)=NEXT(NSU1)
NEXT(NSU1)=NSU
320 ITPB(NSU) = N2 + 16
D(NSU)=N3
RETURN
170 RETURN 1
END

SYMBOL ENTRY

D(I) - ALPHABETIC NAME OF THE SYMBOL

```

*****
* BIT POSITION * ITS VALUE * DESCRIPTION *
* IN C(I) * *
*****
* 0 - 2 * 000 * UNDEFINED LOCAL SYMBOL *
* *-----*
* * 001 * NON-SUBSCRIPTED VARIABLE OR *
* * * DEFINED FUNCTION *
* *-----*
* * 011 * SUBSCRIPTED VARIABLE OR *
* * * DEFINED FUNCTION *
* *-----*
* * 100 * INDIRECT REFERENCE, BITS *
* * * 6-20 POINT TO SYMBOL ENTRY *
* *-----*
* * 101 * INDIRECT REFERENCE, BITS *
* * * 6-20 POINT TO SUBSCRIPT *
*-----*
* 4 * 0 * LOCAL VARIABLE OR DEFINED *
* * * FUNCTION *
* *-----*
* * 1 * THE SYMBOL IS REFERENCED *
* * * INDIRECTLY FROM ANOTHER *
* * * SUBPROGRAM *
*-----*
* 6 - 20 * K * POINTER TO C(K)-D(K) WHICH *
* * * IS: *
* * * (1) A SYMBOL OR SUBSCRIPT *
* * * ENTRY IF BITS(0-2)=100 *
* * * (2) EXPRESSION ENTRY *
* * * IF BITS(0-2)=001 *
* * * (3) A SUBSCRIPT ENTRY *
* * * IF BITS(0-2)=011 *
*-----*
* 21-35 * J * POINTER TO NEXT SYMBOL ENTRY *
* * * C(J)-D(J) *
* * * FOR LAST SYMBOL, J=0 *
*****

```

SUBSCRIPT ENTRY

D(I)=SUBSCRIPT WORD
BITS 0-8 VALUE OF FIRST SUBSCRIPT
BITS 9-17 VALUE OF SECOND SUBSCRIPT
BITS 18-26 VALUE OF THIRD SUBSCRIPT
BITS 27-35 VALUE OF FOURTH SUBSCRIPT

* BIT POSITION * ITS VALUE * DESCRIPTION
* IN C(I) *

* 0-2 * 010 * BITS 6-20 POINT TO EXPRESSION
* * * C(K)-D(K)

* UNASSIGNED SUBSCRIPTED * 110 *
* VARIABLE , BITS 6-20 POINTS *
* BACK TO ITS SYMBOL ENTRY, *
* WHICH SHOULD BE AN INDIRECTLY *
* REFERENCED SYMBOL *

* DUMMY ENTRY TO INDICATE THE * 111 *
* END OF A SUBSCRIPT ENTRY LIST *
* BITS 6-20 POINT BACK TO ITS *
* SYMBOL ENTRY, *

* 3-5 * 001 * NUMBER OF SUBSCRIPTS: 1,2,3,4
* * 010 *
* * 011 *
* * 100 *

* 6-20 * K * POINTER TO C(K)-D(K) WHICH IS
* * * (1) EXPRESSION ENTRY
* * * IF BITS 0-2 = 010
* * * (2) ITS SYMBOL ENTRY
* * * IF BITS 0-2 = 110
* * * 111

* 21-35 * J * POINTER TO NEXT SUBSCRIPT ENTRY
* * * C(J)-D(J)
* * * FOR LAST ENTRY, J=0

3.1.8. STRING AND ARRAY UTILITY ROUTINES

FIVE UTILITY ROUTINES ARE COLLECTED IN SUBPROGRAM PACK TO FACILITATE STRING AND ARRAY HANDLING. THE CALLING SEQUENCES AND DESCRIPTIONS ARE LISTED BELOW.

(1) CALL PACK(<N>,<ARRAY>,<STRING>)

THIS ROUTINE PACKS AN ARRAY <ARRAY> OF <N> SINGLE CHARACTERS INTO ARRAY <STRING>;

(2) CALL UNPACK(<N>,<STRING>,<ARRAY>)

THIS ROUTINE UNPACKS AN <N> CHARACTER STRING IN ARRAY <STRING> INTO AN ARRAY OF SINGLE CHARACTERS <ARRAY>;

(3) CALL MOVE(<N>,<ARRAY1>,<ARRAY2>)

THIS ROUTINE TRANSFERS THE <N> WORDS IN ARRAY <ARRAY1> TO ARRAY <ARRAY2>;

(4) CALL STRCON(<N>,<VALUE>,<ARRAY>)

THIS ROUTINE LOADS THE <N> LOCATIONS OF ARRAY <ARRAY> WITH THE VALUE <VALUE>;

(5) I=ISRCH(<N>,<ARRAY>,<VALUE>,<NOFIND>)

THIS FUNCTION SEARCHES THE <N> WORDS OF ARRAY <ARRAY> FOR <VALUE> AND LOADS I WITH THE ARRAY LOCATION OF THE FIRST OCCURENCE. IT TRANSFERS TO <NOFIND> IF <VALUE> IS NOT FOUND.

THE LISTINGS APPEAR BELOW.

FORMAL.PACK
 UTILITY PROGRAM TO. (UN)PACK CHARACTERS
 MOVE WORDS
 STORE CONSTANTS INTO ARRAYS

CALL PACK (N, FROM, TO)
 CALL UNPACK (N, FROM, TO)
 CALL STRCON (N, FROM, TO)
 CALL MOVE (N, FROM, TO)
 I = ISRCH(N, ARRAY, TARGET, \$NOFIND)

AXRS

P	PROC	
GET*	NAME	A0
STORE*	NAME	A2
	EX	0,*P(0,0)
	END	
\$ (2)		
	L,S6	A4,,*A1
	LMJ	A0,,A0
GETTBL*	L,S1	A4,,A1
	L,S2	A4,,A1
	L,S3	A4,,A1
	L,S4	A4,,A1
	L,S5	A4,,A1
	LMJ	A0,GETTBL-2
	S,S6	A4,,*A3
	LMJ	A2,,A2
STRTBL*	S,S1	A4,,A3
	S,S2	A4,,A3
	S,S3	A4,,A3
	S,S4	A4,,A3
	S,S5	A4,,A3
	LMJ	A2,STRTBL-2
GETWD*	+	1,GETTBL
STRWD*	+	1,STRTBL
K10	+	1,0
\$ (1)		
GETARG		
	L	A1,K10
	L	A3,A1
	L	R1,*0,X11
	A,U	A1,*1,X11
	A,U	A3,*2,X11
	J	0,A0
PACK*		
	LMJ	A0,GETARG
	L	A2,STRWD
	J	S+3
	L	A4,,*A1
	STORE	
JGD		R1,S-2


```
J      4,X11
UNPACK*
LMJ    A0,GETARG
L      A0,GETWD
J      $+3
GET
S      A4,,*A3
JGD    R1,$-2
J      4,X11
STRCON*
LMJ    A0,GETARG
L      A1,,A1
L,U    A2,A1
BT     A3,,*A2
J      4,X11
MOVE*
LMJ    A0,GETARG
BT     A3,,*A1
J      4,X11
ISRCH*
LMJ    A0,GETARG
L      A3,,A3
A,U    A1,0400000
S,H2   A1,$+2
L      A1,K10
SE     A3,,*A1
J      3,X11
L,U    A0,,A1
J      5,X11
END
```

3.1.9. CONSTANT ARITHMETIC OPERATION ROUTINES

THE FUNCTION SUBPROGRAM CONSOP PERFORMS ARITHMETIC OPERATIONS WITH FORMAL CONSTANTS. THE CALLING SEQUENCE

```
CALL CONSOP($<INT>,$<RAT>,$<REAL>,$<ERR>,<IOP>,<ARG1>,<ARG2>,<RESULT>)
```

PERFORMS THE OPERATION <IOP> ON THE ARGUMENTS POINTED TO BY <ARG1> AND <ARG2> AND STORES THE RESULTS IN THE LOCATION POINTED TO BY <RESULT> AND RETURNS

NORMALLY	IF THE RESULT IS ZERO,
TO <INT>	IF THE RESULT INTEGER,
TO <RAT>	IF THE RESULT IS RATIONAL,
TO <REAL>	IF THE RESULT IS FLOATING POINT,
TO <ERR>	IF ANY ERROR OCCURS (E.G., OVERFLOW),

THE SUBPROGRAM USES FOUR FUNCTIONS: EUCLID, INTADD, INTMUL, AND INTEXP. THESE FUNCTIONS ARE LOCATED IN SUBPROGRAM INTEGER AND ARE DESCRIBED BELOW.

I=EUCLID(\$<NONE>,<J>,<K>)	RETURNS THE VALUE OF THE GREATEST COMMON DENOMINATOR OF <J> AND <K>, IF ONE EXISTS (>1), ELSE RETURNS TO <NONE>.
I=INTADD(\$<OV>,<J>,<K>)	RETURNS SUM OF <J> AND <K>.
I=INTMUL(\$<OV>,<J>,<K>)	RETURNS PRODUCT OF <J> AND <K>.
I=INTEXP(\$<OV>,<J>,<K>)	RETURNS <J> ** <K>.
	(IN THE OCCURENCE OF AN OVERFLOW THE ROUTINE RETURNS CONTROL TO <OV>.)

OVERFLOW WITH INTEGER OPERATIONS CAUSES THE CONVERSION OF INTEGERS TO REALS, OVERFLOW WITH REAL OPERATIONS GIVES ERROR RETURN.

THE LISTINGS APPEAR BELOW.

FORMAL,CONSOP

FUNCTION CONSOP (*, *, *, *, IOP, I1, I2, I3)

ARITHMETIC OPERATIONS WITH FORMAL CONSTANTS

I1, IOP, I2 ==> I3

WHERE I1 AND I2 ARE THE POINTERS FOR THE INPUT
CONSTANTS, I3 IS THE POINTER TO THE RESULT AND IOP IS

1, 2 OR 3 FOR +, * OR **, RESPECTIVELY,

THE RETURNS ARE AS FOLLOWS:

RETURN 0 = RESULT IS ZERO

RETURN 1 = NON ZERO INTEGER RESULT

RETURN 2 = RATIONAL RESULT

RETURN 3 = REAL RESULT

RETURN 4 = ERROR

THE PROGRAM USES INTEGER ARITHMETIC ROUTINES

INTADD, INTMUL, INTEXP WITH ERROR RETURN

IN CASE OF OVERFLOW, FUNCTION EUCLID

GIVES THE COMMON DIVISOR OF TWO INTEGERS.

INCLUDE CMMN
INCLUDE PWORD
REAL FI,FJ,FK
EQUIVALENCE (CONSOP,K)
EQUIVALENCE (II,FI), (JJ,FJ), (KK,FK)
LOGICAL RAT,SIGN

INITIALIZE
RESULT AS ZERO

KF=1
KFF=1
K=IGETF1(\$990)
I3=K
II=1
JJ=1
IO=0
JO=0

GET TYPES OF INPUT CONSTANTS AS
1, 2 OR 3 FOR INTEGER, RATIONAL OR REAL

IF (I1 .EQ. 0) GO TO 1
II=D(I1)
IO=ITYP(I1)
K1=NEXT(I1)
1 IF (I2 .EQ. 0) GO TO 2
JJ=D(I2)
JO=ITYP(I2)
K2=NEXT(I2)
2 IF (IO .LT. 2) IO=IO+1
IF (JO .LT. 2) JO=JO+1
III=II
JJJ=JJ

DEFINE IJMP FOR BRANCHING BY TYPES OF INPUT:

1 = INT-INT, 4 = RAT-INT, 7 = REAL-INT
 2 = INT-RAT, 5 = RAT-RAT, 8 = REAL-RAT
 3 = INT-REAL, 6 = RAT-REAL, 9 = REAL-REAL

BRANCH BY OPERATOR

SPECIAL RESULTS

... RESULT IS ONE

... RESULT IS I1

... RESULT IS I2

... ERROR RETURN

... GET REAL EQUIVALENCE OF INPUTS
 CONVERT I1 TO REAL

CONVERT I2 TO REAL

--- ADDITION

BRANCH BY TYPES

IJMP=JO+3*(IO-1)

GO TO (100,300,500),IOP

20 D(K)=1
 RETURN 1

30 IF (I1 .EQ. 0) RETURN
 D(K)=I1
 IF (I1 .EQ. 0) RETURN 1
 ITYP(K)=ITYP(I1)
 IF (IO .NE. 2) GO TO 35
 KK=ILINK1(K,2,D(K1))
 35 RETURN IO

40 IF (JJ .EQ. 0) RETURN
 D(K)=JJ
 IF (I2 .EQ. 0) RETURN 1
 ITYP(K)=ITYP(I2)
 IF (JO .NE. 2) GO TO 45
 KK=ILINK1(K,2,D(K2))
 45 RETURN JO

990 CALL FMLERR (\$991, I1, 2, 16)
 991 RETURN 4

50 IF (IO .EQ. 3) GO TO 55
 FI=I1
 IF (IO .NE. 2) GO TO 55
 FK=D(K1)
 FI=FI/FK
 55 GO TO (132,352,552),IOP

60 IF (JO .EQ. 3) GO TO 65
 FJ=JJ
 IF (JO .NE. 2) GO TO 65
 FK=D(K2)
 FJ=FJ/FK
 65 GO TO (134,354,553),IOP

100 IF (I1 .EQ. 0) GO TO 40
 IF (JJ .EQ. 0) GO TO 30

GO TO (110,120,130,140,150,130,130,130,130),IJMP

```

110      KK=INTADD($130,II,JJ)
        IF (KK .EQ. 0) RETURN
        D(K)=KK
        RETURN 1

120      KF=EUCLID($125,II,JJ)
        III=II/KF
        JJJ=JJ/KF
        KK=INTMUL($130,III,D(K2))
        KK=INTADD($130,KK,JJJ)
        KK=INTMUL($130,KK,KF)
        GO TO 126

125      KK=INTMUL($130,II,D(K2))
        KK=INTADD($130,KK,JJ)

126      ITYP(K)=1
        D(K)=KK
        KK=ILINK1(K,2,D(K2))
        RETURN 2

140      KF=EUCLID($145,II,JJ)
        III=II/KF
        JJJ=JJ/KF
        KK=INTMUL($130,JJJ,D(K1))
        KK=INTADD($130,KK,III)
        KK=INTMUL($130,KK,KF)
        GO TO 146

145      KK=INTMUL($130,JJ,D(K1))
        KK=INTADD($130,KK,II)

146      ITYP(K)=1
        D(K)=KK
        KK=ILINK1(K,2,D(K1))
        RETURN 2

150      IIII=D(K1)
        JJJJ=D(K2)
        KF=EUCLID($152,II,JJ)
        III=III/KF
        JJJ=JJJ/KF

152      KFF=EUCLID($154,IIII,JJJJ)
        IIII=IIII/KFF
        JJJJ=JJJJ/KFF

154      KK=INTMUL($130,III,JJJJ)
        KKK=INTMUL($130,JJJ,IIII)
        KK=INTADD($130,KK,KKK)
        KKK=INTMUL($130,IIII,JJJJ)
        IF (KK .EQ. 0) RETURN
        IF (KFF .EQ. 1) GO TO 156
        KF1=EUCLID($156,KFF,KK)
        KFF=KFF/KF1

```

```

.... INTEGER + REAL
.... RATIONAL + REAL
.... REAL + INTEGER
.... REAL + RATIONAL
.... REAL + REAL

```

```

RETURN FROM CONVERSIONS

```

MULTIPLICATION

```

BRANCH BY TYPES:

```

```

.... INTEGER * INTEGER
.... INTEGER * RATIONAL

```

```

.... RATIONAL * INTEGER

```

```

.... RATIONAL * RATIONAL

```

```

KK=KK/KF1
156 IF (KF .NE. 1) KK=INTMUL($130, KK, KF)
IF (KFF .NE. 1) KKK=INTMUL($130, KKK, KFF)
D(K)=KK
IF (KKK .EQ. 1) RETURN 1
ITYP(K)=1
KK=ILINK1(K, 2, KKK)
RETURN 2

```

```

130 GO TO 50
132 GO TO 60

```

```

134 D(K)=BOOL(FI+5J)
IF (D(K) .EQ. 0) RETURN
ITYP(K)=3
RETURN 3

```

```

300 IF ((II .EQ. 0) .OR. (JJ .EQ. 0)) RETURN
GO TO (310, 320, 350, 330, 340, 350, 350, 350, 350), IJMP

```

```

310 D(K)=INTMUL($350, II, JJ)
RETURN 1

```

```

320 IF (II .EQ. 1) GO TO 40
KF=EUCLID($322, II, D(K2))
III=II/KF
KK=D(K2)/KF
GO TO 324

```

```

322 KK=D(K2)
324 D(K)=INTMUL($350, III, JJJ)
IF (KK .EQ. 1) RETURN 1
ITYP(K)=1
KF=ILINK1(K, 2, KK)
RETURN 2

```

```

330 IF (JJ .EQ. 1) GO TO 30
KF=EUCLID($332, JJ, D(K1))
JJJ=JJ/KF
KK=D(K1)/KF
GO TO 324
332 KK=D(K1)
GO TO 324

```

```

340 IIII=D(K1)
JJJJ=D(K2)

```



```

... INTEGER ** RATIONAL
... RATIONAL ** RATIONAL
... REAL ** RATIONAL
      IF THE SIGN OF THE BASE IS NEGATIVE
      AND THE DENOMINATOR OF THE EXPONENT
      IS EVEN, THEN THE RESULT IS
      IMAGINARY, THUS GIVE ERROR RETURN. IF
      THE DENOMINATOR IS ODD THEN THE SIGN
      OF THE RESULT IS MINUS WHEN THE
      NUMERATOR IS ALSO ODD, OTHERWISE IT
      WILL BE POSITIVE
... RATIONAL ** INTEGER
... INTEGER ** REAL
... RATIONAL ** REAL
... REAL ** REAL
... REAL ** INTEGER
      IF (RAT) GO TO 511
      D(K)=KK
      RETURN 1
520   IF (II .EQ. 1) GO TO 20
      IF (II .NE. -1) GO TO 525
      IF (MOD(D(K2),2) .EQ. 0) GO TO 990
521   IF (MOD(JJ,2) .EQ. 0) GO TO 20
      GO TO 30
525   IF (.NOT. SIGN) GO TO 551
      IF (MOD(JJ, 2) .EQ. 0) SIGN = .FALSE;
      IF (MOD(D(K2),2) .EQ. 1) GO TO 551
      GO TO 990
530   IF (JJ .NE. 1) GO TO 532
      IF (.NOT. RAT) GO TO 30
      D(K)=D(K1)
      IF (SIGN) D(K) = - D(K)
      IF (ABS(II) .EQ. 1) RETURN 1
      ITYP(K)=1
      KF=ILINK1(K,2,IABS(II))
      RETURN 2
532   IF (IABS(II) .EQ. 1) III=1
      IF (IABS(II) .NE. 1) III = INTEXP ($550, IABS(II), JJ)
      JJJ = INTEXP ($550, D(K1), JJ)
      IF (.NOT. RAT) GO TO 534
      KF=III
      III=JJJ
      JJJ=KF
534   IF (SIGN .AND. (MOD(JJ,2) .EQ. 1)) III=-III
      D(K)=III
      IF (JJJ .EQ. 1) RETURN 1
      ITYP(K)=1
      KF=ILINK1(K,2,JJJ)
      RETURN 2
550   IF (SIGN) GO TO 990

```


END

END

FORMAL, INTEGER

INTEGER ARITHMETIC OPERATIONS

I = EUCLID (S., J, K)

GIVES LARGEST COMMON FACTOR

RETURN 1 IF NO COMMON FACTOR

I = INTADD (S., J, K)

I = INTMUL (S., J, K)

I = INTEXP (S., J, K)

SETS I TO J + K, J * K, AND J ** K RESPECTIVELY

RETURN 1 IF OVERFLOW OCCURS

HANS, MARCH 24, 1970

```

S(1)  AXRS
EUCLID*
      LM      A1,*1,X11
      LM      A2,*2,X11
      TG      A2,A1
LOOP   DSC     A1,36
      SZ      A0
      DI      A0,A2
      TG,U    A1,2
      J       LOOP
      JNZ     A1,0,X11
      L       A0,A2
      J       4,X11
INTADD*
      L       A0,*1,X11
      A       A0,*2,X11
      JNO     4,X11
      J       0,X11
INTMUL*
      L       A0,*1,X11
      MI      A0,*2,X11
      JNZ     A0,0,X11
      XOR     A1,A0
      JN      A2,0,X11
      L       A0,A1
      J       4,X11
INTEXP*
      L       A3,*1,X11
      L       A4,*2,X11
      AN,U    A4,1
      L       A1,A3
      J       ENDL
LOOPE  MI      A0,A3
      JNZ     A0,0,X11
      XOR     A1,A0
      JN      A2,0,X11
ENDL   L       A0,A1
      JGD     A4,LOOPE
      J       4,X11
      END

```

3.1.10. VARIABLE LIST HANDLING ROUTINES

THE LIST OF VARIABLES IN GSCAN'S ARRAY ARE HANDLED BY TWO ROUTINES, FMLSCN AND GETARG. FMLSCN PLACES THE LIST OF VARIABLES INTO A CHAIN AND RETURNS THE CHAIN'S LOCATION (IN COMMON VARIABLE NK). GETARG IS USED TO FETCH THE NEXT VARIABLE FROM THE CHAIN EACH TIME ONE IS DESIRED. THE ROUTINE CALL

CALL GETARG

RETURNS THE NAME OF THE VARIABLE IN N1, THE VARIABLE'S TYPE IN N2, AND ITS SUBSCRIPT POINTER IN N3.

THE LISTINGS FOLLOW:


```

:
:
END
200 IF (NK .NE. 0) CALL FMLCHD ($201, NK)
201 RETURN 1
END

```

FORMAL,GETARG

```

*****
ROUTINE TO OBTAIN THE FIRST VARIABLE FROM THE LIST
*****
SUBROUTINE TO PUT THE FIRST ARGUMENT FROM THE CHAINED LIST,
STARTING AT NK, INTO N1, N2, N3.

```

GET THE ARGUMENT IDENTIFIER

TEST WHETHER IT IS SUBSCRIPTED VARIABLE

YES, GET THE SUBSCRIPTS

NO, FREE THE CELL

RETURN

```

SUBROUTINE GETARG
INCLUDE CMMN
INCLUDE PWORD

```

```

N1 = D(NK)
N2 = ITYP(NK) - 6

```

IF (N2 .EQ. 0) GO TO 10

```

NL = NEXT(NK)
N3 = D(NL)
N2 = ITYP(NL)

```

10 NK = FMLCHC (\$999, NK)

```

RETURN
999 N1 = 0
N2 = 0
RETURN
END

```

3-90

3.2. EXPRESSION TRANSLATION FROM INFIX TO PREFIX

SYMBOLIC EXPRESSIONS ARE STORED WITH THEIR PREFIX FORM IN THE COMMON BLOCK STORAGES, FMLCM1 AND FMLCM2 (ARRAYS C AND D). EACH PAIR OF WORDS, C AND D, CORRESPONDS TO AN ITEM, SUCH THAT THE LAST 15 BITS OF THE C-WORD CONTAIN THE LOCATION OF THE NEXT ITEM, THE PRECEDING 15 BITS CONTAIN THE LOCATION OF THE PRECEDING ITEM. THE ITEM ITSELF IS DESCRIBED BY THE FIRST 6 BITS OF THE C-WORD (ITYP) AND BY THE D-WORD. TABLE \$TS GIVES THE SUMMARY OF THIS REPRESENTATION. THE ORDER OF ITYP IS ARRANGED SUCH THAT IT CORRESPONDS TO THE OPERATOR PRECEDENCE.

INTERNAL REPRESENTATION OF EXPRESSIONS.

```

*****
*ITYP*      D(I)      *      REPRESENTS      *
*****
*  0 *      INTEGER    *  A SIGNED INTEGER AS CONSTANT  *
*-----*-----*-----*
*  1 *      INTEGER    *  A SIGNED INTEGER AS THE NUMERAT.*
*      *              *  OF A RATIONAL CONSTANT.      *
*-----*-----*-----*
*  2 * POSITIVE INTEGER *  A POSITIVE NON-ZERO INTEGER AS *
*      *              *  THE DENOMINATOR OF A RATIONAL *
*      *              *  CONSTANT, *
*      *              *  NUMERATOR AND DENOMINATOR ARE *
*      *              *  ALWAYS APPEARING CONSECUTIVELY. *
*-----*-----*-----*
*  3 * FLOATING-POINT NUMBER *  A REAL CONSTANT. *
*-----*-----*-----*
*  5 * POSITIVE INTEGER=I *  I'ITH ARGUMENT OF A FUNCTION *
*      *              *  DEFINED BY AN EXPRESSION. *
*-----*-----*-----*
*  6 * ALPHANUMERIC NAME *  NON-SUBSCRIPTED VARIABLE WHOSE *
*      *              *  VALUE IS ITS NAME, D(I). *
*-----*-----*-----*
*  7 * ALPHANUMERIC NAME *  SUBSCRIPTED VARIABLE WHOSE VALUE*
*      *              *  IS ITS NAME, D(I), WITH ITS *
*      *              *  SUBSCRIPTS DEFINED IN THE NEXT *
*      *              *  TERM, *
*-----*-----*-----*
*  8 * INTEGER IN FIELD 0-8 *  SUBSCRIPT WORD WITH 1 SUBSCRIPT *
*      *      FIELD 9-35 = 0 * *
*-----*-----*-----*
*  9 * INTEGER FIELDS 0-8 *  SUBSCRIPT WORD WITH 2 SUBSCRIPTS*
*      * AND 9-17, * *
*      *      FIELD 18-26 = 0 * *
*-----*-----*-----*
* 10 * INTEGER FIELDS 0-8, *  SUBSCRIPT WORD WITH 3 SUBSCRIPTS*
*      * 9-17 AND 18-26, * *
*      *      FIELD 27-35 = 0 * *
*-----*-----*-----*

```


EXPRESSION REPRESENTATION (CONTINUED)

```

*-----*-----*-----*-----*
* 11 * INTEGER FIELDS 0-8, * SUBSCRIPT WORD WITH 4 SUBSCRIPTS*
*    * 9-17, 18-26 AND 27-35 *
*    *
*    * SUBSCRIPTED VARIABLE IS ALWAYS *
*    * FOLLOWED BY A SUBSCRIPT WORD. *
*-----*-----*-----*-----*
* 16 * POSITIVE INTEGER * A COMMA (,), WHERE H1 IS THE *
*    * H1 = FIELD 0-17 * NUMBER OF FOLLOWING EXPRESSIONS,*
*    * H2 = FIELD 18-35 * H2=0 IF THE EXPRESSIONS ARE IN *
*    * * THE CORE, *
*    * * H2=INDEX IF THE EXPRESSIONS ARE *
*    * * ON DRUM. *
*-----*-----*-----*-----*
* 17 * NOT USED * REPRESENTS ( , IT IS USED ONLY *
*    * * DURING TRANSLATION FROM INFIX *
*    * * TO PREFIX *
*-----*-----*-----*-----*
* 18 * POSITIVE INTEGER * ADD (+) OPERATOR WHERE D(I) IS *
*    * GREATER THAN 1 * THE NUMBER OF FOLLOWING SUB- *
*    * * EXPRESSIONS TO BE ADDED. *
*-----*-----*-----*-----*
* 19 * POSITIVE INTEGER * MULTIPLY (*) OPERATOR WHERE D(I)*
*    * GREATER THAN 1 * IS THE NUMBER OF FOLLOWING SUB, *
*    * * EXPRESSIONS TO BE MULTIPLIED. *
*-----*-----*-----*-----*
* 19 * -2 * DIVISION /, USED ONLY DURING *
*    * * TRANSLATION FROM INFIX TO PREFIX*
*-----*-----*-----*-----*
* 20 * 2 * EXPONENTIALIZATION SUCH THAT THE*
*    * * FOLLOWING TWO SUB-EXPRESSIONS *
*    * * REPRESENT THE BASE AND EXPONENT,*
*    * * RESPECTIVELY. *
*-----*-----*-----*-----*

```


EXPRESSION REPRESENTATION (CONTINUED)

```

* 21 * INTEGER FIELDS * MATHEMATICAL FUNCTIONS WHERE *
*      * 0-17 AND 18-35 * FIELDS 18-35 REPRESENTS THE *
*      * * NUMBER OF FOLLOWING SUB-EXPR. *
*      * * AS ARGUMENTS AND FIELD 0-17 IS *
*      * * AN IDENTIFIER INTEGER AS FOLLOWS*
*      * * 32+128 EXP *
*      * * 33+128 LOG *
*      * * 34+128 TNH *
*      * * 35+128 SIN *
*      * * 36+128 COS *
*      * * 37+128 ATN *
*      * * 50+128 FAC *
*      * * 51+256 BIN *
*      * * 52+384 STEP *
*      * * THE FOLLOWING IDENTIFIER INTEG. *
*      * * APPEAR ONLY DURING TRANSLATION! *
*      * * (EXCEPT THE # ONES) *
*      * * 64+128 EXPAND *
*      * * 65+128 CODEM *
*      * * 66+256 COEFF # *
*      * * 67+128 NUM # *
*      * * 68+128 DENOM # *
*      * * 69+128 EXPON # *
*      * * 70+128 BASE # *
*      * * 71+128 ARG # *
*      * * 72+640 SUBST *
*      * * 73+640 REPLAC *
*      * * 75+128 PRODEX *
*      * * 76+128 FLOAT *
*      * * 77+128 FLOATB *
*      * * 78+256 FIX *
*      * * 79+256 FIXE *
*      * * 80+128 LDOP *
*      * * 81+128 NARG *
*-----*

```

EXPRESSION REPRESENTATION (CONTINUED)

```

*-----*-----*-----*-----*
* 22 * FIELD 0-17:      * DEFINED FUNCTION, USED ONLY *
*   * EXPRESSION POINTER * DURING TRANSLATION FROM INFIX *
*   * FIELD 18-35:      * TO PREFIX *
*   * NUMBER OF ARGUMENTS * *
*-----*-----*-----*-----*
* 23 *      2          * DIFFERENTIATION OF A FUNCTIONAL *
*   *                  * EXPRESSION BY A VARIABLE; IT IS *
*   *                  * FOLLOWED BY A SUBSCRIPTED OR *
*   *                  * NONSUBSCRIPTED VARIABLE AND BY *
*   *                  * AN UNDEFINED FUNCTION OR OTHER *
*   *                  * DIFFERENTIAL OPERATOR. *
*-----*-----*-----*-----*
*25- * ALPHANUMERIC NAME * AN UNDEFINED FUNCTION WHOSE NAME*
* 31 *                  * IS D(I). IT IS FOLLOWED BY *
*   *                  * ITYP(I)-24 SUB-EXPRESSIONS AS *
*   *                  * ITS ARGUMENTS *
*****

```

IN ITS FINAL INTERNAL FORM, EACH EXPRESSION STARTS WITH A PAIR OF C-D WORDS REPRESENTING A COMMA. THE UPPER HALF OF THIS D-WORD CONTAINS THE NUMBERS OF EXPRESSIONS LINKED TOGETHER, WHICH IS USUALLY ONE UNLESS IT REPRESENTS THE VALUE OF A LIST VARIABLE. THE LOWER HALF OF THE C-WORD IS ZERO, UNLESS THE ACTUAL EXPRESSION IS ROLLED OUT TO THE DRUM, IN WHICH CASE IT CONTAINS THE SECTOR INDEX.

THE PREFIX FORM OF THE ALGEBRAIC EXPRESSIONS,

$$C1 - C2 - \dots - CN$$

HAS THE PROPERTY THAT STARTING AT ANY ITEM, CJ, THERE IS A UNIQUE ITEM, CK, WITH

$$J .LE. K .LE. N$$

SUCH THAT

$$CJ - \dots - CK$$

IS A VALID EXPRESSION, I.E. A SUBEXPRESSION OF C1-CN. THE PROGRAM ELEMENT SBEXP PROVIDES THE VALUE K TO A GIVEN J.

FORMAL,SBEXP

CALCULATION OF THE END OF SUBEXPRESSION IN PREFIX

AXRS.

FUNCTION LASTXX(S,J,J1,J2)

S IS AN ERROR RETURN

J IS THE STARTING LOCATION

J1 IS THE STARTING GENERALIZED DEGREE

IF J2 IS NOT ZERO SET THE PRECEDENCE BITS OF THE SUBEXPRESSION
INITIALIZE

```

S(1)
LASTXX*
S,H2 X11,WB+1
L A0,*1,X11. SET J IN A0

PLACE DEGREE IN A5AND SUBTRACT ONE
N1 L A5,*2,X11.
AN,U A5,1.
L A4,*3,X11. GET PRECENCE BIT IN A4

LOOP AND,U A0,077777. GET LOCATION IN A1
JZ A1,ERR. ERROR IF IT IS ZERO
L,S1 A2,FMLCM2-1,A1 GET TYPE
JZ A4,S+3. SET PRECEDENCE BIT ONLY IF IT IS NEEDED
OR,U A2,040.
S,S1 A3,FMLCM2-1,A1.
AND,U A2,037
EX TBLE,A3
L,H2 A0,FMLCM2-1,A1. GET NEXT WORD
JGD A5,LOOP. GO BACK IF DEGREE IS NOT ZERO

JOB IS FINISHED L A0,A1. GET RESULT BACK TO A0
J S,X11.

ERROR CALLING ERR LMJ X11,ERRERR.
J N2
+ K0
+ S-LASTXX,WB
N2 LX,H2 X11,WB+1.
J 0,X11
WB 'LASTXX'
+4,0
+ 'FOR',0

TABLE FOR DEGREE MODIFICATION BY TYPE
S(2)
TBLE NOP: (0)
A,U A5,1. (1)
NOP: (2)
NOP: (3)
J ERR, (4)
NOP: (5)
NOP: (6)

```

A,U	A5,1.	(7)	
NOP.		(8)	
NOP.		(9)	
NOP.		(10)	
NOP.		(11)	
J	ERR.	(12)	
J	ERR.	(13)	
J	ERR.	(14)	
J	ERR.	(15)	
A,H2	A5,FMLCM3-1,A1.		(16)
J	ERR.	(17)	
A,H2	A5,FMLCM3-1,A1.		(18)
A,H2	A5,FMLCM3-1,A1.		(19)
A,H2	A5,FMLCM3-1,A1.		(20)
A,H2	A5,FMLCM3-1,A1.		(21)
A,H2	A5,FMLCM3-1,A1.		(22)
A,H2	A5,FMLCM3-1,A1.		(23)
NOP.		(24)	
A,U	A5,1.	(25)	
A,U	A5,2.	(26)	
A,U	A5,3.	(27)	
A,U	A5,4.	(28)	
A,U	A5,5.	(29)	
A,U	A5,6.	(30)	
A,U	A5,7.	(31)	
K0	+		
	0		
END			

THE MAIN PROGRAM FOR THE TRANSLATION IS EXPRES WHICH IS SIMILAR TO A STANDARD INFIX TO PREFIX TRANSLATION USING AN OPERATOR PUSH DOWN STACK. PROGRAM ELEMENT STOUT0 CORRESPONDS TO THE STANDARD 'POP-UP' OPERATION, I.E. PLACING THE TOP ELEMENT FROM THE PUSH DOWN STACK INTO THE OUTPUT STACK. WHENEVER THIS ROUTINE IS CALLED THE PROGRAM CHECKS WHETHER A SIMPLIFICATION OF THE PARTIALLY WRITTEN OUT EXPRESSION (WHICH MUST BE AN EXPRESSION ITSELF) CAN BE SIMPLIFIED. THIS SIMPLIFICATION DEPENDS PRIMARILY ON THE OPERATOR TO BE WRITTEN OUT. BECAUSE OF THE COMPLEXITY OF SIMPLIFICATIONS, STOUT0 HANDLES DIRECTLY ONLY THE SIMPLIFICATIONS CORRESPONDING TO ARITHMETIC OPERATIONS; AND CALLS STOUT1-5 TO HANDLE THE OTHER FUNCTION OPERATORS.

PROGRAM ELEMENT FNCT GIVES A CONVERSION BETWEEN THE FORMAL BCD FUNCTION NAMES AND ITS INTERNAL INTEGER IDENTIFIER.

FORMAL.FNCT

FUNCTION NAME AND ID, NUMBER CONVERSION ROUTINE

FUNCTION ROUTINE WHICH GIVES THE IDENTIFIER
NUMBER OF THE MATHEMATICAL OR FORMAL FUNCTION

'SMB', AND VICE VERSA.

THERE ARE TWO SUBROUTINES IN THIS DECK.

1. IFUNCT(SMB)

THIS SUBROUTINE CONVERTS A GIVEN FUNCTION NAME TO
ITS CORRESPONDING IDENTIFICATION NUMBER.

2. IFUNC1(SMB)

THIS SUBROUTINE CONVERTS A GIVEN IDENTIFICATION NUMBER
TO ITS FUNCTION NAME.

INTERNAL REPRESENTATION AND LEAD OPERATOR ID.

THE INTERNAL REPRESENTATION OF THE MATH. AND FORMAL FUNCTIONS IS
EQUAL TO ID.N + (N * 128), WHERE N IS THE NUMBER OF REQUIRED
ARGUMENTS (IF N IS EQUAL TO OR LESS THAN 3). WHEN N=4, IT
INDICATES AN ARBITRARY BUT EVEN NUMBER OF ARGUMENTS, WHILE IF
N=5 IT INDICATES AN ARBITRARY BUT ODD NUMBER OF ARGUMENTS.

LEAD OP.

INTERNAL REPRESENTA

ID.			
0	CONSTANT ZERO	0	= ITP(I)
1	INTEGER CONSTANT	0	..
2	RATIONAL CONSTANT	1	..
3	REAL CONSTANT	3	..
5	FUNCTION ARGUMENT UNDEFINED	5	..
6	NON-SUBSCRIPTED VARIABLE	6	..
7	SUBSCRIPTED VARIABLE	7	..
16	LIST	16	..
18	+	18	..
19	*	19	..
20	**	20	..
23	DIFFERENTIAL OPERATOR	23	..
25-31	UNDEFINED FUNCTION	25-31	..
32	EXP	32+128	= H1(I)
33	LOG	33+128	..
34	TNH	34+128	..
35	SIN	35+128	..
36	COS	36+128	..
37	ATN	37+128	..
50	FAC	50+128	..
51	BIN	51+256	..
52	STEP	52+384	..
--	EXPAND	64+128	..
--	CODEM	65+128	..
--	COEFF	66+256	..
--	NUM	67+128	..
--	DENOM	68+128	..
--	EXPON	69+128	..

[illegible][illegible]

```

*****
IFUNC1 (SMB)
*****

```

10 RETURN

```

IFUNCT = 0
IF (SMB .EQ. 'DIF') IFUNCT = 1
IFUNCT = IDC (ISRCH (DIM, IFC, SMB, $10))
RETURN

```

```
ENTRY IFUNC1 (SMB)

IFUNCT = 0

IFUNCT = IFC (ISRCH (DIM, IDC, SMB, $10))
GO TO 10

END
```


3.2.1. EXPRES

THE TRANSLATION ROUTINE EXPRES ASSUMES THAT THE INPUT INFIX ITEMS ARE PROVIDED BY THE SCANNER, GSCAN, WHICH IS INITIALIZED BY THE CALLING ROUTINE. EXPRES USES 2 STACKS DURING THE TRANSLATION, OPERATOR AND OUTPUT STACKS. THE TWO STACKS ARE FORWARD AND BACKWARD LINKED ARRAYS IN C-D COMMON STORAGE AND THE TOP ELEMENT OF THE PUSH DOWN STACK IS LINKED TO THE END OF THE OUTPUT STACK. THE INDEX OF THIS TOP ELEMENT IS CONTAINED IN NP. THUS ANY NEW ITEM LINKED IN DURING THE TRANSLATION IS INSERTED AFTER NP; AND NP IS RESET TO THE ITEM LOCATION INDEX IF THAT ITEM IS TO BE PLACED IN THE PUSH DOWN STACK. AT THE END OF THE TRANSLATION, THE PUSH DOWN STACK CONTAINS ONLY THE INITIALIZING ITEM, A COMMA, WITH ITS D-WORD CONTAINING THE NUMBER OF EXPRESSIONS TRANSLATED. SINCE THE OUTPUT STACK IS LINKED TO THE PUSH DOWN STACK, THE TRANSLATED EXPRESSION CAN BE READ IN FORWARD LINK DIRECTION. THE REVERSION OF THE ARGUMENTS OF NON-SYMMETRIC FUNCTIONS IS PROVIDED IN THE STOUT0-5 ROUTINES TO PROVIDE CORRECT PREFIX FORM. THE MINUS (-) AND DIVIDE (/) OPERATORS ARE TRANSFORMED TO (+) AND (**), RESPECTIVELY:

-A	=>	(-1)*A
- CONST	=>	(-CONST)
A - B	=>	A + (-1) * B
A - CONST	=>	A + (-CONST)
A / B	=>	A * (B ** (-1))

THE TRANSFORMATION OF THE DIVIDE OPERATOR IS TAKEN CARE IN STOUT0 ROUTINE. THE TRANSFORMATION OF THE UNARY AND BINARY (-) OPERATOR IS TAKEN CARE IN EXPRES USING LOGICAL VARIABLE NEG.

THE DEPENDENCE TREE AND THE LISTING OF EXPRES ARE AS FOLLOWS:

```

*****
*                               *
* EXPRES *.....>* FMLERR *
*                               *
*****
.
.                               *****
.                               *
.                               *
. ....>* FMLCHA- *
.                               *
.                               *
.                               *****
.                               *
.                               *
. ....>* FMLCHA- *
.                               *
.                               *
.                               *****
.                               *
.                               *
. ....>* FNCT *.....>* PACK *
.                               *
.                               *
.                               *****
.                               *
.                               *
. ....>* SBEXP *
.                               *
.                               *
.                               *****
.                               *
.                               *
. ....>* GSCAN *
.                               *
.                               *
.                               *****
.                               *
.                               *
. ....>* SYMBOL *
.                               *
.                               *
.                               *****
.                               *
.                               *
. ....>* STOUT0 *
.                               *
.                               *
.                               *****

```

FORMAL,EXPRES

TITLE : EXPRESSION TRANSLATION

DATA STRUCTURE : CMMN

PWORD

ARGUMENT : * ERROR RETURN
ISW INPUT ARGUMENT

SPECIFICATION :

THIS IS A GENERALIZED EXPRESSION TRANSLATION ROUTINE FROM INFIX TO PREFIX FORM. IT ASSUMES THAT THE CALLING ROUTINE INITIALIZED THE SCANNER, THUS GSCANR GIVE THE CONSECUTIVE LOGICAL SYMBOLS. THE ROUTINE MAY BE CALLED FROM 4 DIFFERENT PLACES DEPENDING ON JSW:

- ISW = 0 PROCESS AN ASSIGN STATEMENT --- VARIABLE = EXPRESSION ;
- = 1 TRANSLATE THE EXPRESSION PART FROM A READ-IN DATA WHICH MAY BE IN THE FORM: EXPRESSION ;
OR VARIABLE = EXPRESSION ;
- = 2 PROCESS SUBSCRIPT EXPRESSION IN THE FORM:
EXPRESSON)
- = 3 PROCESS AN EXPRESSION IN THE FORM:
EXPRESSON ;

IN THE FIRST CASE THE INFORMATIONS FOR THE VARIABLE ARE STORED IN N1,N2 AND N3. IN THE SECOND CASE, ONLY THE EXPRESSION PART IS RETAINED UPON RETURN. IN ALL CASES, THE TRANSLATED EXPRESSION IS PLACED ABOVE THE PUSH-DOWN STACK WITH THE THE PUSH-DOWN STACK CONTAINING ONLY ONE ENTRY: A COMMA WITH A COUNT CORRESPONDING TO THE NUMBER OF EXPRESSIONS TO ACCOMODATE LISTS.

LOGICAL VARIABLE 'SB' IS TRUE WHENEVER THE SCANNED SYMBOL IS IN SUBSCRIPT LEVEL. 'SBC' VARIABLE CONTAINS THE DEPTH OF THIS LEVEL.

LOGICAL VARIABLE 'EQL' IS TRUE WHEN AN '=' HAD BEEN PROCESSED ALREADY, THUS IT MAY NOT APPEAR AGAIN. '=' MAY ALSO NOT APPEAR ON SUBSCRIPT LEVEL.

THE SYNTAX OF EXPRESSIONS IS CHECKED AT EVERY SCANNED SYMBOL BY MASKING 'TEST' WHICH WAS SET BY THE PREVIOUS SYMBOL. IF THE RESULT IS NOT ZERO THEN THE EXPRESSION HAS SYNTACTIC ERROR. IN THE FOLLOWING TABLE, 'A' DENOTES AN ALPHANUMERIC NAME, 'N' DENOTES A NUMERIC CONSTANT, 'I' DENOTES POSITIVE INTEGER:

SYMBOL	MASKING BITS (DEC.)	RESET TEST (DEC.)
INITIAL ASSIGN	---	1000000 (64)
INITIAL OTHERS	---	0100000 (32)
A	0001110 (14)	0001000 (8)
A(	0001110 (14)	0100000 (32)
AC	1001110 (78)	0100000 (32)
N	1001110 (78)	0000100 (4)
[I]	1001110 (78)	0000100 (4)
NI	1001110 (78)	0000100 (4)

SUBROUTINE EXPRES (*, ISW)

INCLUDE CMMN

INCLUDE PWORD

```

(      1001110 (78)      0100000 (32)
=      1110101 (117)     0000001 ( 1)
UNARY +- 1011110 (94)     0010000 (16)
BINARY +- 1110001 (113)   0010000 (16)
* / **    1110001 (113)   0010000 (16)
,          1110001 (113)   0100000 (32)
) AS SEPARATOR 1110001 (113) 0000100 ( 4)
]          1110001 (113)   0000100 ( 4)
) AS END OF SUBS. 1110001 (113) 0000010 ( 2)
;          1110001 (113)   ---

```

'BRT' AND 'PAR' ARE USED TO COUNT THE BRACKETS AND PARENTHESIS, RESPECTIVELY.
LOGICAL 'NEG' IS SET TO TRUE BY '-' FOR THE NEXT CHARACTER SCANNED ONLY.

INITIALIZE

PUSH-DOWN STACK WITH COMMA

SUBSCRIPT LEVEL

LOGICAL VARIABLES EQL AND END, INITIAL TEST

LOOP

* TO GET NEXT SYMBOL

```

*
*
* JUMP ACCORDING TO SYMBOL

```

IND = 1,2,3,4 FOR

INTEGER, REAL, IDENTIFIER, SPECIAL CHARACTER

```

*
*--- INTEGER

```

```

*
*
*--- REAL

```

```

*
*
*
*
*
*

```

LOGICAL SB,EQL,NEG

```

NP=IGETF1($990)
NP0=NP
C(NP)=20K10
D(NP)=1

```

```

SB= .FALSE.
BRT = 0
PAR = 0
SBC=0

```

```

EQL= ISW .GE. 2
TEST=32
IF (ISW .EQ. 0) TEST=64
NEG= .FALSE.
IF (ISW .EQ. 2) GO TO 180

```

```

30 CONTINUE
CALL GSCANR($990,IND,N1,ITC,ICC)

```

GO TO (100,110,40,60),IND

```

100 I=0
GO TO 120

```

```

110 I=3
120 IF (AND(TEST,78) .NE. 0) CALL FMLERR($990,NI,I,1)
TEST=4
J=ILINK1(NP,I,N1)
IF (NEG) D(J)=-D(J)
NEG= .FALSE.

```


SUBSCRIPTED FUNCTION

```

: ... YES, GO TO FUNCTION PART
: OF DEFINITION

```

```

: ... NO, IT IS A DUMMY ARGUMENT,
: THEN IT MUST BE FOLLOWED BY
: AN INTEGER AND LEFT BRACKET

```

```

IF (TEST .EQ. 2) GO TO 220

```

```

INEG = 2
GO TO 500

```

```

285 CALL GSCANR($990,IND,IDT,ITC,ICC)
IF (IND .NE. 1 .OR. IDT .LE. 0) CALL FMLERR($990,ITC,1,17)
I=ILINK1(NP,5,IDT)
CALL GSCANR($990,IND,IDT,ITC,ICC)
IF (IND .NE. 4 .OR. IDT .NE. 3) CALL FMLERR($990,ITC,1,17)
TEST=4
GO TO 30

```

```

*--- RIGHT BRACKET ]

```

```

END OF FUNCTION ARGUMENTS

```

```

290 IF (AND(TEST,113) .NE. 0) CALL FMLERR($990,ITC,1,1)
BRT = BRT-1
IF (BRT .LT. 0) CALL FMLERR($990,ITC,1,4)
TEST=4
CALL STKOUT($990,17)
IF (ITYP(NP) .NE. 16) CALL FMLERR($990,ITC,1,1)
I=D(NP)
J=NP
NP=LAST(NP)
CALL RMOVF1(J)
J=ITYP(NP)
IF (J .LT. 21) CALL FMLERR($990,ITC,1,4)
IF (J .EQ. 24) ITP(NP)=I+24
IF ((J .EQ. 24) .AND. ((I+24) .GT. 31))
1 CALL FMLERR($990,D(NP),1,3)
IF (J .LT. 24) H2(NP)=I
CALL STKOUT($990,21)
GO TO 30

```

```

*--- RIGHT PARENTHESIS )

```

```

CHECK IF THIS IS AN END OF SUBSCRIPT
OR THE END OF A SUBEXPRESSION

```

```

300 IF (AND(TEST,113) .NE. 0) CALL FMLERR($990,ITC,1,1)

```

```

PAR = PAR-1
IF (PAR .LT. 0) CALL FMLERR($990,ITC,1,4)
CALL STKOUT($990,18)
IF (ITYP(NP) .NE. 17) CALL FMLERR($990,ITC,1,4)
IF (D(NP) .NE. 0) GO TO 310

```

```

I=NP
NP=LAST(NP)
CALL RMOVF1(I)

```

```

: ... END OF SUBEXPRESSION,
: REMOVE MATCHING '('

```



```

: ... END OF A SUBSCRIPT LIST
: CHECK AND PACK SUBSCRIPTS

```

END FOR SUBSCRIPT LIST,
RETURN FOR 'ISW' = 2
GO TO IDENTIFIER IF ISW ISN'T 2

*--- MINUS ---

*--- PLUS *

UNARY PLUS OR MINUS

BINARY OPERATOR

```
*--- MULTIPLICATION *
*      GO TO BINARY OPERATOR
```

```

** EXPONENTIAL **
*      GO TO BINARY OPERATOR

```

TEST=4
GO TO 30

```

310  TEST=2
      N2=D(NP)
      IF (N2 .GT. 4) CALL FMLERR($990,N2,0,5)
      N3=0
      SBC=SBC-1
      IF (SBC .EQ. 0) SB= .FALSE.
      DO 320 KK=N2-1,0,-1
      K=NEXT(NP)
      IF (ITYP(K) .NE. 0) CALL FMLERR ($990, D(K), 2, 29)
      IF (D(K) .LT. 0 .OR. D(K) .GT. 511) CALL FMLERR($990,D(K),0,15)
      FLD(9*KK,9,N3) = D(K)
320  CALL RMOVF1(K)
      IF ((ISW .EQ. 2) .AND. (.NOT. SB)) GO TO 330
      J=NEXT(NP)
      N1=D(J)
      K=NP
      NP=LAST(NP)
      CALL RMOVFN(K,J)
      GO TO 140

```

```

330  CALL IFREE0(NP0)
      RETURN

```

340 NEG= .TRUE.

```
350 IF (AND(TEST,94) .NE. 0) GO TO 360
```

CALL STKOUT(\$990,18)
TEST=16
GO TO 30

```

360 J=18
370 I=2
380 IF (AND(TEST,113) .NE. 0) CALL FMLERR($990,ITC,1,1)
TEST=16
CALL STKOUT($990,J)
NP=ILINK1(NP,J,I)
GO TO 30

```

390 J=19
GO TO 370

END OF TRANSLATION

```

:
:
:
RETURN FOR 'ISW' = 1 AND 3
:
:
TRANSFER ASSIGNED VARIABLE INFORMATION
:
:
:
RETURN FOR 'ISW' = 0
:
ERROR RETURN
:
:

```

```

IF (PAR .NE. 0 .OR. BRT .NE. 0) CALL FMLERR($990, '() L]', 1, 4)
CALL STKOUT ($990, 18)
IF ((ITYP(NP) .NE. 16) .OR. (LAST(NP) .NE. 0))
1 CALL FMLERR($990, ITC, 1, 1)

```

```

IF (ISW .NE. 0) RETURN
IF (.NOT. EGL) CALL FMLERR($990, ITC, 1, 1)

```

```

N1=NN1
N2=NN2
N3=NN3

```

RETURN

```

990 CALL IFREE0(NP0)
RETURN 1
END

```

3.2.2. STOUT0

STOUT0 CORRESPONDS TO THE POP-UP PROCESS IN A PRECEDENCE GRAMMAR TRANSLATION, I.E., GIVEN A PRECEDENCE NUMBER IIOP, THE TOP ELEMENTS OF THE PUSH DOWN STACK IS WRITTEN OUT UNTIL THE PRECEDENCE NUMBER OF THE TOP ELEMENT IS LESS THAN THE GIVEN NUMBER IIOP. IN THE FORMAL DATA STRUCTURE, THIS 'WRITING OUT' CORRESPONDS TO THE RESETTING THE PUSH DOWN STACK POINTER NP TO ITS PRECEDING ITEM: $LAST(NP) \Rightarrow NP$. FURTHER COMPLICATION ARISES FROM THE AUTOMATIC SIMPLIFICATION PROVIDED IN THE STOUT0-ROUTINES. THESE SIMPLIFICATIONS ARE ALWAYS CHECKED IN RESPECT TO THE TOP ELEMENT OF THE PUSH DOWN STACK. WHEN A SIMPLIFICATION IS EXECUTED, THE RESULTING EXPRESSION IS THE LAST VALID EXPRESSION ABOVE THE PUSH DOWN STACK. IN THIS CASE, THE OPERATOR IS SIMPLY DISCARDED INSTEAD OF WRITING IT OUT. FURTHERMORE, THE SIMPLIFIED RESULTING EXPRESSION MAY BE FURTHER SIMPLIFIED, NOW WITHOUT THE DISCARDED OPERATOR. TO PROVIDE A SYSTEMATIC SIMPLIFICATION PROCEDURE, THE EXPRESSION IS PUSHED BACK TO THE PUSH DOWN STACK, THUS DURING CONSECUTIVE POP-UP OPERATIONS, FURTHER SIMPLIFICATIONS MAY BE EXECUTED. THE NECESSITY OF THIS PUSH BACK OPERATION DEPENDS ON THE INDIVIDUAL SIMPLIFICATION EXECUTED, AND LOGICAL VARIABLE SIMPSW IS USED TO INDICATE THE NECESSITY OF PUSH BACK OPERATION. WHEN AN EXPRESSION IS PUSHED BACK, THE PRECEDENCE NUMBER OF ITS ITEMS ARE INCREASED BY 32. THIS TRICK SERVES TWO PURPOSES. FIRST, THE PUSHED BACK ITEMS ALWAYS HAVE LARGER PRECEDENCE NUMBER THAN THE GIVEN ARGUMENT IIOP FOR THE STOUT0 ROUTINE; THUS A COMPLETE SIMPLIFICATION IS ALWAYS EXECUTED BEFORE THE CONTROL GOES BACK TO THE CALLING ROUTINE. SECOND, IT PROVIDES A 2-WAY LOGIC, WHETHER AN OPERATOR WAS ENCOUNTERED THE FIRST TIME. THIS FEATURE IS USED TO REORDER ARGUMENTS FOR NON-SYMMETRIC OPERATORS TO PROVIDE CORRECT PREFIX FORM, AND IT IS USED FOR THE EXECUTION OF CERTAIN FORMAL FUNCTIONS REQUIRING MORE THAN ONE STEP.

STOUT0 CHECKS FOR SIMPLIFICATION ONLY WHEN THE TOP OPERATOR IS +, * OR **. WHEN THE TOP ITEM IS AN OPERAND, CONSTANT OR VARIABLE, ITS INCREASED PRECEDENCE IS DECREASED BY 32, AND IT IS WRITTEN OUT. WHEN THE TOP OPERATOR IS A MATHEMATICAL, FORMAL, DEFINED, DIFFERENTIAL OR UNDEFINED FUNCTION OPERATOR, THEN STOUT0 CALLS STOUT1-5, RESPECTIVELY.

STOUT0 TRANSFERS THE DIVISION OPERATOR INTO EXPONENTIALIZATION, I.E. IN PREFIX ORDER

$$/ \ B \ A \Rightarrow \ * \ * \ B \ (-1) \ A$$

SUCH THAT THE DIVIDE OPERATOR IS REPLACED BY THE OPERATORS * AND ** IN THE PUSH DOWN STACK, AND THE CONSTANT (-1) IS INSERTED BETWEEN THE TWO SUBEXPRESSIONS B AND A IN THE OUTPUT STACK.

STOUT0 MAKES THE + AND * OPERATORS N-ARY OPERATORS. E.G. WHEN

THE TOP ELEMENT OF THE PUSH DOWN STACK IS +, AND ONE OF ITS ARGUMENT SUBEXPRESSION IN THE OUTPUT STACK IS A SUM (LEAD OPERATOR IS +), THEN THE TWO + OPERATORS ARE COMBINED INTO ONE WITH THE PROPER COUNT OF ARGUMENTS. AS A RESULT, THE SUM OF N EXPRESSIONS WILL BE SUCH THAT NONE OF THE SUBEXPRESSIONS ARE SUMS THEMSELVES. THE SAME IS TRUE FOR PRODUCTS, I.E., NONE OF ITS FACTORS ARE PRODUCTS.

COMPARING SUBEXPRESSIONS AND SIMPLIFYING THEM IN RESPECT TO THE TOP OPERATOR IS PROVIDED IN PROGRAM ELEMENT COMP1. THIS ROUTINE COMPARES TWO SUBEXPRESSIONS. IF THEY ARE NOT EQUAL, I.E. SIMPLIFICATION IS NOT POSSIBLE, THEN IT RETURNS WITH AN INDICATION WHETHER THE FIRST SUBEXPRESSION IS CONSIDERED TO BE 'SMALLER' OR 'LARGER' THAN THE SECOND ONE. THIS RELATION IS USED FOR SORTING THE ARGUMENTS OF THE SYMMETRIC N-ARY OPERATORS + AND *. WHEN SIMPLIFICATION IS POSSIBLE, IT IS DONE IN COMP1.

COMP1 INVESTIGATES FIRST THE GIVEN TWO SUBEXPRESSIONS ACCORDING TO THE FOUR COMBINATIONS, WHETHER THEY ARE CONSTANTS OR NOT; THEN EACH GROUP IS BROKEN DOWN ACCORDING TO THE TOP OPERATOR. ARITHMETIC OPERATIONS WITH FORMAL CONSTANTS ARE PROVIDED BY PROGRAM ELEMENT CONSOP, COMPARING NON-CONSTANT SUBEXPRESSIONS FOR IDENTITY IS PROVIDED BY PROGRAM ELEMENT COMP2.

THE DEPENDENCE TREES OF STOUT0 AND COMP1 ARE GIVEN BELOW FOLLOWED BY THE LISTINGS OF STOUT0, COMP1 AND COMP2:

```

.
.
.
.
.
.
*****
*
. . . . . > * ERRWBK *
*
*****

```

```

.                                     *****
.                                     *
.                                     *
. ....>* SBEXP *
.                                     *
.                                     *****

```

```

.                                     *****
.                                     *
.                                     *
. . . . . > * FMLCHA *
.                                     *
.                                     *
.                                     *****

```

```

.                                     *****
.                                     *
.                                     *
. . . . . > * STOUT5 *
.                                     *
.                                     *****

```

```

      *****
      *                               *
.....>* STOUT3  *
      *                               *
      *****

```

3-114

```

.
.
.
.....>* STOUT2 *
.
.
*****
.
.
.
*****
.
.
.
.....>* COMP1 *
.
.
*****
.
.
.
*****
.
.
.
.....>* STOUT4 *
.
.
*****
.
.
.
.....>* STOUT1 *
.
.
*****

```

```

*****
* COMP1 *.....>* FMLERR *
*
*****

.
.
.
.....>* COMP2 *.....>* SBEXP *
.
.
.
.....>* FMLCHA- *
.
.
.
.....>* FMLCHA- *
.
.
.
.....>* ERRWBK *
.
.
.
.....>* SBEXP *
.
.
.
.....>* CONSOP *
.
.
.

```


FORMAL,STOUTO
OPERATOR WRITE-OUT FROM PUSH-DOWN STACK
SIMPLIFICATION WITH ARITHMETIC OPERATIONS

```
SUBROUTINE STKOUT($,IIOP)
INCLUDE CMMN
INCLUDE PWORD
LOGICAL BITSW,SIMPSW
```

```
*****
MAIN SUBROUTINE TO WRITE OUT OPERATORS (AND OPERANDS WITH
INCREASED PRECEDENCE) FROM THE PUSH-DOWN STACK UNTIL THE TOP
ELEMENT HAS PRECEDENCE NUMBER LESS THAN IIOP.
```

 WHEN AN OPERATOR IS WRITTEN OUT FROM THE PUSH-DOWN STACK, OR
 EXECUTED, A SIMPLIFICATION MAY OCCUR ON THE ALREADY WRITTEN OUT
 SUBEXPRESSION. THIS SIMPLIFICATION ON THE OUTER LEVEL OF A SUB-
 EXPRESSION MAY INDUCE THE NEED OF FURTHER SIMPLIFICATION FOR THE
 INNER LEVELS OF THE SUBEXPRESSION. IN THIS CASE, THE SUBEXPRESSION
 IS PUSHED BACK TO THE STACK WITH INCREASED PRECEDENCE NUMBER. THIS
 IS INDICATED BY THE SETTING OF SIMPSW, AN ITEM TO BE WRITTEN OUT
 OUT FROM THE STACK WHICH HAS AN INCREASED PRECEDENCE, IS INDICATED
 BY SETTING 'BITSW. TO .TRUE..

THE POINTER TO THE TOP ELEMENT IN THE STACK IS 'NP'.
THIS SUBROUTINE HANDLES THE ARITHMETIC OPERATORS WITH USING
SUBROUTINE COMPSF FOR COMPARING TWO EXPRESSIONS. THE MATHEMATICAL,
FORMAL, DEFINED AND UNDEFINED FUNCTIONS ARE HANDLED BY CALLING
SUBROUTINES STMATH, STFORM, STCTD AND STUNFC, RESPECTIVELY.
THE DIFFERENTIAL OPERATOR IS HANDLED BY STDIFF.

OPERANDS WITH INCREASED PRECEDENCE NUMBERS ARE SIMPLY WRITTEN OUT

START

COMPARE THE TOP ELEMENT OF THE STACK TO IIOF
RETURN, IF LESS THAN IOF

```

INITIALIZE BITSW AND SIMPSW

```

BRANCH BY IOP, THE TOP ELEMENT

```
*
*
*
*
*
*--- WRITE OUT OPERAND
*      :
*      :
BRANCH BY OPERATOR IOP
*
*
```

```

10      IOP=ITYPB(NP)
      IF (IOP.LT.IIOP) RETURN

```

```
SIMPSW=.FALSE.
BITSW=.FALSE.
```

```

      IF (IOP.LT.32) GO TO 20
      IOP=IOP-32
      BITSW=.TRUE.
20    IF (IOP.GT.11) GO TO 30
      IBIT(NP)=0

```

```
NP=LAST(NP)
GO TO 10
```

```

30      IF (IOP.LE.17) CALL ERROR($990,30)
      K=IOP-17

```



```

*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :

```

NOW COMPARE I-IK AND J-JK

```

*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :
*      :      :

```

... THE TWO EXPRESSIONS WERE
SIMPLIFIED WITH THE RESULT IN
I-IK, THUS DECREASE THE NUMBER
OF SORTED ARGUMENTS KN AND THE
DEGREE OF THE OPERATOR

CALL COMPSF (\$80, \$90, \$990, IOP, I, IK, J, JK)


```

*
*---- EXPONENTIALIZATION
*      GET BASE AND EXPONENT POINTERS
*      :
*      :
*      :
*      :
*      SINCE BITSW IS FALSE,
*      THIS IS THE FIRST ENCOUNTER
*      OF THIS EXPONENTIAL OPERATOR, THUS THE
*      BASE AND EXPONENT MUST BE INTERCHANGED TO
*      GIVE THE CORRECT PREFIX FORM
*      :
*      :
*      :
*      NOW CHECK FOR SIMPLIFICATION
*      :
*      NORMAL RETURN MEANS TO REMOVE **
*      :
*---- MATHEMATICAL AND FORMAL FUNCTIONS
*      CALL THE APPROPRIATE ROUTINES
*      :
*      :
*---- DEFINED FUNCTION
*      :
*---- DIFFERENTIAL OPERATOR
*      :
*---- UNDEFINED FUNCTION
*      :
*      :
SPECIAL RETURNS TO START
*---- REMOVE OPERATOR
*      :
*      :
*      :
*      :
*---- PUSH BACK LAST EXPRESSION FOR FURTHER SIMPLIFICATION
*      :
*      :
*      :
*      :
*---- WRITE OUT OPERATOR
*      :
*      :
*      :
ERROR RETURN

```

```

GO TO 140

130  I=NEXT(NP)
      IK=LASTXX($990,I,1,0)
      J=NEXT(IK)
      JK=LASTXX($990,J,1,0)
      IF (BITSW) GO TO 140

      CALL FMLCH7 ($990, I, IK, J, JK)
      IBIT(NP)=1
      BITSW=.TRUE.

140  CALL COMPSF ($910, $910, $990, IOP, I, IK, J, JK)

      GO TO 900

150  IF (FLD(11,7,D(NP)) .LT. 64) CALL STMTM ($900, $910, $990)
      CALL STFORM ($900, $910, $990)

160  CALL STFCTD ($900, $910, $990)

170  CALL STDIFF ($900, $910, $990)

180  K=K-7
      CALL STUNFC ($900, $910, $990, K)

900  K=LAST(NP)
      CALL FMLCHI ($990, NP)
      NP=K
      IF (.NOT.SIMPSW) GO TO 10

      NP=NEXT(NP)
      IBIT(NP)=1
905  NP=LASTXX($990,NP,1,1)
      GO TO 10

910  IF (SIMPSW) GO TO 905
      IBIT(NP)=0
      NP=LAST(NP)
      GO TO 10

990  RETURN 1
      END

```

FORMAL,COMP1

SUBROUTINE COMPSF (*, *, *, IOP, I, IK, J, JK)
 INCLUDE CMMN
 INCLUDE PWORD

 COMPSF (*, *, *, IOP, I, IK, J, JK)

 THE SUBROUTINE COMPARES TWO EXPRESSIONS STARTING AT
 I AND J, ENDING AT IK AND JK, RESPECTIVELY. 'IOP' IS
 THE LEAD OPERATOR FOR WHICH THE COMPARISON IS TO BE
 DONE: IOP = 18 FOR +
 IOP = 19 FOR -
 IOP = 20 FOR **
 THE LEAD OPERATOR OF THE TWO SUBEXPRESSIONS MAY NOT
 BE THE SAME AS IOP, EXCEPT FOR IOP = 20(**).
 IF THE TWO EXPRESSIONS SIMPLIFY WITH THE LEAD
 OPERATOR IOP, THEN THE RESULTING EXPRESSION IS
 LINKED INTO THE PLACE I-IK AND THE ORIGINAL EXPRESSIONS
 ARE RELEASED. SIMPSW IS SET TO TRUE IF SIMPLIFICATION
 OCCURED SUCH THAT FURTHER SIMPLIFICATION MAY OCCURE
 IN THE RESULT.
 THE RETURNS ARE AS FOLLOWS:
 RETURN 0 = EXPRESSIONS SIMPLIFIED WITH RESULT IN I-IK
 RETURN 1 = EXPRESSION I-IK IS SMALLER THAN J-JK
 RETURN 2 = EXPRESSION J-JK IS SMALLER THAN I-IK
 RETURN 3 = ERROR RETURN
 FOR IOP=20, RETURN 1 AND 2 ARE EQUIVALENT.

INITIALIZE POINTERS

;
 ;
 ;
 ;
 ;
 ;
 ;
 ;
 ;

BRANCH ACCORDING TO THE TYPE OF EXPRESSIONS:
 CONSTANT OR NON-CONSTANT

*
 *
 *
 *
 *--- I-IK IS A CONSTANT
 *--- J-JK IS A CONSTANT
 * EXECUTE IOP OPERATION WITH CONSTANTS
 *

REAL FLPT
 REAL F(1), FKONE
 EQUIVALENCE (F(1), D(1))
 DATA FKONE / 2.7182818 /

II=I
 JJ=J
 IIK=IK
 JJK=JK
 IO=ITYP(II)
 JO=ITYP(JJ)
 K=IOP-17

KJUMP=1
 IF (JO,LE,3) KJUMP=KJUMP+1
 IF (IO,LE,3) KJUMP=KJUMP+2
 GO TO (2400,2200,2300,2100),KJUMP

2100 CALL CONSOP(\$2110,\$2120,\$2110,\$2000,K,II,JJ,IJ)

```

*
*
*
*
*---- I-IK IS NOT A CONSTANT
*---- J-JK IS CONSTANT
*
* J-JK IS ZERO, THUS RESULT IS :
*       I-IK FOR +
*       J-JK FOR *
*       1   FOR **
*
* J-JK IS NOT ZERO
* RETURN 2 IF IOP IS +
*
* IF J-JK IS 1 THEN THE RESULT IS I-IK FOR *
* AND **
*
* NONE OF ABOVE, BRANCH BY IOP
* CHECK THE OPERATOR * AND **
* ... MULTIPLICATION
*       CHECK IF THE DISTRIBUTIVE LAW IS
*       APPLICABLE
*       YES, INSERT J-JK INTO THE
*       SUM I-IK
*
* ... EXPONENTIALIZATION,
*       PROCEED BY THE LEAD OPERATOR OF
*       BASE I-IK
*       ... PRODUCT
*       GO TO RAISING ITS FACTORS
*       TO THE EXPONENT
*       ... EXPONENTIALIZATION
*       GO TO FORM THE PRODUCT OF
*       EXPONENTS
*
2110 IJJ = IJ
      GO TO 2130
2120 IJJ=NEXT(IJ)
2130 CALL MOVEXX (I, IK, IJ, IJJ)
      GO TO 2020

2200 IF (D(J).NE.0) GO TO 2220

      GO TO (2020,2040,2210),K

2220 IF (K.EQ.1) RETURN 2

      IF ((JJ.EQ.0).AND.(D(JJ).EQ.1)) GO TO 2020
      IF (JO .EQ. 3 .AND. F(JJ) .EQ. 1.0) GO TO 2020
      IF (K .EQ. 3) GO TO 2235

      IF (EXPDSW .EQ. 0 .OR. IO .NE. 18) RETURN 2

2225 N=D(II)
      K=NEXT(II)
      DO 2230 L=1,N
      II=ILINK1(II,19,2)
      II=IADDNN(II,J,JK)
      II=LASTXX($2000,K,1,0)
2230 K=NEXT(II)
      GO TO 2010

2235 IF (IO .EQ. 18) GO TO 2240

      IF (IO.EQ.19) GO TO 2050

      IF (IO.EQ.20) GO TO 2290
      RETURN 1

```

```

SUM
CHECK IF EXPONENT IS
INTEGER AND EXPAND SWITCH
IS ON

```

```

YES, THUS EXECUTE
EXPONENTIALIZATION THE
BINOMIAL COEFF, FCT.
HAS ID,= 51+256 DIVIDE
THE SUM I-IK INTO TWO
TERMS

```

```

NOW THE SUM OF TWO
TERMS ARE ESTABLISHED
IN II-IK AND INK-IK
WHICH MUST BE RAISED
TO THE NTH POWER

```

```

LINK IN LEAD OPERATOR
+, AND GENARTE THE
TERMS

```

```

2240 IF (JO .NE. 0 .OR. D(JJ) .LE.1) RETURN 1
IF (EXPDSW .GE. 0 .AND. EXPDSW .LT. D(JJ)) RETURN 1

```

```

ILL=LAST(II)
CALL REMOVN(II,IK)
IF (D(II).LE.1) CALL ERROR($2000,2250)
D(II)=D(II)-1
IIKK=LASTXX($2000,II,1,0)
INK=NEXT(IIKK)
IF (D(II) .EQ. 1) II = RM0VF1 (II)

```

```

N=D(JJ)
M1 = N+1

```

```

ILL = ILINK1(ILL,18,M1)
I=ILL
K=0
NN=N
DO 2280 L = 1,M1
IF (L.EQ.1) GO TO 2250
IF (L .EQ. M1) GO TO 2260
ILL=ILINK1(ILL,19,3)
ILL = ILINK1 (ILL, 21, 307*2**18+2)
ILL = ILINK1(ILL,0,N)
ILL=ILINK1(ILL,0,K)
2250 ILL=ILINK1(ILL,20,2)
ILL=IADDNN(ILL,II,IIKK)
ILL=ILINK1(ILL,0,NN)
IF (L.EQ.1) GO TO 2270
2260 ILL=ILINK1(ILL,20,2)
ILL=IADDNN(ILL,INK,IIK)
ILL=ILINK1(ILL,0,K)
2270 NN=NN-1
2280 K=K+1
IK=ILL

```

```

*      :      :      :      :      :
*      :      :      :      :      :
*---- I-IK IS A CONSTANT
*---- J-JK IS NOT A CONSTANT
*      :
*      I-IK IS ZERO, THUS THE RESULT IS:
*          J-JK FOR +
*          I-IK FOR * AND **
*
*      I-IK IS NOT ZERO
*      RETURN 1 FOR IOP = +
*
*      IF I-IK IS 1, THEN THE RESULT IS I-IK
*
*      OTHERWISE PROCEED ACCORDING TO
*      IOP = * OR **
*
*      ... MULTIPLICATION
*      CHECK IF DISTRIBUTIVE LAW IS
*      APPLICABLE
*
*          YES, MULTIPLY WITH I-IK INTO THE
*          SUM J-JK
*
*      ... EXPONENTIALIZATION
*      CHECK IF THE CONSTANT BASE I-IK
*      SHOULD BE CHANGED ACCORDING TO OPTION
*      BASE
*
*          BRANCH ACCORDING TO BASE
*          ... BASE MUST BE 2
*
*              CHANGE I-IK TO IRR
*
*      CALL IFREEN(II,I IK)
*      GO TO 2010
*
*      2300 IF (D(II).NE.0) GO TO 2310
*
*      GO TO (2040,2020,2020),K
*
*      2310 IF (K.EQ.1) RETURN 1
*
*      IF ((I0.EQ.0) .AND. (D(II).EQ.1)) GO TO 2070
*      IF (I0.EQ.3 .AND. F(II).EQ.1.0) GO TO 2070
*
*      IF (K.EQ.3) GO TO 2330
*
*      IF (EXPDSW .EQ. 0 .OR. JO .NE. 18) RETURN 1
*
*      2315 N=D(JJ)
*           K=NEXT(JJ)
*           DO 2320 L=1,N
*           JJ=ILINK1(JJ,19,2)
*           JJ=IADDNN(JJ,I,IK)
*           JJ=LASTXX($2000,K,1,0)
*
*      2320 K=NEXT(JJ)
*           GO TO 2030
*
*      2330 IF (BASE.EQ.0) RETURN 1
*           IF (D(II).LT.0) RETURN 1
*           FLPT=0.
*
*           GO TO (2340,2350,2360),BASE
*
*      2340 IF ((I0.EQ.0).AND.(D(II).EQ.2)) RETURN 1
*           IR=0
*           IRR=2
*
*      2345 IF (I0.EQ.0) FLPT=FLOAT(D(II))
*           IN=NEXT(II)
*           IF (I0.EQ.1) FLPT=FLOAT(D(II))/FLOAT(D(IN))
*           IF (I0.EQ.3) FLPT = F(II)

```



```

IF (FLPT,LE,0) CALL FMLERR('S2000,FLPT,2,6)
FLPT=LOG(FLPT)
IF (BASE,EQ,1) FLPT=FLPT/,.69314718
IF (BASE,EQ,2) FLPT=FLPT/2,.3025851
JN=LAST(JJ)
J=ILINK1(JN,19,2)
JN=ILINK1(J,3,FLPT)
IN=LAST(II)
CALL RMOVFN(II,IJK)
I=ILINK1(IN,IR,IRR)
IK=I
SIMPSW=,TRUE.
RETURN 2

2350 IF ((IO,EQ,0).AND,(D(II);EQ,10)) RETURN 1
IR=0
IRR = 10
GO TO 2345

2360 IF (IO .EQ. 3 .AND. F(II) .EQ. FKONE) GO TO 2370
IR=3
IRR = BOOL(FKONE)
GO TO 2345

2370 ITYP(II) = 21
D(II) = 1
H1(II) = 160
SIMPSW = ,TRUE.
RETURN

2400 GO TO (2410,2600,2700),K

2410 ICD1=1
IF (IO,NE,19) GO TO 2420
ICD1=2
I3=NEXT(II)
IF (ITYP(I3),GT,3) GO TO 2420
ICD1=3
I4=I3

```



```

      ... RATIONAL FACTOR RESULT

```

```

      ... REAL FACTOR RESULT

```

```

      WHEN EXPRESSIONS HAD SIMILAR
      STRUCTURES:

```

```

      ... NO FACTORS, THUS THE NEW
      FACTOR IS 2

```

```

      ... BOTH HAD FACTORS, GET THE
      SUM OF THEM, AND GO TO
      SET THE RESULT

```

```

      ... EXPRESSIONS ARE NOT EQUIVALENT, SET
      RETURN INDEX

```

```

      LINK BACK REMOVED FACTORS AND
      RETURN

```

```

      ... MULTIPLICATION

```

```

      CHECK IF DISTRIBUTIVE LAW IS APPLICABLE
      ... IF YES, GO TO THE PROPER PLACE

```

```

      I1=I
      I=NEXT(I)
      CALL RMOVF1(I1)
      GO TO 2450

```

```

2490 J2=NEXT(J1)
2500 IF (ICD1.GT.1) GO TO 2510

```

```

      I1=LAST(I)
      I=ILINK1(I1,19,1)
2510 D(I)=D(I)+1
      I1=ILINKN(I,J1,J2)
      GO TO 2450

```

```

2520 J2=J1
      IF (F(J1).EQ.1.0) GO TO 2430
      GO TO 2500

```

```

2530 IF (ICD1.EQ.3) GO TO 2550
      IF (ICD1.EQ.2) GO TO 2540

```

```

      I1=LAST(I)
      I=ILINK1(I1,19,1)
2540 D(I)=D(I)+1
      I1=ILINK1(I,0,2)
      GO TO 2460

```

```

2550 CALL CONSOP($2470,$2490,$2520,$2000,1,I3,J3,J1)
      GO TO 2440

```

```

2560 IRT=1
      GO TO 2580

```

```

2570 IRT=2

```

```

2580 IF (ICD1.NE.3) GO TO 2590
      I1=ILINKN(I,I3,I4)
      D(I)=D(I)+1

```

```

2590 IF (ICD2.NE.3) RETURN IRT
      I1=ILINKN(I,J3,J4)
      D(J)=D(J)+1
      RETURN IRT

```

```

2600 IF (EXPDSW.EQ.0) GO TO 2610

```



```

2000    RETURN 3

2070    IF (K.EQ. 2) GO TO 2040
        GO TO 2020

2010    SIMPSW=.TRUE.
2020    CALL RMOVFN(J,JK)
        RETURN

2030    SIMPSW=.TRUE.
2040    CALL MOVEXX (I, IK, J, JK)
        RETURN

2210    K=LAST(I)
        CALL RMOVFN(II,IJK)
        I=ILINK1(K,0,1)
        IK=I
        D(I)=1
        GO TO 2020

2290    I=NEXT(II)
        IK=LASTXX($2000,I,1,0)
        CALL RMOVF1(II)
        J=ILINK1(IK,19,2)
        SIMPSW=.TRUE.
        RETURN 2

2050    IF (.NOT. PRODEX) RETURN 1
        SIMPSW = .TRUE.
        N=D(I)
        D(I)=2
        D(NP)=N
        ITYPB(NP)=51
        ITYP (I) = 20
        K=NEXT(I)
        N=N-1
        DO 2060 L=1,N
            KK = LASTXX($2000,K,1,0)
            K = NEXT(KK)
            I1 = IADDDN(KK,JJ,JJK)
2060    KK = ILINK1(I1,20,2)
            IK=JJK
            RETURN 2

        END

```

RETURN 0 IF C(I).EQ.C(J)

```
1 IF C(I).LT.C(J)
```

```
2 IF C(I).GT.C(J)
```

3 IF ERROR OCCURS

BEFORE RETURN, SET IL AND JL TO THE LAST LINKS OF EXPRESSIONS.

SET POINTERS

• • • • •

COMPARE THE GENERALIZED DEGREES

*--- C(I) LESS THAN C(J)

*--- C(I) GREATER THAN C(J)

*--- DEGREES ARE EQUAL

* INDIVIDUAL TERMS MUST BE CHECKED

```

*      : IF TERMS ARE NOT EQUAL,
*      : EXIT TO 'LESS' OR 'GREATER'

```

SUBROUTINE COMPAR (S,S,I,ID,J,JD,IL,JL)

INCLUDE CMMN

INCLUDE PWORD

LOGICAL RET1

IN=I

JN=J

IDN=ID

JDN=JD

IF (IDN-JDN) 10,30,20

```
10      RET1=.TRUE.  
      GO TO 150
```

```
20      RET1=.FALSE.  
      GO TO 150
```

```
30 IF (IDN.EQ.0) RETURN
```

```

40 IF ((IN*JN).LE.0) RETURN 3

```

IF (ITYP(IN)-ITYP(JN)) 10,50,20

50 IF (H1(IN)-H1(JN)) 10,60,20

60 IF (H2(IN)-H2(JN)) 10,70,20

```

70      K=ITYP(IN)

```

```
IF ((K.EQ.1).OR.(K.EQ.7).OR.(K.EQ.17).OR.(K.EQ.23)) GO TO 100
```

IDN=IDN-1

IF (K,LT,16) GO TO 90

IF (K,LT,24) GO TO 80

IDN=IDN+K-24

GO TO 90

```

80      IDN = IDN + H2(IN)

```

```

80      IDN = IDN
90      JDN=IDN

```


3.2.3. STOUT1

STOUT1 ROUTINE IS USED TO EXECUTE FORMAL FUNCTIONS. THUS ITS MAJOR BREAKDOWN IS ACCORDING TO THE TYPE OF THE FUNCTION AFTER CHECKING THE CORRECTNESS OF THE NUMBER OF ARGUMENTS.

STOUT1 USES THREE AUXILIARY PROGRAMS. PROGRAM ELEMENT INCL IS USED FOR THE EVALUATION OF A COMMON DENOMINATOR (CODEM). THE ROUTINE COLLECTS THE FACTORS FOR THE COMMON DENOMINATOR. WHEN A NEW FACTOR IS GIVEN TO IT, THE ROUTINE CHECKS WHETHER THE FACTOR WAS ALREADY GIVEN BEFORE, OR IF IT IS A MULTIPLE OF A PREVIOUSLY GIVEN FACTOR. PROGRAM ELEMENTS COMPINC AND DIFIN ARE USED IN CONJUNCTION WITH SUBST AND REPLAC OPERATIONS. THEY ARE USED WHEN THE SUBEXPRESSION TO BE REPLACED IS A SUM/PRODUCT OR DIFFERENTIAL EXPRESSION, RESPECTIVELY. THEY CHECK FOR INCLUSION OF SUBEXPRESSIONS. IF THE SUBEXPRESSION TO BE REPLACED IS A SUM OF N TERMS AND THE SUBEXPRESSION TO BE INVESTIGATED IS A SUM OF M TERMS, THE ROUTINE CHECKS WHETHER ALL N TERMS HAVE IDENTICAL TERMS IN THE INVESTIGATED SUBEXPRESSION. IF YES, THEN THE N TERMS FROM THE M TERMS OF THE INVESTIGATED SUBEXPRESSION ARE REMOVED. DIFIN HAS A SIMILAR STRUCTURE. SINCE MULTIPLE DIFFERENTIATIONS APPEAR AS A CONSECUTIVE LIST OF SIMPLE DIFFERENTIATIONS IN THE OUTPUT STACK SUCH THAT A SIMPLE DIFFERENTIATION IS COMPOSED OF THE DIFFERENTIAL OPERATOR FOLLOWED BY THE VARIABLE DIFFERENTIATED BY, THE SAME INCLUSION LOGIC IS USED TO DETECT EQUIVALENCE FOR REPLACEMENT. OBVIOUSLY, DIFIN ALSO CHECKS THE EQUIVALENCE OF THE EXPRESSIONS TO BE DIFFERENTIATED.

DETAILS OF STOUT1, INCL, COMPINC AND DIFIN ARE DESCRIBED IN THEIR LISTINGS.


```
*****  
*                               *  
* STOUT1   *.....>* DPND      *  
*                               *  
*****  
* FNLERR    *  
*                               *  
*                               *  
*****
```

```

.                                     *****
.                                     *
.                                     *
. . . . . > * FMLCHA- *
.                                     * IN *
.                                     *****

```

```

*****
*                                     *
*                                     *
*.....>* DIFIN *.....>* COMP2 *
*                                     *
*****

```

```

*****
*
*
* FNLCHB
*
*****

```


3-134

```

.
.
.
.....>* INCL *.....>* COMP2 *
*
*****
*****

.
.
.
.
.....>* FMLCHA- *
* IN *
*****

.
.
.
.
.....>* FMLCHA- *
* IN1 *
*****

.
.
.
.
.....>* SBEXP *
*
*****

.
.
.
.
.....>* CONSOP *
*
*****
```

2164

FORMAL,STOUT1
EVALUATION OF FORMAL FUNCTIONS

SUBROUTINE STFORM (***,*)
LOGICAL PRODEX
INCLUDE CMMN
INCLUDE PWORD

EXECUTE FORMAL FUNCTIONS

FUNCTION IDENTIFICATION IS IN H1(NP) AND THE ACTUAL
NUMBER OF ARGUMENTS IS H2(NP). THE IDENTIFICATION FIELD IS
IS AS FOLLOWS:

64 + 128 = EXPAND	USE EXPAND OPTION
65 + 128 = CODEM	GET COMMON DENOMINATOR FORM
66 + 256 = COEFF	GET THE COEFFICIENT
67 + 128 = NUM	GET THE NUMERATOR
68 + 128 = DENOM	GET THE DENOMINATOR
69 + 128 = EXPON	GET THE EXPONENT
70 + 128 = BASE	GET THE BASE
71 + 256 = ARG	GET K'ITH ARGUMENT OF LEAD OP.
72 + 640 = SUBST	SUBSTITUTE FOR EXPRESSIONS
73 + 640 = REPLAC	REPLACE EXPRESSIONS
74 + 128 = POWER	...
75 + 128 = PRODEX	DISTRIBUTE EXPONENT OVER PRODUCT
76 + 128 = FLOAT	FLOAT INTEGERS AND RATIONALS
77 + 128 = FLOATB	FLOAT ONLY IN BASE EXPRESSIONS
78 + 256 = FIX	FIX REAL CONSTANTS
79 + 256 = FIXE	FIX REALS ONLY IN EXPONENTS
80 + 128 = LDOP	LEAD OPERATOR ID. AS INTEGER
81 + 128 = NARG	NUMBER OF ARGUMENTS OF LEAD OPERATOR

THE SECOND TERM OF THE IDENTIFICATION, N*128, INDICATES
THE REQUIRED NUMBER OF ARGUMENTS: FOR N=1,2,3 THE REQUIRED
NUMBER OF ARGUMENTS ARE 1,2,3, RESPECTIVELY; FOR N=4 THE
REQUIRED NUMBER OF ARGUMENTS IS EVEN, FOR N=5 THE REQUIRED
NUMBER OF ARGUMENTS IS ODD.

CHECK FOR CORRECT NUMBER OF ARGUMENTS IF THE OPERATOR
HAS BEEN FIRST ENCOUNTERED

.....
BRANCH BY FUNCTION

*
*
*

DIMENSION IA(4,20)

N=FLD(0,11,D(NP))
I=H2(NP)
IF (BITSW) GO TO 5
II=FLD(35,1,I)
IF ((N.LE.3).AND.(N.NE.1)) CALL FMLERR(\$990,'',1,3)
IF ((N.EQ.4).AND.(II.NE.0)) CALL FMLERR(\$990,'',1,3)
IF ((N.EQ.5).AND.(II.NE.1)) CALL FMLERR(\$990,'',1,3)

5 CONTINUE
K=FLD(11,7,D(NP))-63
IF (K .LT. 3 .OR. K .GT. 8) GO TO 6


```

OTHERWISE GET THE POINTER OF ITS
EXPONENT I1-I2 FOR THE BASE, I3-I4
FOR THE EXPONENT AND SET NC FOR THE
NUMBER OF TERMS IN THE EXPONENT

LOOP 3 = FOR THE NC TERMS IN THE
EXPONENT
    TERM POINTER I3-I4
    IF NEGATIVE CONSTANT THEN
        INCLUDE IT IN IC-ICK
    IF IT IS NOT A PRODUCT OR
    PRODUCT WITH NOT A NEGATIVE
    CONSTANT, THEN SKIP THIS TERM
    INCLUDE EXPONENT TERM      BASE
    IN IC-ICK
END OF LOOP 3
END OF LOOP 2
END OF LOOP 1
NOW CHECK COMMON DENOMINATOR
... IT IS EMPTY, REMOVE INITIALIZATION AND
RETURN
... NOT EMPTY, SET M AS NUMBER OF ITS FACTORS

REMOVE INITIAL * IF M=1

```

```

IF (ITYP(J1).NE.20) GO TO 80

I1=NEXT(J1)
I2=LASTXX($990,I1,1,0)
I3=NEXT(I2)
NC=1
IF (ITYP(I3).NE.18) GO TO 50
NC=D(I3)
I3=NEXT(I3)

50 DO 70 LLL=1,NC
   I4=LASTXX($990,I3,1,0)

   KK=1

   IF ((ITYP(I3).LE.3).AND.(D(I3).LT,0)) GO TO 60
   KK=2

   IF (ITYP(I3).NE.19) GO TO 70
   I5=NEXT(I3)
   IF ((ITYP(I5).GT.3).OR.(D(I5).GE,0)) GO TO 70

60 CALL INCLUD($990,IC,ICK,I1,I2,I3,I4,KK)

70 I3=NEXT(I4)

80 J1=NEXT(J2)
90 CONTINUE

IF (D(IC).NE.0) GO TO 100

CALL RMOVF1(IC)
RETURN 1

100 I1=NEXT(IC)
    M=D(IC)
    IF (M.GT. 1) GO TO 110

CALL RMOVF1(IC)
IC=I1

```

```

*      : CLEAR UP THE M FACTORS IN IC-ICK
*      : SINCE THE EXPONENTS ARE IN SUM FORMS
*      :
*      : GET POINTER I2 TO THE + IN EXPONENT
*      :
*      :
*      : IF THE DEGREE OF + IS ONE, THEN REMOVE IT
*      :
*      :
*      :
*      : NOW WE HAVE A CLEARED COMMON DENOMINATOR
*      : IN IC-ICK
*      :
*      : MULTIPLY THE N TERMS OF THE ARGUMENT
*      : BY THE COMMON DENOMINATOR IC-ICK
*      :
*      :
*      : RETURN TO SIMPLIFY THE TERMS MULTIPLIED
*      : BY THE COMMON DENOMINATOR
*      :
*      : SECOND ENCOUNTER OF CODEM (BITSW=.TRUE.)
*      : INSERT EXPONENTIAL OPERATOR AND -1 EXPONENT
*      : FOR THE COMMON DENOMINATOR IC-ICK,
*      :
*      :
*      : INSERT MULTIPLY OPERATOR
*      : AND RETURN TO REMOVE FUNCTION
*      :
*      :
*---- COEFF (K=3)
*      : COEFF[E1,E2] GIVES THE COEFFICIENT OF E2 IN E1
*      : FIND THE POINTERS FOR E1 (J-JK), AND E2 (I-IK)
*      :
*      :
*      :

```

```

110 DO 120 L=1,M
      IF (ITYP(I1).NE.20) CALL ERROR($990,110)
      I2=NEXT(I1)
      I1=LASTXX($990,I2,1,0)
      I2=NEXT(I1)

      IF (ITYP(I2).NE.18) CALL ERROR($990,111)
      I1=NEXT(I2)
      I1=LASTXX($990,I1,1,0)
      IF (D(I2).GT.1) GO TO 120
      CALL RMOVF1(I2)
120 I1=NEXT(I1)

      I2=I

      DO 130 L=1,N
      I1=NEXT(I2)
      I3=ILINK1(I2,19,2)
      I4=IADDNN(I3,IC,ICK)
130 I2=LASTXX($990,I1,1,0)

      H2(NP) = 2
      SIMPSW=.TRUE.
      RETURN 2

140 I=NEXT(NP)
      IK=LASTXX($990,I,1,0)
      IC=NEXT(IK)
      ICK=LASTXX($990,IC,1,0)
      IC=ILINK1(IK,20,2)
      ICK=ILINK1(ICK,0,-1)

      I=ILINK1(NP,19,2)
      RETURN 1

150 I=NEXT(NP)
      IK=LASTXX($990,I,1,0)
      J=NEXT(IK)
      JK=LASTXX($990,J,1,0)

```



```

* IF EITHER E1 OR E2 ARE CONSTANTS, THEN RESULT
* IS ZERO
*

```

```

*
* OTHERWISE GET THE COEFFICIENT AS FOLLOWS:
* MAKE E1 A SUM WITH ONE TERM, UNLESS IT IS
* ALREADY A SUM, THEN CHECK ALL TERMS OF E1 BY
* COMPARING ITS FACTORS TO THE FACTORS OF E2.
* REMOVE THE COMMON FACTORS FROM THE TERM OF E2:
* THE REMAINING PART OF THE TERM IS A TERM OF THE
* COEFFICIENT UNLESS THERE WERE NO MATCHING
* FACTOR. INITIALIZE
* MAKE E1 TO A SUM UNLESS IT IS ALREADY A SUM
* AND GET ITS NUMBER OF TERMS: N
*

```

```

* GET THE NUMBER OF FACTORS IN E2: M
*

```

```

* LOOP1 = FOR THE NUMBER OF TERMS IN E1
*

```

```

* TERM POINTERS: J1-J2
* CHECK THE TYPE OF TERM
*

```

```

* --- THIS TERM IS NOT A PRODUCT, THUS IT
* GIVES A COEFFICIENT 1 IF IT IS
* IDENTICAL TO E2, OTHERWISE IT GIVES
* ZERO
*

```

```

* THEY ARE IDENTICAL, THUS CHANGE
* CONSTANT TO BE INSERTED (K) TO 1
*

```

```

* NOW REMOVE THE TERM AND INSERT THE
* RESULT K
*

```

```

* SKIP THE INSIDE LOOPS AND GET NEXT
* TERM
*

```

```

* --- THE TERM IS A PRODUCT
*

```

```

170 IF ((ITYP(I).GT.3).AND.(ITYP(J).GT.3)) GO TO 180
    L = ILINK1 (NP, 0, 0)
    CALL RMOVFN(J,JK)
    CALL RMOVFN(I,IK)
    RETURN 1

```

```

180 IF (ITYP(J) .NE. 18) J=ILINK1(IK,18,1)
    N=D(J)

```

```

    M=1
    IF (ITYP(I).EQ.19) M=D(I)
    J2=J

```

```

    DO 250 L=1,N
    J1=NEXT(J2)
    J2=LASTXX($990,J1,1,0)

```

```

    K=0
    IF (ITYP(J1).EQ.19) GO TO 200

```

```

    IF (M.NE.1) GO TO 190
    CALL COMPAR($190,$190,$990,J1,1,I,1,J2,IK)

```

```

    K=1

```

```

190 JJ=ILINK1(J2,0,K)
    CALL RMOVFN(J1,J2)
    IF (JK.EQ.J2) JK=JJ
    J2=JJ

```

```

    GO TO 250

```



```

*      : IF ITS NUMBER OF FACTORS, NJ, IS LESS THAN
*      : M, THE NUMBER OF FACTORS IN E2 THEN THE
*      : COEFFICIENT MUST BE ZERO FROM THIS TERM.

```

```

*      : OTHERWISE PREPARE FOR COMPARISON

```

```

*      : LOOP 2 FOR THE NUMBER OF FACTORS IN E2

```

```

*      : FACTOR POINTERS: I1-I2

```

```

*      : LOOP 3 FOR THE NJ FACTORS OF THE TERM
*      : J1-J2

```

```

*      : FACTOR POINTERS: J3-J4
*      : COMPARE THE FACTORS

```

```

*      : IDENTICAL FACTORS: REMOVE IT
*      : FROM THE TERM OF E1 AND EXIT
*      : FROM THE INNER LOOP TO GET NEXT
*      : FACTOR

```

```

*      : NOT IDENTICAL, CONTINUE INSIDE
*      : LOOP

```

```

*      : END OF LOOP 3 INDICATES THAT FACTOR
*      : I1-I2 IS NOT IN THE TERM J1-J2; THUS
*      : THE TERM GIVES ZERO COEFFICIENT

```

```

*      : GET NEXT FACTOR IN E2

```

```

*      : END OF LOOP 2 INDICATES THAT THE TERM
*      : GIVES NON-ZERO COEFFICIENT. THE EQUIVALENT
*      : PART OF E2 HAD BEEN REMOVED FROM THE TERM
*      : J1-J2 CHECK THE NUMBER OF FACTORS LEFT

```

```

*      : ... NO FACTORS LEFT IN J1-J2, THUS THE
*      : TERM GIVES COEFFICIENT 1

```

```

*      : ... ONE FACTOR LEFT IN J1-J2, THUS IT

```

```

200    NJ=D(J1)
        IF (NJ.LT.M) GO TO 190

```

```

        I1=I
        IF (M.GT.1) I1=NEXT(I1)

```

```

        DO 220 LL=1,M
        I2=LASTXX($990,I1,1,0)

```

```

        NJ=D(J1)
        J3=NEXT(J1)

```

```

        DO 210 LLL=1,NJ
        J4=LASTXX($990,J3,1,0)

```

```

        CALL COMPAR($210,$210,$990,I1,1,J3,1,I2,J4)

```

```

        IF (J4.EQ.J2) J2=LAST(J3)
        CALL RMOVFN(J3,J4)
        D(J1)=D(J1)-1
        GO TO 220
        J3=NEXT(J4)

```

```

210

```

```

        GO TO 190

```

```

220    I1=NEXT(I2)

```

```

        IF (D(J1)-1) 230,240,250

```

```

230    K=1
        GO TO 190

```

```

*      :      :      IS A COEFFICIENT,
*      :      :      REMOVE UNNECESSARY * OPERATOR
*      :      :      :
*      :      :      ... IF MORE THAN ONE FACTOR LEFT AS
*      :      :      COEFFICIENT THEN IT IS IN
*      :      :      CORRECT FORM
*      :      :
*      :      :      END OF LOOP 1: E1 CONTAINS THE FINAL
*      :      :      COEFFICIENT
*      :      :
*      :      :      REMOVE OPERATOR + FROM E1 IF IT HAS ONLY ONE
*      :      :      TERM
*      :      :      AND GO TO REMOVE OP. COEFF
*      :      :
*      :      :      --- NUM (K=4)
*      :      :      --- DENOM (K=5)
*      :      :      GET THE NUMERATOR/DENOMINATOR OF AN EXPRESSION
*      :      :      I=IK
*      :      :
*      :      :      IS IT A RATIONAL CONSTANT?
*      :      :      :
*      :      :      ... YES, RETURN NUM OR DENOM
*      :      :      :
*      :      :      :
*      :      :      :
*      :      :      NO, BRANCH BY TYPE
*      :      :      :
*      :      :      ... EXPRESSION IS NEITHER A PRODUCT NOR AN
*      :      :      EXPONENTIAL
*      :      :      ... NUMERATOR IS THE EXPRESSION
*      :      :      :
*      :      :      ... DENOMINATOR IS 1
*      :      :      :
*      :      :      :
*      :      :      EXPRESSION IS AN EXPONENTIAL EXPRESSION
*      :      :      MAKE IT A PRODUCT WITH ONE FACTOR
*      :      :      :
*      :      :      EXPRESSION IS A PRODUCT OF N FACTORS
*      :      :      :
*      :      :      LOOP FOR THE N FACTORS

```

```

240  CALL RMOVF1(J1)

250  CONTINUE

      SIMPSW=.TRUE.
      IF (D(J).GT.1) GO TO 170

      CALL RMOVF1(J)

      GO TO 170

260  I=NEXT(NP)
      IK=LASTXX($990,I,1,0)

      IF (ITYP(I) .NE. 1) GO TO 280

      IF (K .EQ. 5) GO TO 270
      ITP(I) = 0
      CALL RMOVF1 (IK)
      RETURN 1

270  ITP(IK) = 0
      CALL RMOVF1 (I)
      RETURN 1

280  IF (ITYP(I) .EQ. 19) GO TO 300
      IF (ITYP(I).EQ.20) GO TO 290

      IF (K.EQ.4) RETURN 1

      II = ILINK1 (NP, 0, 1)
      CALL RMOVF1 (I,IK)
      RETURN 1

290  I=ILINK1(NP,19,1)

300  N=D(I)
      JK=I

```

THE FACTOR IS AN EXPONENTIAL EXPRESSION,
GET POINTERS

J1-J2 = BASE, J3-J4 = EXPONENT
IF THE EXPONENT IS NOT , CHANGE IT TO A
SUM A SUM BY INSERTING OPERATOR + WITH
DEGREE 1

GET M, THE NUMBER OF TERMS IN THE EXPONENT

LOOP FOR THE M TERMS IN THE EXPONENT

```

THE TERM POINTERS: I1-I2
CHECK THE SIGN OF THE TERM:
    POSITIVE AND K=4   ---> STAYS
    POSITIVE AND K=5   ---> REMOVE
    NEGATIVE AND K=4   ---> REMOVE
    NEGATIVE AND K=5   ---> CHANGE TO +

```

IF IT IS A PRODUCT, THEN STEP UP WITH
POINTER

NOW DETERMINE SIGN BY THE TYPE OF II
IF IT IS NOT A CONSTANT, THEN IT IS
POSITIVE OTHERWISE THE SIGN OF THE
CONSTANT

BRANCH BY COMBINATION

... REMOVE THE TERM

```
DO 390 L=1,N
J=NEXT(J,K)
JK=LASTXX($990,J,1,0)
```

```
IF (ITYP(J),NE,20) GO TO 380
```

```
J1=NEXT(J)
J2=LASTXX($990,J1,1,0)
J3=NEXT(J2)
J4=LASTXX($990,J3,1,0)
```

```
IF (ITYP(J3),NE,18) J3=ILINK1(J2,18,1)
```

```

M=D(J3)
I2=J3

```

```
DO 350 LL=1,M
  I1=NEXT(I2)
  I2=LASTXX($990,I1,1,0)
```

II=I1

```
IF (ITYP(I1).EQ.19) II=NEXT(I1)
```

```
IF (ITYP(II).GT.3) GO TO 310
IF (D(II).LT.0) GO TO 340
```

```
310 IF (K.EQ.4) GO TO 350
```

```

320      II=LAST(I1)
      CALL RMOVFN(I1,I2)
      IF (J4,NE,I2) GO TO 330
      IF (JK,EQ,I2) JK=II
      J4=II

```

[illegible]

```

330      I2=II
          D(J3) = D(J3) - 1
          GO TO 350
340      IF (K.EQ.4) GO TO 320

          D(II)=-D(II)
350      CONTINUE

          IF (D(J3)-1) 360,370,390

360      II=LAST(J)
          CALL RMOVFN(J,JK)
          JK=II
          D(I)=D(I)-1
          GO TO 390

370      CALL RMOVF1 (J3)
          GO TO 390

380      IF (K.EQ.5) GO TO 360
390      CONTINUE

          IF (D(I)-1) 400,410,420

400      J=ILINK1(NP,0,1)

410      CALL RMOVF1(I)

420      SIMPSW=.TRUE.
          RETURN 1

430      I=NEXT(NP)
          IF (ITYP(I).EQ.20) GO TO 440

          IF (K .EQ. 7) RETURN 1

          IK=LASTXX($990,I,1,0)
          CALL RMOVFN(I,IK)

```

```

*      :
*      :
*      :... EXPRESSION IS EXPONENTIAL
*      : SEPARATE BASE AN EXPONENT
*      :
*      :
*      : RETURN BASE
*      :
*      :
*      : RETURN EXPONENT
*      :
*      :
*---- ARG (K=8)
*---- ARG[E,N] GIVES THE N'TH ARGUMENT IN E
*
*      :
*      : IF N IS NOT AN INTEGER, THEN IT IS AN ERROR
*      :
*      :
*      : EXPRESSION POINTERS: J-JK
*      : WITH J AS THE LEAD OPERATOR POINTER
*      :
*      :
*      : CHECK THE LEAD OPERATOR
*      :
*      :
*      : ... E HAS ONLY ONE ARGUMENT
*      : E IS THE RESULT IF N=1
*      :
*      : OTHERWISE GENERATE ZERO AS RESULT
*      :
*      :
*
*      : OTHERWISE GET THE NUMBER OF ARGUMENTS, M, OF
*      : THE LEAD OPERATOR
*      :
*      :
*      : IF N IS NOT BETWEEN 1 AND M, THEN GENERATE ZERO
*      : FOR RESULT
*      :
*      :
*      : GET THE POINTERS OF THE N'TH ARGUMENT
*      : AND REMOVE THE OTHERS
*      :
*      :
*      :

```

```

I=ILINK1(NP,0,1)
RETURN 1

440 J=NEXT(I)
JK=LASTXX($990,J,1,0)
IF (K.EQ.6) GO TO 450

CALL RMOVF1(I)
I=NEXT(JK)
JK=LASTXX($990,I,1,0)

450 CALL RMOVFN(I,JK)
RETURN 1

460 I=NEXT(NP)

IF (ITYP(I).NE.0) CALL FMLERR($990,'ARG',1,13)
N=D(I)

J=NEXT(I)
CALL RMOVF1(I)
JK=LASTXX($990,J,1,0)

IF (ITYP(J).GE.16) GO TO 490

IF (N.EQ.1) RETURN 1

470 I=ILINK1(NP,0,0)
480 CALL RMOVFN(J,JK)
RETURN 1

490 M=H2(J)
IF (ITYP(J).GE.24) M=ITYP(J)-24

IF ((N.LE.0).OR.(N.GT.M)) GO TO 470

IK=J
DO 510 L=1,N
IF (L.NE.N) GO TO 500
I=NEXT(IK)
CALL RMOVFN(J,IK)

```


REMOVE THE PAIRS

GET THE POINTERS OF THE EXPRESSION: J-JK

INITIALIZE J1-J2 AS THE FIRST SUBEXPRESSION
TO BE COMPARED AS THE FULL EXPRESSION J-JK

LOOP TO COMPARE SUBEXPRESSION J1-J2 WITH THE
ARGUMENTS

BRANCH BY TYPE OF THE I' TH ARGUMENT

... DIFFERENTIAL OPERATOR, CALL DIFCMP

... SUM OR PRODUCT, CALL CMPRMV

... ELSE, CALL COMPAR

RETURNS FROM COMPARES

... WHEN IT WAS NOT A PRODUCT OR SUM AND
THEY ARE EQUIVALENT, THUS MAKE
REPLACEMENT

CHECK IF THIS IS THE END OF THE
EXPRESSION J-JK IF YES, THEN THE JOB
IS FINISHED,

OTHERWISE GET NEXT SUBEXPRESSION FROM
J-JK

... IT IS A PRODUCT OR SUM, AND
EQUIVALENT PART WAS FOUND IN J1-J2
WHICH HAD BEEN REMOVED BY CMPRMV THE
REMAINING PART MUST BE CHECKED. I2
HAD BEEN SET TO THE END OF THE
REMAINDER

INSERT REPLACEMENT

CLEAR IT UP WITH THE NUMBER OF
TERMS/FACTORS LEFT

THE EXPRESSIONS DO NOT COMPARE

CALL REMOVN(IA(1,N),JK)

J=NEXT(NP)
JK=LASTXX(\$990,J,1,0)

J1=J
J2=JK

560 DO 590 I=1,N

I1=ITYP(IA(3,I))

IF (I1 .EQ. 23)
1 CALL DIFCMP(\$570,\$590,\$990,J1,J2,IA(3,I),IA(4,I))

IF ((I1 .EQ. 18) .OR. (I1 .EQ. 19))
1 CALL CMPRMV(\$580,\$590,\$990,J1,J2,IA(3,I),IA(4,I),I2)

CALL COMPAR(\$590,\$590,\$990,J1,1,IA(3,I),1,J2,IA(4,I))

570 J4=J2
J2=IADDNN(J2,IA(1,I),IA(2,I))
CALL RMOVFN(J1,J4)

IF (J4 .EQ. JK) GO TO 650

GO TO 610

580 L=LAST(J1)
IF (D(J1) .NE. 0) L=ILINK1(L,I1,2)

L=IADDNN(L,IA(1,I),IA(2,I))

IF (D(J1)-1) 620,630,640


```

* GET INTEGER ARGUMENT, IJ, AND REMOVE IT
*
*
*
*
*
*

```

```

* GET EXPRESSION POINTERS: J=JK
* SET IK TO J FOR FIX, TO JK FOR FIXE.
*
*
*

```

```

* IS IT FIX
* .... YES, GO TO INNER PART OF LOOP
*
* .... NO, , IT IS FIXE
*

```

```

* OUTER PART OF LOOP
* IS THE ENTRY AN EXPONENTIAL OPERATOR
* .... NO, GO TO CHECK FOR END
*
* .... YES, SKIP BASE PART AND GO TO
* INNER PART OF THE LOOP
*

```

```

* INNER PART OF THE LOOP
* IS THIS A REAL CONSTANT
* .... NO, GO TO GET NEXT ENTRY
*
* .... YES, CHECK IF IT SHOULD BE CONVERTED
*
* .... NO, GO TO GET NEXT ENTRY
* .... YES, CHANGE IT TO INTEGER
*

```

```

* GET NEXT ENTRY, BUT FIRST CHECK FOR END
*
* .... YES, GO TO CHECK FOR FINAL END
* .... NO, GET NEXT ENTRY AND CONTINUE IN
* INNER PART
*

```

```

* END OF INNER PART
* CHECK FOR FINAL END
* .... YES, RETURN
* .... NO, GET NEXT ENTRY AND CONTINUE IN OUTER
* PART
*

```

```

800 I=NEXT(NP)
    IF (ITYP(I) .NE. 0) CALL ERRERR ($990,800)
    IJ=D(I)
    CALL RMOVF1(I)
    I=NEXT(NP)

```

```

    IK=I
    J=I
    JK=LASTXX($990,I,1,0)

```

```

    IF (K .EQ. 15) GO TO 820

```

```

    IK=JK

```

```

810 IF (ITYP(I) .NE. 20) GO TO 840

```

```

    J=NEXT(I)
    J=LASTXX($990,J,1,0)
    J=NEXT(J)
    JK=LASTXX($990,J,1,0)

```

```

820 IF (ITYP(J) .NE. 3) GO TO 830

```

```

    ID = ISIGN(IFIX(BOOL(ABS(D(J))))+5,D(J))
    IF (ABS(BOOL(D(J))- FLOAT(ID)) .GT. 10,**IJ) GO TO 830

```

```

    ITYP(J) = 0
    D(J) = ID

```

```

830 IF (J .EQ. JK) GO TO 840

```

```

    J=NEXT(J)
    GO TO 820

```

```

840 IF (I .EQ. IK) RETURN 1

```


FORMAL,INCL

CALCULATE THE GREATEST COMMON MULTIPLE OF 2 EXPRESSIONS

SUBROUTINE INCLUD(S,IC,ICK,I1,I2,I3,I4,K)
 INCLUDE CMMN
 INCLUDE PWORD

IC-ICK IS A PRODUCT OF EXPONENTIAL EXPRESSIONS

WITH POSITIVE EXPONENTS, MAY BE EMPTY (D(IC)=0)

GIVEN EXPONENTIAL EXPRESSION (BASE = I1-I2, EXP.=I3-I4),

WHERE THE EXP. IS EITHER A NEGATIVE CONSTANT (K=1)

OR A PRODUCT WITH NEGATIVE CONSTANT FACTOR (K=2),

IC-ICK IS MODIFIED (ENLARGED) AS FOLLOWS

1. IF IC-ICK DOES NOT HAVE A BASE IDENTICAL TO I1-I2,

THEN $(I1-I2)^{((-1)*(I3-I4))}$ IS INSERTED IN IC-ICK

(THE EXPONENTIAL EXPR. WITH POSITIVE EXPONENT)

2. IF IC-ICK HAS A BASE IDENTICAL TO I1-I2, THEN THE

EXPONENTS ARE INVESTIGATED

2.A. I3-I4=NEGATIVE CONSTANT C(I).

IF THE SAME BASE DOES NOT HAVE CONSTANT EXPONENT IN

IC-ICK THEN -C(I) IS INCLUDED

IF IT HAS, C(J), THEN IF $(C(J)+C(I)).GE.0$ THEN

DO NOTHING, ELSE REPLACE C(J) WITH -C(I).

2.B I3-I4 = IS A PRODUCT WITH NEGATIVE CONSTANT FACTOR

REMOVE CONSTANT C(I) AND SEARCH FOR THE SAME

EXPRESSION (WITHOUT CONSTANT) IN THE EXPONENTS

OF THE SAME BASE OF IC-ICK. IF FOUND, DO THE

SAME WITH THE CONSTANTS AS BEFORE. IF NOT

FOUND INCLUDE THE NEW EXPONENT WITH -C(I).

THIS ROUTINE IS USED TO OBTAIN THE GREATEST COMMON MULTIPLE

TO DEFINE COMMON DENOMINATOR.

RETURN 1 = ERROR RETURN

CHECK WHETHER IC-ICK IS EMPTY

... YES, THEN INSERT

IF (D(IC).GT.0) GO TO 30

10 D(IC)=D(IC)+1
 ICK=ILINK1(ICK,20,2)
 ICK=IADDNN(ICK,I1,I2)
 ICK=ILINK1(ICK,18,1)
 II=ICK
 ICK=IADDNN(ICK,I3,I4)
 II=NEXT(II)
 IF (K.EQ.2) II=NEXT(II)
 D(II)=-D(II)

RETURN

RETURN

20 RETURN 1

... NO, LOOP TO COMPARE BASES

30 N=D(IC)
 J2=IC


```

*      : RETURN
*      :
*---- K=2, CHECK FOR PRODUCT EXPONENT
:      : CHAIN OUT CONSTANT
:      :
:      : SET IS=2 FOR NOT EQUAL, THEN
:      : COMPARE PRODUCTS
:      :
:      : THEY ARE EQUAL, RESET IS = 1, AND ADD
:      : CONSTANTS
:      :
:      : JUMP BY 'IS'
:..... NO CHANGE IN CONSTANT
:      :
:      : ... EXPONENTS NOT EQUAL, CONTINUE
:      : LOOP
:      :
:      : ... INSERT NEW CONSTANT
:      :
:      : CHECK RELATION OF CONSTANTS AND GO TO
:      : JUMP BY 'IS'
:      :
:      : END OF LOOP
EXPONENT IS NOT IN THE TERMS
INSERT NEW TERM
:
: BRANCH ACCORDING TO 'K'
*
*---- LINK CONSTANT BACK INTO THE PRODUCT

```

```

GO TO 80
95  IF (ITYP(J1).NE.19) GO TO 150
    K3=NEXT(J1)
    K4=LASTXX($20,K3,1,0)
    CALL REMOVN(K3,K4)
    D(J1)=D(J1)-1
    IS=2
    CALL COMPAR($100,$100,$20,I3,1,J1,1,I4,J2)
    IS=1
    CALL CONSOP($140,$140,$140,$20,1,K1,K3,J22)
100  D(J1)=D(J1)+1
    GO TO (110,120,130),IS
110  K2=ILINKN(I3,K1,K2)
    D(I3)=D(I3)+1
    K4=ILINKN(J1,K3,K4)
    GO TO 80
120  K4=ILINKN(J1,K3,K4)
    GO TO 150
130  CALL IFREEN(K3,K4)
    K3=IADDNN(J1,K1,K2)
    K2=ILINKN(I3,K1,K2)
    D(I3)=D(I3)+1
    K3=NEXT(J1)
    D(K3)=-D(K3)
    GO TO 80
140  IF (D(J22).GT.0) GO TO 100
    IS=3
    GO TO 100
150  CONTINUE
    D(J)=D(J)+1
    GO TO (170,160),K

```

```

*      :
*      :
*--- LINK NEW EXPONENT
*      :
*      :
*      :
*      :
*      :
*      :
RETURN
:
END

```

```

160  D(I3)=D(I3)+1
      K2=ILINKN(I3,K1,K2)

170  K3=IADDNN(J2,I3,I4)
      K4=NEXT(J2)
      IF (K.EQ.2) K4=NEXT(K4)
      D(K4)=-D(K4)
      IF (J2.EQ.ICK) ICK=K3

      RETURN

      END

```


FORMAL,COMPINC

 EXPRESSION COMPARISON AND REMOVAL ROUTINE

GIVEN TWO EXPRESSIONS, J1-J2 AND I1-I2, SUCH THAT
 EXPRESSION I1-I2 IS EITHER A SUM OR A PRODUCT. THE
 SUBROUTINE CHECKS WHETHER EXPRESSION I1-I2 IS
 INCLUDED IN EXPRESSION J1-J2 WHEN I1-I2 IS EITHER
 A SUM OR A PRODUCT. IF IT IS NOT INCLUDED, A RETURN
 2 OCCURS. IF IT IS INCLUDED, THEN IT IS REMOVED
 FROM J1-J2, AND J1, J2 ARE MODIFIED BEFORE A
 RETURN 1 OCCURS. RETURN 3 IS AN ERROR RETURN.

SET UP LOOP FOR M TERMS/FACTORS OF I1-I2

LOOP TO COMPARE M SORTED TERMS / FACTORS

GET NEXT TERM / FACTOR IN I1-I2

GET NEXT TERM / FACTOR IN J1-J2
 END OF J1-J2, EXIT TO 'NOT INCLUDED'

COMPARE THEM

... NOT IDENTICAL, GO TO GET NEXT TERM/FACTOR
 IN J1-J2

... IDENTICAL, MARK IT IN IBIT

END OF LOOP

NORMAL EXIT = ALL TERMS/FACTORS WERE FOUND
 I1-I2 IS INCLUDED IN J1-J2

LOOP TO REMOVE MARKED SUBEXPRESSIONS

END OF LOOP

SUBROUTINE CMPRMV(\$,\$,\$,J1,J2,I1,I2,JJ)
 INCLUDE CMMN
 INCLUDE PWORD

IF ((ITYP(J1) .NE. ITP(I1)) .OR. (D(J1) .LT. D(I1))) RETURN 2

M=D(I1)

N=0

I4 = I1

J4 = J1

DO 130 L = 1,M

I3 = NEXT (I4)

I4 = LASTXX (\$990, I3, 1, 0)

100 IF (J2 .EQ. J4) GO TO 150

J3 = NEXT (J4)

J4 = LASTXX (\$990, J3, 1, 0)

CALL COMPAR (\$100, \$100, \$990, I3, 1, J3, 1, I4, J4)

IBIT (J3) = 1

130 N = N+1

J3=NEXT(J1)

K=D(J1)

DO 140 L=1,K

J4=LASTXX(\$990,J3,1,0)

I3=NEXT(J4)

IF (IBIT(J3) .EQ. 1) CALL RMOVFN(J3,J4)

140 J3=I3


```

: CHECK DEGREE AND SPECIFY END OF THE NEW
: EXPRESSION=JJ
:
: RETURN 1 TO INDICATE I1-I2 IS INCLUDED IN J1-J2
:
: ERROR RETURN
:
NOT INCLUDED, CLEAR IBITS
:
:
:
:
RETURN 2
:
END

```

```

D(J1)=K-M
JJ=J1

IF (D(J1) .EQ. 0) RETURN 1
JJ=LASTXX($990,J1,1,0)
RETURN 1

990 RETURN 3

150 IF (N .EQ. 0) RETURN 2
J3=NEXT(J1)
K=D(J1)
DO 160 L=1,K
IBIT(J3)=0
J4=LASTXX($990,J3,1,0)
160 J3=NEXT(J4)

RETURN 2

END

```

FORMAL.DIFIN

COMPARISON ROUTINE FOR DIFF. EXPRESSIONS

EXPRESSION I1-I2 IS A DIFFERENTIAL EXPRESSION. THE ROUTINE
CHECKS IF EXPRESSION J1-J2 IS IDENTICAL TO I1-I2 OR IT
IS A DIFFERENTIAL QUOTIENT OF I1-I2, OR NONE OF ABOVE.

RETURN 0 = I1-I2, J1-J2 HAVE THE SAME NUMBER OF DIFFERENTIAL
OPERATORS. NO CHANGE

RETURN 1 = J1-J2 IS A DIFFERENTIAL QUOTIENT OF I1-I2.
THE COMMON PART OF J1-J2 IS REMOVED FROM IT, J1 IS
RESET TO THE BEGINING OF THE REMAINING PART.

RETURN 2 = CONDITIONS DO NOT SATISFY, NO CHANGE

RETURN 3 = ERROR RETURN

THIS ROUTINE IS USED FOR THE EXECUTION OF SUBST AND REPLAC
FUNCTIONS.

IF J1-J2 IS NOT A DIFFERENTIAL EXPRESSION, THEN
RETURN 2. OTHERWISE INITIALIZE FOR COMPARISON

GET NUMBER OF DIFFERENTIAL OPERATORS

NOW I-IK, J-JK ARE THE EXPRESSIONS DIFFERENTIATED
BY NI, NJ TIMES, RESPECTIVELY. IF NI=NJ THEN RETURN,
IF NI.GT.NJ THEN RETURN 2, IF NI.LT.NJ THEN CONTINUE
CHECKING FOR INCLUSION

COMPARE THE 2 EXPRESSIONS, I-IK, J-JK, FIRST

SUBROUTINE DIFCMP(*,*,*,J1,J2,I1,I2)

INCLUDE CMMN

INCLUDE PWORD

IF (ITYP(J1) .NE. 23) RETURN 2

J=J1

JK=J2

I=I1

IK=I2

NI=0

NJ=0

10 IF (ITYP(I) .NE. 23) GO TO 20

NI=NI+1

I=NEXT(I)

IF (ITYP(I) .EQ. 7) I=NEXT(I)

I=NEXT(I)

GO TO 10

20 IF (ITYP(J) .NE. 23) GO TO 30

NJ=NJ+1

J=NEXT(J)

IF (ITYP(J) .EQ. 7) J=NEXT(J)

J=NEXT(J)

GO TO 20

30 IF (NJ-NI) 40,50,60

40 RETURN 2

50 RETURN

```

:,... NOT IDENTICAL - RETURN 2
:
:,... THEY ARE IDENTICAL, THUS CHECK DIFF. VARIABLES.
: INITIALIZE LOOP FOR NI DIFF OPERATORS
: OUTER LOOP FOR NI DIFF. VARIABLES, I3-I4
:
: SKIP DIFF. OPERATOR AND SET UP LOOP
: TO CHECK VARIABLE
:
: INNER LOOP FOR NI DIFF. VARIABLES.
: J3-J4
:
: NOW CHECK THE TWO VARIABLES FOR
: IDENTITY UNLESS J1-J4 HAD AN
: EQUIVALENCE BEFORE.
:
: ... THEY ARE IDENTICAL, SET
: AND GO OUT FROM THE INNER
: LOOP
:
: THEY ARE NOT IDENTICAL,
: PROCEED TO NEXT J VARIABLE
:
: NORMAL EXIT FROM THE INNER LOOP MEANS.
: AN I-VARIABLE DOES NOT HAVE A
: COUNTERPART IN J-EXPRESSION, THUS
: CLEAR IBITS IN EXPRESSION J AND
: RETURN 2
:
: PROCEED TO NEXT I-VARIABLE
:
: NORMAL EXIT FROM THE OUTER LOOP MEANS THAT
: J1-J2 CONTAINS I1-I2, THE IDENTICAL DIFF.

```

```

60 CALL COMPAR($40,$40,$990,I,1,J,1,IK,JK)

I4=I1
DO 110 L=1,NI

I3=NEXT(I4)
I4=I3
IF (ITYP(I3) .EQ. 7) I4=NEXT(I4)
J4=J1

DO 80 LL=1,NJ
J3=NEXT(J4)
J4=J3
IF (ITYP(J3) .EQ. 7) J4=NEXT(J4)

IF (IBIT(J3) .EQ. 1) GO TO 70
CALL COMPAR($70,$70,$990,I3,1,J3,1,I4,J4)

IBIT(LAST(J3))=1
IBIT(J3)=1
IBIT(J4)=1
GO TO 100

70 J4=NEXT(J4)
80 CONTINUE

J3=J1
90 IF (J3 .EQ. J) RETURN 2
IBIT(J3)=0
J3=NEXT(J3)
GO TO 90

100 I4=NEXT(I4)
110 CONTINUE

```


3.2.4. STOUT2

STOUT2 ROUTINE CHECKS THE MATHEMATICAL FUNCTIONS FOR SIMPLIFICATIONS. THE ROUTINE USES FORTRAN LIBRARY PROGRAMS TO CALCULATE THE FUNCTION VALUES WHEN THEIR ARGUMENTS ARE CONSTANTS. OTHERWISE THE ROUTINE CHECKS FOR SIMPLIFICATION ACCORDING TO THE TYPE OF FUNCTION. DETAILS APPEARS WITH THE LISTING OF THE PROGRAM.

[illegible]

FORMAL,STOUT2
EVALUATION OF MATHEMATICAL FUNCTIONS

SUBROUTINE STMATH (**,*)
INCLUDE CMMN
INCLUDE PWORD

MATHEMATICAL FUNCTIONS

THIS ROUTINE WRITES OUT OR EXECUTES THE MATHEMATICAL FUNCTIONS.
THE FUNCTION IDENTIFICATION IS IN H1(NP), WHICH MUST BE
ONE OF THE FOLLOWING:

32 + 128 FOR EXP
33 + 128 '' LOG
34 + 128 '' TNH
35 + 128 '' SIN
36 + 128 '' COS
37 + 128 '' ATN
50 + 128 '' FAC
51 + 256 '' BIN
52 + 384 '' STEP

WHERE N*128 INDICATES THE CORRECT NUMBER OF ARGUMENTS, N,
THE ACTUAL NUMBER OF ARGUMENTS IS IN H2(NP).
THE RETURNS FROM THE ROUTINE ARE AS FOLLOWS:

RETURN 0 = NOT USED

1 = REMOVE TOP OPERATOR FROM STACK

2 = WRITE OUT OPERATOR FROM STACK

3 = ERROR

GET IDENTIFICATION NUMBER AND CHECK NUMBER OF ARGUMENTS

:

GET FIRST ARGUMENT, I-IK, AND ITS TYPE, IJ

:

BRANCH TO INTEGER FUNCTIONS

:

IF ARGUMENT IS NOT A CONSTANT, BRANCH TO MATH.
FUNCTIONS

:

IF MATHSW OPTION IS OFF, DO NOT EVALUATE
MATHEMATICAL FUNCTION WITH CONSTANT ARGUMENT,

:

CONSTANT MATHEMATICAL FUNCTION

CONSTANT ARGUMENT FOR MATHEMATICAL FUNCTIONS, GET
REAL EQUIVALENCE BEFORE BRANCHING

:

REAL FR

N=H2(NP)

IF (FLD(0,11,D(NP)),NE,N) CALL FMLEHR(\$990,'',1,3)

K=FLD(11,7,D(NP))-31

IF (K.GT.6) K=Y-12

I=NEXT(NP)

IJ=ITYP(I)

IK=LASTXX(\$990,I,1,0)

IF (K.GT.6) GO TO 100

IF (IJ.GT.3) GO TO (280,360,400,400,410,400),K

IF (.NOT. MATHSW) RETURN 2

FR = BOOL (D(I))

3-163

NOW CHECK INTGSW

DO NOT EVALUATE, THUS RETURN

EVALUATE, BRANCH BY INTEGER FUNCTION

... FAC (K=7)
EVALUATE ONLY FOR POSITIVE INTEGER ARGUMENTS
LESS THAN 11

GENERATE INTEGER RESULT FOR THE FACTORIAL

... BIN (K=8)
EVALUATE (N,0), (N,1), (N,N) OR IF BOTH
ARGUMENTS ARE INTEGERS

FIRST ARGUMENT IS NOT AN INTEGER

SECOND ARGUMENT IS AN INTEGER

... IT IS 1, THUS RESULT IS BIN(N,1)=N

... SECOND ARGUMENT IS 0, THUS RESULT IS
BIN(N,0) = 1

... SECOND ARGUMENT IS NOT INTEGER
COMPARE IT TO THE FIRST

... IT IS THE SAME, THUS RESULT IS
BIN(N,N) = 1

... NOT THE SAME, THUS DO NOT EVALUATE

... CONSTANT ARGUMENTS
IF NOT INTEGER, RETURN

NOW TRY TO EVALUATE BIN(N,K)

CHECK THE RANGE OF K

120 IF (.NOT.INTGSW) RETURN 2

IF (K - 8) 130, 150, 220

130 IF (IJ.NE.0) RETURN 2

N=D(I)

IF ((N.LT.0).OR.(N.GT.10)) RETURN 2

J=ILINK1(NP,0,1)

140 IF (N.LE.1) GO TO 95

D(J)=D(J)*N

N=N-1

GO TO 140

150 IF (ITYP(I).EQ.0) GO TO 200

IF (ITYP(J).NE.0) GO TO 180

IF (D(J).NE.1) GO TO 160

CALL RMOVF1(J)

RETURN 1

160 IF (D(J).NE.0) RETURN 2

170 IK=JK

L=ILINK1(NP,0,1)

GO TO 95

180 CALL COMPAR(\$190,\$190,\$990,I,1,J,1,IK,JK)

GO TO 170

190 RETURN 2

200 IF (ITYP(J).NE.0) RETURN 2

N=D(I)

K=D(J)

CHECK THE TYPE OF ARGUMENT

.... IT IS 'LOG', THUS THEY CANCEL

.... NEITHER SUM NOR PRODUCT, RETURN

.... SUM OR PRODUCT

CHECK FOR 'LOG' TERM/FACTOR

GET THE NUMBER OF TERMS/FACTORS, N

LOOP FOR THE N TERMS/FACTORS

TERM/FACTOR POINTERS: J-JK

LOG TERM/FACTOR FOUND, REMOVE 'LOG'
OPERATOR AND ARGUMENT, JJ-JJK

IF N IS 2, THEN REMOVE LEAD OPERATOR
+ OR * OTHERWISE DECREASE ITS DEGREE

NOW BRANCH BY LEAD OPERATOR TO MAKE
TRANSFORMATION

.... +, USE RULE:

$\text{EXP}[X+\text{LOG}[Y]] = Y * \text{EXP}[X]$

.... *, USE RULE: $\text{EXP}[X * \text{LOG}[Y]] = Y^{**}X$

CHECK IF THERE IS A CONSTANT TERM IN
THE SUM

.... CONSTANT TERM IN SUM, USE
TRANSFORMATION

$\text{EXPLC}+X = \text{EXPLC} * \text{EXPEX}$

280 IF ((IJ.NE.21).OR.(H1(I).NE.161)) GO TO 290

CALL RMOVF1(I)
RETURN 1

290 IF (EXPDSW.EQ.0.OR.IJ.LT.18.OR.IJ.GT.19) RETURN 2

N=D(I)
JK=I
NJ=IJ-17

DO 350 L=1,N
J=NEXT(JK)
JK=LASTXX(\$990,J,1,0)

IF ((ITYP(J).NE.21).OR.(H1(J).NE.161)) GO TO 330

SIMPSW=.TRUE.
JJ=NEXT(J)
CALL RMOVF1(J)
CALL REMOVN(JJ,JK)

D(I)=D(I)-1
IF (D(I).GE.2) GO TO 300
CALL RMOVF1(I)

300 GO TO (310,320),NJ

310 I=ILINK1(NP,21,1)
H1(I)=160
I=ILINKN(NP,JJ,JK)
I=ILINK1(NP,19,2)
RETURN 1

320 I=ILINKN(NP,JJ,JK)
I=ILINK1(NP,20,2)
RETURN 1

330 IF ((ITYP(J).GT.3).OR.(NJ.NE.1)) GO TO 350

[illegible]

```

CALL REMOVE(N(J,JK)
D(I)=D(I)-1
IF (D(I).GE.2) GO TO 340
CALL REMOVE1(I)
340 I=ILINK1(NP,21,1)
H1(I)=160
I=ILINKN(NP,J,JK)
I=ILINK1(NP,21,1)
H1(I)=160
I=ILINK1(NP,19,2)
SIMPSW=.TRUE.
RETURN 1
350 CONTINUE

RETURN 2

360 IF ((IJ.NE.21).OR.(H1(I).NE.160)) GO TO 370

CALL REMOVE1(I)
RETURN 1
370 IF (IJ .NE. 19 .OR. EXPDSW ,EQ. 0) GO TO 390


N=D(I)
JK=I
DO 380 L=1,N
J=NEXT(JK)
K=ILINK1(JK,21,1)
H1(K)=161
380 JK=LASTXX($990,J,1,0)
ITYP(I)=18
SIMPSW=.TRUE.
RETURN 1
390 IF (IJ.NE.20) RETURN 2


ITYP(I)=19
K=ILINK1(I,21,1)
H1(K)=161
SIMPSW=.TRUE.
RETURN 1

400 NJ=-1
GO TO 420

```

```

:
: CHECK FOR TRANSFORMATION: COSC-1*X]=COS[X]
:

```

```

: RETURN IF ARGUMENT IS NOT A PRODUCT
:

```

```

: RETURN IF FACTOR IS NOT -1
:

```

```

: PERFORM TRANSFORMATION
:

```

```

: ERROR RETURN
:

```

```

: END
:

```

```

410 NJ=1

```

```

420 IF (IJ.NE.19) RETURN 2
      J=NEXT(I)

```

```

      IF ((ITYP(J).NE.0).OR.(D(J).NE.-1)) RETURN 2

```

```

      CALL RMOVF1(J)

```

```

      D(I)=D(I)-1

```

```

      IF (D(I).GT.1) GO TO 430

```

```

      CALL RMOVF1(I)

```

```

430 I=ILINK1(NP,21,0)

```

```

      D(I)=D(NP)

```

```

      IF (NJ.EQ.1) RETURN 1

```

```

      I=ILINK1(NP,0,NJ)

```

```

      I=ILINK1(NP,19,2)

```

```

      RETURN 1

```

```

990 RETURN 3

```

```

END

```

3.2.5. STOUT3

STOUT3 ROUTINE SUBSTITUTES THE GIVEN ARGUMENT EXPRESSIONS INTO THE DEFINING EXPRESSION OF A DEFINED FUNCTION. THESE EXPRESSIONS WERE PLACED INTO THE OUTPUT STACK BY EXPRES. DETAILS OF THE ROUTINE ARE DESCRIBED IN ITS LISTING.

```
*****  
*                               *  
* STOUT3   *. . . . .>* FMLERR    *  
*                               *  
*****  
  
.  
. .  
. .  
. .  
. .  
*. . . . .>* FMLCHB    *  
*                               *  
*. . . . .>* SBEXP     *  
*                               *  
*. . . . .>* FMLCHA    *  
*                               *
```

FORMAL,STOUT3
EVALUATION OF DEFINED FUNCTIONS

SUBROUTINE STFCTD (**,**)
 INCLUDE CMMN
 INCLUDE PWORD

 DEFINED FUNCTIONS

THIS ROUTINE EXECUTES THE DEFINED FUNCTIONS FROM THE PUSH-
DOWN STACK. BITSW MUST BE FALSE. THE NUMBER OF ARGUMENTS IS H2(NP)
WHICH ARE WRITTEN OUT FOLLOWING NP. THE DEFINING EXPRESSION OF
THE FUNCTION FOLLOWS THE ARGUMENTS. SINCE THE FUNCTION IS EN-
COUNTERED THE FIRST TIME, ITS ARGUMENTS ARE IN REVERSED ORDER.
MAXIMUM 20 ARGUMENTS ARE ALLOWED.

RETURN 3 IS AN ERROR RETURN.

NORMAL RETURN OCCURS ALWAYS WITH RETURN 1 WITH SIMPSW SET TO
TRUE.

GET THE NUMBER OF GIVEN ARGUMENTS, N	DIMENSION ICH(2,20)
:	N=H2(NP)
:	IF (BITSW.OR.(N.GT.20)) CALL FMLERR(\$990,'',1,3)
ESTABLISH POINTER TABLE FOR THE ARGUMENTS IN ICH	
:	JK=NP
:	DO 10 I=1,N
:	K=N+1-I
:	J=NEXT(JK)
:	JK=LASTXX(\$990,J,1,0)
:	ICH(1,K)=J
:	10 ICH(2,K)=JK
REMOVE THE ARGUMENTS	
GET THE POINTERS, I-IK, OF THE EXPRESSION	
:	CALL FMLCHE (\$990, ICH(1,N), ICH(2,1))
:	IL=NP
:	I=NEXT(NP)
:	IK=LASTXX(\$990,I,1,0)
LOOP TO INSERT THE ARGUMENTS	
:	20 I=NEXT(IL)
IF THIS IS NOT AN ARGUMENT, SKIP IT	
:	IF (ITYP(I).NE.5) GO TO 30
:	J=D(I)
CHECK IF IT IS IN THE RANGE	
:	IF ((J.LE.0).OR.(J.GT.N)) CALL FMLERR(\$990,J,0,14)
... YES, REMOVE DUMMY ARGUMENT AND	
INSERT THE TRUE ARGUMENT	
:	IL = FMLCH5 (\$990, IL, ICH(1,J), ICH(2,J))
:	CALL FMLCHI (\$990, I)
:	IF (I.NE.IK) GO TO 20
:	GO TO 40
:	30 IL=I

IF (I,NE,IK) GO TO 20

```

40      SIMPSW=.TRUE.
      CALL FMLCHA ($990, ICH(1,N), ICH(2,1))
      RETURN 1

990    RETURN 3

      END

```


3.2.6. STOUT4

STOUT4 ROUTINE EXECUTES DIFFERENTIATIONS. WHEN THE DIFFERENTIAL OPERATOR TO BE WRITTEN OUT (EXECUTED) IS ENCOUNTERED THE FIRST TIME, IT HAS 3 ARGUMENTS: THE ORDER OF DIFFERENTIATION, THE DIFFERENTIATING VARIABLE, AND THE EXPRESSION TO BE DIFFERENTIATED. AT THIS TIME, THE ORDERS OF DIFFERENTIATION IS REMOVED AND THE DIFF. OPERATOR WITH THE VARIABLE IS INSERTED AS MANY TIMES AS THE REMOVED ORDER SPECIFIED IT.

DIFFERENTIATION IS EXECUTED IN THE SECOND PHASE, ACCORDING TO THE TYPE OF THE EXPRESSION. THE DIFFERENTIATION IS EXECUTED ACCORDING TO THE CHAIN RULE; WHEN THE EXPRESSION TO BE DIFFERENTIATED IS A FUNCTION WHOSE ARGUMENT DEPENDS ON THE DIFFERENTIATING VARIABLE, THEN THE DIFFERENTIATION OF THE FUNCTION BY ITS ARGUMENT IS EXECUTED, THE DIFFERENTIAL QUOTIENT OF THE ARGUMENT ESTABLISHED BY PLACING THE DIFFERENTIAL OPERATOR AND VARIABLE IN FRONT OF IT, AND THE PRODUCT OF THESE EXPRESSIONS IS PUSHED BACK TO THE PUSH DOWN STACK TO EVALUATE THE DIFFERENTIAL QUOTIENT OF THE ARGUMENT.

DIFFERENTIATION OF UNDEFINED FUNCTION AND SPECIAL FUNCTIONS ARE LEFT UNTOUCHED.

THE ROUTINE USES TWO AUXILIARY ROUTINES, DPND AND DIF1. DPND GIVES THE LOGICAL ANSWER WHETHER A GIVEN VARIABLE APPEARS IN AN EXPRESSION. DIF1 IS AN ELEMENTARY DIFFERENTIATION ROUTINE WHICH EITHER SETS THE RESULT TO CONSTANT 0 OR 1, OR IT PLACES THE DIFFERENTIAL OPERATOR IN FRONT OF THE EXPRESSION.


```
*****
*          *
* STOUT4  *.....>* UPND  *
*          *
*****      * FMLERR *
*****      *
```

```
*****
```

```
*****
```

```
*          *
*          *
*.....>* FMLCHA- *
*      IN      *
*****
```

```
*****
```

```
*****
```

```
*          *
*          *
*.....>* FMLCHA- *
*      IN1      *
*****
```

```
*****
```

```
*****
```

```
*          *
*          *
*.....>* ERRWBK  *
*          *
*****
```

```
*****
```

```
*****
```

```
*          *
*          *
*.....>* SBEXP   *
*          *
*****
```

```
*****
```

3-174

```

.
.
.
.....>* DIF1 *.....>* DPND *
*
*
*****
.
.
.
.
*****
*
*.....>* COMP2 *
*
*
*****
.
.
.
.
*****
*
*.....>* FMLCHB *
*
*
*****
.
.
.
.
*****
*
*.....>* FMLCHA *
*
*
*****

```

FORMAL,STOUT4
EVALUATION OF DIFFERENTIATION

SUBROUTINE STDIFF (**,*,*)
INCLUDE CMMN
INCLUDE PWORD

DIFFERENTIATION ROUTINE

THIS ROUTINE WRITES OUT (OR EXECUTES) THE DIFFERENTIAL
OPERATOR FROM THE PUSH-DOWN STACK. WHEN BITSW IS .FALSE., THE
OPERATOR IS ENCOUNTERED THE FIRST TIME. IN THIS CASE, IT HAS 3
ARGUMENTS = ORDER OF DIFFERENTIATION, VARIABLE BY WHICH THE
DIFFERENTIATION IS TO BE DONE AND THE EXPRESSION TO BE DIFFERENT-
IATED. WHEN BITSW IS .TRUE., THE OPERATOR HAS ONLY THE LAST TWO
OF THE ABOVE ARGUMENTS WITH THE ORDER OF DIFFERENTIATION ALWAYS
ONE. DIFFERENTIATION IS ALWAYS EXECUTED, (RECURSIVELY), EXCEPT
WHEN THE EXPRESSION IS AN UNDEFINED FUNCTION OR THE DUMMY
ARGUMENT OF A DEFINED FUNCTION. SUBROUTINE DIF IS USED
TO OBTAIN RESULTS OF DIFFERENTIATION WHEN THE EXPRESSION
IS CONSTANT, VARIABLE, UNDEFINED FUNCTION OR DIFFERENTIAL
EXPRESSION.

RETURN 1 = REMOVE OPERATOR FROM THE STACK
RETURN 2 = WRITE OUT OPERATOR FROM THE STACK
RETURN 3 = ERROR RETURN

IS THIS THE FIRST ENCOUNTER OF DIFFERENTIATION

```

:
: ... YES,
: CHECK THE NUMBER OF ARGUMENTS
:
:
:
: REMOVE ORDER, N
:
:
:
: GET VARIABLE POINTER, J=JK
:
:
:
: IF IT IS NOT A VARIABLE, THEN ERROR
:
: IF N=0 THEN HANDLE IT AS NO-OP, REMOVE VARIABLE,
: OPER, IF N=1 THEN GO TO EXECUTION PHASE
: IF N>1 THEN COPY AND INSERT DIFF. OPERATOR WITH THE
: VARIABLE N-1 TIMES TO FORM REPEATED DIFF.
:
:

```

IF (BITSW) GO TO 40

IF (H2(NP).NE.3) CALL FMLERR(\$990,'DIF',1,3)
J=NEXT(NP)
IF (ITYP(J).NE.0) RETURN 2
N=D(J)

CALL RMOVF1(J)
IBIT(NP)=1
H2(NP)=2
BITSW=.TRUE.

J=NEXT(NP)
JK=J
I=ITYP(J)
IF (I.EQ.7) JK=NEXT(J)

IF (I .LT. 5 .OR. I .GT. 7) CALL FMLERR (\$990, 'DIF',1,12)

IF (N.EQ.1) GO TO 50


```

100      IIK=LASTXX($990,II,1,0)
        CALL DIF ($990, J, JK, II, IIK)
        SIMPSW=.TRUE.
        GO TO 70

110      JI=ILINK1(JK,18,0)
        N=D(I)
        IL=JI
        IIK=I

        DO 120 L=1,N

        II=NEXT(IIK)
        IKI=LASTXX($990,II,1,0)

        IF (IDPND(J,JK,II,IKI).EQ.0) GO TO 120

        JI=IADDNN(JI,I,IIK)

        JI1=JI
        JI=IADDNN(JI,II,IKI)
        JI1=NEXT(JI1)
        CALL DIF ($990, J, JK, JI1, JI)

        D(IL)=D(IL)+1

        IF (L.EQ.N) GO TO 120

        II=NEXT(IKI)
        JI=IADDNN(JI,II,IK)

120      IIK=IKI

        CALL RMOVFN(I,IK)
        SIMPSW=.TRUE.

```

```
*
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*---- EXPONENTIAL EXPRESSION
*      GET POINTERS OF BASE, IB-IBK
*              EXPONENT, IE-IK
*
*:
*:
*:
*      CHECK THE DEPENDENCE OF THE EXPONENT AND
*      BASE ON THE VARIABLE J-JK
*
*:
*:
*... BOTH, EXPONENT AND BASE IS
*    INDEPENDENT GENERATE ZERO AS RESULT
*
*:
*:
*... EXPRESSION IS DEPENDENT
*    INSERT + IN FRONT IF BOTH BASE AND
*    EXPONENT ARE DEPENDENT
*
*:
*:
*NOW USE THE FORMULA:
*(B**E)'=E*(B**(E-1))*B' + LOG(B)*E'*(B**E)
*
*SKIP THE FIRST TERM IF THE BASE IS
*INDEPENDENT
*
*GENERATE FIRST TERM
*
*:
*:
*:
*:
*:
*:
*
*:
*SKIP THE SECOND TERM IF THE EXPONENT IS
*INDEPENDENT
*
*GENERATE SECOND TERM
*
*:
*:
IF (D(IL),GE.2) GO TO 70
IF (D(IL).EQ.0) GO TO 130
CALL RMOVF1(IL)
GO TO 70
130 ITYP(IL)=0
GO TO 70
*
140 IB=NEXT(I)
    IBK=LASTXX($990,IB,1,0)
    IE=NEXT(IBK)
*
    K1>IDPND(J,Jk,IB,IBK)
    K2>IDPND(J,Jk,IE,Ik)
    IF ((K1+K2) .NE. 0) GO TO 170
*
150 IJ=ILINK1(JK,0,0)
160 CALL RMovFN(I,Ik)
    GO TO 70
*
170 IJ=JK
    IF ((K1*K2).NE.0) IJ=ILINK1(IJ,18,2)
*
SIMPSW=.TRUE..
*
IF (K1.EQ.0) GO TO 180
IJ=ILINK1(IJ,19,3)
IJ=IADDNN(IJ,IE,Ik)
IJ=ILINK1(IJ,20,2)
IJ=IADDNN(IJ,IB,IBK)
IJ=ILINK1(IJ,18,2)
IJ=ILINK1(IJ,0,-1)
IJ=IADDNN(IJ,IE,Ik)
IJ=ILINK1(IJ,23,2)
IJ=IADDNN(IJ,J,Jk)
IJ=IADDNN(IJ,IB,IBK)
*
180 IF (K2.EQ.0) GO TO 160
IJ=ILINK1(IJ,19,3)
IJ=ILINK1(IJ,21,1)
```


27467
FORMAL,DPND

DEPENDENCE CHECK SUBROUTINE

DEPENDENCE CHECK OF EXPRESSION I-IK FROM VARIABLE J-JK

IDPND = 0 IF I-IK IS INDEPENDENT

1 IF I-IK CONTAINS J-JK

2 IF I-IK CONTAINS FUNCTION ARGUMENT (TYPE 5)

3 IF I-IK CONTAINS J-JK AND FUNCTION ARGUMENT

INITIALIZE

SET POINTERS

.....

LOOP TO CHECK THE ELEMENTS OF THE EXPRESSION

.....
EXPRESSION CONTAINS THE VARIABLE

.....
CHECK FOR FUNCTION ARGUMENT

..... YES, IT IS

..... NO, CHECK FOR END OF EXPRESSION

END OF LOOP

RETURNS

END

FUNCTION IDPND (J,JK,I,IK)

INCLUDE CMN

INCLUDE PWORD

LOGICAL JUMP

JUMP = .TRUE.

IF (J .NE. JK) JUMP = .FALSE.

II = I

N = ITYP(J)

ND = D(J)

M = ITYP(JK)

MD = D(JK)

K = 0

10 IF (ITYP(II).NE.N) GO TO 30

IF (D(II) .NE. ND) GO TO 40

IF (JUMP) GO TO 20

IF (II.EQ.IK) GO TO 50

II=NEXT(II)

IF (ITYP(II).NE.M) GO TO 30

IF (D(II) .NE. MD) GO TO 40

20 IF (K.EQ.0) K=K+1

IF (K.EQ.2) GO TO 60

GO TO 40

30 IF (ITYP(II).NE.5) GO TO 40

IF (K.NE.2) K=K+2

IF (K.GE.3) GO TO 60

40 IF (II.EQ.IK) GO TO 50

II=NEXT(II)

GO TO 10

50 IDPND=K

RETURN

60 IDPND=3

RETURN

END


```

*--- I-IK IS A FUNCTION OR A ARITHMETIC EXPRESSION
*   CHECK IF I-IK IS INDEPENDENT OF J-JK, IF YES,
*   SET RESULT EQUAL TO ZERO

```

```

*   :
*   :
*   SET SIMPSW TO TRUE FOR
*   ARITHMETIC EXPRESSIONS
*   THEN GO TO INSERT DIFF. OPERATOR
*   :
*   :
*   :

```

```

RESULT IS ZERO

```

```

RESULT IS ONE

```

```

RESULT IS DENOTED BY DIFFERENTIATION
SUCH THAT THE DIFFERENTIAL OPERATORS ARE SORTED
BY THE VARIABLES

```

```

NEXT OPERATOR IS NEITHER A DIFF. OR ITS VARIABLE IS
LESS THAN THE PRESENT ONE, THUS INSERT
DIFF. OPERATOR HERE AND RETURN

```

```

GET NEXT OPERATOR FOR TESTING

```

```

ERROR RETURN

```

```

10  K=IDPND(J,JK,I,IK)
    IF (K.EQ.0) GO TO 100

```

```

    IF (N.GT.21) GO TO 130
    IF ((N.EQ.21) .AND. (H1(I) .GT. 165)) GO TO 130
    SIMPSW=.TRUE.
    GO TO 130

```

```

100 K=0
    GO TO 120

```

```

110 K=1
120 IL=ILINK1(IL,0,K)
    CALL RMOVFN(I,IK)
    I=IL
    IK=IL
    SIMPSW=.TRUE.
    RETURN

```

```

130 IF (N.NE.23) GO TO 150
    NJ=NEXT(I)
    NJK=NJ
    IF (ITYP(NJ) .EQ. 7) NJK=NEXT(NJ)
    CALL COMPAR($150,$160,$140,J,1,NJ,1,JK,NJK)

```

```

150 IL=ILINK1(IL,23,2)
    IL=IADDNN(IL,J,JK)
    I=NEXT(ILL)
    RETURN

```

```

160 IL=NJK
    I=NEXT(IL)
    N=ITYP(I)
    GO TO 130

```

```

140 RETURN 1
    END

```

3-184

3.2.7. STOUT5

STOUT5 ROUTINE INVERTS THE ORDERS OF ARGUMENTS OF AN UNDEFINED FUNCTION TO OBTAIN PROPER PREFIX FORM.

```
*****
*          *
* STOUT5  *.....>* ERRWBK *
*          *
*****
```

```
.
.
.
.
.
.....>* SBEXP *
.
.
.
.
*****
```

```
.
.
.
.
.
.....>* FMLCHA *
.
.
.
*****
```


3.3. EXPRESSION TRANSLATION FROM PREFIX TO INFIX

TRANSLATION OF THE INTERNAL PREFIX FORM OF EXPRESSIONS INTO EXTERNAL INFIX FORM IS PROVIDED BY 3 PROGRAM ELEMENTS, FMLWR, FMLPKS AND FMLPCK, WITH THE HELP OF SOME UTILITY ROUTINES LISTED EARLIER. THE MAIN TRANSLATION ROUTINE IS FMLWR. FMLPKS TRANSLATES THE INTERNAL PACKED SUBSCRIPTS, FMLPCK PACKS THE BCD ITEMS INTO OUTPUT BUFFER.

3.3.1. FMLWR AND FMLPKS

FMLWR IS THE MAIN TRANSLATION ROUTINE. IT IS CALLED FROM THE OUTPUT ROUTINES, AND IT ALSO COLLECTS THE LIST OF VARIABLES WHOSE EXPRESSION VALUES MUST BE TRANSLATED. THE ROUTINE TRANSLATES THOSE EXPRESSIONS IN A LOOP. IN THE BEGINNING OF THE LOOP, THE VARIABLE NAME IS PACKED INTO THE OUTPUT BUFFER BY FMLPCK, THEN ITS EXPRESSION VALUE IS TRANSLATED IN AN INSIDE LOOP WHERE THE CONSECUTIVE ITEMS OF THE PREFIX EXPRESSION IS PROCESSED. THE TRANSLATION USES A PUSH DOWN STACK FOR THE OPERATORS IN C-D ARRAYS WHOSE INITIAL POINTER IS BEGINI AND ITS RUNNING POINTER IS I. THE ROUTINE HAS AN INTERNAL SUBROUTINE, DIVTST. THIS ROUTINE CHECKS THE SIGN OF THE EXPONENT IN AN EXPONENTIAL SUBEXPRESSION. IT GIVES A NORMAL RETURN WHEN THE EXPONENT STARTS WITH A NEGATIVE CONSTANT, THUS MULTIPLICATION OPERATORS MUST BE CONVERTED TO DIVISION OPERATOR, AND THE EXPONENT SHOULD BE USED WITH POSITIVE SIGN (LOGICAL VARIABLE SIGN IS SET TO FALSE). LOGICAL VARIABLE NEG IS USED TO INDICATE WHETHER A NEGATIVE SIGN OF A CONSTANT SHOULD APPEAR AS A BINARY OPERATOR REPLACING +, OR IT SHOULD BE CONSIDERED AS A UNARY OPERATOR.

FMLPKS HANDLES THE NUMERIC SUBSCRIPTS OF THE VARIABLES.

```
*****
*          *
* FMLWR   *.....>* FMLCHA- *
*          *          * IN   *
*****
```

```

.
.
.
.....>* FMLCHA- *
.      * IN1    *
.
*****
```

```

.
.
.
.....>* FNCT    *
.      *        *
.
*****
```

```

.
.
.
.....>* GETARG  *
.      *        *
.
*****
```

```

.
.
.
.....>* E$DRUM  *
.      *        *
.
*****
```

```

.
.
.
.....>* FMLSV1   *
.      *        *
.
*****
```

```

.
.
.
.....>* ERRWBK   *
.      *        *
.
*****
```



```

*****
*                               *
.....>* SBEXP *
*                               *
*****

*****
*                               *
.....>* FMLPKS *.....>* ERRWBK *
*                               *
*****

*****
*                               *
.....>* FMLPCK *.....>* FMLERR *
*                               *
*****
*                               *
.....>* PACK *
*                               *
*****

*****
*                               *
.....>* ERRWBK *
*                               *
*****

*****
*                               *
.....>* FMLPCK *
*                               *
*****

*****
*                               *
.....>* SYMBOL *
*                               *
*****

*****
*                               *
.....>* FMLSCN *
*                               *
*****

```


282
 FORMAL,FMLWR

 EXPRESSION TRANSLATION FROM PREFIX TO INFIX FOR OUTPUT

 THIS SUBROUTINE TRANSLATES AN EXPRESSION IN THE INTERNAL
 PREFIX NOTATION TO THE REGULAR INFIX NOTATION WHICH IS
 SUITABLE FOR OUTPUT.
 AN OPERATOR STACK BEGINING FROM 'BEGINI' AND ENDING AT 'I'
 IS USED TO STORE OPERATORS .
 AT THE END OF EVERY SUBEXPRESSION A SPECIAL MARKER 'M' IS
 USED FOR LATER OUTPUT USE.

INITIALIZE

COLLECT ARGUMENTS

LOOP 1

* GET TOP ARGUMENT

* INITIALIZE PACK ROUTINE AND WRITE OUT VARIABLE ID.

* UNPACK SUBSCRIPTS , IF NEEDED

* FIND EXPRESSION POINTER
 * CHECK WHETHER THE VARIABLE IS DEFINED

* YES,INITIALIZE POINTERS AND SET FLAGS

SUBROUTINE FMLWR (*, SUB, U)

INCLUDE CMMN
 INCLUDE PWORD
 LOGICAL SUBS
 IOUNIT = U

CALL FMLSCN (\$500)

```

10  CALL GETARG
    SUBS = .FALSE.
    CALL SYMBL1 ($500, 1)
    IF (EPTR .NE. 0) GO TO 50
    IF (NSY .EQ. 0) GO TO 30
    SUBS = .TRUE.
    NSU = LAST(NSY)
20  IF (FLD(0,3,C(NSU)) .NE. 2) GO TO 610
    N2 = FLD(3,3,C(NSU))
    N3 = D(NSU)
    EPTR = LAST(NSU)
50  CONTINUE
30  CALL FMLPKI (N1, 1, SUB)

    IF (N2 .NE. 0) CALL FMLPKS ($500, N2, N3)
    CALL FMLPCK ('=', 1)
    IF (XFLG1 .EQ. 0) CALL FMLPKO

    IF (EPTR .EQ. 0) GO TO 700
551 IF (H1(EPTR) .NE. 0) EPTR = ROLLIN ($500, $550, EPTR, 1)

    LOGICAL NEG, SIGN
    I = FMLCH1 ($500)
    BEGINI = I
    C(I) = 0
    NEG = .TRUE.
  
```

```

* :
* : SKIP THE NEXT 4 STATEMENTS
* :

```

```

* : LOOP 2

```

```

* :   DECREMENT THE OPERATOR DEGREE COUNT
* :

```

```

* :   GET NEXT EXPRESSION POINTER
* :

```

```

* :   BRANCH ACCORDING TO DATA TYPE
* :

```

```

* :   ... COMMA, PLUS, MULTIPLICATION, EXPONENTIATION
* :

```

```

* :       , + * **
* :

```

```

* :   ... FUNCTIONS AND DIF
* :

```

```

* :       ... MATH AND FORMAL FUNCTIONS
* :

```

```

* :       ... DIFFERENTIAL OPERATOR
* :

```

```

* :       ... DIF WITH 3 ARGUMENTS
* :

```

```

SIGN = .TRUE.

```

```

GO TO 110

```

```

99  D(I) = ABS(D(I)) - 1

```

```

100 CONTINUE

```

```

EPTR = NEXT(EPTR)

```

```

IF (EPTR .EQ. 0) GO TO 999

```

```

110 S = ITYP (EPTR)

```

```

GO TO (300, 300, 999, 300, 999, 302, 302, 302,

```

```

+ 999, 999, 999, 999, 999, 999, 999, 999,

```

```

+ 130, 999, 130, 130, 130, 200, 999, 200,

```

```

+ 999, 200, 200, 200, 200, 200, 200, 200), S+1

```

```

130 CONTINUE

```

```

II = ITYP(I)

```

```

IF (S - II) , , 180

```

```

I = FMLCH2 ($500, I, 17, ' )#')

```

```

IF (LSTOP .NE. 0) CALL FMLPCK (LSTOP, 1)

```

```

LSTOP = 0

```

```

NEG = .TRUE.

```

```

IF (S .EQ. 18) SIGN = .TRUE.

```

```

CALL FMLPCK ('#(', 1)

```

```

180 I = FMLCH2 ($500, I, S, -H2(EPTR))

```

```

IF (S .NE. 20 .OR. (II .EQ. 19 .AND. .NOT. NEG)) GO TO 100

```

```

CALL DIVTST ($500, EPTR, $100)

```

```

IF (LSTOP .NE. 0) CALL FMLPCK (LSTOP, 1)

```

```

LSTOP = 0

```

```

CALL FMLPCK ('1#/#', 1)

```

```

GO TO 100

```

```

200 IF (LSTOP .NE. 0) CALL FMLPCK (LSTOP, 1)

```

```

LSTOP = 0

```

```

IF (S .GE. 25) GO TO 220

```

```

IF (S .EQ. 23) GO TO 210

```

```

F = IFUNC1 (H1(EPTR))

```

```

DEG = H2 (EPTR)

```

```

GO TO 230

```

```

210 IF (H2(EPTR) .NE. 2) GO TO 215

```

```

CALL FMLPCK ('(Δ/Δ', 1)

```

```

I = FMLCH2 ($500, I, 21, 0)

```

```

GO TO 100

```

```

215 F = 'DIF'

```


BRANCH ACCORDING TO TYPE OF OPERATOR

... RIGHT BRACKET AND RIGHT PARETHSIS
] AND)

... COMMA
,

... PLUS
+

... MULTIPLICATION AND DIVISION
* AND /

... EXPONENTIALIZATION
**

... DIFFERENTIATION

```

401  II = LAST (I)
      CALL FMLCHB ($500, I)
      I = II
      GO TO 400
409  CONTINUE

      GO TO (420, 430, 440, 450, 460, 470), ITYP (I) - 15

430  CALL FMLPCK (D(I), 1)
      GO TO 401

420  CALL FMLPCK ('-', 1)
      NEG = .TRUE.
      GO TO 99

440  LSTOP = 'n+n'
      NEG = .TRUE.
      GO TO 99

450  LSTOP = 'n*n'
      J = NEXT(EPTR)
      IF (ITYP(J) .NE. 20) GO TO 455
      CALL DIVTST ($500, J, $455)
      LSTOP = 'n/n'
455  CALL FMLPCK(LSTOP, 1)
      LSTOP = 0
      NEG = .FALSE.
      GO TO 99

460  II = NEXT(EPTR)
      IF (ITYP(II) .EQ. 0 .AND. ABS(D(II)) .EQ. 1) GO TO 461
      IF (ITYP(II) .EQ. 3 .AND. BOOL(ABS(U(II))) .EQ. 1.0) GO TO 461
      CALL FMLPCK ('n**n', 1)
      NEG = .FALSE.
      SIGN = .FALSE.
      IF (ITYP(II) .NE. 19 .OR. H2(II) .NE. 2) GO TO 99
      II = NEXT(II)
      IF (ITYP(II) .EQ. 0 .AND. D(II) .EQ. -1) EPTR = II
      GO TO 99
461  EPTR = II
      GO TO 401

470  CALL FMLPCK ('d', 1)
      II = LAST(I)
      CALL FMLCHB ($500, I)

```

```
*      :      :      :
```

```
*      :      :      :
```

```
*      END OF LOOP 2
```

```
*      CLEAN UP
```

```
*      CALL FMLPCK ('H:H', 1)
```

```
*      :
```

```
*      :
```

```
*      CHECK WHETHER THE ARGUMENTS ARE EXHAUSTED
```

```
*      IF NOT CONTINUE
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      YES, END OF LOOP1
```

```
*      RETURN
```

```
*      :
```

```
*      UNASSIGNED VARIABLE
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      ERRORS
```

```
*      :
```

```
*      ERROR RETURN
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      :
```

```
*      END
```

```

I = II
GO TO 100

600    CALL FMLPKO
        CALL FMLCHR ($500, BEGINI)

610    IF (.NOT. SUBS) GO TO 611
        NSU = NEXT(NSU)
        IF (NSU .NE. 0) GO TO 20
611    IF (NK .NE. 0) GO TO 10

        RETURN

700    CALL FMLPCK ('UNASSI', 1)
        CALL FMLPCK ('GNED', 1)
        CALL FMLPKO
        GO TO 610

999    CALL ERRERR ($500, EPTR)

500    IF (NK .NE. 0) CALL FMLCHD ($501, NK)
501    NK = 0
        RETURN 1
550    CALL FMLSV1
        GO TO 551

        SUBROUTINE DIVTST (*, JJ, *)
        J = NEXT (LASTXX ($801, NEXT(JJ), 1, 0))
        IF (ITYP(J) .EQ. 19) J = NEXT(J)
        IF (ITYP(J) .GT. 3) GO TO 800
        IF (D(J) .LT. 0) RETURN
800    RETURN 3
801    RETURN 1
        END
        END

```

FORMAL,FMLPKS

 SUBROUTINE TO UN-PACK 'SN' SUBSCRIPTS FROM THE
 SUBSCRIPT-WORD 'SW'.

INITIALIZE

```

:
:
: CHECK FOR ILLEGAL NUMBER OF SUBSCRIPTS
:
:
: CHECK WHETHER MORE THAN ONE SUBSCRIPTS
:
:
LOOP
:
: WRITE OUT THE VALUE OF SUBSCRIPT AND A COMMA
:
:
:
END OF LOOP
WRITE OUT THE REMAINING SUBSCRIPT AND
THE RIGHT PARENTHESIS
:
:
RETURN

```

ERROR RETURN

SUBROUTINE FMLPKS (*,SN,SW)

IMPLICIT INTEGER (A-Z)
 REFERENCES ON

IF (SN * (4-SN) .LT. 0) CALL ERRERR (\$20,SN,0,3)
 CALL FMLPCK ('M', 1)

IF (SN .EQ. 1) GO TO 10

DO 5 J = 0,SN-2

CALL FMLPCK (FLD(9+J,9,SW), 2)
 5 CALL FMLPCK ('M', 1)

10 CALL FMLPCK (FLD(9*(SN-1),9,SW), 2)
 CALL FMLPCK ('M', 1)

RETURN

20 RETURN 1
 END

3.3.2. FMLPCK

ROUTINE FMLPCK PACKS THE OUTPUT BUFFER WITH THE DATA PROVIDED BY FMLWR. THE ROUTINE IS INITIALIZED BY THE I/O ROUTINES TO DEFINE THE FORMAT THROUGH ENTRY FMLPIN. A SECONDARY INITIALIZATION IS INITIATED BY FMLWR THROUGH ENTRY FMLPKI WHICH ALSO DEFINES THE PROPER I/O ROUTINE (SUB) FOR OUTPUTTING THE BUFFER WHEN IT IS FILLED. THE DATA TO BE PACKED IS TRANSMITTED THROUGH ENTRY FMLPCK, WHICH CAN BE INTEGER NUMBER, FLOATING POINT NUMBER OR HOLLERITH WORD. THE NUMBERS ARE CONVERTED TO HOLLERITH DATA BY FMLEDI WHICH CALLS THE EDIT\$ PACKAGE. WHEN THE COMMON VARIABLE XFLG1 IS SET, THE ROUTINE PROVIDES A CONTINUATION NUMBERING TO ACCOMODATE FORTRAN TYPE OF OUTPUT. ALSO, BRACKETS ARE CHANGED TO PARENTHESES. ENTRY FMLPKO WRITES OUT THE BUFFER.

FORMAL,FMLPCK

 OUTPUT PACKING ROUTINE FOR EXPRESSIONS

 ENTRIES:

1. FMLPIN(S,I1,I2)
 INITIALIZATION ENTRY USED BY THE
 PROPER FMLI01-7 TO SPECIFY FORMAT:
 I1=STARTING COLUMN
 I2=NUMBER OF COLUMNS USED
2. FMLPKI(DATA,IND,SUB)
 REINITIALIZE BUFFER AS DEFINED BY FMLPIN
 BEFORE PACKING 'DATA' INTO 'CBUF'
 'SUB' DEFINES I/O ROUTINE
3. FMLPCK(DATA,IND)
 PACK 'DATA' INTO 'CBUF'
 'DATA' IS BCD IF IND=1
 IS INTEGER IF IND=2
 IS REAL IF IND=3
4. FMLPKO
 WRITE OUT BUFFER, AND REINITIALIZE IT
 AS DEFINED BY FMLPIN

 FMLPIN(S,I1,I2)

DEFINE THE DIMENSION OF CBUF BY PARAMETER

DEFINE INITIAL COLUMN

IS1 = STARTING COLUMN
 IS2 = ENDING COLUMN

 FMLPKI(DATA,IND,SUB)

 REINITIALIZE BUFFER

INITIALIZE FOR FIRST COLUMN
 II = COLUMN TO BE NEXT
 III = NUMBER OF CHAR. TO BE PACKED

SUBROUTINE FMLPIN(S,I1,I2)

PARAMETER NCB = 22

IS1=I1
 IF (IS1 .LE. 1) IS1=2
 IS2=IS1+I2-1

IF ((IS1 .GT. (IS2-19)).OR.(IS2 .GT. 132))
 1 CALL FMLERR(S1,'FORMAT',1,12)
 RETURN
 1 RETURN 1

ENTRY FMLPKI(DATA,IND,SUB)
 INCLUDE CMMN
 REAL FDATA
 DIMENSION A(3),B(18),CC(132)
 CALL STRCON(NCB,' ',CBUF)

GO TO PACK DATA

FMLPCK(DATA,IND)

CONVERT IF NECESSARY

SEPARATE CHARACTERS FROM A INTO B

LOOP TO PACK CHARACTERS INTO CBUF

: IGNORE BLANKS

: CHECK FOR LAST COLUMN

: ... YES, PACK AND WRITE OUT BUFFER

: REINITIALIZE BUFFER

: INSERT CONTINUATION CHAR. FOR PUNCH

: INSERT CHARACTER INTO CC

: IS CHARACTER A SUBEXPRESSION SEPARATOR

: ... YES, SKIP IT FOR PUNCH

: INSERT A BLANK OTHERWISE

: INSERT CHARACTER

: IS IT PUNCH

5 II=IS1
III=IS1-1
DO 5 I=1,IS1
CC(I)=5
ICON=0

ENTRY FMLPCK(DATA,IND)

A(1)=DATA

A(2)=' '

A(3)=' '

IF (IND .EQ. 2) CALL FMLEDI (A)

IF (IND .EQ. 3) CALL FMLEDF (A)

2 CALL UNPACK(18,A,B)

DO 20 I=1,18

IF (B(I) .EQ. 5) GO TO 20

IF (II .LE. IS2) GO TO 15

IF (III .EQ. 0) CALL ERRERR(0,0)

CALL PACK(III,CC,CBUF)

CALL SUB

CALL STRCON(NCB,' ',CBUF)

K=II-III-1

IF (K .NE. 0) CALL MOVE(K,CC(III+1),CC(IS1))

IF (XFLG1 .NE. 0) CC(IS1-1)=ICON+49

ICON=ICON+1

II=K+IS1

III=II-1

15 IF (B(I) .NE. '000000') GO TO 16

III=II-1

IF (XFLG1 .NE. 0) GO TO 20

III=II

B(I)=5

16 CC(II)=B(I)

IF (XFLG1 .EQ. 0) GO TO 17

```

:      :... YES, CHECK AND CHANGE BRACKETS
:      :
:      :

```

```

END OF LOOP, RETURN

```

```

*****

```

```

FMLPKO

```

```

*****

```

```

PACK LEFT-OVER INFORMATION
INTO BUFFER AND WRITE IT OUT

```

```

REINITIALIZE BUFFER

```

```

17
20

```

```

IF (CC(II) .EQ. '[[[[[[[') CC(II)='[[[[[[[('
IF (CC(II) .EQ. ']]]]]]]') CC(II)=']]]]]]]'
II=II+1
CONTINUE

```

```

RETURN

```

```

ENTRY FMLPKO

```

```

CALL PACK(II-1,CC,CBUF)
CALL SUB

```

```

CALL STRCON(NCB,' ',CBUF)
II=IS1
III=II-1
RETURN
END

```

FORMAL,FMLEDI

CONVERSION ROUTINE FROM BINARY TO FD
THIS ROUTINE USES THE EUI\$ PACKAGE FROM THE
SYSTEM LIBRARY.

GET THE PARAMETER AND SAVE RETURN

\$ (1)	AXRS	
FMLEDI*	S	X10,A5
	LMJ	X10,PARAM
	L	A0,A4
	LMJ	X11,EDECVS
	J	EXIT1
FMLEDF*	S	X10,A5
	LMJ	X10,PARAM
	L	A1,A4
	L,U	A0,14*/6+8
	LMJ	X11,FFLG1\$
EXIT1	LMJ	X11,EDITXS
	L	X11,EXIT
	L	X10,A5
	J	2,X11
PARAM	SZ	A3
	L,U	A3,*0,X11
	S,H2	A3,EDPKT
	L	A4,A3
	S	X11,EXIT
	L,U	A0,EDPKT
	LMJ	X11,EDITS
	J	0,X10
\$ (2)		
EXIT	+	0
EDPKT	+	3,0
	+	0
	+	1,0,0,0,0,0
	+	0
	+	0
	+	0
	+	'D','E',0,0,0,0
	+	0
	+	0
	+	0
END		

3.4. STATEMENT ROUTINES

THIS SECTION CONTAINS THE DESCRIPTION AND LISTINGS OF THE ROUTINES HANDLING FORMAL STATEMENTS. THE I/O ROUTINES HANDLE THE READ, PRINT, PUNCH AND WRITE STATEMENTS. FMLDMP HANDLES THE STATEMENT. FMLSAY HANDLES THE SAVE AND RESET STATEMENTS. FMLDMP HANDLES THE UIMP STATEMENT. FMLASG HANDLES THE ASSIGN STATEMENT. FMLDPT HANDLES THE OPTION STATEMENT. FMLERS HANDLES THE ERASE STATEMENT. FMLFOR AND FRECT HANDLE THE FORTRAN VALUED FUNCTIONS AND NCOUNT STATEMENT. FMLEQU HANDLES THE EQUIVALENCING SYMBOLIC VARIABLES AS SUBROUTINE ARGUMENTS THROUGH THE SUBROUTINE CALL STATEMENT GENERATED BY THE PRECOMPILER.

ALL OF THESE ROUTINES, EXCEPT FMLSAY, START WITH STATEMENT INITIALIZATION USING FMLINT. FURTHERMORE, MOST OF THEM INITIALIZES THE SCANNER, GSCAN.

3.4.1. I/O STATEMENT

THERE ARE 7 I/O STATEMENT ROUTINES FMLI01-7 WITH EACH ONE HAVING ITS I/O ROUTINE FMLIOA-6:

- FMLI01 = CARD OR TELETYPE READ WITH FMLIOA.
- FMLI02 = UNFORMATED PRINT WITH FMLIOC.
- FMLI03 = FORMATED PRINT WITH FMLIOF.
- FMLI04 = PUNCH CARD WITH FMLIOD.
- FMLI05 = READ FROM UNIT WITH FMLIOB.
- FMLI06 = UNFORMATED WRITE ON UNIT WITH FMLIOE.
- FMLI07 = FORMATED WRITE ON UNIT WITH FMLIOG.

ROUTINE FMLUIT IS USED IN THE INTERACTIVE FORMAL TO EXTRACT THE UNIT AND FORMAT PARAMETERS FROM THE I/O STATEMENT.

THE INPUT ROUTINES, FMLI01 AND FMLIO5, USE FMLRD FOR THE TRANSLATION OF THE INPUT DATA. FMLRD OBTAINS THE LIST OF VARIABLES FROM THE INPUT STATEMENT, THEN IT PROCEEDS TO READ AND TRANSLATE THE INPUT DATA.

THE OUTPUT ROUTINES USE FMLWR FOR GETTING THE LIST OF VARIABLE FROM THE OUTPUT STATEMENT AND FOR OUTPUTTING THE EXPRESSION IN INFIX FORM.

FORMAL,FMLUIT

EXTRACT FORMAT DEFINITIONS IN I/O STATEMENTS

ORDER OF 'N' ARGUMENTS

I1 - UNIT
I2 - STARTING COLUMN
I3 - LAST COLUMN
I4 - DUMMY

INITIALIZE

EXTRACT ARGUMENT CHARACTERS

PROCESS NON-ZERO CHARACTER

IF THIS IS THE FIRST AND NOT A '(' THEN RETURN

OTHERWISE MAKE IT BLANK

SKIP '('

SAVE OTHERWISE

GO OUT FROM LOOP IF ')'

TRANSLATE ARGUMENTS AS SUBSCRIPTS

SUBROUTINE FMLUIT (IN,N,I1,I2,I3,I4)

INCLUDE CMMN

DIMENSION IN(13),INN.

N=0

K=1

KK=-1

INN(1)=' '

INN(2)=' '

INN(3)=' '

DO 20 I=1,13

DO 10 J=1,6

JJ=6*(J-1)

IJ=FLD(JJ,6,IN(I))

IF (IJ .EQ. 5) GO TO 10

IF ((KK .EQ. -1) .AND. (IJ .NE. 41)) RETURN

FLD(JJ,6,IN(I))=5

IF (KK .LT. 0) KK=0

IF (IJ .EQ. 41) GO TO 10

FLD(KK,6,INN(K))=IJ

IF (IJ .EQ. 32) GO TO 30

KK=KK+6

IF (KK .LE. 30) GO TO 10

KK=0

K=K+1

IF (K .GT. 3) GO TO 30

10 CONTINUE

20 CONTINUE

30 CALL ISCANG(INN,3,0)

CALL EXPRES(\$60,2)

N=N2

I1=FLD(0,9,N3)

I2=FLD(9,9,N3)

I3=FLD(18,9,N3)

I4=FLD(27,9,N3)

ERROR IN FORMAT DEFINITION

50 RETURN

```

60 PRINT 61
61 FORMAT (' ERROR IN I/O STATEMENT FORMATING' )
END

```

FORMAL.FMLRD

```

*****
SUBROUTINE TO HANDLE INPUT OF STATEMENTS
EITHER FROM CARDS OR FROM I/O UNITS.
*****

```

```

INITIALIZE
:
: READ IN STATEMENTS
:
: COLLECT ARGUMENTS INTO A CHAIN STARTING FROM 'NK'
:
: INITIALIZE THE SCANNER ROUTINE
:
LOOP
:
: TRANSLATE THE EXPRESSION
:
: GET TOP ARGUMENT FROM THE CHAIN
:
: SET THE EXPRESSION POINTER
:
: STORE IN THE SYMBOL TABLE
:
: CONTINUE LOOP UNTIL ARGUMENTS EXHAUSTED
:
END LOOP
RETURN

```

ERROR RETURN

SUBROUTINE FMLRD (*,SUB,U)

```

INCLUDE CMMN
IOUNIT = U

CALL SUB

CALL FMLSCN ($20)
FTRARG = 0

CALL ISCANG (CBUF,14,SUB)

10 CONTINUE

CALL EXPRES ($20, 1)

CALL GETARG

EPTR = NP

CALL SYMBOL ($20,2)

IF (NK .NE. 0) GO TO 10

RETURN

20 IF (NK .NE. 0) CALL FMLCHD ($11, NK)
21 NK = 0
RETURN 1
END

```

2990
FORMAL,FMLIO1

FORMAL STATEMENT ROUTINE TO HANDLE CARD INPUT
THIS ROUTINE CALLS 'FMLRD', THE READ ROUTINE,
WHICH IN TURN CALLS 'FMLIOA' TO READ IN CARD IMAGES.

INITIALIZE

SET EXTERNAL FLAG

:

GET FORTRAN TYPE ARGUMENTS

:

INITIALIZE SCANNER ROUTINE

:

READ IN INFORMATIONS

RETURN

ERROR OCCURED

SUBROUTINE FMLIO1(Alpha)
REFERENCES ON

EXTERNAL FMLIOA

CALL FMLINT (\$10, 2)

CALL ISCANG (Alpha,220,0)

CALL FMLRD (\$10,FMLIOA,0)

20 RETURN

10 CALL ERRRET ('FMLIO1')
GO TO 20
END

FORMAL,FMLIO2
 FORMAL STATEMENT ROUTINE TO PRINT EXPRESSIONS

 THIS FORMAL STATEMENT ROUTINE CALLS THE WRITE
 ROUTINE 'FMLWR' WHICH IN TURN CALLS 'FMLIOC'
 TO PRINT OUT CONTENTS OF 'CBUF'.

INITIALIZE

SET EXTERNAL FLAG

GET FORTRAN TYPE ARGUMENTS

INITIALIZE SCANNER ROUTINE

INITIALIZE FORMATS FOR PACKING

WRITE OUT THE INFORMATIONS

RETURN

ERROR OCCURED

SUBROUTINE FMLIO2(BCD)

INCLUDE CMMN

EXTERNAL FMLIOC

CALL FMLINT (\$20, 2)

CALL ISCANG (BCD, 220, 0)

XFLG1 = 0

CALL FMLPIN (\$20, 1, 72)

CALL FMLWR (\$20, FMLIOC, 0)

10 RETURN

 20 CALL ERRRET ('FMLIO2')
 GO TO 10
 END

300
FORMAL,FMLIO3

FORMAL STATEMENT ROUTINE TO PRINT FORMATED EXPRESSION

THIS ROUTINE CALLS THE WRITE ROUTINE 'FMLWR' WHICH
IN TURN CALLS 'FMLIOF' TO PRINT OUT CBUF.

INITIALIZE
PROGRAM

SCANNER

FORMAT DEFINITION

GO TO WRITE ROUTINE

SUBROUTINE FMLIO3(I1,I2,BCD)
INCLUDE CMMN
EXTERNAL FMLIOF

CALL FMLINT(\$20,4)

CALL ISCANG(BCD,220,0)

XFLG1 = XFLG2

IF (XFLG1 .EQ. 0) GO TO 2

I1=7

I2 = 66

2 CALL FMLPIN(\$20,I1,I2)

CALL FMLIFF(I1+I2-1)

CALL FMLWR(\$20,FMLIOF,0)

10 RETURN

20 CALL ERRRET('FMLIO3')

GO TO 10

END

FORMAL,FMLIO4

 FORMAL STATEMENT ROUTINE TO PUNCH EXPRESSIONS

 THIS FORMAL STATEMENT ROUTINE CALLS THE WRITE
 ROUTINE 'FMLWR' WHICH IN TURN CALLS 'FMLIOD'
 TO PUNCH OUT THE CONTENTS OF 'CBUF'.

INITIALIZE

SET EXTERNAL FLAG

INITIALIZE SCANNER ROUTINE

DEFINE FORMAT FOR PACKING

PUNCH OUT INFORMATIONS

RETURN

ERROR OCCURED

SUBROUTINE FMLIO4(BCD)
 INCLUDE CMMN

REFERENCES ON

EXTERNAL FMLIOD
 CALL FMLINT (\$20, 2)

CALL ISCANG (BCD, 220, 0)

XFLG1 = XFLG2
 IF (XFLG1 .NE. 0) GO TO 1
 CALL FMLPIN (\$20, 1, 80)
 GO TO 2
 CALL FMLPIN (\$20, 7, 66)
 CONTINUE

1
 2

CALL FMLWR (\$20, FMLIOD, 0)

10 RETURN

20 CALL ERRET ('FMLIO4')
 GO TO 10
 END

FORMAL.FMLIO5

FORMAL STATEMENT ROUTINE TO HANDLE INPUT FROM I/O UNITS

THIS ROUTINE CALLS THE READ ROUTINE 'FMLRD'
WHICH IN TURN CALLS 'FMLIOB' TO READ CARD IMAGES
INTO THE 'CBUF'.

INITIALIZE

SET EXTERNAL FLAG

:

INITIALIZE SCANNER ROUTINE

:

READ IN INFORMATIONS

RETURN

ERROR OCCURED:

1

SUBROUTINE FMLIO5(U,ALPHA)

REFERENCES ON

EXTERNAL FMLIOB

CALL FMLINT (\$10, 3)

CALL ISCANG (ALPHA,220,0)

CALL FMLRD (\$10, FMLIOB, U)

20 RETURN

10 CALL ERRRET ('FMLIO5')

GO TO 20

END

FORMAL,FMLIO6
 FORMAL STATEMENT ROUTINE TO WRITE EXPRESSIONS ON UNIT U

 THIS ROUTINE CALLS THE WRITE ROUTINE 'FMLWR'
 AND AT SAME TIME PASSES THE ARGUMENT 'FMLIOE'.
 'FMLWR' WILL IN TURN CALLS 'FMLIOE' TO WRITE OUT
 THE CONTENTS OF 'CBUF' ON I/O UNIT 'U'.

INITIALIZE

SET EXTERNAL FLAG

:

INITIALIZE SCANNER ROUTINE

:

INITIALIZE FORMAT FOR PACKING

:

WRITE OUT INFORMATIONS

RETURN

ERROR OCCURED

SUBROUTINE FMLIO6(U, BCD)
 INCLUDE CMMN

REFERENCES ON

EXTERNAL FMLIOE
 CALL FMLINT (\$20, 3)

CALL ISCANG (BCD, 220, 0)

XFLG1 = 0
 CALL FMLPIN (\$20, 1, 80)

CALL FMLWR (\$20, FMLIOE, U)

10 RETURN

20 CALL ERRRET ('FMLIO6')
 GO TO 10
 END

034
FORMAL,FMLIO7

FORMAL STATEMENT TO WRITE FORMATED EXPRESSION

THIS ROUTINE CALLS THE WRITE ROUTINE 'FMLWR'
WHICH IN TURN CALLS 'FMLIOG' TO WRITE OUT 'CBUF'.

INITIALIZE
PROGRAM

SCANNER

FORMAT DEFINITION

GO TO WRITE ROUTINE

SUBROUTINE FMLIO7(I1,I2,I3,BCD)
EXTERNAL FMLIOG
INCLUDE CMMN

CALL FMLINT(\$20,5)

CALL ISCANG(BCD,220,0)

XFLG1 = XFLG2
IF (XFLG1 .EQ. 0) GO TO 2
I2 = 7
I3 = 66
2 CALL FMLPIN(\$20,I2,I3)
CALL FMLIGG(I2+I3-1)

CALL FMLWR(\$20,FMLIOG,I1)
10 RETURN
20 CALL ERRRET('FMLIO7')
GO TO 10
END

FORMAL,FMLIOA
CARD INPUT ROUTINE

THIS ROUTINE READS ONE CARD AT A TIME INTO CBUF
IT IS CALLED BY 'FMLRD' ROUTINE TO PERFORM
THE INPUT OPERATION FROM CARD READER. THE 'FMLRD'
ROUTINE ITSELF IS CALLED BY 'FMLIO1',

INPUT A CARD IMAGE

⋮
RETURN
⋮

SUBROUTINE FMLIOA
INCLUDE CMMN

100 READ 100, (CBUF(I), I=1,14)
FORMAT (13A6,A2)

RETURN
END

FORMAL,FMLIOB

I/O DEVICE INPUT ROUTINE

THIS ROUTINE READS ONE CARD AT A TIME FROM UNIT U
INTO 'CBUF'. IT IS CALLED BY 'FMLRD' TO PERFORM
THE INPUT OPERATION FROM I/O UNIT 'IOUNIT'.
THE 'FMLRD' ITSELF IS CALLED BY 'FMLIO5',

INPUT A LINE IMAGE FROM UNIT 'IOUNIT'

⋮
RETURN
⋮

SUBROUTINE FMLIOB
INCLUDE CMMN

100 READ (IOUNIT, 100) (CBUF(I), I=1,14)
FORMAT (13A6,A2)

RETURN
END

3050

FORMAL,FMLIOC

PRINTED OUTPUT ROUTINE

FORMAL SUBROUTINE TO PRINT CARD IMAGES

THIS ROUTINE IS CALLED BY 'FMLWR' TO PERFORM
PRINT OPERATION ON PRINTER. THE 'FMLWR'
ITSELF IS CALLED BY FORMAL STATEMENT ROUTINE,
'FMLIO2'.

SUBROUTINE FMLIOC

INCLUDE CMMN

PRINT 100, (CBUF(I), I=1,12)

RETURN

100 FORMAT (13A6,A2)

END

FORMAL,FMLIOD

PUNCHED OUTPUT ROUTINE

FORMAL SUBROUTINE TO PUNCH CARD IMAGES

THIS ROUTINE IS CALLED BY 'FMLWR' TO PERFORM
THE OUTPUT OPERATION THROUGH PUNCHER. -
'FMLWR' ITSELF IS CALLED BY THE FORMAL STATEMENT
ROUTINE 'FMLIO4'.

SUBROUTINE FMLIOD

INCLUDE CMMN

PUNCH 100, (CBUF(I), I=1,14)

RETURN

100 FORMAT (13A6,A2)

END

FORMAL,FMLIOE

I/O DEVICE OUTPUT ROUTINE

FORMAL SUBROUTINE TO WRITE CARD IMAGES ON IOUNIT
 THIS ROUTINE IS CALLED BY 'FMLWR' TO PERFORM
 THE OUTPUT OPERATION THROUGH THE I/O UNIT 'IOUNIT'.
 THE 'FMLWR' ROUTINE ITSELF IS CALLED BY THE
 FORMAL STATEMENT ROUTINE 'FMLIO6'.

```

SUBROUTINE FMLIOE
INCLUDE CMMN
WRITE (IOUNIT, 100) (CBUF(I), I=1,14)
RETURN
100  FORMAT (13A6,A2)
END

```

FORMAL,FMLIOF

PRINT PARTIAL BUFFER ROUTINE

```

SUBROUTINE FMLIFF(II)
INCLUDE CMMN
JJ=(II+5)/6
RETURN
ENTRY FMLIOF
PRINT 100,(CBUF(K),K=1,JJ)
100  FORMAT (22A6)
RETURN
END

```


FORMAL,FMLIOG
WRITE PARTIAL BUFFER TO I/O DEVICE ROUTINE

```
SUBROUTINE FMLIGG(I)
INCLUDE CMMN
J=(I+5)/6
RETURN
ENTRY FMLIOG
WRITE (IOUNIT,100) (CBUF(K),K=1,J)
100 FORMAT (22A6)
RETURN
END
```

3-214

3.4.2. ROLOUT STATEMENT

THE ROLOUT STATEMENT IS HANDLED BY PROGRAM ELEMENT FMLOUT.
AFTER INITIALIZATION, THE ROUTINE OBTAINS THE LIST OF VARIABLES BY
FMLSCN. THE EXPRESSION VALUES OF THE VARIABLES ARE WRITTEN OUT TO
DRUM BY PROGRAM ELEMENT E\$DRUM.

FORMAL.FMLOUT
FORMAL OPTION STATEMENT ROUTINE

```
SUBROUTINE FMLOUT (ARR)
  INCLUDE CMVN
  INCLUDE PWORD
  CALL FMLINT ($10,2)
  CALL ISCAN (ARR, 220, 0)
  CALL FMLSCN ($10)
20  CALL GETARG
  CALL SYMBOL ($10, 1)
  IF (EPTR .EQ. 0) GO TO 10
  CALL ROLOUT ($10, EPTR)
  IF (NK .NE. 0) GO TO 20
30  RETURN
10  IF (NK .NE. 0) CALL FMLCHD ($11, NK)
11  NK = 0
  CALL ERRRET ('FMLOUT')
  GO TO 30
END
```

3.4.3. SAVE AND RESET STATEMENTS

THESE STATEMENTS ARE HANDLED BY FMLSAV. BOTH SUBROUTINES ARE CALLED WITH ONE ARGUMENT, WHICH CAN BE OF THE FORM:

```
'FILENAME;'  
'FILENAME.ELTNAME;'  
'ELTNAME;'  
'ELTNAME;'
```

IN THE FIRST CASE THE DATA WILL BE SAVED IN (OR READ FROM) A DATA-FILE, IN ALL OTHER CASES AN ABSOLUTE ELEMENT IS CREATED OR READ. IN THE FIRST TWO CASES, THE FILENAME IS TAKEN FROM THE SPECIFICATION, IN CASES THREE AND FOUR IT IS ASSUMED TO BE TPF\$.

FILENAME MAY CONTAIN A COMPLETE EXEC-8 FILE SPECIFICATION, WITH ALL THE DROP-OUT RULES APPLYING:

<QUALIFIER>*<FILE>(<FCYCLE>)/<READKEY>/<WRITEKEY>

AND ELTNAME MAY BE

<ELEMENT>/<VERSION>(<CYCLE>)

HOWEVER, <CYCLE> WILL BE IGNORED.

THE FOLLOWING ERRORS CAN OCCUR:

A. IF NO SPECIFICATION IS GIVEN, OR IF THE ELEMENT OR VERSION ARE LONGER THAN 12 CHARACTERS, OR IF BAD PUNCTUATION IS USED, THE PROGRAM WILL PRINT

SYNTAX ERROR

AND RETURN TO THE CALLER.

B. WHEN READING FROM OR WRITING INTO A PROGRAM-FILE, ANY OF THE FOLLOWING THREE ERROR MESSAGES CAN OCCUR:

PFSERR	(1)
PFWERR	(2)
PFIERR	(3)

MEANING ERRORS ON AN ER TO (1) PFSS (PROGRAM FILE SEARCH), (2) PFWL\$ (DETERMINATION OF NEXT FREE WRITE LOCATION), OR (3) PFIS (INSERTION OF ELEMENT INTO PROGRAM FILE). THEY WILL BE FOLLOWED BY ANY OF THE FOLLOWING, AFTER WHICH THE PROGRAM WILL RETURN:

ELEMENT NOT FOUND	(1)
I/O ERROR	(2)
NOT PF	(3)
BAD STATUS 4	(4)
PF OVERFLOW	(5)
BAD STATUS>5	(6)

MEANING:

- (1) THE ELEMENT SEARCHED FOR WAS NOT FOUND
- (2) EXEC 8 ENCOUNTERED AN I/O ERROR ON HANDLING THE FILE
(POSSIBLY READ- OR WRITE- ONLY, OR KEYS NEEDED)
- (3) THE FILE MAY NOT BE USED AS A PROGRAM FILE
- (5) THE ELEMENT MADE THE PROGRAM-FILE OVERFLOW
- (4) OR (6) SHOULD NEVER OCCUR

C. IF THE ABSOLUTE ELEMENT/FILE READ IS NOT IN THE PROPER FORMAT, AS SHOWN BY WORDS 15 AND 16 OF THE LABEL-BLOCK, THE MESSAGE

FILE/ELEMENT INCORRECT FORMAT

WILL BE PRINTED, AND THE PROGRAM WILL RETURN.

D. IF AN I/O ERROR OCCURS IN THE PROCESS OF READING FROM OR WRITING TO THE FILE, THE PROGRAM WILL RETURN AFTER PRINTING THE FOLLOWING MESSAGE:

I/O ERROR
ST= XX

WHERE XX IS THE ABNORMAL I/O STATUS RETURNED BY EXEC-8

E. IF AN ATTEMPT IS MADE TO QXQT AN ABSOLUTE ELEMENT CREATED BY FORMAL, EXEC-8 WILL RESPOND WITH:

ERROR LOADING PROGRAM
ABNORMAL I/O STATUS = 25
POSSIBLE ERROR IN ABS ELEMENT OF FILE OVERWRITTEN

F. THE FILE USED FOR SAVE/RESET MUST BE ASSIGNED OR CATALOGUED. IF THE PROGRAM CANNOT ASSIGN THE FILE EXCLUSIVELY, IT WILL PRINT:

FILE CANNOT BE ASSIGNED XXXXXXXXXXXX

WHERE XXX IS THE FAC-REJECT CODE IN OCTAL. UPON RETURNING (NORMALLY OR ABNORMALLY) THE PROGRAM RESETS THE FILE'S ASSIGN-STATUS TO WHAT IT WAS BEFORE THE STATEMENT WAS EXECUTED.

3-218

AFTER A CALL TO FMLSAV THE COMMON STORAGE WILL BE COMPLETELY RE-INITIALIZED. THIS MEANS RE-CHAINING THE C-AREA, RESETTNG THE OPTIONS TO THEIR ASSUMED VALUES, AND PRINTING THE SIGNON MESSAGE.

THE PROGRAM CALLS PACK AND UNPACK FOR CHARACTER MANIPULATIONS, AND SETS A FLAG IN FMLINT, A LISTING FOLLOWS.

308

FORMAL,FMLSAV

SAVE/RESET ROUTINE

CALL FMLSAV (ARRAY)

CALL FMLRES (ARRAY)

WHERE ARRAY CONTAINS FILE. , FILE.ELEMENT, OR ELEMENT

CONVERSION FROM BINARY TO OCTAL

PACKET FOR I/O OPERATIONS

RS	AXRS		
WS	COMMON		
P	EQU	020	. FUNCTION CODE FOR READ
SNZ*	EQU	010	. FUNCTION CODE FOR WRITE
	PROC	1,1	
	NAME		
	SR	R3,P(1,1),P(1,2),P(1,3)+P(0,1)	
	END		
OCTAL*	PROC	1,5	
	L,U	R1,5	
	USL	OCTAL(1,1),3	
	SSL	OCTAL(1,1)+1,3	
	JGD	R1,5-2	
	A	OCTAL(1,1)+1,ZEROS	
	END		
MAXLEN	EQU	84	
/S(2)	LIT		
WB	'FMLSAV'		
	+	1,0	
	+	'FOR',K0	
K0	+	0	
K10	+	1,0	
HLANKS	'		
ZEROS	'000000'		
KMAXLN	+	MAXLEN	
ARG			
FILBEG			
ELT1	+	0	
FILLEN			
ELT2	+	0	
ELT3	+	0	
NEWADD			
SAVA3	+	0	
FLAG(1)	+	1	
VERFLG	EQUF	FLAG(1),,S1	
ELTFLG	EQUF	FLAG(1),,S2	
ASGFLG	EQUF	FLAG(1),,S3	
SAVFLG	EQUF	FLAG(1),,S6	
FLAG(2)	+	0	
MAXCOL	EQUF	FLAG(2),,H2	
INTFLG	EQUF	*FMLINT,,H2	
IOPKT	'FMLSSV'		
	+	0	

PACKET FOR SCRATCH FILE I/O

PACKET FOR PROGRAM-FILE MANIPULATIONS

ERROR MESSAGES

IOFUNC	+	0	
IOACW	+	0	
IOADDR	+	0	
IOPKT1			'FORMAL\$SAVES'
	+	0	
	+	0	
	+	56,BUF	
	+	0	
IOPKTS	+	0	
	+	IOPKT,IOPKT1	. SWITCH TELLS WHERE TO READ AND W
PFPKT			'FML\$SV
PFELT			'
	+	0	
	+	6,0	
PFVER			'
	+	0	
	+	0	
PFADDR	+	0	
	+	0	
USE			'QUSE FML\$SV,'
DO 8	,		'
ASG			'QASG,AX FML\$SV . '
FREE			'QFREE,A FML\$SV . '
/			
CSFST1			' SCRATCH '
CSFSTP			'FILE CANNOT BE ASSIGNED XXXXXXXXXXXX'
CSFERR			
	L,U	A1,'A	'
	S,H1	A1,FREE+1	
	OCTAL	A0	
	S	A1,CSFSTP+5	
	OCTAL	A0	
	S	A1,CSFSTP+4	
	LMJ	A0,PRET	
	7	0106,CSFSTP	
CSFER1	LMJ	A0,PRET	
	+	0106,CSFST1	
PFPCWS	+	0103,PFMSG(1)	
	+	0102,PFMSG(2)	
	+	0101,PFMSG(3)	
	+	0102,PFMSG(4)	
	+	0102,PFMSG(5)	
	+	0102,PFMSG(6)	
PFMSG(1)			'ELEMENT NOT FOUND'
PFMSG(2)			'I/O ERROR'
PFMSG(3)			'NOT PF'
PFMSG(4)			'BAD STATUS 4'


```

PFMSG(5) 'PF OVERFLOW'
PFMSG(6) 'BAD STATUS>5'
PFSERR LMJ A0,PFERR
        'PFSERR'
PFWERR LMJ A0,PFERR
        'PFWERR'
PFIERR LMJ A0,PFERR
        'PFIERR'
PFERR LXI,U A0,0101
      ER PRINT$
      TG,U A2,6
      L,U A2,6
      L A0,PFPCWS-1,A2
      J PRET+1

ELTERR
VERERR
ERRLON
      LMJ A0,PRET
      + 0102,$+1
      'SYNTAX ERROR'

ABSERR
      LMJ A0,PRET
      + 0105,$+1
      'FILE/ELEMENT INCORRECT FORMAT'

IOERR DL A1,0,A0
      DS A1,IOER2+2
      L A0,3,A0
      DSL A0,33
      LSSL A0,3
      LDSL A0,3
      A,U A0,' 00'
      S,H2 A0,IOER2
      PSRINT PFPCWS+1
      LMJ A0,PRET
      + 0104,IOER2
      'ST= 00'
      DO 3 , ' '

PRINT ERROR MESSAGE AND RETURN

PRET L A0,,A0
     ER PRINT$
     J RETURN

ACWDS
      + CDIM,FMLCM3
      + CDIM,FMLCM2
      + PTRLG,FMLCM1
      EQU $-ACWDS
      ACWDA + 28,LABELA
      ACWDB + 28,LABELB
      LABELA
      DO 14 , + 0

```

ENTRY, SET FLAG IF THIS IS SAVE, THEN UNPACK
THE ARGUMENT.

SEARCH FOR A TERMINATING ';'. IF NONE IS FOUND
USE THE MAXIMUM FOR THE LENGTH OF THE ARGUMENT, AND
STORE THE LENGTH IN MAXCOL.

SEARCH FOR FIRST NON-BLANK CHARACTER

IF IT IS ; NO SPECIFICATION WAS GIVEN

IF IT IS A . ASSUME TPF\$ AS FILENAME, AND SET
FLAG FOR ELEMENT NEEDED

	'FORMAL'	
	+	PTRLG,CDIM
DO 12	+	0
LABELB	RES	28
BUF	RES	MAXLEN
/\$ (1)		
FMLSAV*	SZ	SAVFLG
FMLRES*	S,H2	X11,WB+1
	SZ	ASGFLG
	SZ	A3
	L	R3,BLANKS
	L,U	A0,*0,X11
	S,H2	A0,ARG
	LMJ	X11,UNPACK
	+	KMAXLN
	IND	ARG
	+	BUF
	+	\$-FMLSAV,WB
	L	X11,K10
	L	A3,X11
	L,U	A2,; ;
	L,U	R1,MAXLEN
	SE	A2,BUF,*A3
	A,U	A3,1
	AN,U	A3,2
	S	A3,MAXCOL
SC1	S	X11,SAVA3
	L	A5,BUF,*X11
	TNE,U	A5,' '
	J	SC1
	L,U	A0,BUF-1,X11
	S	A0,FILBEG
	TNE,U	A5,; ;
	J	ELTERR
	TE,U	A5,; ;
	J	NOLP
	S	X11,SAVA3
GETTPF	L	A0,('TPF\$')
	S	A0,USE+2
	SZ	ELTFLG
	J	GETFIL

. PACK, UNPACK, ETC. DON'T USE R3

SEARCH FOR A . IN THE ARGUMENT, IF NONE IS FOUND,
ASSUME TPF\$ AS FILENAME.

NOLP

L,U	A2,'.'
L	A3,X11
AN,U	X11,1
L	R1,MAXCOL
SE	A2,BUF,*A3
J	GETTPF

SET FLAG FOR POSSIBLE FILE-ONLY, AND PACK FILENAME
INTO USE-IMAGE.

SNZ	ELTFLG
L,U	A2,BUF,A3
AN	A2,FILBEG
S	A2,FILLEN
SNZ	BUF-1,A3
S	A3,SAVA3
LMJ	X11,PACK
+	FILLEN
IND	FILBEG
+	USE+2
+	\$-FMLSAV,WH

USE INTERNAL NAME FML\$SV ON FILE, AND TRY TO
ASSIGN EXCLUSIVELY. GO TO CSFERR IF ASSIGN WAS
REJECTED, OTHERWISE FIGURE OUT THE CORRECT @FREE-OPTIONS
AND PUT THEM INTO THE @FREE IMAGE.

GETFIL

L,U	A0,USE
ER	CSF\$
L,U	A0,ASG
ER	CSF\$
SNZ	ASGFLG
JN	A0,CSFERR
SSL	A0,18
L,U	A1,'AR '
TG,U	A0,0100000
L,U	A1,'A '
TNE,U	A0,0100000
L,U	A1,'AX '
S,H1	A1,FREE+1
L	A3,SAVA3

NOW TEST IF FILENAME IS FOLLOWED BY ELEMENT

SC2

L	A2,BUF,A3
TE,U	A2,' '
J	GETELT
TLEM	A3,MAXCOL
J	SC2

IF ELTFLG WAS NOT SET, USE FILE, OTHERWISE SYNTAX ERROR

TNZ	ELTFLG
J	ELTERR

SET WRITE/READ ADDRESS TO ZERO, AND BRANCH ON COMMAND

NOELT

SZ
TNZ
J
J

PFADDR
SAVFLG
COPOUT
COPIN

GET ELEMENT AND VERSION. FIRST BLANK OUT THE FIELDS
IN THE PF-PACKET, AND SET ELT3 TO POINT TO THE
ELEMENT FIELD

GETELT

DL
DS
DS
L,U
S
SZ

A0,BLANKS
A0,PFELT
A0,PFVER
A0,PFELT
A0,ELT3
VERFLG

SAVE STARTING ADDRESS IN ELT1

GET0

L,U
S
J

A0,BUF,A3
A0,ELT1
GET11

SCAN IMAGE, AND LOOK FOR DELIMITER

IGNORE CYCLE.

GET1

L

A2,BUF,A3

GET11

TE,U
TNE,U
J
TNE,U
J
TLEM
J

A2, ' '
A2, ' '
GET3
A2, ' '
GET
A3
GF

END OF IMAGE OR BLANK MEANS NO MORE FIELDS
SO PRETEND WE ALREADY HAD A VERSION

GET3

SZ
J

VERFLG
GET2

/ FOUND, CHECK IF WE ALREADY HAD VERSION
THEN SET FLAG

GET4

TZ
J
SNZ

VERFLG
VERERR
VERFLG

PACK NAME INTO AREA GIVEN IN ELT3

GET2

AN,U

A0,BUF,A3

ERROR IF > 12 CHARACTERS

TLE,XU
J
SM
A,U
S
LMJ
+
IND

A0,-12
ERRLON
A0,ELT2
A3,1
A3,SAVA3
X11,PACK
ELT2
ELT1

GO BACK FOR VERSION IF WE HAD ELT/

BRANCH ON COMMAND

GET NEXT FREE LOCATION IN PROGRAM FILE

WRITE THE COMMON BLOCKS, USING IOSUB
THEN WRITE THE SAVED DRUM AREA
AND FINALLY THE LABEL-BLOCK

INSERT ABSOLUTE ELEMENT INTO PROGRAM-FILE.

	IND	ELT3
	+	S-FMLSAV, WB
	TNZ	VERFLG
	J	NOVERS
	L, U	A0, PFVER
	S	A0, ELT3
	L	A3, SAVA3
GET5	L	A2, BUF, A3
	TE, U	A2, ' '
	J	GET0
	TLEM	A3, MAXCOL
	J	GET5
NOVERS		
	TZ	SAVFLG
	J	ELTIN
	L, U	A0, PFPKT
	ER	PFWL\$
	JNZ	A2, PFWERR
	S	A1, PFAUDR
/.		
COPOUT		
	L, U	A1, W\$
	LMJ	X11, IOSUB
	L	A0, IOPKTS+1
	LMJ	X11, GETSEC
	SZ	LABELA+16
	L	A1, IOADDR
	S	A1, NEWADD
	AN	A1, PFAUDR
	S	A1, PFPKT+9
	MSI, U	A1, 28
	S	A1, LABELA
	L	A1, ACWDA
	S	A1, IOACW
	L	A1, PFAUDR
	S	A1, IOADDR
	ER	IOW\$
	TZ, S1	IOFUNC
	LMJ	A1, IOERR
	TNZ	PFADDR
	J	RET1
	SZ	PFPKT+8
	SZ	PFPKT+11

MARK FMLINT TO RE-INITIALIZE STORAGE, AND RETURN

RET1

SEARCH PROGRAM-FILE FOR THE ELEMENT

ELTIN

READ LABEL-BLOCK AND TEST IF IT HAS THE RIGHT FORMAT.
THEN USE IOSUB TO READ THE COMMON-BLOCKS, AND FINALLY
COPY DRUM-AREA AND SET INTFLG IN CASE IT WASN'T SET.

CGPIN

L
LN,U
ER
JNZ
A1,NEWADD
A0,PFPKT
PFI\$
A2,PFIERR

SZ
J
INTFLG
RETURN

L,U
ER
JNZ
A0,PFPKT
PFS\$
A2,PFSERR

L
S
L
S
L,U
S,S2
L,U
ER
TZ,S1
LMJ
DL
DTE
J
L,U
LMJ
TNZ
J
LMJ
+
+
J
+
L
SSC
LMJ
A0,PFAUDR
A0,IOAUDR
A1,ACWUB
A1,IOACW
A1,R\$
A1,IOFUNC
A0,IOPKT
IOW\$
IOFUNC
A1,IOERR
A0,LABELB+14
A0,LABELA+14
ABSERR
A1,R\$
X11,IOSUB
ROLFLG+2
COPIN1
X11,S2INIT
IOPKT1
0
CSFER1
\$-FMLSAV,WH
A0,IOPKTS+1
A0,18
X11,GETSEC

COPIN1

SNZ
INTFLG

COMMON RETURN. FREE FILE IF NECESSARY, RESTORE X11,
RESET SAVFLG, AND RETURN

RETURN

L,U
TZ
A0,FREE
ASGFLG

THIS SUBROUTINE READS OR WRITES THE NACWD AREAS GIVEN
BY THE ACCES-WORDS AT ACWDS

IOSUB

IOLoop

ER	CSF\$
L	X11, WB+1
SNZ	SAVFLG
J	2, X11
L, U	A3, NACWD-1
L, U	A0, IOPKT
S, S2	A1, IOFUNC
L	A1, PFAUDR
A, U	A1, 1
S	A1, IOAUDR
L	A1, ACWDS, A3
S	A1, IOACW
ER	IOW\$
TZ, S1	IOFUNC
LMJ	A1, IOERR
L, H1	A1, IOACW
A, U	A1, 27
DSL	A1, 36
DI, U	A1, 28
A	A1, IOADDR
S	A1, IOADDR
JGD	A3, IOLoop
J	0, X11

SUBROUTINE TO COPY DRUM-SAVE AREA TO OR FROM
THE ABSOLUTE ELEMENT.

CHECK IF THERE IS ANYTHING
THEN INITIALIZE THE I/O PACKETS

GETSEC

TNZ	ROLFLG+2
J	0, X11
S	A1, IOPKTS
L, U	A2, R\$*/6
S, H1	A2, 3, A1
SSL	A1, 18
L, U	A2, W\$*/6
S, H1	A2, 3, A1
L	A1, IOPKT1+4
S	A1, IOPKT+4
L	A1, K10
LN, U	A4, 1

PICK UP ONE PLANE WORD

GETS1

L	A2, PLANE, A1
LSSL	A2, 1
L, U	A3, 34
A, U	A4, 35

SEE IF BIT IS SET

CALCULATE SECTOR ADDRESS
THEN READ AND WRITE THE SECTORS

INCREMENT ADDRESS IN SAVE-FILE

LOOP UNTIL WORD IS EXHAUSTED
THEN GET NEXT UNTIL WE ARE DONE

GETS2	JZ	A2,GETS4
	JPS	A2,GETS3
	ANU	A4,A3
	A	A5,A5
	S	A5,IOPKT1+5
	L,H2	A0,IOPKTS
	ER	IOW\$
	TZ,S1	3,A0
	J	IOERR
	L,H1	A0,IOPKTS
	ER	IOW\$
	TZ,S1	3,A0
	J	IOERR
	L	A0,IOPKT+5
	A,U	A0,2
	S	A0,IOPKT+5
GETS3	JGD	A3,GETS2
GETS4	TLEM,U	A1,PLNSZ1-1
	J	GETS1
	J	0,X11
	END	

3.4.4. DUMP STATEMENT

THE DUMP STATEMENT IS HANDLED BY FMLDMP. THE ROUTINE GOES THROUGH THE SYMBOL TABLE, AND PRINTS OUT THE NAMES OF VARIABLES BY FMLIOC WITH THE USE OF OUTPUT PACKING ROUTINES FMLPCK AND FMLPKS. THE EXPRESSION VALUES ARE PROCESSED BY PROGRAM ELEMENT FMLDMQ. FMLDMQ PICKS UP THE CONSECUTIVE ITEMS OF THE EXPRESSION, IT DEFINES THE CORRESPONDING HOLLERITH DATA AND TRANSMITS IT TO THE OUTPUT PACKING ROUTINE.

FORMAL,FMLDMP

SUBROUTINE LEVEL

SYMBOL LEVEL

INDIRECT

UNDEFINED VARIABLE

NONSUBSCRIPTED, DEFINED VARIABLE

SUBSCRIPTED VARIABLE

SUBSCRIPT LEVEL

ASSIGNED SUBSCRIPTED VARIABLE

SUBROUTINE FMLDMP(XX)

INCLUDE CMMN

INCLUDE PWORD

EXTERNAL FMLIOC

LOGICAL ALL,EXP

ALL = XX .GT. 20

EXP = ((XX/2)*2) .EQ. XX

IF (NS .EQ. 0) GO TO 99

SR = NS

IF (ALL) GO TO 1

CALL FMLINT (\$99,0)

SR = NSB

1 CONTINUE

EX = LAST(SR)

IF (EX .EQ. 0) GO TO 1000

2 CONTINUE

CALL FMLPKI (D(EX),1,FMLIOC)

IF(C(EX).LT,0) GO TO 21

I = ITYP(EX)

IF (AND(I,24)-8) 22,23,24

21 CONTINUE

CALL FMLPKO

PRINT '(1H+,15X, 'DUMMY VARIABLE '))'

GO TO 2000

22 CONTINUE

CALL PRTPCK (1)

CALL PRTPCK (2)

GO TO 2000

23 CONTINUE

IT = LAST(EX)

IF (IT .EQ. 0) GO TO 1900

ASSIGN 2000 TO JUMP

GO TO 4

24 CONTINUE

IF (LAST(EX) .EQ. 0) GO TO 2000

SS = LAST(EX)

ASSIGN 3000 TO JUMP

3 CONTINUE

I = FLD (3,3,C(SS))

CALL FMLPKS (\$99,I,D(SS))

IF (FLD(0,3,C(SS)) - 6) 31, 32, 33

EXPRESSION LEVEL

UNASSIGNED SUBSCRIPTED VARIABLE

END OF SUBSCRIPTS

```

31  CONTINUE
    IF (LAST(SS) .EQ. 0) GO TO 2900
    IT = LAST(SS)

4   CONTINUE
    IF (.NOT. EXP ) GO TO 1900
    H1IT = H1(IT)
    IF (H1IT .NE. 0) GO TO 41
    CALL PRTPCK (1)
    CALL FMLDEX ($99,IT)
    GO TO JUMP

41  PRINT '(1H+,10X',EXPRESSON ON DRUM',15)',H1IT
    GO TO JUMP

32  CONTINUE
    CALL PRTPCK (2)
    GO TO 3000

33  CONTINUE
2900 CALL FMLPKO
3000 SS = NEXT (SS)
    IF (NEXT(SS) .EQ. 0) GO TO 2000
    SS = NEXT(SS)
    CALL FMLPCK (D(EX),1)
    GO TO 3

1900 CALL FMLPKO
2000 EX = NEXT(EX)
    IF (EX .EQ. 0) GO TO 1000
    GO TO 2

1000 IF (NEXT (SR) .EQ. 0 .OR% .NOT. ALL) RETURN
    SR = NEXT(SR)
    GO TO 1

99  CALL ERRRET ('FMLDMP')
    RETURN
    SUBROUTINE PRTPCK (KKK)
    GO TO (100,200),KKK

100 CALL FMLPCK ('=',1)
110 CALL FMLPKO
    RETURN

200 CALL FMLPCK ('UNASSI',1)
    CALL FMLPCK ('GNED ',1)
    GO TO 110
END

```


CONTINUE LOOP

```

35      ID = D(IT)
      DEG = S-24
40      CALL FMLPCK (ID,1)
      CALL FMLPCK ('<',1)
      CALL FMLPCK (DEG,2)
      CALL FMLPCK ('>',1)
100     IT = NEXT(IT)
      CALL FMLPCK ('π',1)
      S = ITYP(IT)

      IF (IT .NE. 0) GO TO 1
      CALL FMLPKO
      RETURN
999     CALL ERRERR ($99,IT)
99      CALL ERRRET('FMLDEX')
      RETURN 1
      END

```

3-234

3.4.5. ASSIGN STATEMENT

THE ASSIGN STATEMENT IS HANDLED BY FMLASG. AFTER INITIALIZA-
TION, THE ROUTINE TRANSLATES THE EXPRESSION BY EXPRES AND STORES
THE RESULT BY SYMBOL.

32940
FORMAL,FMLASG

FORMAL ASSIGNMENT STATEMENT

PROCESS FORMAL ASSIGNMENT STATEMENT

BCD IS THE SYMBOLIC ARKAY OF THE ORIGINAL STATEMENT,

N IS THE NUMBER OF FORTRAN TYPE ARGUMENTS IN BCD.

GET FORTRAN TYPE ARGUMENT LIST,
AND INITIALIZE SCANNER

:
:
CALL EXPRESSION TRANSLATOR

:
:
STORE IN THE SYMBOL TABLE

RETURN

ERROR OCCURS

SUBROUTINE FMLASG(BCD)

INCLUDE CMMN

INCLUDE PWORD

CALL FMLINT (\$100,2)
CALL ISCANG (BCD, 220, 0)

CALL EXPRES (\$100, 0)
EPTR = NP

CALL SYMBOL (\$100,2)

101 RETURN

100 CALL ERRRET ('FMLASG')
GO TO 101
END

3-236

3.4.6. OPTION STATEMENT

THE OPTION STATEMENT IS HANDLED BY PROGRAM ELEMENT FMLOPT.
AFTER INITIALIZATION, THE ROUTINE SCANS THE OPTION WORDS, AND SETS
THE CORRESPONDING SWITCH WORDS IN FMLCM1.

FORMAL.FMLOPT
 FORMAL OPTION STATMENT ROUTINE

SUBROUTINE FMLOPT(BCD)
 INCLUDE CMMN

 OPTION STATEMENT PROGRAM

THIS ROUTINE SETS THE OPTION SWITCHES INTO POSITIONS
 SPECIFIED BY AN OPTION STATEMENT. THE LIST OF THE
 OPTIONS IS IN BCD. THE OPTIONS WITH THEIR INITIAL
 SETTINGS ARE AS FOLLOWS:

INTGSW=.TRUE. CORRESPONDS TO INT OPTION CAUSING THE
 EVALUATION OF THE INTEGER VALUED
 MATHEMATICAL FUNCTIONS WHENEVER THEIR
 ARGUMENTS ARE CONSTANTS. NOINT OPTION
 TURNS THE SWITCH TO .FALSE., THUS PRO-
 HIBITING THE EVALUATION OF THE INTEGER
 VALUED FUNCTIONS.

MATHSW=.TRUE. CORRESPONDS TO MFCT OPTION CAUSING THE
 EVALUATION OF THE MATHEMATICAL FUNCTIONS
 WHENEVER THEIR ARGUMENTS ARE CONSTANTS.
 NOMFCT TURNS THE SWITCH TO .FALSE., THUS
 PROHIBITING THE EVALUATION OF THE MATHE-
 MATICAL FUNCTIONS.

PRODEX=.TRUE. CAUSES THE DISTRIBUTION OF THE EXPONENT
 ON A PRODUCT TO ITS FACTORS. NOPREX TURNS
 THE SWITCH TO .FALSE. WHICH LEAVES THE
 EXPONENTIALIZATION OF A PRODUCT UNCHANGED.

EXPDSW=N CORRESPONDS TO EXPAND(N) OPTION WITH
 INITIAL SETTING N=1, WHICH CORRESPONDS
 TO THE USE OF THE DISTRIBUTIVE LAW, BUT
 LEAVES SUMS RAISED TO INTEGERS UNCHANGED.
 WITH OPTION EXPAND(N), N>1, THE SUMS
 RAISED TO POSITIVE INTEGERS UP TO N TO
 BE EXTENDED BY THE BINOMIAL LAW. N=0
 PROHIBITS THE USE OF THE DISTRIBUTIVE
 LAW.

BASE=0 INITIAL SETTING CORRESPOND TO BASE(0)
 OPTION WHICH LEAVES CONSTANTS RAISED TO
 NON-CONSTANT POWERS UNCHANGED. BASE=1
 CORRESPONDS TO BASE(2) OPTION, BASE=2
 TO BASE(10) OPTION, BASE=3 TO BASE(E)
 OPTION. THESE OPTIONS CAUSES THE
 CONSTANT BASES TO BE CHANGED TO 2, 10 OR
 E, RESPECTIVELY.

NOFORT/FORTRN OPTION 'FORTRN' FORCES THE FORMATED
 OUTPUT STATEMENTS TO GENERATE FORTRAN
 TYPE OF OUTPUT, I.E. OUTPUT IS ARRANGED
 BETWEEN COL. 7-72 WITH COL. 6 AS CONTI-
 NUATION COLUMN, AND BRACKETS ARE REPLACED
 BY PARENTHESES.

INITIALIZE FORMAL CALLING AND SCANNER

GET OPTION SYMBOL FROM THE LIST

IF THIS IS NOT A BCD WORD, SKIP IT
IF IT IS A BCD WORD TERMINATED BY (
THEN GO TO CHECK BASE AND EXPAND

NOW CHECK BCD WORD AGAINST THE LIST

... IN THE LIST
... SET LOGICAL VARIABLE AND GO TO PROPER
... SWITCH

... NOT IN LIST, MUST BE AN ERROR

... INT, NOINT
... MFCT, NOMFCT
... PRODEX, NOPREX
... FORTRAN TYPE OUTPUT SWITCH

THIS MUST BE THE BASE OR EXPAND, GET SUBSCRIPT
THEN GO TO PROPER PLACE

... EXPAND, SET SWITCH TO THE SUBSCRIPT

... BASE, SET BASE TO THE CODED SUBSCRIPT

LOGICAL IS
PARAMETER NOPT = 4
PARAMETER NOPT2 = NOPT * 2
DIMENSION H(NOPT2)
DATA H /'INT MFCT PRODEXFOTRNNPOINT NOMFCTNOPREXNOFORT'/

CALL FMLINT (\$100, 2)
CALL ISCANG (BCD, 220, 0)

10 CALL GSCANR (\$100, IND, IOP, J, IL)

IF (IND .NE. 3) GO TO 120
IF (J .EQ. 1) GO TO 70

I = ISRCH (NOPT2, H, IOP, \$30)

IS = I .LE. NOPT
GO TO (40, 50, 60, 65, 40, 50, 60, 65), I

30 CALL FMLERR (\$100, IOP, 1, 11)

40 INTGSW = IS
GO TO 120

50 MATHSW = IS
GO TO 120

60 PRODEX = IS
GO TO 120

65 XFLG2 = 0
IF (IS) XFLG2 = 1
GO TO 120

70 CALL GSCANR (\$100, IND, JOP, J, IL)
IF (IOP .EQ. 'BASE') GO TO 80
IF (IOP .NE. 'EXPAND') CALL FMLERR(\$100,IOP,1,11)

IF (IND .NE. 1) CALL FMLERR (\$100,'EXPAND',1,12)
EXPDSW = JOP
GO TO 110

80 BASE = 0
IF (IND .EQ. 3) GO TO 90

```

      :
      :
      :
      :
      :
      :
      :
      :
      :
      :
SKIP RIGHT PARENTHESES
      :
      :
CHECK FOR COMMA OR END OF LIST
      :
      :
COMMA, GO BACK TO GET NEXT OPTION SYMBOL
      :
      :
SEMICOLON, END OF LIST
      :
      :
ERROR RETURN

END

```

```

      IF (IND .NE. 1) CALL FMLERR ($100,'BASE',1,12)
      IF (JOP .EQ. 2) BASE = 1
      IF (JOP .EQ. 10) BASE = 2
      GO TO 110
90    IF (JOP .NE. 'E') CALL FMLERR ($100,'BASE',1,12)
      BASE = 3

110   CALL GSCANR ($100, IND, IOP, J, IL)
      IF (IND .NE. 4 .OR. IOP .NE. 5) CALL FMLERR ($100,'',1,10)

120   CONTINUE
      CALL GSCANR ($100, IND, IOP, J, IL)

      IF (IND .EQ. 4 .AND. IOP .EQ. 20) GO TO 10

      IF (IND .EQ. 4 .AND. IOP .EQ. 23) RETURN

100   CALL ERRRET ('FMLOPT')
      RETURN

      END

```

3-240

3.4.7. ERASE STATEMENT

THE ERASE STATEMENT IS HANDLED BY PROGRAM ELEMENT FMLERS.
AFTER INITIALIZATION, THE ROUTINE COLLECTS THE NAMES OF VARIABLES
BY FMLSCN, AND THEY ARE DELETED TOGETHER WITH THEIR EXPRESSION
VALUES BY SYMBOL.

FORMAL,FMLERS

FORMAL ERASE STATEMENT

THIS ROUTINE PROCESSES THE FORMAL STATEMENT

'ERASE'.

IT FIRST COLLECTS ALL ARGUMENTS INTO A CHAIN AND
THEN PROCESSES ONE BY ONE.

INITIALIZE AND GET FORTRAN TYPE ARGUMENT LIST

:
:
COLLECT ARGUMENT LIST INTO A CHAIN STARTING AT NK
:

LOOP

: GET THE TOP ARGUMENT FROM THE CHAIN

:
: ERASE THE POINTER FROM THE SYMBOL TABLE
:

CONTINUE THE LOOP UNTIL ARGUMENT LIST EXHAUSTED

RETURN

ERROR OCCURS

SUBROUTINE FMLERS(BCD)
INCLUDE CMVNCALL FMLINT(\$10,2)
CALL ISCANG(BCD,220,0)

CALL FMLSCN(\$10)

1 CALL GETARG

CALL SYMBOL(\$10,3)

IF (NK .NE. 0) GO TO 1

20 RETURN

10 IF (NK .NE. 0) CALL FMLCHD (\$11, NK)

11 NK = 0

CALL ERRRET ('FMLERS')

GO TO 20

END

3-242

3.4.8. DO AND END STATEMENTS

THE DO AND END STATEMENTS ARE HANDLED BY THE SUBPROGRAMS FMLD00 AND FMLD01. FMLD01 IS ESSENTIALLY A BUFFERING ROUTINE THAT BUFFERS THE LOOP UPON INPUT AND RETURNS THE STATEMENTS AND MAINTAINS A CURRENT POINTER DURING EXECUTION OF THE LOOP. THE SUBPROGRAM FMLD00 HAS TWO ENTRY POINTS FMLD0 AND FMLEND. EACH HAS TWO FUNCTIONS, ONE DURING INPUT AND ONE DURING EXECUTION. DURING INPUT THE TWO ENTRY POINTS HANDLE THE DO AND END STATEMENTS RESPECTIVELY, SETTING UP THE LOOP LINKAGES. DURING EXECUTION, FMLD0 INTERPRETS THE DO STATEMENT, EVALUATES THE LOOP PARAMETERS AT EACH ITERATION, AND MAKES THE DECISION AS TO WHETHER OR NOT TO CONTINUE; WHILE FMLEND SIMPLY TRANSFERS BACK TO THE TOP OF THE LOOP.

FORMAL,FMLD00

DO STATEMENT HANDLER FOR INTERACTIVE FORMAL
 THIS SUBROUTINE PROCESSES THE DO AND END STATEMENTS FOR
 INTERACTIVE FORMAL. THE METHOD USED HAS NO LIMIT ON
 THE NUMBER OF LEVELS IN A NEST, BUT DOES PLACE A LIMIT ON
 THE NUMBER OF CARD IMAGES IN A NEST. THE METHOD USED CONSISTS
 OF TWO PHASES: INPUT -- DURING WHICH ALL THE DO AND END
 CARDS IN ONE NEST ARE MODIFIED TO CONTAIN POINTERS AND
 ARE STACKED WITH THE REST OF THE FORMAL IMAGES -- AND
 EXECUTION -- DURING WHICH THE NEST IS UNSTACKED AND EVALUATED
 TO COMPLETION. DURING EXECUTION THE INDEX VALUE, INITIAL
 VALUE, STEP-SIZE, AND LAST VALUE ARE EVALUATED AT EACH ITERATION
 OF THE LOOP. THESE MAY BE ANY INTEGER-VALUED FORMAL EXPRESSION
 AND MAY VARY DYNAMICALLY DURING LOOP EXECUTION. IN ADDITION,
 THE LOOP MAY BE THOUGHT OF AS A CONDITIONAL STATEMENT.
 IF THE INITIAL VALUE FAILS THE END-VALUE TEST UPON ENTERING
 THE LOOP, THEN THE LOOP IS BYPASSED.

... INPUT DO STATEMENT

POINT DO STATEMENT AT PREVIOUS DO STATEMENT

SCAN FOR INDEX NAME

CHECK TO SEE IF INDEX HAS BEEN PREVIOUSLY USED

5

... EXECUTE DO STATEMENT

SUBROUTINE FMLD0(IN, IDIM, INPUT, EXECUT)
 INCLUDE CMMN
 COMMON/DOSTAK/B(1)
 INCLUDE PWORD
 LOGICAL INPUT, EXECUT, ANSWER
 DIMENSION IN(IDIM)
 DEFINE DOPART(J)=FLD(0,18,B(J))
 DEFINE MARK(J)=FLD(0,18,B(J+IDIM-2))
 DEFINE CURTOP(J)=FLD(18,18,B(J+IDIM-2))
 DEFINE CURVAL(J)=B(J+IDIM-1)
 DATA SEMIC/' ;', DOWORD/'DO DO ', CRTOP/0/
 DOLEV=DOLEV+1
 IF(EXECUT)GO TO 100

INPUT=.TRUE.
 CALL DSTACK(IN, IDIM, J, \$822)
 DOPART(J)= '
 MARK(J)=SEMIC

CURTOP(J)=CRTOP
 CRTOP=J

CALL FMLSC1(B(J), IDIM, 0)
 CALL FMLSC2(\$801, IND, CURVAL(J), IT, ICT)
 DOPART(J)=DOWORD

J=CURTOP(J)
 IF(J.EQ.0)RETURN
 IF(CURVAL(J).EQ.CURVAL(CRTOP))GO TO 802
 GO TO 5

EVALUATE EXPRESSION

EXIT IF INDEX IS SUBSCRIPTED

TEST FOR STEP-SIZE

EXIT IF LESS THAN TWO OR MORE THAN THREE
EXPRESSIONS
ARE IN THE LIST

TEST FOR INTEGER VALUED STEP-SIZE

SET STEP-SIZE

SET LAST VALUE

SET INITIAL VALUE

UPDATE INDEX

SKIP DO LOOP IF CONVERSATIONALLY INDICATED

TEST CURRENT VALUE AGAINST LAST VALUE

STORE INDEX VALUE

```

100  CALL LSTSTK(K)
      DOPART(K)=0

      CALL FMLSC1(B(K),IDIM,0)
      CALL EXPRES($803,0)

      IF(N2,NE,0)GO TO 807
      J=NEXT(NP)
      J1=J
      II=U(NP)
      INC=1

      IF(II,EQ,2)GO TO 110

      IF(II,NE,3)GO TO 804
      ERR=3

      IF(ITYP(J),NE,0)GO TO 805

      INC=D(J)
      J=NEXT(J)
110  J2=J
      ERR=2

      IF(ITYP(J),NE,0)GO TO 805
      LVAL=D(J)
      J=NEXT(J)
      J3=J
      ERR=1

      IF(ITYP(J),NE,0)GO TO 805
      IVAL=D(J)
      CALL FMLCHH ($800, J1, J2)
      IF(CURVAL(K),EQ,N1)GO TO 120

      CURVAL(K)=CURVAL(K)+INC
      GO TO 130
120  CURVAL(K)=IVAL
130  DOPART(K)=DOWORD

      IF(.NOT.ANSWER)GO TO 140

      IF(INC,LT,0,AND,CURVAL(K),LT,LVAL)GO TO 140
      IF(INC,GE,0,AND,CURVAL(K),GT,LVAL)GO TO 140
      D(J)=CURVAL(K)
      D(NP)=1
      EPTR=NP

```


IF THE NEST IS COMPLETE, RESUME INTERACTIVE
FORMAL

... INPUT END STATEMENT

POINT CORRESPONDING DO STATEMENT TO NEXT
STATMENT

PROCESS END STATEMENT BY BRANCHING TO THE TOP OF THE LOOP

QUERY THE USER AS TO WHETHER THE LOOP IS TO BE

```

140    CURVAL(K)=N1
      ANSWER=.TRUE.
      DOLEV=DOLEV-1
      CALL FMLCHD($800, NP)
      CALL SETSTK(CURTOP(K))
      IF(CURTOP(K).GT.0)RETURN

```

```
ENTRY FMLEND(IN,IDIM,INPUT,EXECUT)
DOLEV=DOLEV-1
IF(DOLEV,LT.6)GO TO 821
IF(IN(1).EQ.'END T ')GO TO 822
IF(EXECUT)GO TO 300
```

```
CALL DSTACK(IN,IDIM,J,$822)
MARK(J)=SEMIC
```

```

CURTOP(J)=CRTOP
SAVE=CURTOP(CRTOP)
CALL NXTSTK(K)
IF(SAVE.EQ.0)K=0

```

CURTOP (CRTOP) = K
CRTOP = SAVE

```
IF (DOLEV.NE.0) RETURN
INPUT=.FALSE.
EXECUT=.TRUE.
ANSWER=.TRUE.
CALL INISTK
RETURN
```

```

300  CALL LSTSTK(K)
      CALL SETSTK(CURTOP(K))
      IF(IN(1).NE.'END Q')RETURN

```

CONTINUED

```

301  WRITE(6,922)IN(2)
      READ(5,923)FLDANS
      ANSWER=.TRUE.
      IF(FLDANS.EQ.'N')ANSWER=.FALSE.
      IF(FLDANS.NE.'Y'.AND.ANSWER.EQ..TRUE.)GO TO 301
      RETURN

800  CALL ERRRET('FMLD0')
      DOPART(K)=DOWORD
      GO TO 140
801  DOPART(J)=DOWORD
      CALL ERRRET('FMLD0')
      RETURN
802  WRITE(6,902)CURVAL(J)
      RETURN
803  GO TO 800
804  WRITE(6,904)
      GO TO 800
805  WRITE(6,905)ERR
      GO TO 800
806  GO TO 800
807  WRITE(6,907)
      GO TO 800
821  WRITE(6,921)
      DOLEV=0
      CALL ERRRET('FMLEND')
      EXECUT=.FALSE.
      RETURN

822  DOLEV=0
      CALL INISTK
      CRTOP=0
      EXECUT=.FALSE.
      INPUT=.FALSE.
      RETURN
902  FORMAT(' VARIABLE ',A6,' ALREADY USED TO INDEX A DO-LOOP IN ',
      ' THIS NEST')
904  FORMAT(' WRONG NUMBER OF DO LOOP PARAMETERS')
905  FORMAT(' THE ',I2,'-TH DO LOOP PARAMETER IS NOT AN INTEGER')
907  FORMAT(' DO LOOP INDEX IS A SUBSCRIPTED VARIABLE')
921  FORMAT(' END DISCOVERED THAT DOES NOT HAVE A CORRESPONDING'
      ' DO')
922  FORMAT(' CONTINUE LOOP: ',A6,'?(YES OR NO)')
923  FORMAT(A1)
      END

```

ERASE DO LOOP THAT IS CURRENTLY BEING COMPILED

FORMAL,FMLU01
DO LOOP STACK ROUTINE

INITIALIZE STACK

GET NEXT STACK LOCATION

GET LAST STACK LOCATION

SET STACK LOCATION

LOAD A STACK LOCATION

READ A STACK LOCATION

SUBROUTINE INISTK
IMPLICIT INTEGER (A-Z)
PARAMETER NIMAGE=50 @ MAX. NUMBER OF FORMAL IMAGES PER DO-LOOP
PARAMETER ILNG=14
PARAMETER SSIZE=ILNG*NIMAGE
COMMON/DOSTAK/B(SSIZE)
DATA TOP/1/,NEXT/1/

NEXT=TOP
RETURN

ENTRY NXTSTK(LOC)
LOC=NEXT
RETURN

ENTRY LSTSTK(LOC)
LOC=NEXT-ILNG
RETURN

ENTRY SETSTK(LOC)
NEXT=LOC
RETURN

ENTRY DSTACK(BUF,N,LOC,S)
DIMENSION BUF(1)
IF(NEXT.GT.SSIZE)GO TO 800
LOC=NEXT
DO 10 J=1,N
10 B(LOC+J-1)=BUF(J)
NEXT=NEXT+ILNG
RETURN

ENTRY USTACK(BUF,N,LOC)
LOC=NEXT
DO 20 J=1,N
20 BUF(J)=B(LOC+J-1)
NEXT=NEXT+ILNG
RETURN

800 WRITE(6,900)
BUF(1)='END T'
RETURN 4
900 FORMAT(' DO LOOP STACK EXCEEDED, STACK ERASED')
END

3-248

3.4.9. FORTRAN VALUED FUNCTIONS AND NCOUNT

THE FORTRAN VALUED FUNCTIONS ARE HANDLED BY PROGRAM ELEMENT
FMLFOR. THE NCOUNT STATEMENT IS HANDLED BY PROGRAM ELEMENT FRECT;
WHICH IS ALSO CALLED BY FMLFOR.

FORMAL,FMLFOR

FORTRAN VALUED FUNCTION ROUTINES

LDOP('F-EXPR') GIVES AN INTEGER CORRESPONDING TO THE LEAD
OPERATOR OF THE 'F-EXPR'.
NARG('F-EXPR') GIVES THE NUMBER OF ARGUMENTS OF THE LEAD
OPERATOR OF THE 'F-EXPR' AS AN INTEGER
IVALUE('F-EXPR') GIVES THE TRUNCATED INTEGER VALUE OF THE
'F-EXPR' WHEN IT IS CONSTANT, ZERO OTHERWISE.
VALUE('F-EXPR') GIVES THE FLOATING-POINT VALUE OF THE 'F-EXPR'
WHEN IT IS A CONSTANT, ZERO OTHERWISE
IDENT('F-EXPR1,F-EXPR2') GIVES INTEGER 1 WHEN THE TWO EXPR.'S
ARE IDENTICAL, ZERO OTHERWISE

NCOUNT (0) GIVES THE AVAILABLE NUMBER OF CORE STORAGE
THE INTERNAL REPRESENTATION OF THE MATH. AND FORMAL
FUNCTIONS IS EQUAL TO ID.# + (N * 128), WHERE N IS THE
NUMBER OF REQUIRED ARGUMENTS , IF N IS EQUAL TO OR LESS
THAN 3. WHEN N = 4 IT INDICATES AN ARBITRARY BUT EVEN NUMBER
OF ARGUMENTS. WHEN N = 5 IT INDICATES ARBITRARY BUT ODD NUMBER
OF ARGUMENTS.

INTERNAL REPRESENTATION AND LEAD OPERATOR ID.

LEAD OP.

INTERNAL REPRESENTATION

ID.			
0	CONSTANT ZERO	0	= ITP(I)
1	INTEGER CONSTANT	0	"
2	RATIONAL CONSTANT	1	"
3	REAL CONSTANT	3	"
5	FUNCTION ARGUMENT UNDEFINED	5	"
6	NON-SUBSCRIPTED VARIABLE	6	"
7	SUBSCRIPTED VARIABLE	7	"
16	LIST	16	"
18	+	18	"
19	*	19	"
20	**	20	"
23	DIFFERENTIAL OPERATOR	23	"
25-31	UNDEFINED FUNCTION	25-31	"
32	EXP	32+128	= H1(I)
33	LOG	33+128	"
34	TNH	34+128	"
35	SIN	35+128	"
36	COS	36+128	"
37	ATN	37+128	"
50	FAC	50+128	"
51	BIN	51+256	"
52	STEP	52+384	"
--	EXPAND	64+128	"
--	CODEM	65+128	"
--	COEFF	66+256	"

--	NUM	67+128	''
--	DENOM	68+128	''
--	EXPON	69+128	''
--	BASE	70+128	''
--	ARG	71+256	''
--	SUBST	72+640	''
--	REPLAC	73+640	''

```

FUNCTION LDOP (BCD)
  INCLUDE CMMN
  INCLUDE PWORD
  DIMENSION NAME (5)
  DATA NAME /'LDOP' NARG IVALUEVALUE IDENT '/'
  EQUIVALENCE (LDOP,I)
  REAL FP

```

```

*****
LDOP (BCD)
*****

```

```

*****
NARG (BCD)
*****

```

```

*****
IVALUE (BCD)
*****

```

```

*****
VALUE (BCD)
*****

```

```

*****
IDENT (BCD)
*****

```

```

INITIALIZE
  GET FORTRAN-TYPE ARGUMENTS
  :
  INITIALIZE SCANNER ROUTINE
  :
  :
  BRANCH ACCORDING TO ENTRY POINTS
*
```

```

II=1
GO TO 10

```

```

ENTRY NARG (BCD)
II=2
GO TO 10

```

```

ENTRY IVALUE (BCD)
II=3
GO TO 10

```

```

ENTRY VALUE (BCD)
II=4
GO TO 10

```

```

ENTRY IDENT (BCD)
II = 5

```

```

10  CALL FMLINT ($20, 2)
    CALL ISCANG (BCD, 220, 0)
    CALL EXPRES ($20, 3)
    GO TO (100,200,300,400,600),II

```

*--- LDOP

```

*      :
*      : CHECK IF IT IS A LIST
*      :
*      :
*      : IF IT IS A MATHEMATICAL FUNCTION, CHECK
*      : IDENTIFICATION IN D(L).
*      : IF IT IS NOT AN INTEGER OR RATIONAL CONSTANT,
*      : THEN ITS TYPE IS ITS IDENTIFICATION
*      :
*      :
*      :
*      :
*      :
*      :
*      :
*      :
*      :

```

*--- NARG

```

*      :
*      : CHECK FOR LIST
*      :
*      :
*      : CHECK FOR CONSTANT OR VARIABLE
*      : IT IS AN OPERATOR OR FUNCTION
*      :
*      :
*      :
*      :
*      :

```

*--- IVALUE

```

*      :
*      : CHECK # OF SUBEXPRESSIONS
*      :
*      :
*      : IS IT A NUMBER
*      :
*      : YES, INTEGER
*      :
*      :
*      :
*      : RATIONAL
*      :
*      :
*      :
*      : REAL
*      :
*      :

```

100 I = 16

```

IF (H2(NP).GT.1) GO TO 500
L=NEXT(NP)
I=ITYP(L)

```

```

IF (I.EQ.21) GO TO 110
IF (I.GT.1) GO TO 500
I=I+1
IF (D(L).NE.0) GO TO 500
I=0
GO TO 500
110 I=FLD(11,7,D(L))
GO TO 500

```

200 I=H2(NP)

```

IF (I.GT.1) GO TO 500
L=NEXT(NP)
K=ITYP(L)
I=0

```

IF (K.LE.17) GO TO 500

```

I=H2(L)
IF (K.GT.24) I=K-24
GO TO 500

```

300 I=0

```

IF (H2(NP).NE.1) GO TO 500
L=NEXT(NP)
K=ITYP(L)

```

IF (K.GT.3) GO TO 500

```

I=D(L)
IF (K.EQ.0) GO TO 500
IF (K.EQ.3) GO TO 310

```

```

K=NEXT(L)
I=1/D(L)
GO TO 500

```

```

310 FP=BOOL(I)
I=FP

```


END

```

GO TO 500

400  I=0

      IF (H2(NP),NE,1) GO TO 500
      L=NEXT(NP)
      K=ITYP(L)

      IF (K,GT,3) GO TO 500

      I=D(L)
      IF (K,EG,3) GO TO 500
      IF (K,NE,0) GO TO 410

      FP=I
      I=BOOL(FP)
      GO TO 500

410  K=NEXT(L)
      FP=FLOAT(I)/FLOAT(D(K))
      I=BOOL(FP)
      GO TO 500

600  IF (H2(NP),NE,2) CALL FMLERR($20,'IDENT',1,3)

      K=NEXT(NP)
      KL=LASTXX($20,K,1,0)
      JL=NEXT(KL)
      J = LASTXX($20, JL, 1, 0)

      I=0
      CALL COMPAR($500,$500,$20,K,1,JL,1,KL,J)
      I=1
      GO TO 500

20   CALL ERRRET (NAME(II))
      I = 0

500  CALL IFREE0(NP)
      RETURN

ENTRY NCOUNT (III)
  I = COUNT(II)
  RETURN

END

```


481

FORMAL,FRECT
CT FREE STORAGE INVENTORY ROUTINE

COUNT FREE CHAIN

COUNT SYMBOL TABLE
SUBPROGRAM LEVEL

SYMBOL LEVEL

SUBSCRIPT LEVEL

EXPRESSION LEVEL

```

INTEGER FUNCTION COUNT (IJ)
INCLUDE CMMN
INCLUDE PWORD
N = 0

100 K = NXNXNX
110 N = N+1
    K = NEXT(K)
    IF (K .NE. 0) GO TO 110
    IFREE = N

    KS = NS
120 IF (KS .EQ. 0) GO TO 190
    N = N+1
    KY = LAST(KS)

130 IF (KY .EQ. 0) GO TO 180
    N = N+1
    IF (IBIT(KY) .EQ. 1) GO TO 170
    KE = LAST(KY)
    IR = 1
    IF (FLD(1,2,C(KY)) .NE. 3) GO TO 150

    IR = 2
    KB = KE
140 IF (KB .EQ. 0) GO TO 170
    N = N+1
    IF (FLD(0,3,C(KB)) .EQ. 7) GO TO 170
    IF (FLD(0,3,C(KB)) .EQ. 6) GO TO 160
    KE = LAST(KB)

150 IF (KE .EQ. 0) GO TO (170,160),IR
    N = N+1
    KE = NEXT(KE)
    GO TO 150
160 KB = NEXT(KB)
    GO TO 140
170 KY = NEXT(KY)
    GO TO 130
180 KS = NEXT(KS)
    GO TO 120
190 IJ = N
    COUNT = IFREE
    RETURN
END

```

3-254

3.4.10. FMLEQU ROUTINE

THIS ROUTINE IS CALLED WHEN FORMAL VARIABLE IS INCLUDED IN A SUBROUTINE HEADING AS AN ARGUMENT. THE CALLING STATEMENT IS GENERATED BY THE PREPROCESSOR. THE ROUTINE INITIALIZES THE SCANNER THROUGH PROGRAM ELEMENT SCANIN.

3497
 FORMAL FMLEQU
 ARGUMENT EQUIVALENCING ROUTINE

INITIALIZE

CALLING ARGUMENT

*... WALK BACK TO THE CALLING STATEMENT

*
 * READ IN ARGUMENT LIST

* SET TYPE FLAG

* FIND THIS VARIABLE IN SYMBOL TABLE

* :
 * :... IF SUBSCRIPTED, THEN LOOK INTO SUBSCRIPTED
 * : ENTRY

* :
 * :... NOT FOUND, STORE INTO SYMBOL TABLE

* :
 * :... SET INDIRECT BIT ON

* :
 * :... REMEMBER THE LOCATION OF THIS VARIABLE IN SYMBOL
 * : TABLE

* :
 * :... NON-SUBSCRIPTED VARIABLE

* :
 * :... SUBSCRIPTED VARIABLE

CALLED ARGUMENT

*... LINK TO THE CALLED ROUTINE

* :
 * IS THIS VARIABLE ALREADY IN SYMBOL TABLE ?

* :
 * :... YES, MUST NOT BE AN INDIRECT ALREADY

* :
 * : SET TYPE

* :
 * :... NO, STORE INTO SYMBOL TABLE

* :
 * :... SET THIS VARIABLE POINTING TO CALLING ARGUMENT

SUBROUTINE FMLEQU (ZZN,SYM)

INCLUDE CMMN
 INCLUDE PWORD

CALL FMLIN1 (\$999,0)
 CALL SCANIN

CALL FMLSCN (\$999)
 CALL GETARG

FLAG = 4
 IF (N2 .NE. 0) FLAG = 5

CALL FINDP1 (\$100)

IF (N2 .NE. 0) CALL FINDP2(\$100)
 GO TO 200

100 IY = 0
 IF (N2 .NE. 0) IY = 3
 CALL STSYEN (\$999)

200 FLD(4,1,C(NSY)) = 1

LOCN = NSY

IF (N2 .NE. 0) LOCN = NSU

CALL FMLINT (\$999,0)
 N1 = SYM
 N2 = 0

CALL FINDP1 (\$300)

IF (C(NSY) .GE. 0) CALL ERRERR(\$999,200)

FLD(0,3,C(NSY)) = FLAG
 GO TO 400

300 IY = FLAG
 CALL STSYEN (\$999)

```
*  
*... RETURN UNNEEDED STACK SPACE  
RETURN  
:  
:  
ERROR DETECTED  
:  
:  
END
```

```
400 LAST(NSY) = LOCN  
500 IF (NK .NE. 0) CALL FMLCHD ($501, NK)  
501 NK = 0  
RETURN  
999 CALL ERRRET ('FMLEQU')  
GO TO 500  
  
END  
END
```

35 FORMAL,SCANIN

ALTERNATE WAY OF INITIALIZING SCANNER FOR FMLEQU

\$ (1)	AXRS	
SCANIN*	S,H2	X11,WB+1
	L	A0,0,X11
	L,H2	A1,1,A0
	L	A0,*0,A1
	L,H2	A1,2,A1
	L,H2	A1,1,A1
	A,U	A1,0,A0
	AN,U	A1,1
	L,U	A1,*0,A1
	S	A1,LOC
	LMJ	X11,ISCANG
	+	A3,LOC+0200000
	+	SIZE
	+	K0
	+	\$-SCANIN,WB
	L	X11,WB+1
	J	1,X11
\$ (2)		
SIZE	+	20
LOC	+	0
K0	+	0
WB	'SCANIN'	0
	+	'FOR',K0
	END	

3-258

3.5. INTERACTIVE MAIN PROGRAM

THE INTERACTIVE MAIN PROGRAM IS PROGRAM ELEMENT FMLMNI. THE ROUTINE INITIALIZES THE OPTION SWITCHES BY CALLING THE OPTION STATEMENT ROUTINE. FOLLOWING THE INITIALIZATION, THERE IS A LOOP IN WHICH THE STATEMENTS ARE READ, RECOGNIZED AND EXECUTED BY CALLING THE PROPER STATEMENT HANDLING ROUTINE. STATEMENTS NOPRINT AND REPRINT ARE HANDLED DIRECTLY IN FMLMNI.

.....

```

*****
*                                     *
* FMLOPT                            *
*                                     *
*****

```

```

*****
*                                     *
>* FMLD00                          *
*                                     *
*****

```

```

*****
*                                     *
>* FMLERS *
*                                     *
*****

```

```

*****
*                                     *
>* FMLIO2                          *
*                                     *
*****

```

* *
* FML103 *
* *



.....> *****
* FML104 *
* *

.....> *****
* FML106 *
* *

.....> *****
* FML107 *
* *

.....> *****
* FMLOUT *
* *

.....> *****
* FMLDMP *
* *

FORMAL,FMLMNI
 MAIN PROGRAM FOR INTERACTIVE FORMAL
 THE PROGRAM RECOGNIZES THE TYPE OF
 STATEMENT BY THE FIRST 2 OR 6 CHARACTERS.
 IF THESE DO NOT MATCH BY A CORRESPONDING
 DATA THEN THE PROGRAM ACCEPTS IT AS AN
 ASSIGN STATEMENT.

STATEMENT IDENTIFIER WORDS

I/O STATEMENT (1-4)

FILE STATEMENT (5-7)

PROGRAM AID STATEMENTS (8-10)

OPTION, ERASE ST. (11-12)

OPTIONS FOR REPRINTING STATEMENT

INITIALIZE SYSTEM THROUGH OPTION STATEMENT

READ STATEMENT

CHECK IF THE FIRST 2 CHAR. ARE CA
 FOR A COMMENT STATEMENT

CHECK FOR DO OR END STATEMENTS

IF DO LOOP SWITCH IS SET, STACK INPUT & CONTINUE
 READING

CHECK IF THE FIRST 2 CHAR. ARE PA
 FOR A PRINT STATEMENT

NO, DETERMINE STATEMENT BY MATCHING FIRST WORD

INTEGER COUNT.
 PARAMETER IDIM = 14
 LOGICAL SW1/.FALSE./,SW2/.FALSE./
 DIMENSION IN(14),INN(14),ITAB(IDIM)
 EQUIVALENCE (IN(2),INN(1))
 DATA INN(14)/';'/

DATA ITAB/

+ 'READ ', 'PRINT ', 'PUNCH ', 'WRITE ',
 + 'ROLOUT', 'SAVE ', 'RESET ',
 + 'DUMP ', 'NCOUNT', 'COMMENT',
 + 'OPTION', 'ERASE ',
 + 'REPRIN', 'NOPRIN' /
 LOGICAL RPRINT
 DATA RPRINT/.TRUE./

CALL FMLOPT('INT,MFCT,PRODEX,EXPAND(1),BASE(0),NOFORT;',0)

10 IF(.NOT.SW2)READ 20, END=250, IN
 IF(SW2)CALL USTACK(IN,IDIM,DUMMY)
 20 FORMAT (13A6,A2)

IF (FLD(0,12,IN(1)) .EQ. 1005K) GO TO 10
 J=2

IF(FLD(0,18,IN(1)).EQ.'@@@@@')GO TO 240
 IF(FLD(0,24,IN(1)).EQ.'@END')GO TO 245

IF(SW1)CALL DSTACK(IN,IDIM,DUMMY,5245)
 IF(SW1)GO TO 10

IF(FLD(0,12,IN(1)).EQ.'@@@@P')GO TO 100

J=0

BRANCH BY STATEMENT TYPE

... ASSIGN STATEMENT

... I/O STATEMENTS
CHECK FOR FORMAT ARGUMENTS

BRANCH BY I/O TYPE

... READ

... READ FROM TTY

--- READ FROM UNIT I1

... PRINT

... UNFORMATED PRINT

--- FORMATED PRINT

... PUNCH

--- WRITE

... FORMATED WRITE

--- UNFORMATED WRITE

... ROLOUT

DO 30 I=1,IDIM
30 IF (IN(1) .EQ. ITAB(I)) J = I

GO TO (40,60,60,60,60,150,160,170,180,190,10,
+ 200,210,220,230), J+1

40 IF (RPRINT) PRINT 50, IN
50 FORMAT (1X,14A6)
CALL FMLASG(IN,0)
GO TO 10

60 CALL FMLUIT(INN,N,I1,I2,I3,I4)

GO TO (70,90,120,130), J

70 IF (RPRINT) PRINT 50, IN
IF (N .NE. 0) GO TO 80

CALL FMLIO1(INN,0)
GO TO 10

80 CALL FMLIO5(I1,INN,0)
GO TO 10

90 IF (N .NE. 0) GO TO 110

CALL FMLIO2(INN,0)
GO TO 10

100 FLD(0,6,IN(1))=05K
CALL FMLIO2(IN,0)
GO TO 10

110 CALL FMLIO3(I1,I2,INN,0)
GO TO 10

120 CALL FMLIO4(INN,0)
GO TO 10

130 IF (N .EQ. 1) GO TO 140

CALL FMLIO7(I1,I2,I3,INN,0)
GO TO 10

140 CALL FMLIO6(I1,INN,0)
GO TO 10

150 IF (RPRINT) PRINT 50, IN
CALL FMLOUT(INN,0)
GO TO 10

```

... SAVE
... RESET
... DUMP
... NCOUNT
... OPTION
... ERASE
... SET REPRINT SWITCH
--- PROHIBIT REPRINT
DO STATEMENT
END STATEMENT
END OF FILE

```

```

160 IF (RPRINT) PRINT 50, IN
    CALL FMLSAV(INN)
    GO TO 10

170 IF (RPRINT) PRINT 50, IN
    CALL FMLRES(INN)
    GO TO 10

180 IF (RPRINT) PRINT 50, IN
    IF (INN(1) .EQ. 'EXPRES') INN(1) = 20
    IF (INN(1) .EQ. 'SYMBOL') INN(1) = 21
    CALL FMLDMP(INN)
    GO TO 10

190 I = COUNT(J)
    PRINT 191, I, J
191 FORMAT (' FREE / TOTAL = ', I5, ' / ', I5)
    GO TO 10

200 IF (RPRINT) PRINT 50, IN
    CALL FMLOPT(INN,0)
    GO TO 10

210 IF (RPRINT) PRINT 50, IN
    CALL FMLERS(INN,0)
    GO TO 10

220 RPRINT=.TRUE.
    GO TO 10

230 RPRINT=.FALSE.
    GO TO 10

240 CALL FMLDO(IN,IDIM,SW1,SW2)
    GO TO 10

245 CALL FMLEND(IN,IDIM,SW1,SW2)
    GO TO 10

250 STOP
    END

```

4. INSTALLATION

THE FORMAL SYSTEM IS DISTRIBUTED AS A TEN-FILE TAPE, SUITABLE FOR MANIPULATION WITH STANDARD FURPUR COMMANDS. THIS SECTION DESCRIBES THE FORMAT OF THE TAPE AND THE INSTALLATION OF THE FORMAL SYSTEM FROM THE TAPE.

4.1. TAPE FORMAT

THE INPUT TAPE HAS AT LEAST NINE AND GENERALLY TEN FILES. THE FIRST FIVE CONTAIN INFORMATION PERTAINING TO A BASE LEVEL OF THE FORMAL SYSTEM, WHILE THE NEXT FIVE CONTAIN THE CURRENT LEVEL. THE TEN FILES (WITH THEIR INTERNAL NAMES) ARE

- (1) T UTILITY FILE
@ADD ELEMENTS AND ABSOLUTE PROGRAMS
- (2) SGS SGS STATEMENTS TO DIRECT @SSG FOR
COMPILATIONS, ASSEMBLIES, COLLECTIONS, ETC.
- (3) RLIB RALPH DEPENDENT ROUTINES
- (4) SI BASE-LEVEL SYMBOLICS
- (5) DOC BASE-LEVEL DOCUMENTATION
- (6) PCF THIS FILE CONTAINS A LIST OF ALL THE
ELEMENT NAMES, FOLLOWED BY THEIR CORRECTIONS,
IF ANY.
- (7) LIB1 FORMAL LIBRARY (RELOCATABLES AND ABSOLUTES
-- WITHOUT RLIB).
- (8) REL OTHER RELOCATABLES
- (9) DOC1 LATEST VERSION OF DOCUMENTATION (INCLUDING ALL
UPDATES).
- (10) PRINTFILE OF THE RUN GENERATING THIS TAPE IF
'PRINTFILE' STATEMENT WAS USED

FILES 1,3,4,7, AND 8 ARE IN 'COPOUT' FORMAT, WHILE FILES 2,5,6,9 AND 10 ARE IN 'COPY,F' FORMAT.

4-2

4.2, MANIPULATION OF THE TAPE

4.2.1. INSTALLATION

A. IF YOU DO NOT HAVE THE 'RALPH' COMPILER IN SYSS*LIBS.:

```
QASG,T    INPUT,8C,<YOUR TAPE>
QASG,CPR  LIB//<WKEY>,F
QMOVE     INPUT,,2
QCOPIN    INPUT,,LIB,
QMOVE     INPUT,,3
QCOPIN    INPUT,,LIB,
QFREE     INPUT.
QPREP     LIB.
QFREE     LIB.
QPRT,T    LIB.
```

B. IF YOU HAVE 'RALPH' IN SYSS*LIBS.

```
QASG,T    INPUT,8C,<YOUR TAPE>
QASG,CPR  LIB//<WKEY>,F
QMOVE     INPUT,,6
QCOPIN    INPUT,,LIB,
QFREE     INPUT.
QPREP     LIB.
QFREE     LIB.
QPRT,T    LIB.
```

4.2.2. PRINT THE USER'S MANUAL

IN ORDER TO PRINT N COPIES OF THE USER'S MANUAL:

```
QASG,T    INPUT,8C,<YOUR TAPE>
QASG,T    T
QCOPIN    INPUT,,T.
QASG,T    DOCIN,F
QMOVE     INPUT,,7
QCOPY,F   INPUT,,DOCIN.
QFREE     INPUT
QASG,CPR  DOCPRINT,F
QBRKPT    PRINTS/DOCPRINT
QT.DOC,ADFILS  A,B
QADD      DOCIN.
QBRKPT    PRINTS.
QFREE     DOCPRINT
QSYM,U    DOCPRINT       . REPEAT N TIMES FOR N COPIES
```

4.2.3. LISTING THE PRINT FILE

TO LIST THE PRINTFILE OF THE LAST UPDATE RUN:

```
@ASG,T INPUT,8C,<YOUR TAPE>
@MOVE INPUT,,9
@ASG,CP FORMALPRINT,F///256
@COPY,F INPUT,FORMALPRINT
@FREE FORMALPRINT
@FREE INPUT
@SYM,U FORMALPRINT . REPEAT N TIMES FOR N COPIES
```


5. UPDATING PROCEDURE

5.1. UPDATING PHILOSOPHY -- SSG

THE UPDATING PROCEDURE IS PROVIDED SO THAT THE USER MAY INCORPORATE CHANGES IN THE FORMAL OBJECT ROUTINES AND/OR THE FORMAL USER'S MANUAL, CREATE COPIES OF THE FORMAL SYSTEM, AND OBTAIN LISTINGS OF THE FORMAL OBJECT ROUTINES AND DOCUMENTATION. THE USER NEED HAVE NO KNOWLEDGE OF THE DETAILS OF THE SYSTEM OR THE METHOD OF UPDATING. HOWEVER, IF THE USER WISHES TO INCORPORATE CHANGES INTO THE SYSTEM OTHER THAN THOSE SUPPLIED BY THE UNIVERSITY OF MARYLAND, THE UPDATING PROCEDURE NEED NOT BE MODIFIED.

THE BASIC PHILOSOPHY OF THE PROCEDURE IS TO MERGE THE PRESENT UPDATES WITH A FILE OF PERMANENT UPDATES (THE PAST UPDATES). THIS FILE IS THEN USED TO UPDATE THE BASE-LEVEL SYMBOLICS TO PRODUCE THE NEW LIBRARY. THE PHILOSOPHY IS IMPLEMENTED BY PRODUCING A NEW RUN STREAM WITH EACH UPDATE, IN ORDER TO FREE THE USER FROM THIS TASK THE SYMBOLIC STREAM GENERATOR (SSG - SEE UNIVAC 1108 SYSTEMS MEMORANDUM 5.12) IS USED. IN ORDER TO PERFORM UPDATES ON THE FORMAL SYSTEM, HOWEVER, THE USER NEED HAVE NO KNOWLEDGE OF SSG OR THE METHOD IN WHICH SSG IS EMPLOYED.

5.2. STANDARD UPDATING PROCEDURE

THE STANDARD PROCEDURE FOR UPDATING THE FORMAL SYSTEM IS INITIATED WHEN THE USER RECEIVES THE UPDATE PACKET FROM THE UNIVERSITY OF MARYLAND. THIS PACKET CONSISTS OF THREE SUBPACKETS:

- (1) ANY CHANGES THAT ARE TO BE MADE TO THE PERMANENT STREAM GENERATION STATEMENT (SGS) FILE (IF ANY),
- (2) ANY CHANGES TO BE MADE TO THE FORMAL LIBRARY,
- (3) NECESSARY 'STREAM GENERATION STATEMENTS' (E.G., LEVEL NUMBER, ELEMENT DELETION, INSERTION, AND NAME CHANGE), AND SKELETON CORRECTIONS (IF ANY),

THE USER IS RESPONSIBLE FOR

- (1) PROVIDING THE INPUT AND OUTPUT TAPES,
- (2) SPECIFYING THE TYPE OF UPDATE,
- (3) SPECIFYING THE TYPE OF PRINT-OUT (INCLUDING A PRINT-FILE (IF ANY)),
- (4) ASSEMBLING THE SUPPLIED UPDATE PACKET INTO THE STANDARD UPDATE DECK,

THE INPUT TAPE IS THE MOST RECENTLY GENERATED UPDATE TAPE (IN THE CASE OF THE INITIAL UPDATE IT IS THE MASTER TAPE SUPPLIED BY THE UNIVERSITY OF MARYLAND). THE OUTPUT TAPE WILL BE THE UPDATED SYSTEM AND SHOULD NOT BE THE SAME AS THE INPUT TAPE. THE USER DESCRIBES THE TYPE OF UPDATE AND PRINT-FILES THROUGH 'STREAM GENERATION STATEMENTS' (HEREAFTER CALLED 'SGS'). THE SGS STATEMENTS ARE DESCRIBED IN SECTION 6.2.2.

WHEN A USER IS SUPPLIED THE MASTER-LEVEL TAPE, AN UPDATE DECK IS ALSO PROVIDED, CONSISTING OF THE FOLLOWING 20 CARDS:

```

1  @RUN      <RUNID>,<ACCOUNT>,<FORMAL>,<TIME>,<PAGES>
2  ..      THIS CARD IS REPLACED BY THE @ASG AND @SETC IMAGES
3  @ASG,T   T
4  @COPIN   INPUT-TAPE,,T.
5  @REWIND  INPUT-TAPE,
6  ..      @ADD T,RALPH1 IF YOU DO NOT HAVE RALPH IN SYSS*LIBS.
7  @MSG,N   REPLACE FOLLOWING CARD TO DO YOUR OWN FILE-SETUP
8  @ADD     T,SETUP
9  @SETC,I
10 @DATA,L  SGS,,SGS1.
11 ..      INSERT UPDATES FOR PERMANENT SGS HERE
12 @END
13 @DATA,LI TCF,
14 ..      NEW CORRECTIONS TO FORMAL LIBRARY
15 @END
16 @ADD     T,SSG1
17 ..      SGS (AND POSSIBLY SKEL-CORRECTIONS) REPLACE THIS CARD
18 @EOF
19 @EOF
20 ..      USER CLEANUP TASKS REPLACE THIS CARD

```

THE UPDATE DECK IS ASSEMBLED IN THE FOLLOWING STEPS:

- (1) INSERT THE PROPER RUNID, ACCOUNT NUMBER, TIME AND PAGE ESTIMATES IN THE APPROPRIATE FIELDS IN CARD 1. THE TIME AND PAGE ESTIMATES SHOULD BE GENEROUS (E.G., FOR A COMPLETE UPDATE, 6 MINUTES AND 800 PAGES ARE A MINIMUM).

- (2) CARD 2 IS REPLACED BY AN ASSIGNMENT STATEMENT

```
@ASG,T INPUT-TAPE,8C,<REEL>
```

TO ASSIGN THE INPUT TAPE <REEL>. IN ADDITION, IF A COMPLETE UPDATE IS DESIRED THE CONTROL CARD

```
@SETC 1/54
```

IS ALSO INSERTED;

- (3) CARD 6 IS REPLACED BY

```
@ADD T.RALPH1
```

IF THE RALPH COMPILER DOES NOT EXIST IN THE USER'S SYS\$*LIB\$; OTHERWISE IT SHOULD BE DELETED;

- (4) CARD 8 SHOULD BE REPLACED BY THE APPROPRIATE STATEMENTS IF THE USER WISHES TO GET HIS INPUT FILES FROM ANY SOURCE OTHER THAN A STANDARD INPUT-TAPE;
- (5) CARD 11 IS REPLACED BY SUBPACKET 1, SUPPLIED BY THE UNIVERSITY OF MARYLAND (IF ANY);
- (6) CARD 14 IS REPLACED BY THE NEW UPDATES, SUBPACKET 2, SUPPLIED BY THE UNIVERSITY OF MARYLAND.
- (7) CARD 17 IS REPLACED BY THE SGS STATEMENTS PROVIDED BY THE USER AND THOSE PROVIDED BY THE UNIVERSITY OF MARYLAND IN SUBPACKET 3 (WHICH MAY ALSO INCLUDE SKELETON CORRECTIONS). USER PROVIDED SGS STATEMENTS MUST PRECEDE THE FIRST @EOF CARD IN THIS PACKET;
- (8) CARD 20 IS REPLACED BY ANY USER TASKS RELATED TO THE UPDATE.

5.2.1. STREAM GENERATION STATEMENTS

THE STREAM GENERATION STATEMENTS (SGS) ARE DESCRIBED BELOW. THE STATEMENTS EACH CONSIST OF ONE OR MORE FIELDS THAT ARE SEPARATED BY BLANKS. SOME FIELDS ARE DIVIDED INTO SUBFIELDS THAT ARE SEPARATED BY COMMAS. THE FIRST FIELD DEFINES THE OPERATION AND MAY START IN ANY COLUMN. IF AN SGS STATEMENT EXTENDS BEYOND ONE CARD LENGTH, A SEMI-COLON MUST BE USED TO TERMINATE THE FIRST CARD BEFORE CONTINUING TO THE SECOND.

IN GENERAL, THE USER IS SUPPLIED 'INSERT', 'CHANGE',

'COMPILER', 'DELETE', AND 'LEVEL' STATEMENTS IN THE UPDATE PACKET. IN ADDITION, THE USER WILL SUPPLY 'PRINTFILE', 'OUTTAPE', 'PERFORM', AND 'NEWLIB' STATEMENTS, ALTHOUGH HE MAY FIND USES FOR ALL OF THE STATEMENTS. THERE MAY BE ANY NUMBER OF INSERT, DELETE, AND CHANGE STATEMENTS.

5.2.1.1. INSERT

THE 'INSERT' STATEMENT IS OF THE FORM

INSERT <ELT1>,<ELT2>,...,<ELTN>

AND SIGNIFIES THAT ELEMENTS <ELT1>,<ELT2>,...,<ELTN> ARE NEW AND ARE TO BE INSERTED INTO THE SOURCE OUTPUT AND RELOCATABLE BINARY FILES.

5.2.1.2. CHANGE

THE 'CHANGE' STATEMENT IS OF THE FORM

CHANGE <ELTA> TO <ELTB>

AND SIGNIFIES THAT THE NAME OF ELEMENT <ELTA> IN THE SOURCE INPUT IS TO BE CHANGED TO <ELTB> IN THE SOURCE OUTPUT AND RELOCATABLE BINARY FILES.

5.2.1.3. DELETE

THE 'DELETE' STATEMENT IS OF THE FORM

DELETE <ELT1>,<ELT2>,...,<ELTN>

AND SIGNIFIES THAT ELEMENTS <ELT1>,<ELT2>,...,<ELTN> ARE TO BE DELETED FROM THE SOURCE OUTPUT AND RELOCATABLE BINARY FILES.

5.2.1.4. LEVEL

THE 'LEVEL' STATEMENT IS OF THE FORM

LEVEL <N>

WHERE <N> IS A STRING OF UP TO FIVE CHARACTERS AND IS USED TO SPECIFY THE LEVEL OF THE FORMAL SYSTEM AT SIGN-ON AND ON LISTING HEADINGS.

5.2.1.5. OUTTAPE

THE 'OUTTAPE' STATEMENT IS OF THE FORM

OUTTAPE <FILENAME> <REEL> <N>

WHERE <FILENAME> IS THE INTERNAL FILENAME OF THE TAPE (UP TO 12 CHARACTERS),

<REEL> IS THE REEL NUMBER OF THE TAPE TO BE USED FOR THE OUTPUT TAPE,

<N> IS THE NUMBER OF FILES TO BE SPACED OVER BEFORE WRITING (ZERO IF ABSENT).

THIS STATEMENT SPECIFIES THE OUTPUT TAPE FOR THE UPDATE. AT THOSE INSTALLATIONS THAT USE A TRAILING R ON THE REEL NUMBER TO SIGNIFY THE 'RING-IN' CONDITION, AN 'R' SHOULD BE APPENDED TO THE REEL NUMBER. THE ABSENCE OF AN 'OUTTAPE' STATEMENT SIGNIFIES THAT NO OUTPUT TAPE IS TO BE WRITTEN.

5.2.1.6. PRINTFILE

THE 'PRINTFILE' STATEMENT IS OF THE FORM

PRINTFILE <FILENAME> <N>

WHERE <FILENAME> IS THE NAME OF THE FILE UPON WHICH THE PRINTFILE WILL BE PLACED,

<N> IS AN INTEGER SPECIFYING THE NUMBER OF TIMES THAT THE FILE IS TO BE PRINTED (IF <N>=0 OR IS LEFT OUT, 1 COPY WILL BE PRINTED).

THE 'PRINTFILE' STATEMENT CAUSES A FILE <FILENAME> TO BE CATALOGUED (READ-ONLY, PUBLIC). IN ADDITION, IF AN OUTPUT TAPE IS SPECIFIED, THEN A COPY OF THIS FILE IS WRITTEN TO THE OUTPUT TAPE AS FILE 10. THE FILE CONTAINS THE LISTING GENERATED BY THE UPDATE (I.E., COMPILATIONS, ASSEMBLIES, COLLECTIONS)

5.2.1.7. NO

THE 'NO' STATEMENT HAS ANY OF THE FOLLOWING FORMS:

NO MAPS

SIGNIFIES THAT NO COLLECTIONS ARE TO BE PERFORMED.

NO COMPIL

SIGNIFIES THAT NO COMPILATIONS OR ASSEMBLIES ARE TO BE PERFORMED; (SPECIAL FORMAT LISTINGS WILL BE MADE WITHOUT APPLYING THE ACCUMULATED CORRECTIONS)

NO LISTIN

MEANS THAT NO SPECIAL FORMAT LISTINGS SHOULD BE MADE.

5-6

5.2.1.8. COMPILER

THE 'COMPILER' STATEMENT IS OF THE FORM
COMPILER <NAME>, <OPTIONS>

IT DIRECTS SSG TO USE THE COMPILER <NAME> FOR ALL FORTRAN COMPILATIONS, AND TO USE <OPTIONS> (IF ANY ARE SPECIFIED), TOGETHER WITH THE STANDARD OPTIONS. IF NO 'COMPILER' STATEMENT IS GIVEN,

COMPILER RALPH

WILL BE ASSUMED

5.2.1.9. CDIM, PLNSIZ, AND DOSIZE

ALL THESE HAVE THE FORM

<LABEL> <NUMBER>

AND ARE USED TO SPECIFY CONFIGURATIONS DIFFERENT FROM THE STANDARD DISTRIBUTION.

CDIM IS THE SIZE OF THE CHAINED STORAGE AREA, (STANDARD 2048) INCREASING CDIM WILL ALLOW LARGER EXPRESSIONS AND/OR MORE EXPRESSIONS IN CORE SIMULTANEOUSLY. THIS MAY SPEED UP THE EXECUTION OF VERY LARGE FORMAL PROGRAMS.

PLNSIZ (STANDARD 100) IS THE SIZE OF THE BIT-MAP FOR THE ROLLOUT-FILE. EACH WORD CORRESPONDS TO 35 BLOCKS OF 2 SECTORS EACH ON DRUM. PLNSIZ SHOULD BE INCREASED IF THE DRUM-AREA IS TOO SMALL TO HOLD ALL DATA IN A PARTICULAR APPLICATION.

DOSIZE (STANDARD 50) IS THE MAXIMUM NUMBER OF STATEMENTS FOR A DO-LOOP IN THE INTERACTIVE FORMAL SYSTEM.

IF ANY OF THESE STATEMENTS IS USED TOGETHER WITH A 'PERFORM COMPLETE UPDATE' SGS, THE GENERATED LEVEL WILL INCLUDE THE NEW VALUES AS STANDARD.

5.2.1.10. ASSUME

THE 'ASSUME' STATEMENTS ARE USED TO CHANGE THE FORMAL OPTIONS ASSUMED BY THE SYSTEM INITIALLY. THEY ARE OF THE FORM:

ASSUME <OPTION> <VALUE>

WHERE <OPTION> MAY BE PRODEX, INTGSW, MATHSW, EXPDSW, POWER, BASE, OR FORTRN, AND <VALUE> MAY BE ANY NUMBER, TRUE, OR FALSE. THESE STATEMENTS MAY BE IN ANY ORDER. ONLY THE FIRST OCCURRENCE OF EACH OPTION IS SIGNIFICANT.

5.2.1.11. LIB

THIS SGS HAS THE FORM

LIB <FILENAME>

AND IS USED TO SPECIFY ANY PROGRAM FILES WHICH SHOULD BE INCLUDED IN THE COLLECTIONS THROUGH A LIB STATEMENT.

5.2.1.12. INCLUDE

THE 'INCLUDE' STATEMENT IS OF THE FORM

INCLUDE RLIB

AND IS USED IF THE USER DOES NOT HAVE THE RALPH COMPILER ON THE SYSTEM. THE STATEMENT CAUSES SEVERAL RALPH DEPENDENT RELOCATABLES TO BE PUT INTO THE FORMAL LIBRARY.

5.2.1.13. PERFORM

THE 'PERFORM' STATEMENT IS EITHER OF THE FORM

PERFORM COMPLETE UPDATE

OR

PERFORM COMPLETE LISTING.

IN EITHER CASE, ALL ELEMENTS ARE EDITED, AND COMPILED OR ASSEMBLED. HOWEVER, THE 'PERFORM COMPLETE UPDATE' STATEMENT REPLACES ALL SYMBOLICS WITH THEIR UPDATED VERSIONS ON THE OUTPUT TAPE (AND EMPTIES THE CORRECTION FILE) WHILE THE 'PERFORM COMPLETE LISTING' ONLY REPLACES THE RELOCATABLES AND COLLECTIONS. IF THE 'PERFORM' STATEMENT IS NOT USED, THEN ONLY THOSE ELEMENTS ACTUALLY CHANGED IN THIS RUN ARE LISTED AND REPLACED IN THE RELOCATABLE OUTPUT FILE.

5.2.1.14. NEWLIB

THE 'NEWLIB' STATEMENT IS OF THE FORM

NEWLIB <FILE>

AND CREATES A PUBLIC, READ-ONLY, CATALOGUED FILE WITH FILE DESCRIPTION <FILE>. THIS FILE WILL CONTAIN ALL THE RELOCATABLES AND ABSOLUTES, AND IF 'INCLUDE RLIB' IS SPECIFIED, IT WILL CONTAIN RLIB. THE <FILE> FIELD CAN INCLUDE THE COMPLETE FILE DESCRIPTION (QUALIFIER*FILENAME(CYCLE)/KEY/KEY).

5.2.2. USER IMPLEMENTED UPDATES

THE SSG PROCESSOR PROVIDES THE CAPABILITY OF MERGING CORRECTIONS. IN ORDER TO IMPLEMENT USER UPDATES, THE USER SHOULD

- (1) BECOME FAMILIAR WITH SSG,
- (2) UPDATE THE FORMAL SYSTEM TO ITS CURRENT LEVEL (AS GIVEN BY UPDATES FROM THE UNIVERSITY OF MARYLAND), AND
- (3) APPLY HIS OWN CORRECTIONS AS TEMPORARY CORRECTIONS TO THAT LEVEL.

WHEN A COMPLETE UPDATE IS MADE, SSG MAY BE USED TO CREATE A RE-NUMBERED TEMPORARY CORRECTION FILE.

5.3. EXAMPLE

THE FOLLOWING EXAMPLE DEMONSTRATES AN UPDATE FROM INPUT TAPE 1000 TO TAPE 1001, WHICH WILL PRODUCE 10 COPIES OF THE COMPLETE LISTING. THE NEW VERSION IS TITLED ABCDE AND IS PLACED ON CATALOGUED FILE 'FORMAL*LIB.' WITH THE WRITE-KEY '1234'.

```
(1) @RUN      UPDATE,000,FORMAL,10,1000
(2) @ASG,T    INPUT-TAPE,8C,1000
    @ASG,T    T
    @COPIN    INPUT-TAPE,,T.
    @REWIND   INPUT-TAPE.
    @MSG,N    REPLACE FOLLOWING CARD TO DO YOUR OWN FILE-SETUP
    @ADD      T.SETUP
    @SETC,I
    @DATA,L   SGS.,SGS1.

(3)      :   SGS STATEMENTS PROVIDED BY UNIVERSITY OF MARYLAND
    :
    @END
    @DATA,LI TCF.

(4)      :   CORRECTIONS PROVIDED BY UNIVERSITY OF MARYLAND
    :
    @END
    @ADD      T.SSG1
(5)      LEVEL  ABCDE
(6)      PRINTFILE  PRINT  10
(7)      OUTTAPE  OUT  1001R
(8)      PERFORM COMPLETE LISTING
(9)      NEWLIB   LIB(+1)//1234
    @EOF
```


EOF
FIN

IN THIS EXAMPLE THE USER WOULD ONLY HAVE SUPPLIED THE CARDS NUMBERED 1,2,6,7,8,9 WHILE THE UNIVERSITY OF MARYLAND WOULD HAVE SUPPLIED THE PACKETS OF CARDS 3, 4 AND 5 (IN ADDITION TO THE ORIGINAL DECK).

6. CROSS-REFERENCE TABLES

IN ORDER TO FACILITATE UNDERSTANDING AND MAINTENANCE OF THE FORMAL SYSTEM PACKAGE, THE FOLLOWING FIVE TABLES SUMMARIZE THE ESSENTIAL INFORMATION ABOUT RELOCATABLE ELEMENT SIZES, PROGRAM ENTRY POINTS, PROGRAM REFERENCES, COMMON BLOCKS AND CROSS-REFERENCES. THE FIVE TABLES ARE:

(1) IBANK - DBANK SIZES.

AN ALPHABETIC LIST OF ALL RELOCATABLE ELEMENT WITH THEIR CORRESPONDING I-BANK AND D-BANK REQUIREMENTS (IN DECIMAL).

(2) ENTRY POINTS AND EXTERNAL REFERENCES.

AN ALPHABETIC LIST OF ALL RELOCATABLE ELEMENTS WITH THE ENTRY POINTS AND EXTERNAL REFERENCES (INCLUDING SYSTEM LIBRARY REFERENCES).

(3) ENTRY POINT REFERENCES.

AN ALPHABETIC LIST OF ALL ENTRY POINTS IN THE FORMAL PACKAGE WITH A LIST OF THE ELEMENTS THAT REFERENCE EACH.

(4) COMMON BLOCK SIZE.

AN ALPHABETIC LIST OF DEFINED COMMON BLOCKS AND THEIR SIZES.

(5) COMMON BLOCK REFERENCES.

A LIST OF ALL COMMON BLOCKS WITH THE ELEMENTS THAT REFERENCE EACH.

AS FORMAL IS SUCH A LARGE PROGRAM IT IS USEFUL TO SUMMARIZE SOME OF THE INFORMATION CONTAINED IN THE TABLES THAT FOLLOW.

(1) THERE ARE 88 ELEMENTS, 228 ENTRY POINTS, AND 5 COMMON BLOCKS.

(2) THE TOTAL I-BANK AND D-BANK REQUIREMENTS ARE:
PREPROCESSOR:

I-BANK = 4359 WORDS

D-BANK = 7017 WORDS

OBJECT TIME ROUTINES: I-BANK = 17903 WORDS

D-BANK = 2123 WORDS

INTERACTIVE FORMAL: I-BANK = 24396 WORDS

D-BANK = 10029 WORDS

6-2

THE VALUES FOR THE PREPROCESSOR AND FOR THE INTERACTIVE
FORMAL CONTAIN THE SYSTEM LIBRARY.

(3) THE TOTAL COMMON STORAGE = 4983 WORDS

IBANK - DBANK LIMITS

```

*****
*      ELEMENT/VERSION      *      IBANK      *      DBANK      *
*****
*      BCDALL                *      49        *      19        *
*      BCD-BIN               *      67        *      7         *
*      CHAR                  *      737       *      2832      *
*      COMPINC               *      263       *      16        *
*      COMP1                 *      1380      *      59        *
*      COMP2                 *      218       *      11        *
*      CONSOP                *      1201      *      27        *
*      DIFIN                 *      342       *      21        *
*      DIF1                  *      257       *      17        *
*      DPND                  *      141       *      12        *
*      DUMPO                 *      12        *      8         *
*      ERRKET                *      32        *      11        *
*      ERRWBK                *      32        *      23        *
*      EXPRES                *      1037      *      48        *
*      E$DRUM                *      211       *      16        *
*      FMLASG                *      49        *      6         *
*      FMLCHA                *      511       *      20        *
*      FMLCHAIN              *      187       *      8         *
*      FMLCHAIN1             *      224       *      8         *
*      FMLCHB                *      455       *      27        *
*      FMLUMP                *      296       *      32        *
*      FMLUMQ                *      268       *      23        *
*      FMLU00                *      495       *      92        *
*      FMLU01                *      155       *      17        *
*      FMLEDI                *      22        *      11        *
*      FMLENT                *      2         *      0         *
*      FMLEQU                *      127       *      8         *
*      FMLERR                *      90        *      251       *
*      FMLERS                *      57        *      7         *
*      FMLFOR                *      317       *      23        *
*      FMLINT                *      107       *      11        *
*      FML10A                *      36        *      7         *
*      FML10B                *      37        *      7         *
*      FML10C                *      36        *      7         *
*      FML10D                *      36        *      7         *
*      FML10E                *      37        *      7         *
*      FML10F                *      54        *      7         *
*      FML10G                *      55        *      7         *
*      FML101                *      44        *      6         *
*      FML102                *      50        *      8         *
*      FML103                *      75        *      7         *
*      FML104                *      58        *      10        *
*      FML105                *      48        *      6         *
*****

```

IBANK - DBANK LIMITS

(CONT.)

```

*****
*      ELEMENT/VERSION      *      IBANK      *      DBANK      *
*****
*  FMLIO7      *      79      *      7      *
*  FMLMNI      *     310     *     59     *
*  FMLOPT      *     210     *     31     *
*  FMLOUT      *     63      *      7     *
*  FMLPCK      *     263     *     174    *
*  FMLPKS      *     114     *     14     *
*  FMLRD       *     80      *      6     *
*  FMLSAV      *     244     *     289    *
*  FMLSCN      *     134     *     13     *
*  FMLSV1      *     168     *     32     *
*  FMLUIT      *     182     *     28     *
*  FMLVER      *      0      *      0     *
*  FMLWR       *     764     *     47     *
*  FNCT        *     61      *     60     *
*  FORMAL      *     273     *     366    *
*  FRECT       *     143     *     12     *
*  FUNTST      *     61      *      5     *
*  GET         *     105     *     10     *
*  GETARG      *     56      *      5     *
*  GIT         *     109     *     17     *
*  GSCAN       *     433     *     54     *
*  INCL        *     494     *     26     *
*  INTEGER     *     34      *      0     *
*  IOINTF      *     166     *      3     *
*  LIST        *     332     *     54     *
*  LOAD        *     38      *      7     *
*  NONARITH    *     118     *     110    *
*  PACK        *     37      *     19     *
*  SBEXP       *     26      *     33     *
*  SCANIN      *     17      *      6     *
*  SEC2IO      *     247     *     150    *
*  SKIP        *     60      *      5     *
*  SPL-LINK    *     60      *      0     *
*  STACK       *     126     *     218    *
*  STDET       *     639     *     214    *
*  STOUT0      *     492     *     29     *
*  STOUT1      *     2162    *     145    *
*  STOUT2      *     941     *     29     *
*  STOUT3      *     223     *     61     *
*  STOUT4      *     739     *     42     *
*****

```

IBANK - DBANK LIMITS

(CONT.)

```
*****
*      ELEMENT/VERSION      *      IBANK      *      DBANK      *
*****
*      SYMBFIND              *      103      *      3          *
*      SYMBOL                *      620      *      16         *
*      SYMBSTOR              *      179      *      6          *
*      TYPE                  *      119      *      10         *
*      XXX                   *      5        *      0          *
*****
```

ENTRY POINTS AND EXTERNAL REFERENCES

```

*****
* ELEMENT/VERSION * ENTRY POINTS * EXT. REFRNCs *
*****
* BCDALL * FLDBIN * *
* * FLDECI * *
* * FLDOCT * *
* * FWORD * *
*-----*
* BCD-RI * BCDBIN * EFR$ *
* * BCDNPT * L$PWF8 *
* * BCDPNT * *
* * BCDSTR * *
* * DFETCH * *
*-----*
* CHAR * BACKSP * CRDIN *
* * CHAR * CRDOUT *
* * CHARIN * ERR *
* * FTROUT * HERFNS *
* * FTRSTK * MOVE *
* * FTRUNS * PACK *
* * ICRD * STRCON *
* * IDEC * UNPACK *
* * NEW * *
* * RESET * *
* * RESTOR * *
* * SAVE * *
*-----*
* COMPINC * CMPRMV * COMPAR *
* * * FMLCHH *
* * * HERFNS *
* * * LASTXX *
*-----*
* COMP1 * COMPSF * ALOG *
* * * COMPAR *
* * * CONSOP *
* * * ERRERR *
* * * FMLERR *
* * * HCGTES *
* * * HERFNS *
* * * IADDNN *
* * * IFREEN *
* * * IFREE1 *
* * * IGETF1 *
* * * ILINKN *
* * * ILINK1 *
* * * LASTXX *
*****

```

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  COMP1                    *                * MOVEXX          *
*  (CONT.)                  *                * REMOVN          *
*                            *                * RMOVFN          *
*                            *                * RMOVF1          *
*-----*-----*-----*
*  COMP2                    *  COMPAR        * HERFN$          *
*                            *                * LASTXX          *
*-----*-----*-----*
*  CONSOP                   *  ISOP          * EUCLID          *
*                            *                * FMLCH1          *
*                            *                * FMLCH2          *
*                            *                * FMLERR          *
*                            *                * HCGTES          *
*                            *                * HERFN$          *
*                            *                * INTADD          *
*                            *                * INTEXP          *
*                            *                * INTMUL          *
*                            *                * NEXP5$          *
*                            *                * NEXP6$          *
*-----*-----*-----*
*  DIFIN                    *  DIFCMP        * COMPAR          *
*                            *                * FMLCHI          *
*                            *                * HERFN$          *
*-----*-----*-----*
*  DIF1                     *  DIF           * COMPAR          *
*                            *                * FMLCHH          *
*                            *                * FMLCH2          *
*                            *                * FMLCH5          *
*                            *                * HERFN$          *
*                            *                * IDPND           *
*-----*-----*-----*
*  DPND                    *  IDPND         * HERFN$          *
*-----*-----*-----*
*  DUMPO                   *  DUMPO         * CDIM            *
*                            *                * FMLCM1          *
*                            *                * FMLCM2          *
*                            *                * FMLCM3          *
*                            *                * PRINT$          *
*                            *                * PTRLG           *
*                            *                * SNAP$           *
*****

```


ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  ERRRET                    * ERRRET          * HERFNS          *
*                            *                * NIO1$           *
*                            *                * NIO2$           *
*                            *                * NPRT$           *
*-----*
*  ERRWBK                    * ERRERR          * FLDECI          *
*                            * SNPERR          * FMLCM1          *
*                            *                * FWORD           *
*                            *                * PCHCAS          *
*                            *                * PCHCNS          *
*                            *                * PRINTS          *
*                            *                * PRTCAS          *
*                            *                * PRTCNS          *
*                            *                * PUNCH$          *
*                            *                * READS           *
*                            *                * SNAPS           *
*                            *                * TRACE           *
*-----*
*  EXPRES                    * EXPRES          * FMLERR          *
*                            *                * GSCANR          *
*                            *                * HCGTES          *
*                            *                * HERFNS          *
*                            *                * ICOPY0          *
*                            *                * IFREE0          *
*                            *                * IFREE1          *
*                            *                * IFUNCT          *
*                            *                * IGETF1          *
*                            *                * ILINKN          *
*                            *                * ILINK1          *
*                            *                * LASTXX          *
*                            *                * RMOVFN          *
*                            *                * RMOVF1          *
*                            *                * STKOUT          *
*                            *                * SYMBOL          *
*-----*
*  E$DRUM                    * ROLLIN          * EXPOUT          *
*                            * ROLOUT          * EXPRIN          *
*                            *                * FMLCHB          *
*                            *                * FMLCHD          *
*                            *                * FMLERR          *
*****

```


ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  E$DRUM                   *                   * HCGTES$      *
*  (CONT.)                  *                   * HERFNS$      *
*                           *                   * PURGE        *
*                           *                   * S2INIT       *
*                           *                   * WCOLNT       *
*-----*
*  FMLASG                   * FMLASG          * ERRRET       *
*                           *                   * EXPRES       *
*                           *                   * FMLINT       *
*                           *                   * HERFNS$      *
*                           *                   * ISCAN$       *
*                           *                   * SYMBOL       *
*-----*
*  FMLCHA                   * FMLCH1          * FMLSV1       *
*                           * FMLCH2          * HCGTES$      *
*                           * FMLCH3          * HERFNS$      *
*                           * FMLCH4          * ROLLIN       *
*                           * FMLCH5          * WCOUNT      *
*                           * FMLCH6          *              *
*                           * FMLCH7          *              *
*-----*
*  FMLCHAIN                 * EXCHXX          * FMLCH1       *
*                           * IADDNN          * FMLCH2       *
*                           * ICOPY0          * FMLCH3       *
*                           * IGETF1          * FMLCH4       *
*                           * ILINKN          * FMLCH5       *
*                           * ILINK1          * FMLCH6       *
*                           * ILINK2          * FMLCH7       *
*                           *                 * HERFNS$      *
*                           *                 * NI02$        *
*                           *                 * NPRT$        *
*                           *                 * NSTOP$       *
*-----*
*  FMLCHAIN1                * IFREEN          * FMLCHA       *
*                           * IFREE0          * FMLCHB       *
*                           * IFREE1          * FMLCHC       *
*                           * IFREE2          * FMLCHD       *
*                           * IFREE9          * FMLCHE       *
*                           * MOVEXX          * FMLCHF       *
*                           * REMOVN          * FMLCHG       *
*                           * REMOV1          * FMLCHH       *
*****

```

(CONT.)

ELEMENT/VERSION	ENTRY POINTS	EXT. REFRNCS
FMLCHAIN1 (CONT.)	REMOV2 RMOVFN RMOVF1 RMOVF2	FMLCHI FMLCHK FMLCHL FMLCHM HERFNS NIO2S NPRTS NSTOPS
FMLCHB	FMLCHA FMLCHB FMLCHC FMLCHD FMLCHE FMLCHF FMLCHG FMLCHH FMLCHI FMLCHK FMLCHL FMLCHM	HCGTES HERFNS NIO2S NPRTS NSTOPS ROLLIN
FMLDMP	FMLDMP	ERRRET FMLDEX FMLINT FMLIOC FMLPCK FMLPIN FMLPKI FMLPKO FMLPKS HCGTES HERFNS NIO1S NIO2S NPRTS
FMLDMQ	FMLDEX	ERRERR ERRRET FMLPCK FMLPKO

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  FMLDMQ                    *                   * FMLPKS          *
*  (CONT.)                  *                   * HCGTES         *
*                            *                   * HERFNS         *
*                            *                   * IFUNC1         *
*-----*-----*-----*
*  FMLD00                    * FMLD0          * DSTACK         *
*                            * FMLEND         * ERRRET         *
*                            *                   * EXPRES         *
*                            *                   * FMLCHD         *
*                            *                   * FMLCHH         *
*                            *                   * FMLSC1         *
*                            *                   * FMLSC2         *
*                            *                   * HERFNS         *
*                            *                   * INISTK         *
*                            *                   * LSTSTK         *
*                            *                   * NIO1$          *
*                            *                   * NIO2$          *
*                            *                   * NRDUS          *
*                            *                   * NWDUS          *
*                            *                   * NXTSTK         *
*                            *                   * SETSTK         *
*                            *                   * SYMBOL         *
*-----*-----*-----*
*  FMLD01                    * DSTACK         * HERFNS         *
*                            * INISTK         * NIO2$          *
*                            * LSTSTK         * NWDUS          *
*                            * NXTSTK         *                   *
*                            * SETSTK         *                   *
*                            * USTACK         *                   *
*-----*-----*-----*
*  FMLEUI                    * FMLEDF         * EDECV$         *
*                            * FMLEUI         * EDITX$         *
*                            *                   * EDIT$          *
*                            *                   * EFLG1$         *
*-----*-----*-----*
*  FMLENT                    * FMLSC1         * GSCANR         *
*                            * FMLSC2         * ISCANR         *
*-----*-----*-----*
*  FMLEQU                    * FMLEQU         * ERRERR         *
*                            *                   * ERRRET         *
*                            *                   * FINDP1         *
*****

```

ENTRY POINTS AND EXTERNAL REFERENCES (CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
* FMLEQU                     *                     *
* (CONT.)                   *                     *
*                             *                     *
*                             *                     *
*                             *                     *
*                             *                     *
*                             *                     *
*                             *                     *
*                             *                     *
*                             *                     *
*-----*
* FMLERR                     * FMLERR          * HERFNS          *
*                             *                     * HMTX4$          *
*                             *                     * NI01$           *
*                             *                     * NI02$           *
*                             *                     * NPRT$           *
*-----*
* FMLERS                     * FMLERS          * ERRRET          *
*                             *                     * FMLCHD          *
*                             *                     * FMLINT          *
*                             *                     * FMLSCN          *
*                             *                     * GETARG          *
*                             *                     * HERFNS          *
*                             *                     * ISCANG          *
*                             *                     * SYMBOL          *
*-----*
* FMLFOR                     * IDENT           * COMPAR          *
*                             * IVALUE          * COUNT           *
*                             * LDOP            * ERRRET          *
*                             * NARG            * EXPRES          *
*                             * NCOUNT          * FMLCHD          *
*                             * VALUE           * FMLERR          *
*                             *                     * FMLINT          *
*                             *                     * HCGTES          *
*                             *                     * HERFNS          *
*                             *                     * ISCANG          *
*                             *                     * LASTXX          *
*-----*
* FMLINT                     * ARG             * CDIM            *
*                             * FMLINT          * ERRERR          *
*                             * FMLIN1          * FMLCH1          *
*                             * K10             * FMLCM1          *
*****

```

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  FMLINT                    *                  * FMLCM2          *
*  (CONT.)                  *                  * FMLCM3          *
*                            *                  * OPT$           *
*                            *                  * PRINT$         *
*                            *                  * VERSPR         *
*-----*
*  FMLIOA                    * FMLIOA          * HERFNS$        *
*                            *                  * NI01$          *
*                            *                  * NI02$          *
*                            *                  * NRDC$          *
*-----*
*  FMLIOB                    * FMLIOB          * HERFNS$        *
*                            *                  * NI01$          *
*                            *                  * NI02$          *
*                            *                  * NRDU$          *
*-----*
*  FMLIOC                    * FMLIOC          * HERFNS$        *
*                            *                  * NI01$          *
*                            *                  * NI02$          *
*                            *                  * NPRT$          *
*-----*
*  FMLIOD                    * FMLIOD          * HERFNS$        *
*                            *                  * NI01$          *
*                            *                  * NI02$          *
*                            *                  * NWDC$          *
*-----*
*  FMLIOE                    * FMLIOE          * HERFNS$        *
*                            *                  * NI01$          *
*                            *                  * NI02$          *
*                            *                  * NWDU$          *
*-----*
*  FMLIOF                    * FMLIFF          * HERFNS$        *
*                            * FMLIOF          * NI01$          *
*                            *                  * NI02$          *
*                            *                  * NPRT$          *
*-----*
*  FMLIOG                    * FMLIGG          * HERFNS$        *
*                            * FMLIOG          * NI01$          *
*                            *                  * NI02$          *
*                            *                  * NWDU$          *
*****

```

(CONT.)

* ELEMENT/VERSION		* ENTRY POINTS	* EXT. REFRNCS

* FMLI01	* FMLI01	* ERRRET	*
*	*	* FMLINT	*
*	*	* FMLIOA	*
*	*	* FMLRD	*
*	*	* HERFNS	*
*	*	* ISCANG	*

* FMLI02	* FMLI02	* ERRRET	*
*	*	* FMLINT	*
*	*	* FMLIOC	*
*	*	* FMLPIN	*
*	*	* FMLWR	*
*	*	* HERFNS	*
*	*	* ISCANG	*

* FMLI03	* FMLI03	* ERRRET	*
*	*	* FMLIFF	*
*	*	* FMLINT	*
*	*	* FMLIOF	*
*	*	* FMLPIN	*
*	*	* FMLWR	*
*	*	* HERFNS	*
*	*	* ISCANG	*

* FMLI04	* FMLI04	* ERRRET	*
*	*	* FMLINT	*
*	*	* FMLIOD	*
*	*	* FMLPIN	*
*	*	* FMLWR	*
*	*	* HERFNS	*
*	*	* ISCANG	*

* FMLI05	* FMLI05	* ERRRET	*
*	*	* FMLINT	*
*	*	* FMLIOB	*
*	*	* FMLRD	*
*	*	* HERFNS	*
*	*	* ISCANG	*

* FMLI06	* FMLI06	* ERRRET	*

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  FMLI06                    *                   *                   *
*  (CONT.)                  *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*-----*-----*-----*
*  FMLI07                    *  FMLI07          *  ERRRET           *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*                            *                   *                   *
*-----*-----*-----*
*  FMLMNI                    *                   *  COUNT            *
*                            *                   *  DSTACK           *
*                            *                   *  FMLASG           *
*                            *                   *  FMLDMP           *
*                            *                   *  FMLDO            *
*                            *                   *  FMLEND           *
*                            *                   *  FMLERS           *
*                            *                   *  FMLI01           *
*                            *                   *  FMLI02           *
*                            *                   *  FMLI03           *
*                            *                   *  FMLI04           *
*                            *                   *  FMLI05           *
*                            *                   *  FMLI06           *
*                            *                   *  FMLI07           *
*                            *                   *  FMLOPT           *
*                            *                   *  FMLOUT           *
*                            *                   *  FMLRES           *
*                            *                   *  FMLSAV           *
*                            *                   *  FMLUIT           *
*                            *                   *  HCGTES           *
*                            *                   *  HCONTS           *
*                            *                   *  NI01$            *
*                            *                   *  NI02$            *
*                            *                   *  NPRT$            *
*                            *                   *  NRDC$            *
*****

```


ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*   FMLMNI                  *                * NSLIOS          *
*   (CONT.)                 *                * NSTOPS          *
*                            *                * USTACK          *
*-----*
*   FMLOPT                  * FMLOPT        * ERRRET          *
*                            *                * FMLERR          *
*                            *                * FMLINT          *
*                            *                * GSCANR          *
*                            *                * HCGTES          *
*                            *                * HERFNS          *
*                            *                * ISCANR          *
*                            *                * ISRCH           *
*-----*
*   FMLOUT                  * FMLOUT        * ERRRET          *
*                            *                * FMLCHD          *
*                            *                * FMLINT          *
*                            *                * FMLSCN          *
*                            *                * GETARG          *
*                            *                * HERFNS          *
*                            *                * ISCANR          *
*                            *                * ROLOUT          *
*                            *                * SYMBOL          *
*-----*
*   FMLPCK                  * FMLPCK        * ERRERR          *
*                            * FMLPIN        * FMLEDF          *
*                            * FMLPKI        * FMLEDI          *
*                            * FMLPKO        * FMLERR          *
*                            *                * HERFNS          *
*                            *                * MOVE            *
*                            *                * PACK            *
*                            *                * STRCON          *
*                            *                * UNPACK          *
*-----*
*   FMLPKS                  * FMLPKS        * ERRERR          *
*                            *                * FMLPCK          *
*                            *                * HERFNS          *
*-----*
*   FMLRD                   * FMLRD         * EXPRES          *
*                            *                * FMLCHD          *
*                            *                * FMLSCN          *
*                            *                * GETARG          *
*****

```


ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*   FMLRD                   *               * HERFNS          *
*   (CONT.)                 *               * ISCAN$         *
*                           *               * SYMBOL         *
*-----*
*   FMLSAV                   * FMLPES        * CDIM           *
*                           * FMLSAV        * CSF$          *
*                           *               * FMLCM1         *
*                           *               * FMLCM2         *
*                           *               * FMLCM3         *
*                           *               * FMLINT         *
*                           *               * IOW$          *
*                           *               * PACK          *
*                           *               * PCHCA$        *
*                           *               * PCHCN$        *
*                           *               * PFIS$         *
*                           *               * PFSS$         *
*                           *               * PFWL$         *
*                           *               * PLNSZ1        *
*                           *               * PRINT$        *
*                           *               * PRTCAS        *
*                           *               * PRTCNS        *
*                           *               * PTRLG         *
*                           *               * PUNCH$        *
*                           *               * READ$         *
*                           *               * S2INIT        *
*                           *               * UNPACK        *
*-----*
*   FMLSCN                   * FMLSCN        * EXPRES         *
*                           *               * FMLCHB         *
*                           *               * FMLCHD         *
*                           *               * FMLCH1         *
*                           *               * FMLCH2         *
*                           *               * FMLERR         *
*                           *               * GSCANR         *
*                           *               * HERFNS         *
*-----*
*   FMLSV1                   * FMLSV1        * DUMP           *
*                           *               * ERRRET         *
*                           *               * HERFNS         *
*                           *               * NIO2$         *
*                           *               * NPRT$         *
*****

```

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  FMLSV1                    *                    * NSTOPS$          *
*  (CONT.)                   *                    * ROLCUT           *
*-----*
*  FMLUIT                    * FMLUIT           * EXPRES           *
*                    *                    * HERFNS$         *
*                    *                    * ISCANG           *
*                    *                    * NIO2$           *
*                    *                    * NPRT$           *
*-----*
*  FMLVER                    * FMLVER           *                    *
*-----*
*  FMLWR                    * FMLWR           * ERRERR           *
*                    *                    * FMLCHB           *
*                    *                    * FMLCHD           *
*                    *                    * FMLCH1           *
*                    *                    * FMLCH2           *
*                    *                    * FMLPCK           *
*                    *                    * FMLPKI           *
*                    *                    * FMLPKO           *
*                    *                    * FMLPKS           *
*                    *                    * FMLSCN           *
*                    *                    * FMLSV1           *
*                    *                    * GETARG           *
*                    *                    * HCGTES$         *
*                    *                    * HERFNS$         *
*                    *                    * IFUNC1           *
*                    *                    * LASTXX           *
*                    *                    * ROLLIN           *
*                    *                    * SYMBL1           *
*-----*
*  FNCT                    * IFUNCT           * HERFNS$         *
*                    * IFUNC1           * ISRCH           *
*-----*
*  FORMAL                    * CRDIN            * CLOSRC           *
*                    * CRDOUT           * CSF$             *
*                    * ERR              * ELT$             *
*                    * LISTOP           * ESOR             *
*                    * PARTBL           * EXITS$           *
*                    * WARN             * FMLVER           *
*                    *                    * GETSRC           *
*                    *                    * HCONT$           *
*****

```

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

ELEMENT/VERSION	ENTRY POINTS	EXT. REFRNCS
FORMAL (CONT.)		INFOR\$ IOW\$ OPNSRC POSTPR PREINF PREPRO PRINT\$ R\$ SDFI SDFIC SDFIO SDFO SDFOC SDFOO SELT\$ SOR SSOR STDET W\$
FRECT	COUNT	HCGTE\$ HERFNS
FUNTST	FUNTST	HERFNS RESTOR SAVE SKIP TYPE
GET	GET	HERFNS TYPE
GETARG	GETARG	FMLCHC HERFNS
GIT	GIT GIT1 GIT2	ERR GET HERFNS TYPE
GSCAN	GSCANR	ARG

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  GSCAN                     *  ISCANG      *  BCDRIN      *
*  (CONT.)                  *              *  BCDNPT      *
*                          *              *  BCDPNT      *
*                          *              *  BCDSTR      *
*                          *              *  FMLERR      *
*                          *              *  HERFNS$     *
*                          *              *  LDSL        *
*                          *              *  LOAD        *
*                          *              *  LSSC        *
*                          *              *  LSSL        *
*-----*
*  INCL                      *  INCLUD     *  COMPAR      *
*                          *              *  CONSOP      *
*                          *              *  HCGTES$     *
*                          *              *  HERFNS$     *
*                          *              *  IADDNN      *
*                          *              *  IFREEN      *
*                          *              *  IFREE2      *
*                          *              *  ILINKN      *
*                          *              *  ILINK1      *
*                          *              *  LASTXX      *
*                          *              *  REMOVN      *
*-----*
*  INTEGER                   *  EUCLID     *              *
*                          *  INTADD     *              *
*                          *  INTEXP     *              *
*                          *  INTMUL     *              *
*-----*
*  IOINTF                    *  EXPOUT     *  FMLCM1      *
*                          *  EXPRIN     *  FMLCM2      *
*                          *  PURGE      *  FMLCM3      *
*                          *  S2INIT     *  GETB$F      *
*                          *  WCOUNT    *  IBCH$N      *
*                          *              *  INIT$O      *
*                          *              *  PURG$F      *
*                          *              *  READ$F      *
*-----*
*  LIST                      *  FMLST      *  ERR         *
*                          *  FMLST1     *  FTROUT      *
*                          *  LIST       *  FTRSTK      *
*                          *              *  FTRUNS      *
*****

```

(CONT.)

ELEMENT/VERSION	ENTRY POINTS	EXT. REFRNCS
LIST		HERFNS
(CONT.)		NEW
		STCK
		STCKTM
		TYPE
		UNSTCK
		WARN
LOAD	LOAD	
NONAKITH	STLONG	HERFNS
	STSHRT	ISRCH
		SKIP
		TYPE
PACK	GETTBL	
	GETWD	
	ISRCH	
	MOVE	
	PACK	
	STRCON	
	STRTBL	
	STRWD	
	UNPACK	
SBEXP	LASTXX	ERRERR
		FMLCM2
		FMLCM3
SCANIN	SCANIN	ISCANG
SEC210	GETB\$F	CSF\$
	IBCH\$N	FMLCM1
	INIT\$0	IOW\$
	PLNSZ1	IOS\$
	PURG\$F	R\$
	READ\$F	SNAP\$
		WAIT\$
		W\$
SKIP	SKIP	HERFNS

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  SKIP                      *              * TYPE          *
*  (CONT.)                  *              *              *
*-----*-----*-----*
*  SPL-LINK                  * IDSA          *              *
*                          * IDSC          *              *
*                          * IDSL          *              *
*                          * IRH           *              *
*                          * IRHX          *              *
*                          * ISSA          *              *
*                          * ISSC          *              *
*                          * ISSL          *              *
*                          * IS1           *              *
*                          * IS2           *              *
*                          * IS3           *              *
*                          * IS4           *              *
*                          * IS5           *              *
*                          * IS6           *              *
*                          * IT1X          *              *
*                          * IT2X          *              *
*                          * IT3X          *              *
*                          * LDSC          *              *
*                          * LDSL          *              *
*                          * LH            *              *
*                          * LHX           *              *
*                          * LSSC          *              *
*                          * LSSL          *              *
*-----*-----*-----*
*  STACK                     * STCK          * ERR           *
*                          * STCKTM        * HERFNS        *
*                          * UNSTCK        * WARN          *
*-----*-----*-----*
*  STDET                     * STDET         * BACKSP        *
*                          *              * CHARIN        *
*                          *              * ERR           *
*                          *              * FMLST         *
*                          *              * FMLST1        *
*                          *              * FUNTST        *
*                          *              * GET           *
*                          *              * GIT           *
*                          *              * GIT2          *
*                          *              * HCGTES        *
*****

```

ENTRY POINTS AND EXTERNAL REFERENCES
(CONT.)

* ELEMENT/VERSION	* ENTRY POINTS	* EXT. REFRNCS	*

* STDET	*	* HERFNS	*
* (CONT.)	*	* ICRD	*
*	*	* IDEC	*
*	*	* ISRCH	*
*	*	* LIST	*
*	*	* LISTOP	*
*	*	* NEW	*
*	*	* RESET	*
*	*	* RESTOR	*
*	*	* SAVE	*
*	*	* SKIP	*
*	*	* STLONG	*
*	*	* STSHRT	*
*	*	* TYPE	*
*	*	* WARN	*

* STOUT0	* STKOUT	* COMPSF	*
*	*	* ERRERR	*
*	*	* FMLCHE	*
*	*	* FMLCHI	*
*	*	* FMLCH2	*
*	*	* FMLCH3	*
*	*	* FMLCH7	*
*	*	* HCGTES	*
*	*	* HERFNS	*
*	*	* LASTXX	*
*	*	* STDIFF	*
*	*	* STFCTD	*
*	*	* STFORM	*
*	*	* STMATH	*
*	*	* STUNFC	*

* STOUT1	* STFORM	* CMPRMV	*
*	*	* COMPAR	*
*	*	* DIFCMP	*
*	*	* ERRERR	*
*	*	* FMLERR	*
*	*	* HCGTES	*
*	*	* HERFNS	*
*	*	* HMTX4S	*
*	*	* IADDNN	*

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*  STOUT1                    *              *              *
*  (CONT.)                   *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*-----*
*  STOUT2                    *  STMATH      *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*-----*
*  STOUT3                    *  STFCTD      *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*                            *              *              *
*-----*
*  STOUT4                    *  STDIFF      *  DIF          *
*****

```


ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```

*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*   STOUT4                  *          *          *
*   (CONT.)                 *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*-----*-----*-----*
*   STOUT5                  * STUNFC    *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*          *                *          *          *
*-----*-----*-----*
*   SYMBFIND                * FINDP1    * HERFN$    *
*          *                * FINDP2    *          *
*-----*-----*-----*
*   SYMBOL                  * SYMBL1    * ERRERR    *
*          *                * SYMBOL    * FINDP1    *
*          *                *          * FINDP2    *
*          *                *          * FMLCHB    *
*          *                *          * FMLCHM    *
*          *                *          * FMLERR    *
*          *                *          * HCGTES    *
*          *                *          * HERFN$    *
*          *                *          * STSYEN    *
*-----*-----*-----*
*   SYMBSTOR                * STSYEN    * FMLCH1    *
*          *                *          * HERFN$    *
*-----*-----*-----*
*   TYPE                    * TYPE      * CHAR      *
*          *                *          * ERR       *
*          *                *          * HERFN$    *
*-----*-----*-----*
*   XXX                    * CDIM      *          *
*          *                * FMLCM1    *          *
*          *                * FMLCM2    *          *
*          *                * FMLCM3    *          *
*****

```

ENTRY POINTS AND EXTERNAL REFERENCES

(CONT.)

```
*****
*      ELEMENT/VERSION      * ENTRY POINTS * EXT. REFRNCS *
*****
*   XXX                    * PTRLG          *
*   (CONT,)                * VERSPR        *
*****
```

```

*****
* ENTRY POINT * ELEMENT *****
* ELEMENTS REFENCING*
* ARG * FMLINT * GSCAN *
* BACKSP * CHAR * STDET *
* BCDBIN * BCD-BIN * GSCAN *
* BCDNPT * BCD-BIN * GSCAN *
* BCDPNT * BCD-BIN * GSCAN *
* BCDSTR * BCD-BIN * GSCAN *
* CDIM * XXX * DUMPO *
* * * FMLINT * FMSAV *
* CHAR * CHAR * TYPE *
* CHARIN * CHAR * STDET *
* CMPRMV * COMPINC * STOUT1 *
* COMPAR * COMP2 * COMPINC *
* * * * *
* COMPSF * COMP1 * STOUT0 *
* CONSOP * CONSOP * COMP1 *
* COUNT * FRECT * FMLFOR *
* CRDIN * FORMAL * CHAR *
* CRDOU * FORMAL * CHAR *
*****

```

ENTRY POINT REFERENCES

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   DFETCH        *   BCD-BIN    *
*-----*
*   DIF           *   DIF1      *   STOUT4      *
*-----*
*   DIFCMP        *   DIFIN     *   STOUT1      *
*-----*
*   DSTACK        *   FMLD01    *   FMLD00      *
*               *               *   FMLMNI      *
*-----*
*   DUMPO         *   DUMPO     *
*-----*
*   ERR           *   FORMAL    *   CHAR        *
*               *               *   GIT         *
*               *               *   LIST        *
*               *               *   STACK       *
*               *               *   STDET       *
*               *               *   TYPE        *
*-----*
*   ERRERR        *   ERRWBK    *   COMP1       *
*               *               *   FMLDMQ      *
*               *               *   FMLEQU      *
*               *               *   FMLINT      *
*               *               *   FMLPCK      *
*               *               *   FMLPKS      *
*               *               *   FMLWR       *
*               *               *   SBEXP       *
*               *               *   STOUT0      *
*               *               *   STOUT1      *
*               *               *   STOUT4      *
*               *               *   STOUT5      *
*               *               *   SYMBOL      *
*-----*
*   ERRKET        *   ERRRET    *   FMLASG      *
*               *               *   FMLDMP      *
*               *               *   FMLDMQ      *
*               *               *   FMLD00      *
*               *               *   FMLEQU      *
*               *               *   FMLERS      *
*               *               *   FMLFOR      *
*               *               *   FMLI01      *
*****

```

(CONT.)

* * *	ENTRY POINT	*	ELEMENT	*ELEMENTS REFERENCEING*
* * *	ERRRET	*	FMLIO2	*
* * *	(CONT.)	*	FMLIO3	*
* * *		*	FMLIO4	*
* * *		*	FMLIO5	*
* * *		*	FMLIO6	*
* * *		*	FMLIO7	*
* * *		*	FMLOPT	*
* * *		*	FMLOUT	*
* * *		*	FMLSVI	*
* * *	EUCOLID	*	INTEGER	*CONSOP
* * *	EXCHXX	*	FMLCHAIN	*STOUT2
* * *	EXPOUT	*	I0INTF	*ESNRUM
* * *	EXPRES	*	EXPRES	*FMLASG
* * *		*	FMLD00	*
* * *		*	FMLFOR	*
* * *		*	FMLRD	*
* * *		*	FMLSCN	*
* * *		*	FMLUIT	*
* * *	EXPHIN	*	I0INTF	*ESDRUM
* * *	FINDP1	*	SYMBFIND	*FMLEQU
* * *		*	SYMBOL	*
* * *	FINDP2	*	SYMBFIND	*FMLEQU
* * *		*	SYMBOL	*
* * *	FIDRIN	*	BCDALL	*
* * *	FLOECI	*	BCDALL	*ERRWBK
* * *	FLOUCT	*	BCDALL	*
* * *	FMLASG	*	FMLASG	*FMLMNI
* * *	FMLCHA	*	FMLCHB	*FMLCHAIN1
* * *		*	STOUT3	*
* * *		*		*

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   FMLCHB        *   FMLCHB   *   E$DRUM           *
*                 *           *   FMLCHAIN1        *
*                 *           *   FMLSCN           *
*                 *           *   FMLWR            *
*                 *           *   SYMBOL           *
*-----*
*   FMLCHC        *   FMLCHB   *   FMLCHAIN1        *
*                 *           *   GETARG           *
*-----*
*   FMLCHD        *   FMLCHB   *   E$DRUM           *
*                 *           *   FMLCHAIN1        *
*                 *           *   FMLD00           *
*                 *           *   FMLEQU           *
*                 *           *   FMLERS           *
*                 *           *   FMLFOR           *
*                 *           *   FMLOUT           *
*                 *           *   FMLRD            *
*                 *           *   FMLSCN           *
*                 *           *   FMLWR            *
*-----*
*   FMLCHE        *   FMLCHB   *   FMLCHAIN1        *
*                 *           *   STOUT0           *
*                 *           *   STOUT3           *
*-----*
*   FMLCHF        *   FMLCHB   *   FMLCHAIN1        *
*-----*
*   FMLCHG        *   FMLCHB   *   FMLCHAIN1        *
*-----*
*   FMLCHH        *   FMLCHB   *   COMPINC          *
*                 *           *   DIF1             *
*                 *           *   FMLCHAIN1        *
*                 *           *   FMLD00           *
*-----*
*   FMLCHI        *   FMLCHB   *   DIFIN            *
*                 *           *   FMLCHAIN1        *
*                 *           *   STOUT0           *
*                 *           *   STOUT3           *
*-----*
*   FMLCHK        *   FMLCHB   *   FMLCHAIN1        *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   FMLCHL        *   FMLCHB    *   FMLCHAIN1        *
*-----*
*   FMLCHM        *   FMLCHB    *   FMLCHAIN1        *
*                   *                   *   SYMBOL            *
*-----*
*   FMLCH1        *   FMLCHA    *   CONSOP            *
*                   *                   *   FMLCHAIN          *
*                   *                   *   FMLINT            *
*                   *                   *   FMLSCN            *
*                   *                   *   FMLWR             *
*                   *                   *   SYMBSTOR          *
*-----*
*   FMLCH2        *   FMLCHA    *   CONSOP            *
*                   *                   *   DIF1              *
*                   *                   *   FMLCHAIN          *
*                   *                   *   FMLSCN            *
*                   *                   *   FMLWR             *
*                   *                   *   STOUT0            *
*-----*
*   FMLCH3        *   FMLCHA    *   FMLCHAIN          *
*                   *                   *   STOUT0            *
*-----*
*   FMLCH4        *   FMLCHA    *   FMLCHAIN          *
*-----*
*   FMLCH5        *   FMLCHA    *   DIF1              *
*                   *                   *   FMLCHAIN          *
*                   *                   *   STOUT3            *
*-----*
*   FMLCH6        *   FMLCHA    *   FMLCHAIN          *
*-----*
*   FMLCH7        *   FMLCHA    *   FMLCHAIN          *
*                   *                   *   STOUT0            *
*                   *                   *   STOUT5            *
*-----*
*   FMLCM1        *   XXX       *   DUMPO             *
*                   *                   *   ERRWBK            *
*                   *                   *   FMLINT            *
*                   *                   *   FMLSAV            *
*                   *                   *   IOINTF            *
*                   *                   *   SEC2IO            *
*****

```


ENTRY POINT REFERENCES

(CONT.)

ENTRY POINT	*	ELEMENT	**ELEMENTS REFERRING*

FMLCM2	*	XXX	DUMPO
	*		FMLINT
	*		FMLSAV
	*		IOINTF
	*		SBEXP

FMLCM3	*	XXX	DUMPO
	*		FMLINT
	*		FMLSAV
	*		IOINTF
	*		SBEXP

FMLDEX	*	FMLDMQ	FMLDMP
	*		FMLMNI

FMLUMP	*	FMLDMP	FMLMNI

FMLDO	*	FMLD00	FMLMNI

FMLEDF	*	FMLEDI	FMLPCK

FMLEDI	*	FMLEDI	FMLPCK

FMLEND	*	FMLD00	FMLMNI

FMLEQU	*	FMLEQU	

FMLERR	*	FMLERR	COMP1
	*		CONSEP
	*		EXPRES
	*		ESDRUM
	*		FMLFOR
	*		FMLOPT
	*		FMLPCK
	*		FMLSCN
	*		GSCAN
	*		STOUT1
	*		STOUT2
	*		STOUT3
	*		STOUT4
	*		SYMBOL

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   FMLERS        *   FMLERS    *   FMLMNI          *
*-----*
*   FMLIFF        *   FMLIOF    *   FMLIO3          *
*-----*
*   FMLIGG        *   FMLIOG    *   FMLIO7          *
*-----*
*   FMLINT        *   FMLINT    *   FMLASG          *
*               *               *   FMLDMP          *
*               *               *   FMLEQU          *
*               *               *   FMLERS          *
*               *               *   FMLFOR          *
*               *               *   FMLIO1          *
*               *               *   FMLIO2          *
*               *               *   FMLIO3          *
*               *               *   FMLIO4          *
*               *               *   FMLIO5          *
*               *               *   FMLIO6          *
*               *               *   FMLIO7          *
*               *               *   FMLOPT          *
*               *               *   FMLOUT          *
*               *               *   FMLSAV          *
*-----*
*   FMLIN1        *   FMLINT    *   FMLEQU          *
*-----*
*   FMLIOA        *   FMLIOA    *   FMLIO1          *
*-----*
*   FMLIOB        *   FMLIOB    *   FMLIO5          *
*-----*
*   FMLIOC        *   FMLIOC    *   FMLDMP          *
*               *               *   FMLIO2          *
*-----*
*   FMLIOD        *   FMLIOD    *   FMLIO4          *
*-----*
*   FMLIOE        *   FMLIOE    *   FMLIO6          *
*-----*
*   FMLIOF        *   FMLIOF    *   FMLIO3          *
*-----*
*   FMLIOG        *   FMLIOG    *   FMLIO7          *
*-----*
*   FMLIO1        *   FMLIO1    *   FMLMNI          *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   FMLI02        *   FMLI02    *   FMLMNI            *
*-----*
*   FMLI03        *   FMLI03    *   FMLMNI            *
*-----*
*   FMLI04        *   FMLI04    *   FMLMNI            *
*-----*
*   FMLI05        *   FMLI05    *   FMLMNI            *
*-----*
*   FMLI06        *   FMLI06    *   FMLMNI            *
*-----*
*   FMLI07        *   FMLI07    *   FMLMNI            *
*-----*
*   FMLOPT        *   FMLOPT    *   FMLMNI            *
*-----*
*   FMLOUT        *   FMLOUT    *   FMLMNI            *
*-----*
*   FMLPCK        *   FMLPCK    *   FMLDMP            *
*               *               *   FMLDMQ            *
*               *               *   FMLPKS            *
*               *               *   FMLWR             *
*-----*
*   FMLPIN        *   FMLPCK    *   FMLDMP            *
*               *               *   FMLI02            *
*               *               *   FMLI03            *
*               *               *   FMLI04            *
*               *               *   FMLI06            *
*               *               *   FMLI07            *
*-----*
*   FMLPKI        *   FMLPCK    *   FMLDMP            *
*               *               *   FMLWR             *
*-----*
*   FMLPKO        *   FMLPCK    *   FMLDMP            *
*               *               *   FMLDMQ            *
*               *               *   FMLWR             *
*-----*
*   FMLPKS        *   FMLPKS    *   FMLDMP            *
*               *               *   FMLDMQ            *
*               *               *   FMLWR             *
*-----*
*   FMLRD        *   FMLRD     *   FMLI01            *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   FMLRD         *           *   FMLI05         *
*   (CONT.)       *           *                   *
*-----*
*   FMLRES        *   FMLSAV    *   FMLMNI         *
*-----*
*   FMLSAV        *   FMLSAV    *   FMLMNI         *
*-----*
*   FMLSCN        *   FMLSCN    *   FMLEQU         *
*                   *           *   FMLERS         *
*                   *           *   FMLOUT         *
*                   *           *   FMLRD          *
*                   *           *   FMLWR          *
*-----*
*   FMLSC1        *   FMLENT    *   FMLD00         *
*-----*
*   FMLSC2        *   FMLENT    *   FMLD00         *
*-----*
*   FMLST         *   LIST      *   STDET          *
*-----*
*   FMLST1        *   LIST      *   STDET          *
*-----*
*   FMLSV1        *   FMLSV1    *   FMLCHA         *
*                   *           *   FMLWR          *
*-----*
*   FMLUIT        *   FMLUIT    *   FMLMNI         *
*-----*
*   FMLVER        *   FMLVER    *   FORMAL         *
*-----*
*   FMLWR         *   FMLWR     *   FMLI02         *
*                   *           *   FMLI03         *
*                   *           *   FMLI04         *
*                   *           *   FMLI06         *
*                   *           *   FMLI07         *
*-----*
*   FTROUT        *   CHAR      *   LIST           *
*-----*
*   FTRSTK        *   CHAR      *   LIST           *
*-----*
*   FTRUNS        *   CHAR      *   LIST           *
*-----*
*   FUNIST        *   FUNTST    *   STDET          *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCE*
*****
*   FWORD         *   BCDALL    *   ERRWBK         *
*-----*
*   GET           *   GET       *   GIT            *
*           *   *           *   STDET         *
*-----*
*   GETARG        *   GETARG   *   FMLEQU         *
*           *   *           *   FMLERS        *
*           *   *           *   FMLOUT        *
*           *   *           *   FMLRD         *
*           *   *           *   FMLWR         *
*-----*
*   GETB$F        *   SEC2IO   *   IOINTF         *
*-----*
*   GETTBL        *   PACK     *                   *
*-----*
*   GETWD         *   PACK     *                   *
*-----*
*   GIT           *   GIT       *   STDET         *
*-----*
*   GIT1          *   GIT       *                   *
*-----*
*   GIT2          *   GIT       *   STDET         *
*-----*
*   GSCANR        *   GSCAN    *   EXPRES         *
*           *   *           *   FMLENT        *
*           *   *           *   FMLOPT        *
*           *   *           *   FMLSCN        *
*-----*
*   IADDNN        *   FMLCHAIN *   COMP1          *
*           *   *           *   INCL           *
*           *   *           *   STOUT1         *
*           *   *           *   STOUT4         *
*-----*
*   IBCH$N        *   SEC2IO   *   IOINTF         *
*-----*
*   ICOPY0        *   FMLCHAIN *   EXPRES         *
*-----*
*   ICRU          *   CHAR     *   STDET         *
*-----*
*   IDEC          *   CHAR     *   STDET         *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   IDENT         *   FMLFOR    *
*-----*
*   IDPND         *   UPND      *   DIF1          *
*               *               *   STOUT1        *
*               *               *   STOUT4        *
*-----*
*   IDSA          *   SPL-LINK  *
*-----*
*   IDSC          *   SPL-LINK  *
*-----*
*   IDSL          *   SPL-LINK  *
*-----*
*   IFREEN        *   FMLCHAIN1 *   COMP1         *
*               *               *   INCL          *
*               *               *   STOUT1        *
*-----*
*   IFREE0        *   FMLCHAIN1 *   EXPRES        *
*-----*
*   IFREE1        *   FMLCHAIN1 *   COMP1         *
*               *               *   EXPRES        *
*-----*
*   IFREE2        *   FMLCHAIN1 *   INCL          *
*               *               *   STOUT2        *
*-----*
*   IFREE9        *   FMLCHAIN1 *
*-----*
*   IFUNCT        *   FNCT      *   EXPRES        *
*-----*
*   IFUNC1        *   FNCT      *   FMLDMQ        *
*               *               *   FMLWR         *
*-----*
*   IGETF1        *   FMLCHAIN  *   COMP1         *
*               *               *   EXPRES        *
*-----*
*   ILINKN        *   FMLCHAIN  *   COMP1         *
*               *               *   EXPRES        *
*               *               *   INCL          *
*               *               *   STOUT2        *
*-----*
*   ILINK1        *   FMLCHAIN  *   COMP1         *
*****

```

ENTRY POINT REFERENCES

(CONT.)

*****	ENTRY POINT	ELEMENT	*****
*****	*****	*****	*****
*	ILINK1	EXPRES	*
*	(CONT.)	INCL	*
*		STOUT1	*
*		STOUT2	*
*		STOUT4	*
*****	*****	*****	*****
*	ILINK2	FMLCHAIN	*
*****	*****	*****	*****
*	INCLUD	INCL	*
*****	*****	*****	*****
*	INISIK	FMLD01	*
*****	*****	*****	*****
*	INIT\$0	SEC210	*
*****	*****	*****	*****
*	INTADD	INTEGER	*
*****	*****	*****	*****
*	INTEXP	INTEGER	*
*****	*****	*****	*****
*	INTMUL	INTEGER	*
*****	*****	*****	*****
*	IRH	SPL-LINK	*
*****	*****	*****	*****
*	IRHX	SPL-LINK	*
*****	*****	*****	*****
*	ISCANG	GSCAN	*
*****	*****	*****	*****
*		FMLAS6	*
*		FMLENT	*
*		FMLERS	*
*		FMLFOR	*
*		FMLI01	*
*		FMLI02	*
*		FMLI03	*
*		FMLI04	*
*		FMLI05	*
*		FMLI06	*
*		FMLI07	*
*		FMLOPT	*
*		FMLOUT	*
*		FMLRD	*
*		FMLUIT	*
*		SCANIN	*
*****	*****	*****	*****

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   ISRCH         *   PACK      *   FMLOPT          *
*               *               *   FNCT            *
*               *               *   NONARITH        *
*               *               *   STDET           *
*-----*
*   ISSA          *   SPL-LINK  *
*-----*
*   ISSC          *   SPL-LINK  *
*-----*
*   ISSL          *   SPL-LINK  *
*-----*
*   IS1           *   SPL-LINK  *
*-----*
*   IS2           *   SPL-LINK  *
*-----*
*   IS3           *   SPL-LINK  *
*-----*
*   IS4           *   SPL-LINK  *
*-----*
*   IS5           *   SPL-LINK  *
*-----*
*   IS6           *   SPL-LINK  *
*-----*
*   IT1X          *   SPL-LINK  *
*-----*
*   IT2X          *   SPL-LINK  *
*-----*
*   IT3X          *   SPL-LINK  *
*-----*
*   IVALUE        *   FMLFOR    *
*-----*
*   K10           *   FMLINT    *
*-----*
*   LASTXX        *   SBEXP     *   COMPINC         *
*               *               *   COMP1           *
*               *               *   COMP2           *
*               *               *   EXPRES          *
*               *               *   FMLFOR          *
*               *               *   FMLWR           *
*               *               *   INCL            *
*****

```


ENTRY POINT	ELEMENT	ELEMENTS REFERENCING
LASTIX		STOUT0
(CONT.)		STOUT1
		STOUT2
		STOUT3
		STOUT4
		STOUT5
LDOP	FMLFOR	
LDSC	SPL-LINK	
LDL	SPL-LINK	GSCAN
LH	SPL-LINK	
LHX	SPL-LINK	
LIST	LIST	STDET
LISTOP	FORMAL	STDET
LOAD	LOAD	GSCAN
LSSC	SPL-LINK	GSCAN
LSSL	SPL-LINK	GSCAN
LSTSTK	FMLD01	FMLD00
MOVE	PACK	CHAR
MOVEXX	FMLCHAIN1	COMPI
NARG	FMLFOR	
NCOUNT	FMLFOR	
NEW	CHAR	LIST
		STDET

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   NXTSTK        *   FMLD01    *   FMLD00          *
*-----*-----*-----*
*   PACK          *   PACK      *   CHAR            *
*               *               *   FMLPCK          *
*               *               *   FMLSAV          *
*-----*-----*-----*
*   PAR1BL        *   FORMAL    *                   *
*-----*-----*-----*
*   PLNSZ1        *   SEC2IO    *   FMLSAV          *
*-----*-----*-----*
*   PTRLG         *   XXX       *   DUMPO           *
*               *               *   FMLSAV          *
*-----*-----*-----*
*   PURGE         *   IOINTF    *   E$DRUM          *
*-----*-----*-----*
*   PURG$F        *   SEC2IO    *   IOINTF          *
*-----*-----*-----*
*   READ$F        *   SEC2IO    *   IOINTF          *
*-----*-----*-----*
*   REMOVN        *   FMLCHAIN1 *   COMP1           *
*               *               *   INCL            *
*               *               *   STOUT1          *
*               *               *   STOUT2          *
*-----*-----*-----*
*   REMOV1        *   FMLCHAIN1 *   STOUT2          *
*-----*-----*-----*
*   REMOV2        *   FMLCHAIN1 *                   *
*-----*-----*-----*
*   RESET         *   CHAR      *   STDET           *
*-----*-----*-----*
*   RES1OR        *   CHAR      *   FUNTST          *
*               *               *   STDET           *
*-----*-----*-----*
*   RMOVFN        *   FMLCHAIN1 *   COMP1           *
*               *               *   EXPRES          *
*               *               *   STOUT1          *
*               *               *   STOUT2          *
*               *               *   STOUT4          *
*-----*-----*-----*
*   RMOVF1        *   FMLCHAIN1 *   COMP1           *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   RMOVF1        *           *   EXPRES           *
*   (CONT.)       *           *   STOUT1           *
*                 *           *   STOUT2           *
*                 *           *   STOUT4           *
*-----*
*   RMOVF2        *   FMLCHAIN1 *
*-----*
*   ROLLIN        *   E$DRUM   *   FMLCHA           *
*                 *           *   FMLCHB           *
*                 *           *   FMLWR            *
*-----*
*   ROLOUT        *   E$DRUM   *   FMLOUT           *
*                 *           *   FMLSV1           *
*-----*
*   SAVE          *   CHAR     *   FUNTST           *
*                 *           *   STDET            *
*-----*
*   SCANIN        *   SCANIN   *   FMLEQU           *
*-----*
*   SETSTK        *   FMLD01   *   FMLD00           *
*-----*
*   SKIP          *   SKIP     *   FUNTST           *
*                 *           *   NONARITH         *
*                 *           *   STDET            *
*-----*
*   SNPERR        *   ERRWBK   *
*-----*
*   STCK          *   STACK    *   LIST             *
*-----*
*   STCKTM        *   STACK    *   LIST             *
*-----*
*   STDET         *   STDET    *   FORMAL           *
*-----*
*   STDIFF        *   STOUT4   *   STOUT0           *
*-----*
*   STFCTD        *   STOUT3   *   STOUT0           *
*-----*
*   STFORM        *   STOUT1   *   STOUT0           *
*-----*
*   STKOUT        *   STOUT0   *   EXPRES           *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   STLONG        *   NONARITH   *   STDET          *
*-----*-----*-----*
*   STMATH        *   STOUT2    *   STOUT0         *
*-----*-----*-----*
*   STRCON        *   PACK      *   CHAR           *
*               *               *   FMLPCK         *
*-----*-----*-----*
*   STRIBL        *   PACK      *                   *
*-----*-----*-----*
*   STRWD         *   PACK      *                   *
*-----*-----*-----*
*   STSHRT        *   NONARITH   *   STDET          *
*-----*-----*-----*
*   STSYEN        *   SYMBSTOR   *   FMLEQU         *
*               *               *   SYMBOL         *
*-----*-----*-----*
*   STUNFC        *   STOUT5     *   STOUT0         *
*-----*-----*-----*
*   SYMBL1        *   SYMBOL     *   FMLWR          *
*-----*-----*-----*
*   SYMBOL        *   SYMBOL     *   EXPRES         *
*               *               *   FMLASG         *
*               *               *   FMLDOO         *
*               *               *   FMLERS         *
*               *               *   FMLOUT         *
*               *               *   FMLRD          *
*-----*-----*-----*
*   S2INIT        *   IOINTF     *   E$DRUM         *
*               *               *   FMLSAV         *
*-----*-----*-----*
*   TYPE          *   TYPE       *   FUNTST         *
*               *               *   GET            *
*               *               *   GIT            *
*               *               *   LIST           *
*               *               *   NONARITH       *
*               *               *   SKIP           *
*               *               *   STDET          *
*-----*-----*-----*
*   UNPACK        *   PACK       *   CHAR           *
*               *               *   FMLPCK         *
*****

```

ENTRY POINT REFERENCES

(CONT.)

```

*****
*   ENTRY POINT   *   ELEMENT   *ELEMENTS REFERENCING*
*****
*   UNPACK        *           *   FMLSAV           *
*   (CONT.)       *           *                       *
*-----*
*   UNSTCK        *   STACK    *   LIST             *
*-----*
*   USTACK        *   FMLD01   *   FMLMNI           *
*-----*
*   VALUE         *   FMLFOR   *                       *
*-----*
*   VERSPR        *   XXX      *   FMLINT           *
*-----*
*   WARN          *   FORMAL   *   LIST             *
*               *           *   STACK            *
*               *           *   STDET            *
*-----*
*   WCOUNT       *   IOINTF   *   E$DRUM           *
*               *           *   FMLCHA           *
*****

```

COMMON BLOCK SIZE

```
*****
*          COMMON          *          SIZE          *
*****
*          BUF            *          21            *
*-----*-----*-----*
*          DOSTAK         *          700           *
*-----*-----*-----*
*          FMLCM1         *          166           *
*-----*-----*-----*
*          FMLCM2         *          2048          *
*-----*-----*-----*
*          FMLCM3         *          2048          *
*****
```

COMMON BLOCK REFERENCES

```

*****
*          COMMON          *          ELEMENTS REFERENCING      *
*****
*          BUF            *          CHAR                      *
*                        *          FUNTST                    *
*                        *          GET                      *
*                        *          GIT                      *
*                        *          LIST                     *
*                        *          NONARITH                   *
*                        *          STDET                     *
*                        *          TYPE                      *
*-----*-----*
*          DOSTAK        *          FMLD00                    *
*                        *          FMLD01                    *
*-----*-----*
*          FMLCM1        *          COMPINC                   *
*                        *          COMP1                     *
*                        *          COMP2                     *
*                        *          CONSOP                     *
*                        *          DIFIN                     *
*                        *          DIF1                      *
*                        *          DPND                      *
*                        *          EXPRES                     *
*                        *          E$DRUM                    *
*                        *          FMLASG                     *
*                        *          FMLCHA                     *
*                        *          FMLCHB                     *
*                        *          FMLDMP                     *
*                        *          FMLDMQ                     *
*                        *          FMLD00                     *
*                        *          FMLEQU                     *
*                        *          FMLERS                     *
*                        *          FMLFOR                     *
*                        *          FMLI0A                     *
*                        *          FMLI0B                     *
*                        *          FMLI0C                     *
*                        *          FMLI0D                     *
*                        *          FMLI0E                     *
*                        *          FMLI0F                     *
*                        *          FMLI0G                     *
*                        *          FMLI02                     *
*                        *          FMLI03                     *
*                        *          FMLI04                     *
*                        *          FMLI06                     *
*                        *          FMLI07                     *
*                        *          FMLOPT                     *
*****

```

COMMON BLOCK REFERENCES

(CONT.)

*****		*****	
*	COMMON	*	ELEMENTS REFERENCING
*****		*****	
*	FMLCM1	*	FMLOUT
*	(CONT.)	*	FMLPCK
*		*	FMLRD
*		*	FMLSCN
*		*	FMLSV1
*		*	FMLUET
*		*	FMLWR
*		*	FRECT
*		*	GETARG
*		*	INCL
*		*	STOUT0
*		*	STOUT1
*		*	STOUT2
*		*	STOUT3
*		*	STOUT4
*		*	STOUT5
*		*	SYMBFIND
*		*	SYMBOL
*		*	SYMBSTOR
*		*	XXX
-----		-----	
*	FMLCM2	*	COMPINC
*		*	COMP1
*		*	COMP2
*		*	CONSOP
*		*	DIFIN
*		*	DIF1
*		*	DPND
*		*	EXPRES
*		*	E\$DRUM
*		*	FMLASG
*		*	FMLCHA
*		*	FMLCHB
*		*	FMLDMP
*		*	FMLDMQ
*		*	FMLDOO
*		*	FMLEQU
*		*	FMLERS
*		*	FMLFOR
*		*	FMLIOA
*		*	FMLIOB
*****		*****	

(CONT.)

*****		*****	
*	COMMON	*	ELEMENTS REFERENCING
*****		*****	
*	FMLCM2	*	FMLIOC
*	(CONT.)	*	FMLIOD
*		*	FMLIOE
*		*	FMLIOF
*		*	FMLIOG
*		*	FMLIO2
*		*	FMLIO3
*		*	FMLIO4
*		*	FMLIO6
*		*	FMLIO7
*		*	FMLOPT
*		*	FMLOUT
*		*	FMLPCK
*		*	FMLRD
*		*	FMLSCN
*		*	FMLSV1
*		*	FMLUIT
*		*	FMLWR
*		*	FRECT
*		*	GETARG
*		*	INCL
*		*	STOUT0
*		*	STOUT1
*		*	STOUT2
*		*	STOUT3
*		*	STOUT4
*		*	STOUT5
*		*	SYMBFIND
*		*	SYMBOL
*		*	SYMBSTOR
*		*	XXX
-----		-----	
*	FMLCM3	*	COMPINC
*		*	COMP1
*		*	COMP2
*		*	CONSOP
*		*	DIFIN
*		*	DIF1
*		*	DPND
*		*	EXPRES
*		*	ESDRUM
*****		*****	

(CONT.)

COMMON		ELEMENTS REFERENCING	
FMLCM3		FMLASG	
(CONT.)		FMLCHA	
		FMLCHB	
		FMLDMP	
		FMLDMO	
		FMLDOO	
		FMLEQU	
		FMLERS	
		FMLFOR	
		FMLIOA	
		FMLIOB	
		FMLIOC	
		FMLIOD	
		FMLIOE	
		FMLIOF	
		FMLIOG	
		FMLIO2	
		FMLIO3	
		FMLIO4	
		FMLIO6	
		FMLIO7	
		FMLOPT	
		FMLOUT	
		FMLPCK	
		FMLRD	
		FMLSCN	
		FMLSV1	
		FMLUIT	
		FMLWR	
		FRECT	
		GETARG	
		INCL	
		STOUT0	
		STOUT1	
		STOUT2	
		STOUT3	
		STOUT4	
		STOUT5	
		SYMBFIND	
		SYMBOL	
		SYMBSTOR	
		XXX	

7. INDEX FOR PROGRAM ELEMENTS

ACOMMON	ASM PROC	OBJECT	3-5
COMMON BLOCKS FOR ASSEMBLY PROGRAMS			
APROCS	FOR PROC	OBJECT	3-3
THIS PROCEDURE DEFINES THE COMMON BLOCK AREAS TO BE USED BY EACH FORMAL OBJECT TIME SUBROUTINES.			
BCDALL	ASM	OBJECT	3-41
BINARY TO FIELDATA CONVERSION ROUTINE			
BCD-BIN	ASM	OBJECT	3-53
FIELDATA-TO-BINARY INTEGER/FLOATING CONVERSION ROUTINE			
BPROCS	FOR PROC	PREPRO	2-21
CHARACTER TABLE FOR SCANNER			
CHAK	FOR	PREPRO	2-25
ROUTINE TO INPUT SOURCE A CHARACTER AT A TIME			
COMPINC	FOR	OBJECT	3-156
EXPRESSION COMPARISON AND REMOVAL ROUTINE			
COMP1	FOR	OBJECT	3-120
EXPRESSION COMPARISON FOR SORTING AND SIMPLIFYING			
COMP2	FOR	OBJECT	3-129
EXPRESSION COMPARISON ROUTINE			
CONSOP	FOR	OBJECT	3-79
ARITHMETIC OPERATIONS WITH FORMAL CONSTANTS			
DIFIN	FOR	OBJECT	3-158
BASIC ROUTINE FOR REPLACING DIFF. QUOTIENTS			

7-2

DIF1	FOR	OBJECT	3-182
BASIC DIFFERENTIATION SUBROUTINE			
DPND	FOR	OBJECT	3-181
DEPENDENCE CHECK SUBROUTINE			
DUMPO	ASM	OBJECT	
TEMPORARY ROUTINE FOR OCTAL DUMP FROM COMMON			
ERRRET	FOR	OBJECT	3-61
ERROR ROUTINE TO PRINT SUBROUTINE NAME			
ERRWBK	ASM	OBJECT	3-40
ERROR AND DEBUGGING ROUTINE FOR OBJECT-TIME F O R M A L			
EXPRES	FOR	OBJECT	3-103
EXPRESSION TRANSLATION FROM INFIX TO PREFIX			
E\$DRUM	FOR	OBJECT	3-20
DRIVER FOR SAVING/RESTORING EXPRESSIONS ON DRUM			
FMLASG	FOR	OBJECT	3-235
FORMAL ASSIGNMENT STATEMENT			
FMLCHA	FOR	OBJECT	3-10
GENERAL CHAIN ROUTINES			
FMLCHAIN	FOR	OBJECT	3-8
TEMPORARY CHAIN ROUTINE I.			
FMLCHAIN1	FOR	OBJECT	3-9
TEMPORARY CHAIN ROUTINE II.			
FMLCHB	FOR	OBJECT	3-14
CONTINUATION OF GENERAL CHAIN ROUTINES			
FMLDMP	FOR	OBJECT	3-230

DUMP STATEMENT ROUTINE

FMLDMQ	FOR	OBJECT	3-232
EXPRESSION PRINT FOR DUMP ROUTINE			
FMLD00	FOR	OBJECT	3-243
DO STATEMENT HANDLER FOR INTERACTIVE FORMAL			
FMLD01	FOR	OBJECT	3-247
DO LOOP STACK ROUTINE			
FMLEDI	ASM	OBJECT	3-199
CONVERSION ROUTINE FROM BINARY TO FD			
FMLENT	ASM	OBJECT	
RE-NAMED ENTRY POINTS FOR FORMAL ROUTINES			
FMLEQU	FOR	OBJECT	3-255
ARGUMENT EQUIVALENCING ROUTINE			
FMLERR	FOR	OBJECT	3-60
FORMAL ERROR MESSAGE ROUTINE			
FMLERS	FOR	OBJECT	3-241
FORMAL ERASE STATEMENT			
FMLFOR	FOR	OBJECT	3-249
FORTRAN VALUED FUNCTION ROUTINES			
FMLINT	ASM	OBJECT	3-36
OVERALL INITAILIZATION ROUTINE AND INITIALIZATION FOR FORMAL STATEMENTS			
FML10A	FOR	OBJECT	3-210
CARD INPUT ROUTINE			
FML10B	FOR	OBJECT	3-210
I/O DEVICE INPUT ROUTINE			

FML10C	FOR	OBJECT	3-211
PRINTED OUTPUT ROUTINE			
FML10D	FOR	OBJECT	3-211
PUNCHED OUTPUT ROUTINE			
FML10E	FOR	OBJECT	3-212
I/O DEVICE OUTPUT ROUTINE			
FML10F	FOR	OBJECT	3-212
PRINT PARTIAL BUFFER ROUTINE			
FML10G	FOR	OBJECT	3-213
WRITE PARTIAL BUFFER TO I/O DEVICE ROUTINE			
FML101	FOR	OBJECT	3-203
FORMAL STATEMENT ROUTINE TO HANDLE CARD INPUT			
FML102	FOR	OBJECT	3-204
FORMAL STATEMENT ROUTINE TO PRINT EXPRESSIONS			
FML103	FOR	OBJECT	3-205
FORMAL STATEMENT ROUTINE TO PRINT FORMATED EXPRESSION			
FML104	FOR	OBJECT	3-206
FORMAL STATEMENT ROUTINE TO PUNCH EXPRESSIONS			
FML105	FOR	OBJECT	3-207
FORMAL STATEMENT ROUTINE TO HANDLE INPUT FROM I/O UNITS			
FML106	FOR	OBJECT	3-208
FORMAL STATEMENT ROUTINE TO WRITE EXPRESSIONS ON UNIT U			
FML107	FOR	OBJECT	3-209
FORMAL STATEMENT TO WRITE FORMATED EXPRESSION			
FMLMNI	FOR	OBJECT	3-262

MAIN PROGRAM FOR INTERACTIVE FORMAL

FMLOPT	FOR	OBJECT	3-237
FORMAL OPTION STATEMENT ROUTINE			
FMLOUT	FOR	OBJECT	3-215
FORMAL EXPRESSION ROLL-OUT ROUTINE			
FMLPCK	FOR	OBJECT	3-196
OUTPUT PACKING ROUTINE FOR EXPRESSIONS			
FMLPKS	FOR	OBJECT	3-194
SUBROUTINE TO UN-PACK 'SN' SUBSCRIPTS FROM THE SUBSCRIPT-WORD 'SW'.			
FMLRD	FOR	OBJECT	3-202
SUBROUTINE TO HANDLE INPUT OF STATEMENTS			
FMLSAV	ASM	OBJECT	3-219
SAVE/RESET ROUTINE			
FMLSCN	FOR	OBJECT	3-88
SUBROUTINE TO PUT THE LIST OF VARIABLES IN GSCANS ARRAY INTO A CHAIN STARTING AT NK			
FMLSV1	FOR	OBJECT	3-17
ROLLOUT ROUTINE FOR ALL EXPRESSIONS			
FMLUIT	FOR	OBJECT	3-201
EXTRACT FORMAT DEFINITIONS IN I/O STATEMENTS			
FMLVER	ASM	PREPRO	2-3
VERSION FOR THE PREPROCESSOR			
FMLWR	FOR	OBJECT	3-189
EXPRESSION TRANSLATION FROM PREFIX TO INFIX FOR OUTPUT			
FNCT	FOR	OBJECT	3-99

FUNCTION NAME AND ID. NUMBER CONVERSION ROUTINE

FORMAL	ASM	PREPRO	2-3
PREPROCESSOR MAIN PROGRAM			
FRECT	FOR	OBJECT	3-253
FREE STORAGE INVENTORY ROUTINE			
FUNTST	FOR	PREPRO	2-50
FUNCTION TEST ROUTINE			
GET	FOR	PREPRO	2-34
ROUTINE TO BUILD IDENTIFIER			
GETARG	FOR	OBJECT	3-89
ROUTINE TO OBTAIN THE FIRST VARIABLE FROM THE LIST			
GIT	FOR	PREPRO	2-35
GSCAN	FOR	OBJECT	3-44
FIELDATA SCANNING ROUTINE			
INCL	FOR	OBJECT	3-152
CALCULATE THE GREATEST COMMON MULTIPLE OF 2 EXPRESSIONS			
INTEGER	ASM	OBJECT	3-86
INTEGER ARITHMETIC OPERATIONS			
IOINTF	ASM	OBJECT	3-22
EXPRESSION PACKING AND UNPACKING FOR ROLLIN/ROLOUT			
LIST	FOR	PREPRO	2-51
SCANS ARITHMETIC STATEMENTS, ARGUMENTS, ETC.			
LOAD	ASM	OBJECT	3-50
GENERAL OBJECT-TIME SCAN ROUTINE			

NONARITH	FOR	PREPRO	2-47
CHECKS FOR SPECIFIC FORTRAN STATEMENTS			
PACK	ASM	OBJECT	3-76
UTILITY PROGRAM TO (UN)PACK CHARACTERS			
MOVE WORDS			
STORE CONSTANTS INTO ARRAYS			
SBEXP	ASM	OBJECT	3-96
CALCULATION OF THE END OF SUBEXPRESSION IN PREFIX			
SCANIN	ASM	OBJECT	3-257
ALTERNATE WAY OF INITIALIZING SCANNER FOR FMLEQU			
SEC2IO	ASM	OBJECT	3-26
ROLLOUT ROUTINE			
SKIP	FOR	PREPRO	2-36
SKIPS CHARACTERS IN THE SCAN			
SPL-LINK	ASM	OBJECT	3-57
FORTRAN LINKAGES TO ASSEMBLER SHIFT INSTRUCTIONS			
AND PARTIAL WORD LOADS.			
STACK	FOR	PREPRO	2-54
STACKS FORTRAN AND FML PARENTHESIS, BRACKETS, ETC. FOR			
PRELIMINARY SYNTAX CHECK			
STDET	FOR	PREPRO	2-39
SCANS ALL FORTRAN AND FORMAL STATEMENTS, DETERMINES TYPE,			
AND DOES THE APPROPRIATE PROCESSING			
STOUT0	FOR	OBJECT	3-116
OPERATOR WRITE-OUT FROM PUSH-DOWN STACK			
SIMPLIFICATION WITH ARITHMETIC OPERATIONS			
STOUT1	FOR	OBJECT	3-135
EVALUATION OF FORMAL FUNCTIONS			

7-8

STOUT2	FOR	OBJECT	3-162
EVALUATION OF MATHEMATICAL FUNCTIONS			
STOUT3	FOR	OBJECT	3-170
EVALUATION OF DEFINED FUNCTIONS			
STOUT4	FOR	OBJECT	3-175
EVALUATION OF DIFFERENTIATION			
STOUT5	FOR	OBJECT	3-185
EVALUATION OF UNDEFINED FUNCTIONS			
SYMBFIND	FOR	OBJECT	3-69
SYMBOL TABLE SEARCHING ROUTINE			
SYMBOL	FOR	OBJECT	3-63
SYMBOL TABLE ROUTINE			
SYMBSTOR	FOR	OBJECT	3-71
ROUTINE TO STORE SYMBOL ENTRIES			
TYPE	FOR	PREPRO	2-32
GETS ONE CHARACTER FROM CHAR AND RETURNS ACCORDING TO ITS TYPE			
XXX	ASM	OBJECT	3-39
SIGN-ON LINE AND COMMON DECLARATION FOR SUBROUTINE			