SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

122002

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM CATALOG, PROGRAM DESCRIPTION SUBMITTAL

(For details on the use of this form, see the SHARE Reference Manual, Section 06)

Input Field Number			Card Columns
1	SDA Number (to be filled in by SDA)	SDA	A13-21
	Date of Submittal	Aug. 15, 1967 New [X] or Revision [ ]	
2	Submitter's Installation Code	STU	ALL CARDS 1-3
3	Program Number or Designation (and Suffix)	FOPL	4-8
4	Submitter's Name	LEACH, LANSE M.	A22-38
5	Submitter's Department (primarily for internal use)	COMPUTER CTR	A39-50
6	Author (if different from above)		A51-67
7	Year Completed (last 2 digits) or Status Code	67	A68-69
8	Title	FORTRAN IV TO PL/1 TRANSLATOR	Title Card O12-68
9	Field of Application	PL/1 CONVERSION	B12-26
10	Primary Subject Code	67	B27-29
11	Secondary Subject Codes		B30-41
12	Principal Source Language	PL/1	B42-48
13	Secondary Source Language		B49-55
14	Type of Routine	MP	B56-57
15	Machine	SYS/360	B58-64
16	Monitor or Operating System Required	OS/360	B65-71
17	Special Machine Requirements	PL/1(F) CMPLR	C12-23
18	Non-Library Routines or Subr. Req.		C24-39
19	Documents Available (indicate page counts):		
	Sent to all Installations: Short Write-up	SW	
	Sent to all Installations: Catalog Cards	19	
	Orderable from SDA: Long Write-up	PA 11	C40-49
	Orderable from SDA: Listing	LS 20	C50-54
20-21	Program Material Avail.		
20	a. Primary Form	PL/1 SOURCE	D12-31
	b. Count	978	D32-39
	c. Medium	CDS	D32-39
21	a. Additional Form	SAMPLE DATA DECK	D40-59
	b. Count	46	D60-67
	c. Medium	CDS	D60-67
22	Search Key	*FORTRAN IV TO*PL/1*T RANSLATOR	Search Card S12-74

Remarks (not to be keypunched)

SHARE PROGRAM CATALOG, PROGRAM DESCRIPTION SUBMITTAL (Continued)

Abstract (Cards 10-99, Columns 12-72)

THIS PROGRAM TRANSLATES FORTRAN IV AS STANDARDIZED BY THE AMERICAN STANDARDS ASSOCIATION (COMMUNICATIONS OF THE ACM, OCT, 1964) TO PL/1 AS DEFINED BY THE IBM PL/1 SPECIFICATIONS (FORM C28-6571). THE TRANSLATOR IS WRITTEN IN PL/1 AND USES THE METHOD OF RECURSIVE DESCENT TO ACCOMPLISH THE TRANSLATION. THE TRANSLATOR PRODUCES READABLE PL/1 OUTPUT THAT REQUIRES MINIMUM PROGRAMMER EFFORT TO OBTAIN A PERFECT TRANSLATION. FORTRAN IS ASSUMED TO HAVE RESERVED WORDS WITH SIGNIFICANT BLANKS. DATA, EQUIVALENCE, AND BACKSPACE STATEMENTS ARE NOT TRANSLATED. THE TRANSLATOR IS WRITTEN IN A WAY TO MAKE IT EASILY MODIFIABLE TO INCLUDE ADDITIONAL FORTRAN STATEMENTS OR ACCEPT A PARTICULAR INSTALLATION VERSION OF FORTRAN IV RATHER THAN THE ASA STANDARD.

(Please attach additional pages, if necessary)

Pages Attached: keypunchable abstract 0
Non-keypunchable short write-up 0

Signature of Submitter Laura M. Faulk Date 15 AUGUST

Signature of Installation Addressee Ryden M. Fredrickson

Printed in U.S.A. 328-1926

FORTRAN IV TO PL/1 TRANSLATOR

AUTHOR: Lanse M. Leach

Direct Inquiries To: Lanse M. Leach
Computation Center
Stanford University
Stanford, California 94305

PART I

DESCRIPTION AND GUIDE TO THE TRANSLATOR PROGRAM

This section contains a detailed description of the translator program with enough facts to enable a programmer to expand the translator to handle additional FORTRAN statements or to modify the translator to accept the FORTRAN of a particular computer installation instead of the ASA Standard FORTRAN.

The translator program, written in PL/1 and compiled under the IBM System/360 F-Level PL/1 Compiler, consists of a main program of four executable statements and fifty-one internal procedures. The main program transfers control to procedure PROGRAM for each FORTRAN main or sub-program in the input stream. Execution terminates when an end of file is encountered in the input stream.

Procedure PROGRAM performs the initialization of the symbol table and various counters used by the translator and calls procedure STATEMENT for each FORTRAN statement until the FORTRAN END statement is encountered. The procedure OUTPUT is called which produces PL/1 declarations from the symbol table information and transfers PL/1 statements from two temporary files or data-sets to the printer and punch or directly to a data-set for input to the PL/1 Compiler.

The text scanning procedure SCAN makes extensive use of PL/1 string operations to return the varying length character string, NEXT, which is composed of either a string of letters, a string of digits, or a special

character. Thus to recognize the FORTRAN floating point number 5.58E-16 six calls to SCAN are required. Also it is the responsibility of procedure FORTIDEN to build valid FORTRAN identifiers from repeated calls to SCAN. SCAN has a one bit argument which controls whether or not to start scanning a new FORTRAN statement or to continue with the current input string, LINE. Other functions performed by SCAN are label processing and comment processing. SCAN determines if a statement number exists on a new statement and if so calls procedure LABEL. If a "C" occurs in column one then procedure COMMENT is called directly from SCAN. SCAN also checks the next card in the input stream for a continuation mark. If the card is a continuation, columns seven to seventy-two are concatenated onto LINE and the next card in the input stream is checked. Thus the reading of card images from the input stream is always one card ahead of the actual processing of FORTRAN statements. Because of this and in order that the translator can have any number of FORTRAN main or subprograms in the input stream, a blank card is required after each END statement in the input stream.

FORTRAN comments have blanks chopped off each end and if a comment card contains all blanks with the exception of the "C" in column one, a blank card is inserted in the PL/1 program to improve appearance. The following is an example of comment processing:

FORTRAN COMMENT

```
C     THIS IS A COMMENT CARD .....................
```

TRANSLATOR PRODUCED PL/1 COMMENT

```
/* THIS IS A COMMENT CARD ..................... */
```


A symbol table is constructed from the information provided in the FORTRAN type statements and consists of four varying length character string arrays as follows:

SYMBOL	NAME OF THE FORTRAN VARIABLE
SYMDIM	DIMENSIONING INFORMATION
SYMTYPE	TYPE INFORMATION
SYMCOM	EXTERNAL AND INITIAL ATTRIBUTE INFORMATION

For each FORTRAN type statement encountered the symbol table is checked to see if the FORTRAN variable is already in the symbol table and if so the symbol table is updated otherwise the new variable name is added to the end of the symbol table and the length counter is increased by one.

Translator input/output is done using PL/1 stream input/output statements. Only when reading FORTRAN card images is a formatted input/output statement used. At all other times either a PUT LIST or a GET LIST input/output statement is used. The following are the five input/output files used by the translator:

SYSIN	INPUT	Fortran source card images
DECLIST	INPUT/OUTPUT	Scratch file for PL/1 declarations and the initial procedure declaration. On the IBM 360/67 this scratch file was on a 2311 disk pack.
PRGLIST	INPUT/OUTPUT	Scratch file for PL/1 statements other than declarations. On the IBM 360/67 this scratch file was on a 2311 disk pack.
SYSPRINT	OUTPUT	Printed output usually consisting of input

		FORTRAN programs, error messages, and the translator produced PL/1 version.
PUNLIST	OUTPUT	Output of translator produced PL/1 program. Punch, data-set for input to the PL/1 Compiler, or dummy if no supplementary output is desired.

In order to produce readable PL/1 output, a tab consisting of a variable number of blanks is concatenated on the front of each output string. Initially the tab is of length one and is increased by three blanks each time a DO group or procedure is entered and decreased by three blanks upon exiting a DO group or procedure.

In order to modify the translator to accept additional FORTRAN statements the following two steps are required:

(1) Write an internal procedure to translate the additional statement. To output the translated statement a call to procedure DISK with the output character string as argument is required.

(2) Insert an IF statement in procedure STATEMENT to produce a call to the added internal procedure when the added statement is encountered.

PART II

LIMITATION AND PROGRAMMER INTERVENTION REQUIRED

The following areas require the user of the translator to modify the translator-produced PL/1 in order to obtain a correctly running PL/1 program:

(1) DATA, EQUIVALENCE, and BACKSPACE statements are not translated and thus require hand translation. For the EQUIVALENCE statement the DEFINED attribute in the DECLARE statement can be used to obtain a correct translation. For the DATA statement the INITIAL attribute in the DECLARE statement can be used. In PL/1 there is no direct translation for the BACKSPACE statement. Thus a FORTRAN program using the BACKSPACE statement would require modification.

(2) Hollerith strings in FORTRAN FORMAT statements can not be placed in PL/1 FORMAT statements. Thus hollerith strings must be converted to character strings and placed in the input or output statement variable lists.

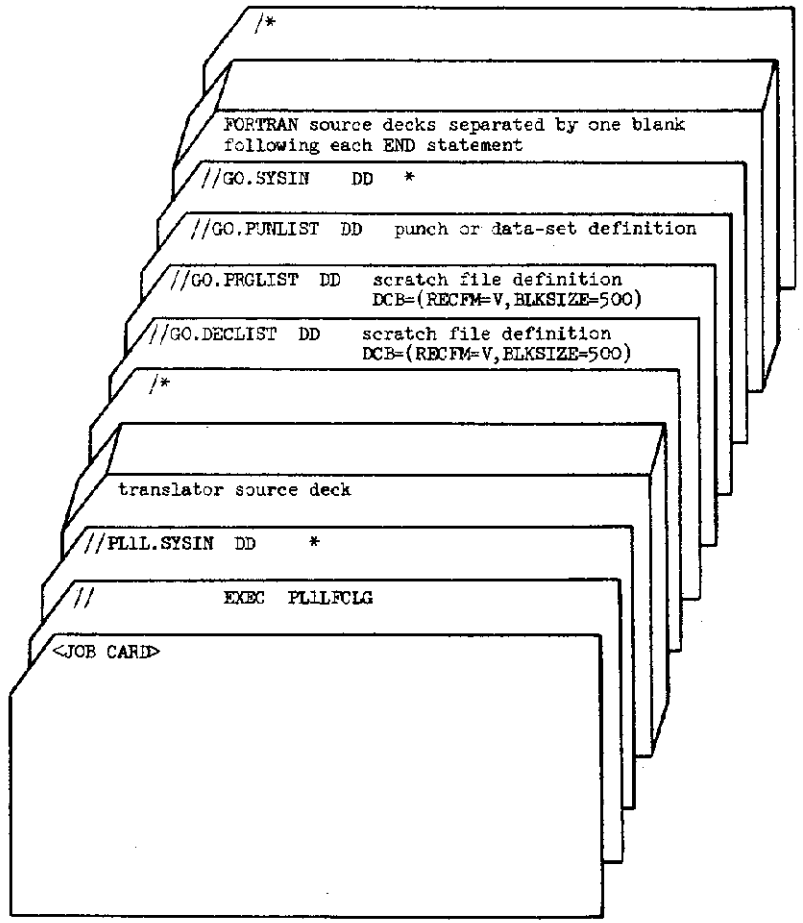
(3) FORTRAN binary input and output statements (unformatted) are translated into GET and PUT LIST statements which are unformatted but not binary in the same sense as in FORTRAN. If this is not acceptable, hand translation to RECORD input and output statements is required.

(4) Input FORMAT lists for card image input must be of length 80 characters (columns) in order to use STREAM input and get the same effect as in FORTRAN. Example one requires this modification in order to execute correctly.

(5) Variables which were in COMMON in the FORTRAN programs and were declared external in the PL/1 version must be checked to be sure that different names for the same location are not used in separate subprograms.

(6) Blanks are significant and FORTRAN is assumed to have reserved words. Any violation of this simplification must be hand translated.

SAMPLE JOB CONTROL LANGUAGE FOR EXECUTION OF THE TRANSLATOR



NOTE: PL1LFCLG is the standard IBM cataloged procedure for PL/1 compile, link, and go.

EXAMPLE 1

FORTRAN SOURCE PROGRAM

```
C       SOLUTION OF SIMULTANEOUS EQUATIONS BY GAUSSIAN ELIMINATION
C
100     FORMAT (I5)
101     FORMAT (8F10.2)
102     FORMAT (I5,F20.2)
        DIMENSION A(50,50),Y(50),X(50)
        READ (5,100) N
        READ (5,101) ((A(I,J),J=1,N),I=1,N)
        READ (5,101) (Y(I),I=1,N)
        M=N-1
        DO 10 I=1,M
        L=I+1
        DO 10 J=L,N
        IF (A(J,I)) 6,10,6
6       DO 8 K=L,N
8       A(J,K)=A(J,K)-A(I,K)*A(J,I)/A(I,I)
        Y(J)=Y(J)-Y(I)*A(J,I)/A(I,I)
10      CONTINUE
        X(N)=Y(N)/A(N,N)
        WRITE (6,102) N,X(N)
        DO 30 I=1,M
        K=N-I
        L=K+1
        DO 20 J=L,N
20      Y(K)=Y(K)-X(J)*A(K,J)
        X(K)=Y(K)/A(K,K)
30      WRITE (6,102) K,X(K)
        RETURN
        END
```


EXAMPLE 1 (CONTINUED)

PL/1 VERSION OF FORTRAN PROGRAM

```
/* FORTRAN PROGRAM TRANSLATED TO PL/1 */
FORT: PROCEDURE OPTIONS (MAIN);
   DECLARE A(50,50);
   DECLARE Y(50);
   DECLARE X(50);
   /* SOLUTION OF SIMULTANEOUS EQUATIONS BY GAUSSIAN ELIMINATION */

   #100: FORMAT (F(5));
   #101: FORMAT (8F(10,2));
   #102: FORMAT (F(5),F(20,2));
   GET FILE (SYSIN)EDIT(N)(R(#100));
   GET FILE (SYSIN)EDIT(((A(I,J) DO J=1 TO N BY 1) DO I=1 TO N BY
    1))(R(#101));
   GET FILE (SYSIN)EDIT((Y(I) DO I=1 TO N BY 1))(R(#101));
   M= N - 1;
   DO I=1 TO M BY 1;
      L= I + 1;
      DO J=L TO N BY 1;
         IF (A(J,I)) > 0 THEN GO TO #6;
         ELSE IF (A(J,I)) = 0 THEN GO TO #10;
         ELSE GO TO #6;
         #6: DO K=L TO N BY 1;
            #8: A(J,K)= A(J,K) - A(I,K)*A(J,I)/A(I,I);
            END;
         Y(J)= Y(J) - Y(I)*A(J,I)/A(I,I);
         #10:
         END;
      END;
   X(N)= Y(N)/A(N,N);
   PUT FILE (SYSPRINT)EDIT(N,X(N))(R(#102));
   DO I=1 TO M BY 1;
      K= N - I;
      L= K + 1;
      DO J=L TO N BY 1;
         #20: Y(K)= Y(K) - X(J)*A(K,J);
         END;
      X(K)= Y(K)/A(K,K);
      #30: PUT FILE (SYSPRINT)EDIT(K,X(K))(R(#102));
      END;
   RETURN;
END;
```

EXAMPLE 2

FORTRAN SOURCE PROGRAM

```
      SUBROUTINE POLAR (X,Y,R,THETA)
C     CONVERT CARTESIAN TO POLAR COORDINATES
      R=SQRT(X*X+Y*Y)
      THETA=ATAN2(Y,X)
      RETURN
      END
```

PL/1 VERSION OF FORTRAN PROGRAM

```
/* FORTRAN PROGRAM TRANSLATED TO PL/1 */
POLAR:PROCEDURE(X,Y,R,THETA);
   /* CONVERT CARTESIAN TO POLAR COORDINATES */
   R= SQRT(X*X + Y*Y);
   THETA= ATAN(Y,X);
   RETURN;
END;
```


EXAMPLE 3

FORTRAN SOURCE PROGRAM

```
      FUNCTION HELP (A,B,C)
      IF (A.GT.B) GO TO 500
      HELP=B*C
      RETURN
500   HELP=A*C
      RETURN
      END
```

PL/1 VERSION OF FORTRAN PROGRAM

```
/* FORTRAN PROGRAM TRANSLATED TO PL/1 */
HELP: PROCEDURE (A,B,C);
   IF (A > B) THEN
   GO TO #500;
   HELP#= B*C;
   RETURN (HELP#);
   #500: HELP#= A*C;
   RETURN (HELP#);
END;
```

EXAMPLE 2

FORTRAN SOURCE PROGRAM

```
      SUBROUTINE POLAR (X,Y,R,THETA)
C     CONVERT CARTESIAN TO POLAR COORDINATES
      R=SQRT(X*X+Y*Y)
      THETA=ATAN2(Y,X)
      RETURN
      END
```

PL/1 VERSION OF FORTRAN PROGRAM

```
/* FORTRAN PROGRAM TRANSLATED TO PL/1 */
POLAR:PROCEDURE(X,Y,R,THETA);
   /* CONVERT CARTESIAN TO POLAR COORDINATES */
   R= SQRT(X*X + Y*Y);
   THETA= ATAN(Y,X);
   RETURN;
END;
```


EXAMPLE 3

FORTRAN SOURCE PROGRAM

```
      FUNCTION HELP (A,B,C)
      IF (A.GT.B) GO TO 500
      HELP=B*C
      RETURN
500   HELP=A*C
      RETURN
      END
```

PL/1 VERSION OF FORTRAN PROGRAM

```
/* FORTRAN PROGRAM TRANSLATED TO PL/1 */
HELP: PROCEDURE (A,B,C);
   IF (A > B) THEN
   GO TO #500;
   HELP#= B*C;
   RETURN (HELP#);
   #500: HELP#= A*C;
   RETURN (HELP#);
END;
```

PL/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--

```
     LOAD,NODECK
```

THE COMPLETE LIST OF OPTIONS USED DURING THIS COMPILATION IS--

```
  EBCDIC
  CHAR60
NOMACRO
  SOURCE2
  COMP
  SOURCE
NOATR
NOXREF
NOEXTREF
NOLIST
  LOAD
NODECK
  FLAGW
NOSTMT
  SIZE=413696
  LINECNT=057
  OPT=00
  SORMGIN=(002,072)
```



```
/* FORTRAN TO PL/1 TRANSLATOR
   BY LANSE M LEACH
      COMPUTER SCIENCE DEPARTMENT
      STANFORD UNIVERSITY, STANFORD, CALIFORNIA
      AUGUST 1,1967

   TRANSLATION IS FROM THE ASA FORTRAN STANDARDIZATION SPECIFICATIONS
   PUBLISHED IN THE 'COMMUNICATIONS OF THE ACM', OCTOBER 1964
   INTO PL/1 (F LEVEL)  AS DESCRIBED IN 'IBM SYSTEM/360 OPERATING
   SYSTEM PL/1 LANGUAGE SPECIFICATIONS', FORM C28-6571.

   THE TRANSLATOR IS WRITTEN IN IBM PL/1 (F LEVEL) USING THE METHOD
   OF RECURSIVE DESCENT AND PL/1 CHARACTER STRINGS AS THE MAJOR TOOLS
   FOR THE TRANSLATION. OUTPUT IS BOTH PRINTED AND PUNCHED.
   DEBUGGING AND TESTING WAS ACCOMPLISHED ON THE IBM SYSTEM/360 AT
   STANFORD UNIVERSITY.

   A COMPLETE DESCRIPTION OF THE TRANSLATOR HAS BEEN PUBLISHED IN
   STANFORD COMPUTATION CENTER REPORT NUMBER 33-70-21,
   STANFORD UNIVERSITY, STANFORD, CALIFORNIA */

1  F#TO#P: PROCEDURE OPTIONS (MAIN);

2     DECLARE (DECLIST,PRGLIST) FILE STREAM;
3     DECLARE PUNLIST FILE STREAM PRINT;
4     DECLARE DISK ENTRY (CHARACTER (200) VARYING );
5     DECLARE (EXPRESSION,LOGFAC,LOGNEG,LOGPRIM,ARITHEXPR,TERM,FACTOR,
      NUMBER,PRIMARY) ENTRY RETURNS (CHARACTER (200) VARYING);
6     DECLARE (LABCOUNT,SYMLENGTH,DOPOINT) FIXED BINARY (15,0);
7     DECLARE EMPTY CHARACTER (71) INITIAL ((71)' ');
8     DECLARE TAB CHARACTER (30) VARYING;
9     DECLARE (LEM,LEN,LAB,FUNCTNAME,LABSTR) CHARACTER (10) VARYING;
10    DECLARE SYMBOL(200) CHARACTER (8) VARYING;
11    DECLARE SYMDIM(200) CHARACTER(20) VARYING;
12    DECLARE SYNTYPE(200) CHARACTER (22) VARYING;
13    DECLARE SYMCOM(200) CHARACTER (30) VARYING;
14    DECLARE (UNIT,LE,FMT#) CHARACTER (12) VARYING;
15    DECLARE DOCOUNT(50) CHARACTER (6) VARYING;
16    DECLARE (WORD,LINE,IOSTRING,VARSTRING) CHARACTER (500) VARYING;
17    DECLARE (OUTPUTLINE,NEXT) CHARACTER (72) VARYING;
18    DECLARE (COL1,COL6,LOGIC) CHARACTER (1);
19    DECLARE COL25 CHARACTER (4),COL772 CHARACTER (66);
20    DECLARE DUMMY CHARACTER (8);
21    DECLARE (ARITHFUNC,NEWCARD,FUNCT) BIT (1);
22    DECLARE BUILTIN(93) CHARACTER (10) VARYING STATIC INITIAL
      ('ABS','IABS','DABS','AINT','INT','IDINT','AMOD','MOD','AMAX0',
      'AMAX1','MAX0','MAX1','DMAX1','AMIN0','AMIN1','MIN0','MIN1','DMIN1',
      'FLOAT','IFIX','SIGN','ISIGN','DSIGN','SNGL','REAL','AIMAG','DBLE',
      'CMPLX','CONJG','EXP','DEXP','CEXP','ALOG','DLOG','CLOG','ALOG10',
      'DLOG10','SIN','DSIN','CSIN','COS','DCOS','CCOS','TANH','SQRT',
      'DSQRT','CSQRT','ATAN','DATAN','ATAN2','DATAN2','DMOD','CABS');
23    DECLARE CBUILTIN (53) CHARACTER (10) VARYING STATIC INITIAL
```

```
          ((3)(1)'ABS',(3)(1)'TRUNC',(2)(1)'MOD',(5)(1)'MAX',(5)(1)'MIN',
          'FLOAT','FIXED',(3)(1)'SIGN',"','REAL','IMAG','FLOAT','COMPLEX',
          'CONJ',(3)(1)'EXP',(3)(1)'LOG',(2)(1)'LOG10',(3)(1)'SIN',
          (3)(1)'COS','TAN',(3)(1)'SQRT',(4)(1)'ATAN','MOD','ABS');

24        PROGRAM: PROCEDURE;
             /* INITIALIZATION AND CONTOL OF ONE PROGRAM OR SUBPROGRAM
                TRANSLATION */
25           PUT FILE (DECLIST) LIST (' /* FORTRAN PROGRAM TRANSLATED TO '
             ||'PL/1 */');
26           PUT FILE (SYSPRINT) LIST ('FORTRAN SOURCE PROGRAM') PAGE;
27           PUT FILE (SYSPRINT) SKIP(3);
28           OUTPUTLINE,LINE='';
29           LABCOUNT=0;
30           SYMLENGTH,DOPOINT,LAC=0;
31           FUNCT,NEWCARD='0'B;
32           TAB='    ';
33           SYMDIM,SYMCOM,SYMTYPE='';
34           CALL SCAN('1'B);
35           IF NEXT¬='FUNCTION' & NEXT¬='SUBROUTINE' THEN
36              PUT FILE (DECLIST) LIST (' FORT: PROCEDURE OPTIONS (MAIN);');
37           LOOP: DO WHILE (NEXT ¬= 'END');
38           CALL STATEMENT;
39           END LOOP;
40           PUT FILE (PRGLIST) LIST (' END;');
41           CALL OUTPUT;
42           END PROGRAM;

43        STATEMENT: PROCEDURE RECURSIVE;
             /* PASSES CONTROL TO APPROPRIATE TRANSLATING PROCEDURE */
44           IF NEXT='DIMENSION' THEN CALL DIMENSION; ELSE
46           IF NEXT='COMMON' THEN CALL COMMON; ELSE
48           IF NEXT='EQUIVALENCE' THEN CALL EQUIVALENCE; ELSE
50           IF NEXT='SUBROUTINE' THEN CALL SUBROUTINE; ELSE
52           IF NEXT='ENTRY' THEN CALL ENTRY; ELSE
54           IF NEXT='READ' THEN CALL READWRITE('GET'); ELSE
56           IF NEXT='WRITE' THEN CALL READWRITE('PUT'); ELSE
58           IF NEXT='COMPLEX' THEN CALL TYP (' COMPLEX FLOAT BINARY '); ELSE
60           IF NEXT='ENDFILE' THEN CALL REWIND; ELSE
62           IF NEXT='REWIND' THEN CALL REWIND; ELSE
64           IF NEXT='BACKSPACE' THEN CALL BACKSPACE; ELSE
66           IF NEXT='LOGICAL' THEN CALL TYP (' BIT (1) '); ELSE
68           IF NEXT='GO' THEN CALL GO; ELSE
70           IF NEXT='ASSIGN' THEN CALL ASSIGN; ELSE
72           IF NEXT='IF' THEN CALL IF; ELSE
74           IF NEXT='DO' THEN CALL DO; ELSE
76           IF NEXT='STOP' THEN CALL STOP; ELSE
78           IF NEXT='PAUSE' THEN CALL PAUSE; ELSE
80           IF NEXT='FORMAT' THEN CALL FORMAT; ELSE
82           IF NEXT='CONTINUE' THEN CALL CONTINUE; ELSE
84           IF NEXT='CALL' THEN CALL CALL; ELSE
86           IF NEXT='RETURN' THEN CALL RETURN; ELSE
98           IF NEXT='FUNCTION' THEN CALL FUNCTION (''); ELSE
90           IF NEXT='REAL' THEN CALL TYP (' FLOAT BINARY '); ELSE
```



```
92           IF NEXT='DOUBLE' THEN CALL DOUBLE; ELSE
94           IF NEXT='INTEGER' THEN CALL TYP (' FIXED BINARY '); ELSE
96           IF NEXT='DATA' THEN CALL DATA; ELSE
98           IF NEXT='EXTERNAL' THEN CALL EXTERNAL; ELSE
100          CALL ASSIGNMENT;
101          END STATEMENT;

102       SCAN: PROCEDURE (W);
             /* SCAN RETURNS EITHER A STRING OF LETTERS, STRING OF DIGITS,
                OR A SPECIAL CHARACTER.        SCAN ALSO PROCESSES COMMENT
                CARDS AND LABELS */
103          DECLARE W BIT(1);
104          ICOUNT=0;
105          NEXT='';
106          NEW: IF W='1'B THEN DO;
108             DO I=1 TO LABCOUNT;
109                CALL DISK ('END;');
110                TAB=SUBSTR(TAB,4);
111             END;
112             LABCOUNT=0;
113             IF NEWCARD='0'B THEN CALL CARD;
115             LINE=COL1:72;
116             IF COL1='C' THEN DO;
118                CALL COMMENT;
119                NEWCARD='0'B;
120                GO TO NEW;
121             END;
122             IF COL1¬=' ' | COL2:5¬='     ' THEN CALL LABEL;
124             ELSE LAB='';
125             NEW1: CALL CARD;
126             IF COL6¬='0' & COL6¬=' ' & COL1¬='C' THEN DO;
128                LINE=LINE||COL7:72;
129                GO TO NEW1;
130             END;
131             NEWCARD='1'B;
132          END;
133          BLANK: IF SUBSTR(LINE,1,1)=' ' THEN DO;
135             LINE=SUBSTR(LINE,2);
136             IF LINE='' THEN RETURN;
138             GO TO BLANK;
139          END;
140          IF LINE='' THEN RETURN;
142          IF SUBSTR(LINE,1,1)>='0' THEN GO TO NUM;
144          IF SUBSTR(LINE,1,1)>='A' THEN GO TO STR;
146          NEXT=SUBSTR(LINE,1,1);
147          LINE=SUBSTR(LINE,2);
148          RETURN;
149          NUM: ICOUNT=ICOUNT+1;
150          IF SUBSTR(LINE,ICOUNT+1,1)>='0' THEN GO TO NUM;
152          NEXT=SUBSTR(LINE,1,ICOUNT);
153          LINE=SUBSTR(LINE,ICOUNT+1);
154          RETURN;
155          STR: ICOUNT=ICOUNT+1;
156          IF SUBSTR(LINE,ICOUNT+1,1)>='A' & SUBSTR(LINE,ICOUNT+1,1)<'0'
```

```
157                     THEN GO TO STR;
158                     NEXT=SUBSTR(LINE,1,ICOUNT);
159                     LINE=SUBSTR(LINE,ICOUNT+1);
160                     END SCAN;

161                  ERROR: PROCEDURE (MESSAGE);
                        /* PRINTS OUT ERROR MESSAGES IN THE FORTRAN LISTING */
162                     DECLARE MESSAGE CHARACTER (60) VARYING;
163                     PUT FILE (SYSPRINT) LIST ('**********'||MESSAGE||
                        '**********') SKIP;
164                     END ERROR;

165                  OUTPUT: PROCEDURE;
                        /* CONTROLS PL/I LISTING AND CARD PUNCHING */
166                     PUT FILE (SYSPRINT) LIST('PL/I VERSION OF FORTRAN PROGRAM')
                        PAGE;
167                     PUT FILE (SYSPRINT) SKIP(3);
168                     DO I=1 TO SYMLENGTH;
169                     PUT FILE (DECLIST) LIST ('    DECLARE '||SYMBOL(I)||
                        SYMDIM(I)||SYMTYPE(I)||SYMCOM(I)||';');
170                     END;
171                     ON ENDFILE (DECLIST) GO TO LABEL2;
173                     ON ENDFILE (PRGLIST) GO TO THATSALL;
175                     ON CONVERSION GO TO THATSALL;
177                     CLOSE FILE (DECLIST);
178                     CLOSE FILE (PRGLIST);
179                     LABEL1: GET FILE (DECLIST) LIST (WORD);
180                     PUT FILE (SYSPRINT) LIST (WORD) SKIP;
181                     PUT FILE (PUNLIST) LIST (WORD) SKIP;
182                     GO TO LABEL1;
183                     LABEL2: GET FILE (PRGLIST) LIST (WORD);
184                     PUT FILE (SYSPRINT) LIST (WORD) SKIP;
185                     PUT FILE (PUNLIST) LIST (WORD) SKIP;
186                     GO TO LABEL2;
187                     THATSALL: CLOSE FILE (DECLIST);
188                     CLOSE FILE (PRGLIST);
189                     END OUTPUT;

190                  CARD: PROCEDURE;
                        /* READS A CARD FROM THE INPUT STREAM */
191                     PUT FILE (SYSPRINT) LIST (OUTPUTLINE) SKIP;
192                     GET FILE (SYSIN) EDIT (COL1,COL25,COL6,COL72,DUMMY)
                        (A(1),A(4),A(1),A(66),A(8));
193                     OUTPUTLINE=COL1||COL25||COL6||COL72;
194                     END CARD;

195                  DISK: PROCEDURE (D);
                        /* TRANSFERS VARYING LENGTH CHARACTER STRINGS TO TEMPORARY FILE
                           ON A 2311 DISK */
196                     DECLARE D CHARACTER (200) VARYING;
197                     OGO: D=TAB||D;
198                     IF LENGTH(D) < 72 THEN DO;
200                        PUT FILE (PRGLIST) LIST (D);
201                        RETURN; END;
```



```
203                     DO I=72 BY -1 TO 1;
204                     IF SUBSTR(D,I,1)=' ' | SUBSTR(D,I,1)=',' THEN GO TO OGG;
206                        END;
207                     I=72;
208                     OGG: PUT FILE (PRGLIST) LIST (SUBSTR(D,1,I-1));
209                     D=SUBSTR(D,I);
210                     GO TO OGO;
211                     END DISK;

212                  COMMENT: PROCEDURE;
                        /* COMMENT PROCESSING -- BLANKS ARE REMOVED FROM THE HEAD AND
                           TAIL OF ALL COMMENTS FOR MORE PRESENTABLE OUTPUT */
213                     IOSTRING=COL25||COL6||COL72;
214                     IF IOSTRING=EMPTY THEN DO;
216                        CALL DISK ('    ');
217                        RETURN;
218                     END;
219                     DO I=71 BY -1 TO 1;
220                        IF SUBSTR(IOSTRING,I,1)^=' ' THEN GO TO OUTP1;
222                        IOSTRING=SUBSTR(IOSTRING,1,I-1);
223                     END;
224                     OUTP1: DO WHILE (SUBSTR(IOSTRING,1,1)=' ');
225                        IOSTRING=SUBSTR(IOSTRING,2);
226                     END;
227                     CALL DISK ('/* '||IOSTRING||' */');
228                     END COMMENT;

229                  ASSIGNMENT: PROCEDURE;
                        /* FORTRAN ASSIGNMENT STATEMENT */
230                     IF FUNCT='1'B & FUNCTNAME=NEXT THEN NEXT=NEXT||'#';
232                     CALL VARIABLE('1'B);
                        /* TEST FOR SIMPLE ARITHMETIC STATEMENT FUNCTION */
233                     IF ARITHFUNC='1'B THEN RETURN;
235                     WORD=VARSTRING;
236                     IF NEXT^='=' THEN DO;
238                        CALL ERROR ('UNRECOGNIZABLE FORTRAN STATEMENT');
239                        CALL SCAN ('1'B);
240                        RETURN;
241                     END;
242                     CALL SCAN ('0'B);
243                     WORD=WORD||'= '||EXPRESSION;
244                     CALL DISK (LAB||WORD||';');
245                     CALL SCAN ('1'B);
246                     END ASSIGNMENT;

247                  EXPRESSION: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
                        /* EXPRESSION AND THE FOLLOWING SIX PROCEDURES PROCESS
                           FORTRAN EXPRESSIONS  */
248                     DECLARE T CHARACTER (200) VARYING;
249                     T=LOGFAC;
250                     DO WHILE (NEXT=',');
251                        CALL SCAN ('0'B);
252                        IF NEXT^='OR' THEN DO;
254                           LINE=NEXT||LINE;
```

```
255                        NEXT='.';
256                        RETURN (T);
257                        END;
258                     CALL SCAN ('0'B);
259                     CALL SCAN ('0'B);
260                     T=T||' | '||LOGFAC;
261                     END;
262                  RETURN (T);
263                  END EXPRESSION;

264               LOGFAC: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
265                  DECLARE T CHARACTER (200) VARYING;
266                  T=LOGNEG;
267                  DO WHILE (NEXT='.');
268                     CALL SCAN ('0'B);
269                     IF NEXT¬='AND' THEN DO;
271                        LINE=NEXT||LINE;
272                        NEXT='.';
273                        RETURN (T);
274                        END;
275                     CALL SCAN ('0'B);
276                     CALL SCAN ('0'B);
277                     T=T||' & '||LOGNEG;
278                     END;
279                  RETURN (T);
280                  END LOGFAC;

281               LOGNEG: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
282                  DECLARE T CHARACTER (200) VARYING;
283                  IF NEXT¬='.' THEN RETURN (LOGPRIM);
285                  CALL SCAN ('0'B);
286                  IF NEXT¬='NOT' THEN DO;
288                     LINE=NEXT||LINE;
289                     NEXT='.';
290                     RETURN (LOGPRIM);
291                     END;
292                  CALL SCAN ('0'B);
293                  CALL SCAN ('0'B);
294                  RETURN ('¬ ('||LOGPRIM||')');
295                  END LOGNEG;

296               LOGPRIM: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
297                  DECLARE T CHARACTER (200) VARYING;
298                  DECLARE OP CHARACTER (2) VARYING;
299                  T=ARITHEXPR;
300                  DO WHILE (NEXT='.');
301                     CALL SCAN ('0'B);
302                     IF NEXT='GT' THEN OP='>'; ELSE
304                     IF NEXT='GE' THEN OP='>='; ELSE
306                     IF NEXT='LT' THEN OP='<'; ELSE
308                     IF NEXT='LE' THEN OP='<='; ELSE
310                     IF NEXT='EQ' THEN OP='='; ELSE
312                     IF NEXT='NE' THEN OP='¬='; ELSE DO;
315                        LINE=NEXT||LINE;
```



```
316                        NEXT='.';
317                        RETURN (T); END;
319                     CALL SCAN ('0'B);
320                     CALL SCAN ('0'B);
321                     T=T||' '||OP||' '||ARITHEXPR;
322                     END;
323                  RETURN (T);
324                  END LOGPRIM;

325               ARITHEXPR: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
326                  DECLARE T CHARACTER (200) VARYING;
327                  IF NEXT='+' | NEXT='-' THEN DO;
329                     T=NEXT;
330                     CALL SCAN ('0'B);
331                  END;
332                  ELSE T='';
333                  T=T||TERM;
334                  DO WHILE (NEXT='+' | NEXT='-');
335                     T=T||' '||NEXT||' ';
336                     CALL SCAN ('0'B);
337                  T=T||TERM;
338                     END;
339                  RETURN (T);
340                  END ARITHEXPR;

341               TERM: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
342                  DECLARE T CHARACTER (200) VARYING;
343                  T=FACTOR;
344                  DO WHILE (NEXT='*' | NEXT='/');
345                     T=T||NEXT;
346                     CALL SCAN ('0'B);
347                     T=T||FACTOR;
348                     END;
349                  RETURN (T);
350                  END TERM;

351               FACTOR: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
352                  DECLARE T CHARACTER (200) VARYING;
353                  T=PRIMARY;
354                  DO WHILE (NEXT='*' & SUBSTR(LINE,1,1)='*');
355                     T='('||T||'**';
356                     CALL SCAN ('0'B);
357                     CALL SCAN ('0'B);
358                     T=T||PRIMARY||')';
359                     END;
360                  RETURN (T);
361                  END FACTOR;

362               PRIMARY: PROCEDURE CHARACTER (200) VARYING RECURSIVE;
363                  DECLARE T CHARACTER (200) VARYING;
364                  IF NEXT='(' THEN DO;
366                     CALL SCAN ('0'B);
367                     T='('||EXPRESSION||')';
368                     CALL SCAN ('0'B);
```

```
369                     RETURN (T); END;
371                  IF NEXT<'0' & NEXT¬='.' THEN DO;
373                     CALL VARIABLE ('0'B);
374                     RETURN (VARSTRING); END;
376                  IF NEXT¬='.' THEN RETURN (NUMBER); ELSE DO;
379                     CALL SCAN ('0'B);
380                     IF NEXT¬='TRUE' & NEXT¬='FALSE' THEN DO;
382                        LINE=NEXT||LINE;
383                        NEXT='.';
384                        RETURN (NUMBER);
385                        END;
386                     ELSE DO;
387                        IF NEXT='TRUE' THEN LOGIC='1'; ELSE LOGIC='0';
390                        CALL SCAN ('0'B);
391                        CALL SCAN ('0'B);
392                        RETURN ('''||LOGIC||'''B');
393                        END;
394                     END;
395               END PRIMARY;

396            NUMBER: PROCEDURE CHARACTER (200) VARYING;
                  /* PROCESSES BOTH FIXED AND FLOATING POINT NUMBERS FOR
                     EXPRESSIONS */
397               DECLARE NUMSTRING CHARACTER (200) VARYING;
398               IF NEXT='.' THEN DO;
400                  NUMSTRING=NEXT;
401                  CALL SCAN ('0'B);
402                  NUMSTRING=NUMSTRING||NEXT;
403                  CALL SCAN ('0'B);
404                  GO TO EXP;
405                  END;
406               NUMSTRING=NEXT;
407               CALL SCAN ('0'B);
408               IF NEXT¬='.' THEN RETURN (NUMSTRING);
410               CALL SCAN ('0'B);
411               IF NEXT='GT' | NEXT='GE' | NEXT='EQ' | NEXT='LT' | NEXT='LE'
                  | NEXT='TRUE' | NEXT='FALSE'
412               | NEXT='NE' | NEXT='AND' | NEXT='OR' | NEXT='NOT' THEN DO;
413                  LINE=NEXT||LINE;
414                  NEXT='.';
415                  RETURN (NUMSTRING);
416                  END;
417               NUMSTRING=NUMSTRING||'.';
418               IF SUBSTR(NEXT,1,1)>='0' THEN DO;
420                  NUMSTRING=NUMSTRING||NEXT;
421                  CALL SCAN ('0'B);
422                  END;
423               EXP: IF NEXT¬='D' & NEXT¬='E' THEN RETURN (NUMSTRING);
425               NUMSTRING=NUMSTRING||'E';
426               CALL SCAN ('0'B);
427               IF NEXT='+' | NEXT='-' THEN DO;
429                  NUMSTRING=NUMSTRING||NEXT;
430                  CALL SCAN ('0'B);
431                  END;
```



```
432               NUMSTRING=NUMSTRING||NEXT;
433               CALL SCAN ('0'B);
434               RETURN (NUMSTRING);
435               END NUMBER;

436            VARIABLE: PROCEDURE (W) RECURSIVE;
                  /* VARIABLE PROCESSING TO INCLUDE BUILTIN FUNCTION CHECKING
                     AND SIMPLE ARITHMETIC FUNCTION STATEMENT CHECKING */
437               DECLARE BFUNCT CHARACTER (200) VARYING;
438               DECLARE (W,DB) BIT (1);
439               CALL FORYIDEN;
440               VARSTRING=NEXT;
441               CALL SCAN ('0'B);
442               IF NEXT¬='(' THEN RETURN;
444               IF W='1'B THEN DO;
                     /* TEST FOR SIMPLE ARITHMETIC STATEMENT FUNCTION */
446                  DO I=1 TO SYMLENGTH;
447                     IF VARSTRING=SYMBOL(I) THEN GO TO NOF;
449                     END;
450                  ARITHFUNC='1'B;
451                  CALL ARGLIST;
452                  CALL DISK (VARSTRING||': PROCEDURE ('||WORD||');');
453                  TAB=TAB||'   ';
454                  CALL SCAN ('0'B);
455                  CALL SCAN ('0'B);
456                  WORD=EXPRESSION;
457                  CALL DISK ('RETURN ('||WORD||');');
458                  CALL DISK ('END;');
459                  TAB=SUBSTR(TAB,4);
460                  CALL SCAN ('1'B);
461                  RETURN;
462                  END;
463               ELSE DO;
                     /* TEST FOR BUILT IN FUNCTION */
464                  IF VARSTRING='DBLE' THEN DB='1'B; ELSE DB='0'B;
467                  DO I=1 TO SYMLENGTH;
468                     IF VARSTRING=SYMBOL(I) THEN GO TO NOF;
470                     END;
471                  DO I=1 TO 53;
472                     IF VARSTRING=BUILTIN(I) THEN VARSTRING=CBUILTIN(I);
474                     END;
475                  BFUNCT=VARSTRING;
476                  DO WHILE (NEXT¬=')');
477                     BFUNCT=BFUNCT||NEXT;
478                     CALL SCAN ('0'B);
479                     BFUNCT=BFUNCT||EXPRESSION;
480                     END;
481               CALL SCAN ('0'B);
482               IF DB='1'B THEN VARSTRING=BFUNCT||',53)';
484               ELSE VARSTRING=BFUNCT||')';
485               RETURN;
486               END;
487               NOF: BFUNCT=VARSTRING;
488               DO WHILE (NEXT¬=')');
```

```
369                     RETURN (T); END;
371                  IF NEXT<'0' & NEXT¬='.' THEN DO;
373                     CALL VARIABLE ('0'B);
374                     RETURN (VARSTRING); END;
376                  IF NEXT¬='.' THEN RETURN (NUMBER); ELSE DO;
379                     CALL SCAN ('0'B);
380                     IF NEXT¬='TRUE' & NEXT¬='FALSE' THEN DO;
382                        LINE=NEXT||LINE;
383                        NEXT='.';
384                        RETURN (NUMBER);
385                        END;
386                     ELSE DO;
387                        IF NEXT='TRUE' THEN LOGIC='1'; ELSE LOGIC='0';
390                        CALL SCAN ('0'B);
391                        CALL SCAN ('0'B);
392                        RETURN ('''||LOGIC||'''B');
393                        END;
394                     END;
395               END PRIMARY;

396            NUMBER: PROCEDURE CHARACTER (200) VARYING;
                  /* PROCESSES BOTH FIXED AND FLOATING POINT NUMBERS FOR
                     EXPRESSIONS */
397               DECLARE NUMSTRING CHARACTER (200) VARYING;
398               IF NEXT='.' THEN DO;
400                  NUMSTRING=NEXT;
401                  CALL SCAN ('0'B);
402                  NUMSTRING=NUMSTRING||NEXT;
403                  CALL SCAN ('0'B);
404                  GO TO EXP;
405                  END;
406               NUMSTRING=NEXT;
407               CALL SCAN ('0'B);
408               IF NEXT¬='.' THEN RETURN (NUMSTRING);
410               CALL SCAN ('0'B);
411               IF NEXT='GT' | NEXT='GE' | NEXT='EQ' | NEXT='LT' | NEXT='LE'
                  | NEXT='TRUE' | NEXT='FALSE'
412               | NEXT='NE' | NEXT='AND' | NEXT='OR' | NEXT='NOT' THEN DO;
413                  LINE=NEXT||LINE;
414                  NEXT='.';
415                  RETURN (NUMSTRING);
416                  END;
417               NUMSTRING=NUMSTRING||'.';
418               IF SUBSTR(NEXT,1,1)>='0' THEN DO;
420                  NUMSTRING=NUMSTRING||NEXT;
421                  CALL SCAN ('0'B);
422                  END;
423               EXP: IF NEXT¬='D' & NEXT¬='E' THEN RETURN (NUMSTRING);
425               NUMSTRING=NUMSTRING||'E';
426               CALL SCAN ('0'B);
427               IF NEXT='+' | NEXT='-' THEN DO;
429                  NUMSTRING=NUMSTRING||NEXT;
430                  CALL SCAN ('0'B);
431                  END;
```



```
432               NUMSTRING=NUMSTRING||NEXT;
433               CALL SCAN ('0'B);
434               RETURN (NUMSTRING);
435               END NUMBER;

436            VARIABLE: PROCEDURE (W) RECURSIVE;
                  /* VARIABLE PROCESSING TO INCLUDE BUILTIN FUNCTION CHECKING
                     AND SIMPLE ARITHMETIC FUNCTION STATEMENT CHECKING */
437               DECLARE BFUNCT CHARACTER (200) VARYING;
438               DECLARE (W,DB) BIT (1);
439               CALL FORTIDEN;
440               VARSTRING=NEXT;
441               CALL SCAN ('0'B);
442               IF NEXT¬='(' THEN RETURN;
444               IF W='1'B THEN DO;
                     /* TEST FOR SIMPLE ARITHMETIC STATEMENT FUNCTION */
446                  DO I=1 TO SYMLENGTH;
447                     IF VARSTRING=SYMBOL(I) THEN GO TO NOF;
449                     END;
450                  ARITHFUNC='1'B;
451                  CALL ARGLIST;
452                  CALL DISK (VARSTRING||': PROCEDURE ('||WORD||');');
453                  TAB=TAB||'   ';
454                  CALL SCAN ('0'B);
455                  CALL SCAN ('0'B);
456                  WORD=EXPRESSION;
457                  CALL DISK ('RETURN ('||WORD||');');
458                  CALL DISK ('END;');
459                  TAB=SUBSTR(TAB,4);
460                  CALL SCAN ('1'B);
461                  RETURN;
462                  END;
463               ELSE DO;
                     /* TEST FOR BUILT IN FUNCTION */
464                  IF VARSTRING='DBLE' THEN DB='1'B; ELSE DB='0'B;
467                  DO I=1 TO SYMLENGTH;
468                     IF VARSTRING=SYMBOL(I) THEN GO TO NOF;
470                     END;
471                  DO I=1 TO 53;
472                     IF VARSTRING=BUILTIN(I) THEN VARSTRING=CBUILTIN(I);
474                     END;
475                  BFUNCT=VARSTRING;
476                  DO WHILE (NEXT¬=')');
477                     BFUNCT=BFUNCT||NEXT;
478                     CALL SCAN ('0'B);
479                     BFUNCT=BFUNCT||EXPRESSION;
480                     END;
481                  CALL SCAN ('0'B);
482                  IF DB='1'B THEN VARSTRING=BFUNCT||',53)';
484                  ELSE VARSTRING=BFUNCT||')';
485                  RETURN;
486                  END;
487               NOF: BFUNCT=VARSTRING;
488               DO WHILE (NEXT¬='('');
```

```
489                     BFUNCT=BFUNCT||NEXT;
490                     CALL SCAN ('0'B);
491                     BFUNCT=BFUNCT||EXPRESSION;
492                     END;
493                  VARSTRING=BFUNCT||')';
494                  CALL SCAN ('0'B);
495                  END VARIABLE;

496               FORTIDEN: PROCEDURE;
                     /* PACKS CHARACTERS TO FORM VALID FORTRAN IDENTIFIERS */
497                  FUN: IF SUBSTR(LINE,1,1)>='A' THEN DO;
499                     NEXT=NEXT||SUBSTR(LINE,1,1);
500                     LINE=SUBSTR(LINE,2);
501                     GO TO FUN;
502                     END;
503                  END FORTIDEN;

504               DIMENSION: PROCEDURE;
                     /* FORTRAN DIMENSION STATEMENT */
505                  CALL SCAN ('0'B);
506                  CALL TYPELIST ('');
507                  CALL SCAN ('1'B);
508                  END DIMENSION;

509               COMMON: PROCEDURE;
                     /* FORTRAN COMMON STATEMENT */
510                  CALL SCAN ('0'B);
511                  BEGCOM: IF NEXT='' THEN GO TO ENDCOM;
513                  IF NEXT='/' THEN DO;
515                     CALL SCAN ('0'B);
516                     CALL SCAN ('0'B);
517                     CALL ERROR ('NAMED COMMON NOT TRANSLATABLE');
518                     CALL SCAN ('0'B);
519                     END;
520                  CALL TYPELIST (' EXTERNAL ');
521                  GO TO BEGCOM;
522                  ENDCOM: CALL SCAN ('1'B);
523                  END COMMON;

524               EQUIVALENCE: PROCEDURE;
                     /* FORTRAN EQUIVALENCE STATEMENT */
525                  CALL ERROR('EQUIVALENCE NOT TRANSLATED IN THIS VERSION');
526                  CALL SCAN ('1'B);
527                  END EQUIVALENCE;

528               EXTERNAL: PROCEDURE;
                     /* FORTRAN EXTERNAL STATEMENT */
529                  CALL SCAN ('0'B);
530                  CALL TYPELIST (' ENTRY ');
531                  CALL SCAN ('1'B);
532                  END EXTERNAL;

533               DOUBLE: PROCEDURE;
                     /* FORTRAN DOUBLE PRECISION DECLARATION */
```



```
534                  CALL SCAN ('0'B);
535                  CALL TYP (' FLOAT BINARY (53) ');
536                  END DOUBLE;

537               DATA: PROCEDURE;
                     /* FORTRAN DATA STATEMENT */
538                  CALL ERROR ('FORTRAN DATA STATEMENT NOT TRANSLATED');
539                  CALL SCAN ('1'B);
540                  END DATA;

541               TYPELIST: PROCEDURE (TYPE);
                     /* SYMBOL TABLE BUILDING */
542                  DECLARE TYPE CHARACTER (22) VARYING;
543                  DO WHILE (NEXT¬='' & NEXT¬='/');
544                     CALL FORTIDEN;
545                     DO I=1 TO SYMLENGTH;
546                        IF SYMBOL(I)=NEXT THEN GO TO GOT;
548                     END;
549                     SYMLENGTH,I=SYMLENGTH+1;
550                  GOT: IF TYPE¬='' & TYPE¬=' EXTERNAL ' THEN SYNTYPE(I)=TYPE;
552                  IF TYPE=' EXTERNAL ' THEN SYNCOM(I)=TYPE;
554                     SYMBOL(I)=NEXT;
555                     CALL SCAN ('0'B);
556                     IF NEXT='(' THEN DO;
558                        WORD='(';
559                        DO WHILE (NEXT¬=')');
560                           CALL SCAN ('0'B);
561                           WORD=WORD||NEXT;
562                        END;
563                        CALL SCAN ('0'B);
564                        SYMDIM(I)=WORD;
565                     END;
566                  IF NEXT='/' THEN RETURN;
568                     CALL SCAN ('0'B);
569                  END;
570                  END TYPELIST;

571               TYPEARGS: PROCEDURE (TYPE);
                     /* ADDING ARGUMENTS TO THE SYMBOL TABLE */
572                  DECLARE TYPE CHARACTER (22) VARYING;
573                  CALL FORTIDEN;
574                  DO I=1 TO SYMLENGTH;
575                     IF SYMBOL(I)=NEXT THEN GO TO GOT;
577                  END;
578                  SYMLENGTH,I=SYMLENGTH+1;
579                  GOT: SYNTYPE(I)=TYPE;
580                  SYMBOL(I)=NEXT;
581                  END TYPEARGS;

582               TYP: PROCEDURE (TYPE);
                     /* SYMBOL TABLE BUILDING */
583                  DECLARE TYPE CHARACTER (22) VARYING;
584                  CALL SCAN ('0'B);
585                  IF NEXT='FUNCTION' THEN DO;
```

```
587                     CALL FUNCTION (TYPE);
588                     RETURN; END;
590                  CALL TYPELIST (TYPE);
591                  CALL SCAN ('1'B);
592                  END TYP;

593               CALL: PROCEDURE;
                     /* FORTRAN CALL STATEMENT */
594                  CALL SCAN ('0'B);
595                  WORD=NEXT;
596                  CALL SCAN ('0'B);
597                  DO WHILE (NEXT¬='');
598                     WORD=WORD||NEXT;
599                     CALL SCAN ('0'B);
600                     WORD=WORD||EXPRESSION;
601                     END;
602                  IF WORD¬='EXIT' THEN WORD='CALL '||WORD||'()';
604                  CALL DISK (LAB||WORD||';');
605                  CALL SCAN ('1'B);
606                  END CALL;

607               SUBROUTINE: PROCEDURE;
                     /* SUBROUTINE DECLARATION */
608                  CALL SCAN ('0'B);
609                  CALL FORTIDEN;
610                  FUNCTNAME=NEXT;
611                  CALL ARGLIST;
612                  PUT FILE (DECLIST) LIST (' '||FUNCTNAME||': PROCEDURE'||WORD
                     ||';');
613                  CALL SCAN ('1'B);
614                  END SUBROUTINE;

615               FUNCTION: PROCEDURE (TYPE);
                     /* FUNCTION DECLARATION */
616                  DECLARE TYPE CHARACTER (22) VARYING;
617                  CALL SCAN ('0'B);
618                  FUNCTNAME=NEXT;
619                  FUNCT='1'B;
620                  CALL ARGLIST;
621                  PUT FILE (DECLIST) LIST (' '||FUNCTNAME||': PROCEDURE '||WORD
                     ||TYPE||';');
622                  CALL SCAN ('1'B);
623                  END FUNCTION;

624               ENTRY: PROCEDURE;
                     /* ENTRY DECLARATION */
625                  DECLARE ENTRYNAME CHARACTER (6) VARYING;
626                  CALL SCAN ('0'B);
627                  CALL FORTIDEN;
628                  ENTRYNAME=NEXT;
629                  CALL ARGLIST;
630                  CALL DISK (LAB||ENTRYNAME||': ENTRY'||WORD||';');
631                  CALL SCAN ('1'B);
632                  END ENTRY;
```



```
633               ARGLIST: PROCEDURE;
                     /* ARGUMENT LIST PROCESSING */
634                  WORD='';
635                  DO WHILE (NEXT¬=')');
636                     CALL SCAN ('0'B);
637                     WORD=WORD||NEXT;
638                     END;
639                  END ARGLIST;

640               RETURN: PROCEDURE;
                     /* FORTRAN RETURN STATEMENT */
641                  IF FUNCT='0'B THEN CALL DISK (LAB||'RETURN;');
643                  ELSE CALL DISK (LAB||'RETURN ('||FUNCTNAME||'0);');
644                  CALL SCAN ('1'B);
645                  END RETURN;

646               READWRITE: PROCEDURE (RW);
                     /* FORTRAN READ AND WRITE STATEMENTS */
647                  DECLARE RW CHARACTER (3);
648                  CALL SCAN ('0'B);
649                  IF NEXT¬='(' THEN CALL ERROR ('MISSING LEFT PAREN IN READ');
651                  CALL SCAN ('0'B);
652                  IF NEXT='6' THEN UNIT='SYSPRINT';
654                  ELSE IF NEXT='5' THEN UNIT='SYSIN';
656                  ELSE DO;
657                     NEXT,UNIT='FILE'||NEXT;
658                     CALL TYPEARGS (' FILE ');
659                     END;
660                  CALL SCAN ('0'B);
661                  IF NEXT=',' THEN DO;
663                     CALL SCAN ('0'B);
664                     LE='EDIT';
665                     FMT#='(R(#'||NEXT||'))';
666                     CALL SCAN ('0'B);
667                  END;
668                  ELSE DO;
669                     LE='LIST';
670                     FMT#='';
671                  END;
672                  CALL SCAN ('0'B);
673                  CALL LIST;
674                  CALL DISK (LAB||RW||' FILE ('||UNIT||
                     ')'||LE||IOSTRING||FMT#||';');
675                  CALL SCAN ('1'B);
676                  END READWRITE;

677               FORMAT: PROCEDURE;
                     /* FORTRAN FORMAT STATEMENTS */
678                  IOSTRING='FORMAT ';
679                  CALL FMTLIST;
680                  CALL DISK (LAB||IOSTRING||';');
681                  CALL SCAN ('1'B);
682                  END FORMAT;
```

```
683                FMTLIST: PROCEDURE;
                      /* FORMAT SPECIFICATION LISTS */
684                   DECLARE HOLL BIT(1);
685                   HOLL='0'B;
686                   FMT1: CALL SCAN ('0'B);
687                   FMT2: IF NEXT='' THEN DO;
689                      IF HOLL='1'B THEN CALL ERROR
                         ('HOLLERITH STRING NOT PERMITTED IN PL/1 FORMAT');
691                      RETURN;
692                   END;
693                   IF NEXT='/' THEN DO;
695                      IF SUBSTR(IOSTRING,LENGTH(IOSTRING),1)=',' |
                            SUBSTR(IOSTRING,LENGTH(IOSTRING),1)='(' THEN
696                      LEN=''; ELSE LEN=',';
698                      CALL SCAN ('0'B);
699                      IF NEXT=',' | NEXT=')' THEN  LEN=''; ELSE LEN=',';
702                      IOSTRING=IOSTRING||LEN||'SKIP'||LEN;
703                      GO TO FMT2;
704                   END;
705                   IF NEXT='(' | NEXT=',' | NEXT=')' THEN DO;
707                      IOSTRING=IOSTRING||NEXT;
708                      GO TO FMT1;
709                   END;
710                   IF NEXT>='0' THEN GO TO NUM;
712                   IF NEXT='I' THEN LEN='F'; ELSE
714                   IF NEXT='G' | NEXT ='D' THEN LEN='E'; ELSE
716                   IF NEXT='L' THEN LEN='B'; ELSE
718                   LEN=NEXT;
719                   IOSTRING=IOSTRING||LEN;
720                   IF NEXT='(' | NEXT='L' | NEXT='A' THEN DO;
722                      CALL SCAN ('0'B);
723                      IOSTRING=IOSTRING||'('||NEXT||')';
724                      GO TO FMT1;
725                   END;
726                   CALL SCAN ('0'B);
727                   IOSTRING=IOSTRING||'('||NEXT||',';
728                   CALL SCAN ('0'B);
729                   CALL SCAN ('0'B);
730                   IOSTRING=IOSTRING||NEXT||')';
731                   GO TO FMT1;
732                   NUM: LEN=NEXT;
733                   CALL SCAN ('0'B);
734                   IF SUBSTR(NEXT,1,1)='H' THEN DO;
736                      I=LEN;
737                      LINE=SUBSTR(LINE,I-LENGTH(NEXT)+2);
738                      CALL SCAN ('0'B);
739                      IF NEXT=',' THEN CALL SCAN ('0'B);
741                      ELSE IF SUBSTR(IOSTRING,LENGTH(IOSTRING),1)=',' THEN
742                      IOSTRING=SUBSTR(IOSTRING,1,LENGTH(IOSTRING)-1);
743                      HOLL='1'B;
744                      GO TO FMT2;
745                   END;
746                   IF NEXT='X' THEN DO;
```



```
748                      IOSTRING=IOSTRING||'X('||LEN||')';
749                      GO TO FMT1;
750                   END;
751                   IF NEXT='P' THEN GO TO FMT1;
753                   IOSTRING=IOSTRING||LEN;
754                   GO TO FMT2;
755                   END FMTLIST;

756                LIST: PROCEDURE;
                      /* READ AND WRITE LISTS */
757                   IOSTRING='';
758                   DO WHILE (NEXT¬='');
759                      CALL ELEMENT;
760                      IOSTRING=IOSTRING||VARSTRING;
761                      CALL SCAN ('0'B);
762                      END;
763                   IF IOSTRING¬='' THEN IOSTRING='('||IOSTRING||')';
765                   END LIST;

766                ELEMENT: PROCEDURE RECURSIVE;
                      /* READ AND WRITE LIST ELEMENTS */
767                   IF NEXT='(' THEN DO;
769                      CALL IMPDO;
770                      CALL SCAN ('0'B);
771                      IF NEXT¬='' THEN VARSTRING=VARSTRING||',';
773                      RETURN;
774                      END;
775                   CALL VARIABLE ('0'B);
776                   IF NEXT¬='' THEN VARSTRING=VARSTRING||',';
778                   END ELEMENT;

779                IMPDO: PROCEDURE RECURSIVE;
                      /* READ AND WRITE IMPLIED DO LISTS */
780                   DECLARE IMPDOVAR CHARACTER (6) VARYING;
781                   IOSTRING=IOSTRING||'(';
782                   CALL SCAN ('0'B);
783                   EL: CALL ELEMENT;
784                   IF NEXT='=' THEN DO;
786                      IMPDOVAR=SUBSTR(VARSTRING,1,LENGTH(VARSTRING)-1);
787                      CALL SCAN ('0'B);
788                      IOSTRING=SUBSTR(IOSTRING,1,LENGTH(IOSTRING)-1);
789                      IOSTRING=IOSTRING||' DO '||IMPDOVAR||'='||NEXT||' TO ';
790                      CALL SCAN ('0'B);
791                      CALL SCAN ('0'B);
792                      IOSTRING=IOSTRING||NEXT;
793                      CALL SCAN ('0'B);
794                      IF NEXT=')' THEN IMPDOVAR='1';
796                      ELSE DO;
797                         CALL SCAN ('0'B);
798                         IMPDOVAR=NEXT;
799                         CALL SCAN ('0'B);
800                      END;
801                      IOSTRING=IOSTRING||' BY '||IMPDOVAR||')';
802                      VARSTRING='';
```

```
803                     RETURN;
804                  END;
805                  IOSTRING=IOSTRING||VARSTRING;
806                  CALL SCAN ('0'B);
807                  GO TO EL;
808                  END IMPOD;

809               BACKSPACE: PROCEDURE;
                     /* FORTRAN BACKSPACE STATEMENT */
810                  CALL ERROR('NO EQUIVALENT FOR BACKSPACE IN PL/1');
811                  CALL SCAN ('1'B);
812                  END BACKSPACE;

813               REWIND: PROCEDURE;
                     /* FORTRAN REWIND AND ENDFILE STATEMENTS */
814                  CALL SCAN('0'B);
815                  CALL DISK (LAB||'CLOSE FILE (FILE'||NEXT||');');
816                  CALL SCAN ('1'B);
817                  END REWIND;

818               DO: PROCEDURE;
                     /* FORTRAN DO STATEMENTS */
819                  CALL SCAN ('0'B);
820                  DOPOINT=DOPOINT+1;
821                  DOCOUNT(DOPOINT)=NEXT;
822                  CALL SCAN ('0'B);
823                  CALL FORTIDEN;
824                  WORD='DO '||NEXT||'=';
825                  CALL SCAN ('0'B);
826                  CALL SCAN ('0'B);
827                  IF SUBSTR(NEXT,1,1)<'0' THEN CALL FORTIDEN;
829                  WORD=WORD||NEXT;
830                  CALL SCAN ('0'B);
831                  CALL SCAN ('0'B);
832                  IF SUBSTR(NEXT,1,1)<'0' THEN CALL FORTIDEN;
834                  WORD=WORD||' TO '||NEXT;
835                  CALL SCAN ('0'B);
836                  IF NEXT=',' THEN DO;
838                     CALL SCAN ('0'B);
839                     IF SUBSTR(NEXT,1,1)<'0' THEN CALL FORTIDEN;
841                  END;
842                  ELSE NEXT='1';
843                  CALL DISK (LAB||WORD||' BY '||NEXT||';');
844                  TAB=TAB||'   ';
845                  CALL SCAN ('1'B);
846                  END DO;

847               IF: PROCEDURE;
                     /* FORTRAN ARITHMETIC AND LOGICAL IF STATEMENTS */
848                  DECLARE (NUM1,NUM2) CHARACTER (6) VARYING;
849                  CALL SCAN ('0'B);
850                  WORD=EXPRESSION;
                     /* TEST FOR LOGICAL IF STATEMENT */
851                  IF SUBSTR(NEXT,1,1)<'0' THEN DO;
```



```
853                     CALL DISK (LAB||'IF '||WORD||' THEN ');
854                     CALL STATEMENT;
855                     RETURN;
856                  END;
                     /* ARITHMETIC IF STATEMENT */
857                  NUM1=NEXT;
858                  CALL SCAN ('0'B);
859                  CALL SCAN ('0'B);
860                  NUM2=NEXT;
861                  CALL SCAN ('0'B);
862                  CALL SCAN ('0'B);
863                  CALL DISK(LAB||'IF '||WORD||' > 0 THEN '||'GO TO #'||NEXT||';');
864                  CALL DISK('ELSE IF '||WORD||' = 0 THEN GO TO #'||NUM2||';');
865                  CALL DISK ('ELSE GO TO #'||NUM1||';');
866                  CALL SCAN ('1'B);
867                  END IF;

868               ASSIGN: PROCEDURE;
                     /* FORTRAN ASSIGN STATEMENT */
869                  DECLARE NUM CHARACTER (6) VARYING;
870                  CALL SCAN ('0'B);
871                  NUM=NEXT;
872                  CALL SCAN ('0'B);
873                  IF NEXT¬='TO' THEN CALL ERROR('MISSING <TO> IN ASSIGN');
875                  CALL SCAN ('0'B);
876                  CALL FORTIDEN;
877                  CALL DISK (LAB||NEXT||'=#'||NUM||';');
878                  CALL SCAN ('1'B);
879                  END ASSIGN;

880               GO: PROCEDURE;
                     /* FORTRAN COMPUTED,ASSIGNED, AND UNCONDITIONAL GO TO */
881                  DECLARE (NUM,NUM) CHARACTER (9);
882                  CALL SCAN ('0'B);
883                  IF NEXT¬='TO' THEN CALL ERROR('MISSING <TO> IN GO TO STATEMENT');
885                  CALL SCAN ('0'B);
886                  IF SUBSTR(NEXT,1,1)>='0' THEN DO;
                        /* UNCONDITIONAL GO TO STATEMENT */
888                     CALL DISK (LAB||'GO TO #'||NEXT||';');
889                     CALL SCAN ('1'B);
890                     RETURN;
891                  END;
                     /* COMPUTED GO TO STATEMENT */
892                  IF NEXT = '(' THEN DO;
894                     I=0;
895                     WORD=NEXT;
896                     DO WHILE (NEXT¬=')');
897                        CALL SCAN ('0'B);
898                        I=I+1;
899                        WORD=WORD||'#'||NEXT;
900                        CALL SCAN ('0'B);
901                        WORD=WORD||NEXT;
902                     END;
903                     CALL SCAN ('0'B);
```

```
904                     IF NEXT¬=',' THEN CALL ERROR('MISSING COMMA IN COMP GO TO');
906                     CALL SCAN ('0'B);
907                     CALL FORTIDEN;
908                     LAC=LAC+1;
909                     NUM=LAC;
910                     J,SYMLENGTH=SYMLENGTH+1;
911                     SYMBOL(J)='LAB'||SUBSTR(NUM,9,1);
912                     NUM=1;
913                     SYMDIM(J)='('||SUBSTR(NUM,9,1)||')';
914                     SYMCOM(J)=' INITIAL '||WORD;
915                     SYMTYPE(J)=' LABEL ';
916                     CALL DISK(LAB||'GO TO LAB'||SUBSTR(NUM,9,1)||'('||NEXT||')');
917                     CALL SCAN ('1'B);
918                     RETURN;
919                  END;
                    /* ASSIGNED GO TO STATEMENT */
920                 CALL TYPEARGS (' LABEL ');
921                 CALL DISK (LAB||'GO TO '||NEXT||';');
922                 CALL SCAN ('1'B);
923                 END GO;

924              STOP: PROCEDURE;
                    /* FORTRAN STOP STATEMENT */
925                 CALL DISK (LAB||'STOP;');
926                 CALL SCAN ('1'B);
927                 END STOP;

928              PAUSE: PROCEDURE;
                    /* FORTRAN PAUSE STATEMENT */
929                 CALL SCAN ('0'B);
930                 CALL DISK (LAB||'DISPLAY (''PAUSE'')NEXT||''');');
931                 CALL SCAN ('1'B);
932                 END PAUSE;

933              CONTINUE: PROCEDURE;
                    /* FORTRAN CONTINUE STATEMENT */
934                 CALL DISK (LAB||'');
935                 CALL SCAN ('1'B);
936                 END CONTINUE;

937              LABEL: PROCEDURE;
                    /* LABEL PROCESSING */
938                 LABCOUNT=0;
939                 IF COL1=' ' THEN LABSTR=''; ELSE LABSTR=COL1;
942                 DO I=1 TO 4;
943                    IF SUBSTR(COL25,I,1)¬=' ' THEN LABSTR=LABSTR||
945                    SUBSTR(COL25,I,1); END;
946                 DO J=1 TO DOPOINT;
947                    IF DOCOUNT(J)=LABSTR THEN LABCOUNT=LABCOUNT+1;
949                 END;
950                 LAB='#'||LABSTR||': ';
951                 END LABEL;

                    /* MAIN DRIVER WHICH CALL PROCEDURE  'PROGRAM' FOR EACH
```



```
                       FORTRAN MAIN OR SUBPROGRAM IN THE INPUT STREAM */
952                 ON ENDFILE (SYSIN) GO TO NEXTB;
954                 NEXTA: CALL PROGRAM;
955                 GO TO NEXTA;
956                 NEXTB: PUT FILE (SYSPRINT) LIST ('END OF INPUT') SKIP(3);
957              END FATOMP;
```

NO ERROR OR WARNING CONDITION HAS BEEN DETECTED FOR THIS COMPILATION.

TIME FOR THIS COMPILATION WAS 1.43 MINUTES