POLISH 77: USER'S GUIDE

by

Chang-Lin Chen* and Lloyd Fosdick**

CU-CS-220-83 July, 1983

*Chang-Lin Chen is a Visiting Scholar from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, The People's Republic of China.

**Department of Computer Science, University of Colorado, Boulder, CO 80309 USA

This research was supported, in part, by the National Science Foundation, grant MCS 8000017; by the Department of Energy, grant DE-AC02-80ER10718; and by the National Bureau of Standards.

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE NATIONAL SCIENCE FOUNDATION.

Errata

page 5, line 1 should read

> After a separator if the parenthesis level is greater than one.

(Note - this erratum reflects a software change that was made after this guide was printed. The illustrations are consistent with the original wording on page 5, line 1).

1. Introduction.

Polish 77 (P77) is a formatter for programs written in Fortran 77 (F77).[3] It provides the usual features found in these tools; for example, indentation of control structures, systematic token spacing, systematic labelling, and line identification. An illustration of the kind of formatting done by P77 is shown in Figures 1a and 1b. Figure 1a is the original program; it is the input for P77. Figure 1b is the same program after formatting; it is the output from P77. Notice that statement labels are ordered, control structures are indented, each DO-range ends on a distinct CONTINUE statement, lines are identified, and a group of comment lines has been decorated with a box.

```
      SUBROUTINE MMM(A, B, C, M, N)
C) MATRIX-MATRIX MULTIPLY.
C INPUT A, B, M, N
C   A,B: MATRICES, DIMENSION(M,M), ORDER N
C OUTPUT C
C   C: C=A*B
C ERROR
C   MESSAGE IF N.LT.1 OR N.GT.M
      INTEGER M, N
      REAL A(M,*),B(M,*),C(M,*)
      COMMON /ERRM/ ERRM1,ERRM2,ERRM3,ERRM4,ERRM5
C LOCAL
      INTEGER I,J,K
      IF((N.LT.1).OR.(N.GT.M))THEN
C ORDER EXCEEDS RANGE
      CALL ERROR(ERRM3)
      RETURN
      ELSE
      DO 10 I=1,N
      DO 10 J=1,N
      C(I,J)=0
      DO 10 K=1,N
10    C(I,J)=C(I,J)+A(I,K)*B(K,J)
      ENDIF
      RETURN
      END
```

Figure 1a: Subroutine before processing by P77.

[3] [ANSI FORTRAN] -- Citations in the running text appear in square brackets. The list of references appears at the end of this text.

```
      SUBROUTINE MMM(A,B,C,M,N)                                          MMM  10
C *************************************************                    MMM  20
C *  MATRIX-MATRIX MULTIPLY.                      *                    MMM  30
C *  INPUT A, B, M, N                             *                    MMM  40
C *    A,B: MATRICES, DIMENSION(M,M), ORDER N     *                    MMM  50
C *  OUTPUT C                                     *                    MMM  60
C *    C: C=A*B                                   *                    MMM  70
C *  ERROR                                        *                    MMM  80
C *    MESSAGE IF N.LT.1 OR N.GT.M                *                    MMM  90
C *************************************************                    MMM 100
      INTEGER M, N                                                       MMM 110
      REAL A(M,*), B(M,*), C(M,*)                                        MMM 120
      COMMON /ERRM/ ERRM1, ERRM2, ERRM3, ERRM4, ERRM5                    MMM 130
C **  LOCAL                                                              MMM 140
      INTEGER I, J, K                                                    MMM 150
C                                                                        MMM 160
      IF ((N.LT.1).OR.(N.GT.M)) THEN                                     MMM 170
C   **  ORDER EXCEEDS RANGE                                            MMM 180
        CALL ERROR(ERRM3)                                                MMM 190
        RETURN                                                           MMM 200
C                                                                        MMM 210
      ELSE                                                               MMM 220
        DO 30 I = 1, N                                                   MMM 230
          DO 20 J = 1, N                                                 MMM 240
            C(I,J) = 0                                                   MMM 250
            DO 10 K = 1, N                                               MMM 260
              C(I,J) = C(I,J) + A(I,K)*B(K,J)                            MMM 270
   10       CONTINUE                                                     MMM 280
   20     CONTINUE                                                       MMM 290
   30   CONTINUE                                                         MMM 300
      END IF                                                             MMM 310
      RETURN                                                             MMM 320
C                                                                        MMM 330
      END                                                                MMM 340
```

Figure 1b: Subroutine after processing by P77.

The following requirements influenced the design of P77.

(1) Textout[4] should be readable by an F77 compiler and should yield the same object code as textin[4].

(2) P77 should be fast. A consequence of this is that P77 uses only lexical analysis, not a full parse of the program.

(3) P77 should be easy to use. Accordingly, the number of parameters is small and default values for all of them are provided. Also, no modification of textin is necessary; i.e. no special formatting marks need be placed in textin before it is given to P77.

(4) For flexibility P77 should be a user callable subroutine. Thus the user must write a main program to call P77. The main program opens files and initializes parameter values when default values are to be overridden. A sample main program is in Appendix 2.

(5) Formatting should be restricted as much as possible to rearranging white space. The only transformations made by P77 that are not a rearrangement of white space are: relabelling, insertion of a CONTINUE statement to terminate a DO-range, insertion of line continuation marks (the character put in column 6 on a continuation line), line identification, relocation of FORMAT statements, and decoration of comment blocks. All of these transformations except the continuation mark transformation can be disabled.

(6) P77 should be portable. Therefore, P77 is written in F77. Machine dependencies are isolated in a few small subprograms described in the "Installation" section.[5]

This guide is designed to serve the needs of the user and of the installer. Sections 2, 3 and 4 are for the user. Section 2 describes the formatting done by P77, Section 3 explains how to use P77, and Section 4 discusses error messages. Section 5 is for the installer. It discusses machine dependent procedures. Appendices contain lists of error messages and other important reference information.

P77 is a descendant of Polish [Dorrenbacher 76], a formatter for Fortran 66 programs which has been widely used since 1974. Many people have contributed to the production of P77. The requirements for P77 and its general design were written by Lloyd Fosdick. A prototype program was written by Jonathan Corbett, Rebecca Sobol, Sue Ross and Charles Manlove as a student project. The current version was written by Chang-Lin Chen. The scanner, the "front end" of P77, was written by Geoffrey Clemm [Clemm 81]. Linda Lindgren of the National Bureau of Standards tested P77. This work is a part of the Toolpack Project [Osterweil 82].

2. Formatting Features.

This section defines fixed formatting features (fff's) and variable formatting features (vff's). Variable formatting features can be controlled by the user by assignment of values to parameters before formatting begins. Fixed formatting features cannot be controlled by the user. The parameters for vff's are summarized in Appendix 1.

The positions on a line in which characters can be written are numbered 1, 2, ..., from left to right. We call the position number the "column index".

---

[4] The words "textin" and "textout" are used throughout this document to denote, respectively, the F77 program that is the input for P77 and the formatted text that is the output from P77.

[5] One departure from strict F77 is present. Hollerith strings are used as in F66. We believe that most F77 compilers support this anachronism.

2.1. Margin control.

Left and right margins for a statement are vff's. They are controlled by the parameters LMARGS and RMARGS: LMARGS is the column index of the left margin for statements; RMARGS is the column index of the right margin for statements. They must satisfy the conditions:

LMARGS >= 7, RMARGS <= 72, RMARGS - LMARGS +1 >= 30.

Their default values are:

LMARGS = 7, RMARGS = 72.

The left margin for a comment is a vff. It is controlled by the parameter LMARGC. It must satisfy the conditions:

LMARGC >= 2, LMARGC <= 60.

This parameter controls the left margin of the comment text. It does not affect the position of the comment line designator (C or *). The default value is:

LMARGC = 3

The right margin for a comment is column index 72 and is a fff.

2.2. Token spacing.

A token is a lexical element of the language: a name, a constant, an arithmetic operator, etc. The spacing of tokens is a fff and the rules governing it are given below. In these rules the following terms are used:

delimiter
: Any symbol in the set {( ) ' /} . The interpretation of / as a delimiter depends on the context. It is a delimiter before and after names of COMMON blocks and value lists in DATA statements.

operator
: The F77 definition applies except that the character = is called the assignment operator here.

separator
: Any symbol in the set {, :} .

parenthesis level
: A non-negative integer denoting the depth of nesting of a token within parentheses. A token that is not contained within parentheses is at parenthesis level zero; one that is contained within a single pair of parentheses is at level 1, and so forth.

precedence
: Operator precedence as defined for F77.

The spacing rules follow. There is one space between adjacent tokens within a statement except there is no space in the following circumstances:

Before a left delimiter if the preceding token is a name;

After a left delimiter;

Before a right delimiter;

Before a separator;

After a separator if the parenthesis level is greater than zero;

After a unary operator;

Before and after a binary arithmetic operator if the operator does not have lowest precedence, or if the parenthesis level is greater than zero;

Before and after the assignment operator if the parenthesis level is greater than zero;

Before and after a relational operator if the parenthesis level is greater than one;

Before and after a binary logical operator if the parenthesis level is greater than one or the operator has highest or next-to-the-highest precedence;

Before and after the concatenation operator.

2.3. Blank lines.

This is a fff. Blank lines appear in textout to mark a break in control flow. In particular, there will be a blank line after the following statements: GO TO; RETURN; PAUSE; STOP; END (except for the very last END). There is a blank line before an ELSE statement and an ELSE IF statement. The idea is that these blank lines indicate an interruption of control flow.

A blank line separates the non-executable specification statements at the front of a program unit from the executable statements. There is one exception: the blank line will come before any statement function specifications.

A blank line appears before and after sequences of FORMAT statements.

2.4. Breaking a statement line.

This is a fff except that the indentation of the continuation line is controlled by the parameter INDNTS (see below). A statement line must be broken, and a continuation line created, if the statement would extend beyond RMARGS. The following rules control the line breaking action.

The token before the break is (in descending order of preference): a separator with parenthesis level zero; a lowest precedence binary operator with parenthesis level zero; a binary operator with parenthesis level zero; a separator; a binary operator; any token except a left delimiter; a left delimiter.

The break is at least half-way between the start of the line and RMARGS. (Note that the start of a line is not necessarily LMARGS because the line may be indented.)

Subject to the above rules, the break is locate as far to the right as possible. Thus, the break occurs at the rightmost separator with parenthesis level zero, if one occurs half-way between the start of the line and RMARGS; otherwise, the break occurs at a lowest precedence binary operator with parenthesis level zero, if one occurs half-way between the start of the line and RMARGS; and so forth.

The continuation line is indented relative to the start of the initial line by the amount specified by the indentation parameter INDNTS. The continuation line designator, the character in line position 6, is the currency symbol.

Statement line continuations in textin are not preserved. When P77 reads a statement from textin it assembles it internally as a single "line", removing all line breaks that might have been present. Before P77 writes the statement in textout it breaks the internal line, if necessary, to create the output line.

2.5. CONTINUE at end of DO range.

This is a fff except as noted below under "Labels". In textout every DO range ends on a CONTINUE statement and nested ranges end on separate CONTINUE statements.

2.6. Indentation.

This is a vff, controlled by the parameters INDNTS for statements and INDNTC for comments. INDNTS specifies the number of line positions used for indentation of statements. As already noted, continuation lines are indented. Also, statements are indented when they fall in the range of a DO or when they fall in an IF-block, an ELSE IF-block, or an ELSE-block. Indentation is relative to the header line: the DO, the IF, etc. Indentation of statements does not extend to the right of line position RMARGS-30. There is no indentation of statements when INDNTS is zero. The default value of INDNTS is 2.

When a comment line falls within the scope of one of the above control blocks it will be indented by the amount specified by INDNTC. See Figure 1b, where INDNTC=2. If indentation of a comment line would cause it to extend to the right of column 72 the comment line is left alone (it is in the same position in textout as it is in textin). There is no indentation of comment lines when INDNTC is zero. The default value of INDNTC is 2.

2.7. Labels.

This is a vff, controlled by the parameter LABELS. If the value of LABELS is x, x not equal to zero, the first label in textout is x, the next is 2x, the next 3x, and so forth. If the value of LABELS is zero then no relabelling is done. Also, if the value of LABELS is zero no change is made to DO loops to make the range end on a CONTINUE (this might force the creation of a new label). The default value of LABELS is 10.

2.8. Line identification.

This is a vff, controlled by the parameter IDENTL. If IDENTL is equal to 1 then identification information is in the column field 73...80. The first half of this field contains the first four characters of the program unit name, and the second half contains a sequence number. If IDENTL is zero then the field is blank in textout. The default value of IDENTL is 1.

When P77 reads textin any information that might have been in columns 73 through 80 is lost (unless the FM STOP comment is used, see below). Therefore this information will always be replaced by something else in textout.

2.9. Move FORMAT statements.

This is a vff, controlled by the parameter MOVESF. If MOVESF is equal to 1 then format statements are located at the end of the program unit in textout. If MOVESF is equal to zero they are left in their original positions. The default value of MOVESF is 0.

2.10. Comments.

Comment lines may be indented and decorated. This is a vff, controlled by two parameters, KCMMNT and MCMMNT. The first of these, KCMMNT, defines the kind of decoration used and the second, MCMMNT, defines the start of the text of the comment line. In explaining the meaning of the values for these parameters the term "comment block" is used: it means an uninterrupted sequence of comment lines. Figures 2-9 illustrate the effect of using these (as well s the other

formatting) parameters. Each figure has three sections: the PARAMS section contains the ten formatting parameters used to produce the results shown -- parameters are in the order listed in Appendix 1; the STDIN section shows textin; the STDOUT section shows textout.

KCMMNT = 1
Dashes precede the text of an indented comment line. See Fig. 2.

KCMMNT = 2
Asterisks precede the text of a comment line. See Fig. 3.

KCMMNT = 3
Half box of asterisks drawn around comment block. See Fig. 4.

KCMMNT = 4
Full box of asterisks drawn around comment block. See Fig. 5.

KCMMNT = 5
If the first character of the text of a comment block is preceded by a right parenthesis then draw a full box around the block (as in KCMMNT = 4) otherwise precede the text of an indented comment line with asterisks (as in KCMMNT = 2). See Fig. 6.

KCMMNT = 6
Leave comment lines alone. They will appear in textout exactly as in textin. See Fig. 7.

MCMMNT = 0
Discard leading blanks on a comment line. See Figs. 2-7.

MCMMNT = 1
The first non-blank character on the comment line is a flag marking the start of the comment text. See Fig. 8.

MCMMNT = 2
Do not discard leading blanks on a comment line. See Fig. 9.

The default values are KCMMNT = 5 and MCMMNT = 2.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT
      22     70      3      4      4      0     10      1      1      0
STDIN:
      SUBROUTINE TEST(M,N)
C   1.
C      MCMMNT = 0
C       KCMMNT = 1
      DO 10 I = 1,N
C   2.
C      MCMMNT = 0
C       KCMMNT = 1
        M = M + I
      DO 10 J = 1,N
C   3.
C      MCMMNT = 0
C       KCMMNT = 1
        M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C 1.                                                                TEST  20
C MCMMNT = 0                                                        TEST  30
C KCMMNT = 1                                                        TEST  40
C                                                                   TEST  50
                      DO 20 I = 1, N                                TEST  60
C --- 2.                                                           TEST  70
C  -- MCMMNT = 0                                                   TEST  80
C  -- KCMMNT = 1                                                   TEST  90
                          M = M + I                                 TEST 100
                          DO 10 J = 1, N                            TEST 110
C -------- 3.                                                      TEST 120
C       -- MCMMNT = 0                                              TEST 130
C       -- KCMMNT = 1                                              TEST 140
                              M = M + J                             TEST 150
   10                     CONTINUE                                  TEST 160
   20                 CONTINUE                                      TEST 170
                      RETURN                                        TEST 180
C                                                                   TEST 190
                      END                                           TEST 200
```

Figure 2: Example showing effect of KCMMNT = 1.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT
      22     70      3      4      4      0     10      1      2      0
STDIN:
      SUBROUTINE TEST(M,N)
C   1.
C      MCMMNT = 0
C       KCMMNT = 2
      DO 10 I = 1,N
C   2.
C      MCMMNT = 0
C       KCMMNT = 2
         M = M + I
      DO 10 J = 1,N
C   3.
C      MCMMNT = 0
C       KCMMNT = 2
         M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C ** 1.                                                             TEST  20
C ** MCMMNT = 0                                                     TEST  30
C ** KCMMNT = 2                                                     TEST  40
C                                                                   TEST  50
                      DO 20 I = 1, N                                TEST  60
C     ** 2.                                                         TEST  70
C     ** MCMMNT = 0                                                 TEST  80
C     ** KCMMNT = 2                                                 TEST  90
                          M = M + I                                 TEST 100
                          DO 10 J = 1, N                            TEST 110
C         ** 3.                                                     TEST 120
C         ** MCMMNT = 0                                             TEST 130
C         ** KCMMNT = 2                                             TEST 140
                              M = M + J                             TEST 150
   10                     CONTINUE                                  TEST 160
   20                 CONTINUE                                      TEST 170
                      RETURN                                        TEST 180
C                                                                   TEST 190
                      END                                           TEST 200
```

Figure 3: Example showing effect of KCMMNT = 2.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT
      22     70      3      4      4      0     10      1      3      0
STDIN:
      SUBROUTINE TEST(M,N)
C   1.
C       MCMMNT = 0
C        KCMMNT = 3
      DO 10 I = 1,N
C   2.
C       MCMMNT = 0
C        KCMMNT = 3
          M = M + I
      DO 10 J = 1,N
C   3.
C       MCMMNT = 0
C        KCMMNT = 3
          M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C * 1.                                                              TEST  20
C * MCMMNT = 0                                                      TEST  30
C * KCMMNT = 3                                                      TEST  40
C ************                                                      TEST  50
C                                                                   TEST  60
                      DO 20 I = 1, N                                TEST  70
C     * 2.                                                         TEST  80
C     * MCMMNT = 0                                                 TEST  90
C     * KCMMNT = 3                                                 TEST 100
C     ************                                                  TEST 110
                          M = M + I                                 TEST 120
                          DO 10 J = 1, N                            TEST 130
C         * 3.                                                     TEST 140
C         * MCMMNT = 0                                             TEST 150
C         * KCMMNT = 3                                             TEST 160
C         ************                                              TEST 170
                              M = M + J                             TEST 180
   10                     CONTINUE                                  TEST 190
   20                 CONTINUE                                      TEST 200
                      RETURN                                        TEST 210
C                                                                   TEST 220
                      END                                           TEST 230
```

Figure 4: Example showing effect of KCMMNT = 3.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT
      22     70      3      4      4      0     10      1      4      0
STDIN:
      SUBROUTINE TEST(M,N)
C   1.
C      MCMMNT = 0
C       KCMMNT = 4
      DO 10 I = 1,N
C   2.
C      MCMMNT = 0
C       KCMMNT = 4
        M = M + I
      DO 10 J = 1,N
C   3.
C      MCMMNT = 0
C       KCMMNT = 4
        M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C **************                                                    TEST  20
C * 1.          *                                                   TEST  30
C * MCMMNT = 0 *                                                   TEST  40
C * KCMMNT = 4 *                                                   TEST  50
C **************                                                    TEST  60
C                                                                   TEST  70
                      DO 20 I = 1, N                                TEST  80
C     **************                                                TEST  90
C     * 2.          *                                             TEST 100
C     * MCMMNT = 0 *                                             TEST 110
C     * KCMMNT = 4 *                                             TEST 120
C     **************                                              TEST 130
                        M = M + I                                   TEST 140
                        DO 10 J = 1, N                              TEST 150
C         **************                                            TEST 160
C         * 3.          *                                         TEST 170
C         * MCMMNT = 0 *                                         TEST 180
C         * KCMMNT = 4 *                                         TEST 190
C         **************                                          TEST 200
                          M = M + J                                 TEST 210
   10                   CONTINUE                                    TEST 220
   20                 CONTINUE                                      TEST 230
                      RETURN                                        TEST 240
C                                                                   TEST 250
                      END                                           TEST 260
```

Figure 5: Example showing effect of KCMMNT = 4.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT
      22     70      3      4      4      0     10      1      5      0
STDIN:
      SUBROUTINE TEST(M,N)
C    )1.
C       MCMMNT = 0
C        KCMMNT = 5
      DO 10 I = 1,N
C    2.
C       MCMMNT = 0
C        KCMMNT = 5
         M = M + 1
      DO 10 J = 1,N
C    )3.
C       MCMMNT = 0
C        KCMMNT = 5
         M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C **************                                                    TEST  20
C * 1.          *                                                   TEST  30
C * MCMMNT = 0 *                                                   TEST  40
C * KCMMNT = 5 *                                                   TEST  50
C **************                                                    TEST  60
C                                                                   TEST  70
                      DO 20 I = 1, N                                TEST  80
C     ** 2.                                                         TEST  90
C     ** MCMMNT = 0                                                 TEST 100
C     ** KCMMNT = 5                                                 TEST 110
                          M = M + I                                 TEST 120
                          DO 10 J = 1, N                            TEST 130
C         **************                                            TEST 140
C         * 3.          *                                         TEST 150
C         * MCMMNT = 0 *                                          TEST 160
C         * KCMMNT = 5 *                                          TEST 170
C         **************                                            TEST 180
                              M = M + J                             TEST 190
   10                     CONTINUE                                  TEST 200
   20                 CONTINUE                                      TEST 210
                      RETURN                                        TEST 220
C                                                                   TEST 230
                      END                                           TEST 240
```

Figure 6: Example showing effect of KCMMNT = 5.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT
      22     70      3      4      4      0     10      1      6      0
STDIN:
      SUBROUTINE TEST(M,N)
C   1.
C       MCMMNT = 0
C       KCMMNT = 6
      DO 10 I = 1,N
C   2.
C       MCMMNT = 0
C       KCMMNT = 6
          M = M + I
      DO 10 J = 1,N
C   3.
C       MCMMNT = 0
C       KCMMNT = 6
          M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C   1.                                                              TEST  20
C       MCMMNT = 0                                                  TEST  30
C       KCMMNT = 6                                                  TEST  40
C                                                                   TEST  50
                      DO 20 I = 1, N                                TEST  60
C   2.                                                              TEST  70
C       MCMMNT = 0                                                  TEST  80
C       KCMMNT = 6                                                  TEST  90
                          M = M + I                                 TEST 100
                          DO 10 J = 1, N                            TEST 110
C   3.                                                              TEST 120
C       MCMMNT = 0                                                  TEST 130
C       KCMMNT = 6                                                  TEST 140
                              M = M + J                             TEST 150
   10                     CONTINUE                                  TEST 160
   20                 CONTINUE                                      TEST 170
                      RETURN                                        TEST 180
C                                                                   TEST 190
                      END                                           TEST 200
```

Figure 7: Example showing effect of KCMMNT = 6.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT
      22     70      3      4      4      0     10      1      1      1
STDIN:
      SUBROUTINE TEST(M,N)
C  *1.
C     *MCMMNT = 1
C      *KCMMNT = 1
      DO 10 I = 1,N
C  *2.
C     *MCMMNT = 1
C      *KCMMNT = 1
         M = M + I
      DO 10 J = 1,N
C  *3.
C     *MCMMNT = 1
C      *KCMMNT = 1
         M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C 1.                                                                TEST  20
C MCMMNT = 1                                                        TEST  30
C KCMMNT = 1                                                        TEST  40
C                                                                   TEST  50
                      DO 20 I = 1, N                                TEST  60
C --- 2.                                                           TEST  70
C  -- MCMMNT = 1                                                   TEST  80
C  -- KCMMNT = 1                                                   TEST  90
                          M = M + I                                 TEST 100
                          DO 10 J = 1, N                            TEST 110
C -------- 3.                                                      TEST 120
C      -- MCMMNT = 1                                               TEST 130
C      -- KCMMNT = 1                                               TEST 140
                              M = M + J                             TEST 150
   10                     CONTINUE                                  TEST 160
   20                 CONTINUE                                      TEST 170
                      RETURN                                        TEST 180
C                                                                   TEST 190
                      END                                           TEST 200
```

Figure 8: Example showing effect of MCMMNT = 1.

```
PARAMS:
  LMARGS RMARGS LMARGC INDNTS INDNTC MOVESF LABELS IDENTL KCMMNT MCMMNT

      22     70      3      4      4      0     10      1      1      2
STDIN:
      SUBROUTINE TEST(M,N)
C1.
C  MCMMNT = 2
C  KCMMNT = 1
      DO 10 I = 1,N
C2.
C  MCMMNT = 2
C  KCMMNT = 1
         M = M + I
      DO 10 J = 1,N
C3.
C  MCMMNT = 2
C  KCMMNT = 1
         M = M + J
 10   CONTINUE
      RETURN
      END

STDOUT:
                      SUBROUTINE TEST(M,N)                          TEST  10
C 1.                                                                TEST  20
C   MCMMNT = 2                                                      TEST  30
C   KCMMNT = 1                                                      TEST  40
C                                                                   TEST  50
                      DO 20 I = 1, N                                TEST  60
C --- 2.                                                           TEST  70
C  --   MCMMNT = 2                                                 TEST  80
C  --   KCMMNT = 1                                                 TEST  90
                          M = M + I                                 TEST 100
                          DO 10 J = 1, N                            TEST 110
C ------- 3.                                                       TEST 120
C      --   MCMMNT = 2                                            TEST 130
C      --   KCMMNT = 1                                            TEST 140
                              M = M + J                             TEST 150
   10                     CONTINUE                                  TEST 160
   20                 CONTINUE                                      TEST 170
                      RETURN                                        TEST 180
C                                                                   TEST 190
                      END                                           TEST 200
```

Figure 9: Example showing effect of MCMMNT = 2.

2.11. Stopping and restarting formatting.

It is possible to stop the formatting action. The stop signal is the comment line: C$ FM STOP

which must be in textin before the program unit, or sequence of program units it guards. In particular it must come before the subroutine, function, block data, or program statement. For example, if the user does not want the subroutine SAVE formatted, then textin would contain the lines

```
C$ FM STOP
    SUBROUTINE SAVE(X,Y)
    {body of the subroutine}
    END
C$ FM START
```

The "FM START" line restarts the formatting. Everything between "FM STOP" and "FM START" is left alone. It is copied, as is, from textin to textout. These special control cards cannot appear within a program unit. Thus formatting cannot be turned off for only part of a program unit.

3. How to Use P77

It is expected that the user will write a main program that will open and close various files used by P77, read a data file containing values for the formatting parameters if the default values are not acceptable, and then call P77. An example of a main program is given in Appendix 2. In this section the files required and how they are used will be described and we will give the invocation rules for P77.

3.1. Files.

There are three "permanent files", one for textin, one for textout, and one for error messages. A fourth file for initializing the formatting parameters is optional-- it is not needed if default values are used or if the values are initialized in the main program by assignment statements. P77 uses the unit names STDIN (textin file), STDOUT (textout file), and STDERR (error message file). If the parameters are on the user's file named P77PAR, if textin is on the user's file named SOURCE, if textout is to go to the file named PRETTY, and if the error messages are to go to the file named ERRORS, then these OPEN statements should be in the user's main program:

```
OPEN(PARAMS,FILE='P77PAR',STATUS=OLD)
OPEN(STDIN,FILE='SOURCE',STATUS=OLD)
OPEN(STDOUT,FILE='PRETTY',STATUS=NEW)
OPEN(STDERR,FILE='ERRORS',STATUS=NEW)
```

See Appendix 2 for a sample main program.

P77 uses seven work files with unit names TMP1, TMP2, ..., TMP7. These must be opened in the main program. They can be removed after executing P77 since they contain no important information for the user.

Ordinarily, the user does not need to know the numerical values that P77 assigns to the I/O units. However, there would be trouble if the user attempted to assign one of these units to some other file. Therefore they have been listed in Appendix 6.

3.2. Common block communication.

The following declarations must appear in the main program.

```
C -- DECLARATIONS FOR FORMATTING PARAMETERS.
      INTEGER RMARGS
      COMMON /USER/ LMARGS, RMARGS, LMARGC, INDNTS, INDNTC,
     $              MOVESF, LABELS, IDENTL, KCMMNT, MCMMNT

C -- DECLARATIONS FOR FILES.
      INTEGER STDIN, STDOUT, STDERR
      INTEGER TMP1, TMP2, TMP3, TMP4, TMP5, TMP6,
     $        TMP7
      COMMON /FILES/ STDIN, STDOUT, STDERR,
     $               TMP1, TMP2, TMP3, TMP4, TMP5,
     $               TMP6, TMP7

C -- DECLARATION FOR NUMBER OF BITS PER WORD (NBTPWD)
      INTEGER NBTPWD
      COMMON /NBTPWC/ NBTPWD
C -- CYBER VALUE OF NBTPWD IS 60
      DATA NBTPWD/60/
```

3.3. Invocation of P77.

The formatting parameters are referred to within P77 by the names introduced above (LMARGS, RMARGS, etc.). If the user's main program does not assign a value to a parameter then the default value will be used. The example in Appendix 2 shows the values being read from a file.

After the permanent files and work files have been opened and new parameter values have been assigned (whenever the default is not to be used) the user gives the command:

```
CALL SCNPOL
```

to invoke execution of P77. The name SCNPOL is a reflection of the fact that P77 consists of two major components: scanner and polisher. The scanner breaks textin into a sequence of tokens. These are written to one of the work files. The polisher reads the tokens from the work file and creates textout.

After control returns from SCNPOL to the main program nothing further needs to be done and execution can be terminated.

4. Error Handling.

Errors are treated differently depending on whether they are detected during scanning or during formatting. If errors are detected during scanning, then control returns to the main program after the scanning process has completed and before the formatting process begins. Error messages will be found on the error message file. These messages all have the form:

```
    SCAN ERROR NO. xx NEAR TOKEN yy
or
    FATAL ERROR NO. xx NEAR TOKEN yy
```

where xx is an error message identification number and yy is a token location number. The meaning of the error number is given in [Clemm 81]. The token location number is to be used with the "listing" to determine where the error was found by the scanner. The listing will be on the file connected to STDOUT (i.e. the file that would contain the formatted program if there were no errors). This file contains textin augmented with token numbers on the left. The listing file for the subroutine in Fig 1a. is shown in Fig. 10. If no erors are detected in scanning, then no listing is on the [STDOUT] file.

```
  1        SUBROUTINE MMM(A, B, C, M, N)
    C) MATRIX-MATRIX MULTIPLY.
    C INPUT A, B, M, N
    C   A,B: MATRICES, DIMENSION(M,M), ORDER N
    C OUTPUT C
    C   C: C=A*B
    C ERROR
    C   MESSAGE IF N.LT.1 OR N.GT.M
 16         INTEGER M, N
 21         REAL A(M,*),B(M,*),C(M,*)
 43         COMMON /ERRM/ ERRM1,ERRM2,ERRM3,ERRM4,ERRM5
    C LOCAL
 58         INTEGER I,J,K
 65         IF((N.LT.1).OR.(N.GT.M))THEN
    C ORDER EXCEEDS RANGE
 82         CALL ERROR(ERRM3)
 88         RETURN
 90         ELSE
 92         DO 10 I=1,N
100         DO 10 J=1,N
108         C(I,J)=0
117         DO 10 K=1,N
125 10      C(I,J)=C(I,J)+A(I,K)*B(K,J)
154         ENDIF
157         RETURN
159         END
```

Figure 10: Example of a listing file.

The number in the first column of Fig. 10 is the number of the first token on the line. Tokens have their usual meaning, that is they are the elementary syntactic elements or lexemes, except that a comment block counts as one token and the end-of-line counts as one token. Thus on the first line there are 13 "visible" tokens, and an end-of-line token, then there is a comment block, the fifteenth token, and the sixteenth token starts the line "INTEGER ..."

If the message is "FATAL ERROR ..." then it is almost certain that a table has overflowed. To fix the error the dimensions of certain arrays in the scanner must be changed. The reader should consult the FSCAN-81 report [Clemm 81] for details. Fortunately, fatal error messages are unlikely.

If the message is "SCAN ERROR ..." then it is almost certain that textin has a syntax error in the neighborhood of the token location given in the message. The user should find the error by looking in the listing file and repair it.

If no errors are detected by the scanner then formatting will take place. If errors are detected during formatting then messages will be written on the error message file Error messages of this type are most likely due to erroneous formatting parameters. When the user has given illegal formatting parameters no formatting will be done. Warning messages are written when a statement cannot be formatted. Examples are statements with long character strings that cannot fit between the left and right margins, and long statements that require more than 19 continuation lines when they are formatted. In cases like this P77

will write a correct, but unformatted, statement. A list of all of the error and warning messages written by the formatter is given in Appendix 3.

5. Machine Dependent Procedures.

There are five machine dependent procedures that must be supplied at installation of P77. Copies of these procedures for a Cyber 750 under NOS are shown in Appendix 5. They are also supplied on the distribution tape.

INTEGER FUNCTION INMAP(C)

This function converts a character to the internal integer representation of the character. The parameter C is the character as read using an A1 format specification. The output (INMAP) is the integer value of the 7-bit ASCII code for the character.

CHARACTER FUNCTION OUTMAP(C)

This function converts the internal integer representation of a character into the external representation of the character. It is the inverse of INMAP. The parameter C is the integer value of the 7-bit ASCII code for the character. The output (OUTMAP) is the representation of this character appropriate for writing it with an A1 format.

INTEGER FUNCTION HOLCHR(HCONST, ICHAR)

This function returns the ith character in A1 format of a Hollerith constant. The value of i is passed in as the value of ICHAR. The value of the Hollerith constant is passed in as the value of HCONST. An example of its usage is given below:

```
MSSG(7) = HOLCHR(5HERROR, 4)
```

This would put the letter "O" in A1 format into the integer array MSSG at position 7. A subsequent write statement like

```
    WRITE (6, 99) MSSG(7)
99  FORMAT(A1)
```

would write the letter "O" to unit 6. Note that passing a Hollerith string in a procedure call and using an integer array to hold characters as shown here is nonstandard F77. However, in Appendix C of the F77 Standard it is recommended that extensions of F77 to support Hollerith strings (as in F66) allow the kind of usage shown above. We believe that most systems support this remnant from F66. We will repair this deviation from strict F77 in a subsequent release of P77.

INTEGER FUNCTION LRS(IVAL, ICOUNT)

This function returns the logical right shift of ICOUNT binary places of the value IVAL. The shift is "zero-fill".

INTEGER FUNCTION LLS(IVAL, ICOUNT)

This function returns the logical left shift of ICOUNT binary places of the value IVAL. The shift is "zero-fill".

6. References.

[ANSI FORTRAN]
ANSI X3.9-1978 FORTRAN. American National Standards Institute, Inc., New York NY 10018.

[Clemm 81]
Geoffrey M. Clemm: FSCAN-81 Report and User's Manual. Tech. Rept. 202(June 1981), Dept. of Computer Science, University of Colorado, Boulder CO 80309.

[Dorrenbacher 76]
J. Dorrenbacher, D. Paddock, D. Wisneski, and L. Fosdick: POLISH, A Fortran Program to Edit Fortran Programs. Tech. Rept. 50(1974, rev. May 1976), Dept. of Computer Science, University of Colorado, Boulder CO 80309.

[Osterweil 82]
Leon J. Osterweil, Stephen Hague, Webb Miller: TOOLPACK Architectural Design - The User's Perspective. (April 15, 1982) Available from Toolpack coordinator - Wayne R. Cowell, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL 60439.

Appendix 1.
List of formatting parameters.

LMARGS
: Left margin for statements. Default value is 7.

RMARGS
: Right margin for statements. Default value is 72.

LMARGC
: Left margin for comments. Default value is 3.

INDNTS
: Indentation for statement lines. Default value is 2.

INDNTC
: Indentation for comment lines. Default value is 2.

MOVESF
: Move format statements to end of program unit. MOVESF = 0 means do not move format statements. MOVESF = 1 means move format statements. Default value is 0.

LABELS
: Value of first label and of the label increment. LABELS = 0 means do not change labels. Default value is 10.

IDENTL
: Fill identification field with line number and abbreviated unit name. IDENTL = 0 means leave identification field blank. IDENTL = 1 means fill in identification field. Default value is 1.

KCMMNT
: Parameter controlling display form of comments. Possible values are 1 through 6. See Figs. 2-7 for examples. Default value is 5.

MCMMNT
: Parameter controlling interpretation of start of comment line. Possible values are 0 through 2. See Figs. 2, 8, 9 for examples. Default value is 2.

Appendix 2.
Example of a main program for invoking P77.

```
      PROGRAM MAINPO                                                    MAIN  10
C     ****************************************************************  MAIN  20
C     * PROGRAM MAINPO                                               *  MAIN  30
C     *                    EXAMPLE OF MAIN PROGRAM                   *  MAIN  40
C     *                                                              *  MAIN  50
C     ****************************************************************  MAIN  60
      INTEGER RMARGS, PARAMS                                            MAIN  70
      COMMON /USER/ LMARGS, RMARGS, LMARGC, INDNTS, INDNTC, MOVESF,     MAIN  80
     $  LABELS, IDENTL, KCMMNT, MCMMNT                                  MAIN  90
      INTEGER STDIN, STDOUT, STDERR, TMP1, TMP2, TMP3, TMP4, TMP5,      MAIN 100
     $  TMP6, TMP7                                                      MAIN 110
      COMMON /FILES/ STDIN, STDOUT, STDERR, TMP1, TMP2, TMP3, TMP4,     MAIN 120
     $  TMP5, TMP6, TMP7                                                MAIN 130
      INTEGER CMNTFL, ERRORS, SOURCE, TABLES                            MAIN 140
      COMMON /LOCNLS/ SOURCE, LISTNG, TABLES, CMNTFL, ERRORS            MAIN 150
C     ************************************                              MAIN 160
C     * NUMBER OF BITS PER WORD - NBTPWD *                              MAIN 170
C     * MACHINE DEPENDENT PARAMETER      *                              MAIN 180
C     ************************************                              MAIN 190
      COMMON /NBTPWC/ NBTPWD                                            MAIN 200
      INTEGER NBTPWD                                                    MAIN 210
C     ************************************                              MAIN 220
C     * VALUE FOR CYBER 60 BITS PER WORD *                              MAIN 230
C     ************************************                              MAIN 240
      DATA NBTPWD/60/                                                   MAIN 250
C                                                                       MAIN 260
      PARAMS = 30                                                       MAIN 270
C     ********************************************                      MAIN 280
C     *                OPEN FILES                *                      MAIN 290
C     * THE FOLLOWING ARE TEMPORARY WORK FILES:  *                      MAIN 300
C     * TMP1, TMP2, TMP3, TMP4, TMP5, TMP6, TMP7 *                      MAIN 310
C     ********************************************                      MAIN 320
      OPEN (PARAMS,FILE='P77PAR',STATUS='OLD')                          MAIN 330
      OPEN (STDIN,FILE='SOURCE',STATUS='OLD')                           MAIN 340
      OPEN (STDOUT,FILE='PRETTY',STATUS='NEW')                          MAIN 350
      OPEN (STDERR,FILE='ERRORS',STATUS='NEW')                          MAIN 360
      OPEN (TMP1,FILE='TMP1',STATUS='NEW')                              MAIN 370
      OPEN (TMP2,FILE='TMP2',STATUS='NEW')                              MAIN 380
      OPEN (TMP3,FILE='TMP3',STATUS='NEW')                              MAIN 390
      OPEN (TMP4,FILE='TMP4',STATUS='NEW')                              MAIN 400
      OPEN (TMP5,FILE='TMP5',STATUS='NEW')                              MAIN 410
      OPEN (TMP6,FILE='TMP6',STATUS='NEW')                              MAIN 420
      OPEN (TMP7,FILE='TMP7',STATUS='NEW')                              MAIN 430
      REWIND PARAMS                                                     MAIN 440
      REWIND STDIN                                                      MAIN 450
      READ (PARAMS,10) LMARGS, RMARGS, LMARGC, INDNTS, INDNTC,          MAIN 460
     $  MOVESF, LABELS, IDENTL, KCMMNT, MCMMNT                          MAIN 470
C                                                                       MAIN 480
   10 FORMAT (10I4)                                                     MAIN 490
```

```
C                                                                       MAIN 500
      CALL SCNPOL                                                       MAIN 510
C                                                                       MAIN 520
C  AT THIS POINT, ON CERTAIN OPERATING SYSTEMS, ALL TEMPORARY WORK      MAIN 530
C  FILES MAY BE REMOVED USING A SYSTEM DEPENDENT SUBROUTINE. ONE CAN    MAIN 540
C  INSERT WHATEVER ROUTINE IS NECESSARY TO REMOVE THESE FILES SINCE     MAIN 550
C  USER WILL NOT NEED THIS INFORMATION AFTER THE PROGRAM TERMINATES.     MAIN 560
C                                                                       MAIN 570
C     CALL RMFILS                                                       MAIN 580
      STOP                                                              MAIN 590
C                                                                       MAIN 600
      END                                                               MAIN 610
```

Appendix 3.
List of Error messages.

Polish Error Messages

*** ERROR 1 ----- (INVALID CONTROL PARAMETER)

One of the formatting parameters (see Appendix 1) has an illegal value.

*** ERROR 2 ----- (RMARGS - LMARGS + 1 < 30)

Line width for statements is too short.

*** ERROR 3 ----- (TABLE 1 EXCEEDS ITS DIMENSION)

Table 1 is defined in the source as TAB1(4,25) within SUBROUTINE POLISH. The second dimension is equal to the maximum number of distinct labels used in DO statements in a program unit. If this error message occurs, it means that more than 25 DO labels are required in textout for the program unit being processed at the time of the message. To remedy this table overflow, change the second dimension of TAB1 as appropriate. Also, the variable NTAB1 must be initialized to whatever value is given to the second dimension of TAB1. This initialization appears near the front of SUBROUTINE POLISH.

*** ERROR 4 ----- (TABLE 2 EXCEEDS ITS DIMENSION)

Table 2 is defined in the source as TAB2(4,50) within SUBROUTINE POLISH. The second dimension is equal to the maximum number of applied occurrences of labels, except in DO statements, in a program unit. The appearance of a lael within a statement is called an "applied occurrance"; the appearance of a label at the left of a statement is called a "defining occurrance". If this error message occurs, it means that more than 50 applied occurrences, excluding DO statements, are required in the program unit being processed at the time of the error message. To remedy this table overflow, change the second dimension of TAB2 as appropriate. Also, the variable NTAB2 must be initialized to whatever value is given to the second dimension of TAB2. This initialization appears near the front of SUBROUTINE POLISH.

*** ERROR 5 ----- (TABLE 3 EXCEEDS ITS DIMENSION)

Table 3 is defined in the source as TAB3(4,125) within SUBROUTINE POLISH. The second dimension is equal to the maximum number of defining occurrences of labels in a program unit. If this error message occurs, it means that more than 125 defining occurrences of labels are required in the program unit being processed at the time of the error message. To remedy this table overflow, change the second dimension as appropriate. Also, the variable NTAB3 must be initialized to whatever value is given to the second dimension of TAB3. This initialization appears near the front of SUBROUTINE POLISH.

*** ERROR 6 ----- (CONTINUE LINES > 19)

Statement with too many continuation lines in source program.

*** MESSAGE 7 --- (CAN NOT BREAK LINE)

Cannot satisfy conditions for breaking a line. Line is not broken in the formatted output.

*** MESSAGE 8 --- (COMMENT BLOCK UUUUSSSS -- UUUUEEEE LEFT AS IS)

Comment block in program unit UUUU (abbreviated name) starting at line identification number SSSS and ending at EEEE is left as it is in textin. This is caused when movement of the block, according to formatting rules, would cause it to extend to the right of column 72.

*** MESSAGE 9 --- (TWO "FM START" IN ROW, LAST AFTER UNIT UUUU)

Improper sequencing of FM START commands in source program; UUUU is abbreviated name of program unit immediately preceding the second of two FM START commands that do not have an intervening FM STOP command.

*** MESSAGE 10 --- (TWO "FM STOP" IN ROW, FIRST AFTER UNIT UUUU)

Improper sequencing of FM STOP commands in source program; UUUU is abbreviated name of program unit immediately preceding the first of two FM STOP commands that do not have an intervening FM START command.

*** MESSAGE 11 --- ("FM START" IN PROGRAM UNIT ????)

Improper placement of FM START command. This command can only be placed between program units.

*** MESSAGE 12 --- ("FM STOP" IN PROGRAM UNIT ????)

Improper placement of FM STOP command. This command can only be placed between program units.

Scanner Error Messages

SCAN ERROR NO. xx NEAR TOKEN yy
FATAL ERROR NO. xx NEAR TOKEN yy

See discussion in text "Error Handling".

Appendix 4.
List of external files.

P77 uses eleven files (one is optional). These files must be opened in the main program (supplied by the user) as illustrated in Appendix 2. The files are identified below by the internal name used for the unit number of the file.

PARAMS
: This file is optional and listed here only for completeness. It is used in the sample main program, Appendix 2, for initializing the formatting parameters. The initialization is unnecessary if default values are used.

STDIN
: This file holds the program to be processed by P77.

STDOUT
: This file holds the output from P77, or if a syntax error is found by the scanner it holds the listing file (see Fig. 10 for example of a listing file).

STDERR
: This file holds error messages and warning messages written by P77.

TMP1-7
: These seven files are work files used by P77.

Appendix 5
Cyber 750 Machine Dependent Procedures

```
      INTEGER FUNCTION INMAP(C)                                          INMA  10
      INTEGER C                                                          INMA  20
      COMMON /MAPBLK/ MAPIN, MAPOUT                                      INMA  30
      INTEGER MAPIN(0:63), MAPOUT(0:63), SMALL                           INMA  40
      DATA MASK/O"7700"/                                                 INMA  50
C                                                                        INMA  60
      SMALL = LRS(C,54)                                                  INMA  70
      IF ((SMALL.LT.0).OR.(SMALL.GT.63)) GO TO 10                        INMA  80
      INMAP = MAPIN(SMALL)                                               INMA  90
      RETURN                                                             INMA 100
C                                                                        INMA 110
   10 PRINT *, 'ERROR IN INMAP, BAD CODE'                                INMA 120
      RETURN                                                             INMA 130
C                                                                        INMA 140
      END                                                                INMA 150
C
      INTEGER FUNCTION OUTMAP(C)                                         OUTM  10
      INTEGER C                                                          OUTM  20
      COMMON /MAPBLK/ MAPIN, MAPOUT                                      OUTM  30
      INTEGER MAPIN(0:63), MAPOUT(0:63), KEY                             OUTM  40
      DATA MASK/O"5555555555555555"/                                     OUTM  50
C                                                                        OUTM  60
      KEY = C - 32                                                       OUTM  70
      IF ((KEY.LT.0).OR.(KEY.GT.63)) GO TO 10                            OUTM  80
      OUTMAP = OR(LLS(MAPOUT(KEY),54),MASK)                              OUTM  90
      RETURN                                                             OUTM 100
C                                                                        OUTM 110
   10 PRINT *, 'ERROR IN OUTMAP, BAD CHARACTER'                          OUTM 120
      RETURN                                                             OUTM 130
C                                                                        OUTM 140
      END                                                                OUTM 150
C
      INTEGER FUNCTION LRS(IVAL,ICOUNT)                                   LRS  10
C                                                                         LRS  20
      LRS = SHIFT(IVAL,-ICOUNT).AND.(.NOT.MASK(ICOUNT))                   LRS  30
      RETURN                                                              LRS  40
C                                                                         LRS  50
      END                                                                 LRS  60
C
      INTEGER FUNCTION LLS(IVAL,ICOUNT)                                   LLS  10
C                                                                         LLS  20
      LLS = SHIFT(IVAL,ICOUNT)                                            LLS  30
      RETURN                                                              LLS  40
C                                                                         LLS  50
      END                                                                 LLS  60
C
      INTEGER FUNCTION HOLCHR(HCONST,ICHR)                               HOLC  10
      INTEGER WORD, HCONST(10)                                           HOLC  20
C                                                                        HOLC  30
```

```
      WORD = (ICHR-1)/10 + 1                                                HOLC  40
      ICHPOS = (LLS(HCONST(WORD),(ICHPOS-1)*6).AND.                         HOLC  50
     $  O"7700000000000000000") .OR. O"0055555555555555555"                 HOLC  60
      RETURN                                                                HOLC  70
C                                                                           HOLC  80
      END                                                                   HOLC  90
C
      BLOCK DATA MAPS
      COMMON /MAPBLK/ MAPIN, MAPOUT
      INTEGER MAPIN(0:63), MAPOUT(0:63)
C
      DATA MAPIN(0)    / 58 /
      DATA MAPIN(1)    / 65 /
      DATA MAPIN(2)    / 66 /
      DATA MAPIN(3)    / 67 /
      DATA MAPIN(4)    / 68 /
             :
             .
      DATA MAPIN(59)   / 62 /
      DATA MAPIN(60)   / 64 /
      DATA MAPIN(61)   / 92 /
      DATA MAPIN(62)   / 94 /
      DATA MAPIN(63)   / 59 /
C
      DATA MAPOUT(0)   / 45 /
      DATA MAPOUT(1)   / 54 /
      DATA MAPOUT(2)   / 52 /
      DATA MAPOUT(3)   / 48 /
      DATA MAPOUT(4)   / 43 /
             :
             .
      DATA MAPOUT(59)  / 49 /
      DATA MAPOUT(60)  / 61 /
      DATA MAPOUT(61)  / 50 /
      DATA MAPOUT(62)  / 62 /
      DATA MAPOUT(63)  / 53 /
C
      END
```

Appendix 6
Table of I/O unit names and numbers.

Name	Number	Remark
STDIN	27	Textin.
STDOUT	28	Textout, or listing if errors in scan.
STDERR	29	Error messages.
PARAMS	30	Formatting parameters.
TMP1	31	
TMP2	32	
TMP3	33	
TMP4	34	
TMP5	35	Tokens from scanner.
TMP6	36	Buffer for comments.
TMP7	37	Listing from scanner.

The unit numbers are assigned in the program unit BLOCK DATA TOKLET.