# FORTRAN-8Ø User's Manual

## FOREWARD

This manual describes how to use the FORTRAN-8Ø compiler and associated software under CP/M or a similar Disk Operating System. Refer to the FORTRAN-8Ø manual for an extensive description of FORTRAN syntax and semantics.

## Table of Contents

# SECTION 1
# Compiling FORTRAN Programs

## 1.1 The FORTRAN-8Ø and MACRO-8Ø Command Scanner

### 1.1.1 Format of Commands

FORTRAN-8Ø and MACRO-8Ø general commands are as follows:

```
objprog-dev:filename.ext,list-dev:filename.ext=source-dev:filename.ext
```

objprog-dev:
    The device on which the object program is to be written.

list-dev:
    The device on which the program listing is written.

source-dev:
    The device from which the source-program input to FORTRAN-8Ø or MACRO-8Ø is obtained. If a device name is omitted, it defaults to A:.

filename.ext
    The filename and filename extension of the object program file, the listing file, and the source file.

If no extension is supplied, it defaults to the following:

| | |
|---|---|
| object-file: | .REL |
| listing-file: | .LST |
| Source-file: | .FOR (FORTRAN-8Ø)<br>.MAC (MACRO-8Ø) |

Either the object file or the listing file or both may be omitted. An object file is always created, unless a listing file is made. If neither a listing file nor an object file are desired, place only a comma to the left of the equal (=) sign. If the names of the object file or the listing file are omitted, they will default to the name of the source file.

Examples:

```
A>F8Ø

*=TEST                  (Compile the program TEST.FOR
```

|  |  |
|---|---|
|  | and place the object in TEST.REL) |
| `*,TTY:=TEST` | (Compile program TEST.FOR and list program on the terminal. No object is generated.) |
| `*TESTOBJ=TEST.FOR` | (Compile program TEST.FOR and put object in TESTOBJ.REL) |
| `*TEST,TEST=TEST` | (Compile TEST.FOR, put object TEST.REL and listing in TEST.LST) |
| `*,=TEST.FOR` | (Compile TEST.FOR but produce no object or listing file. useful for checking for errors.) |

## 2.1.1 FORTRAN-8Ø Compilation Switches

A number of different switches may be given in the command string which affect the format of the listing file, etc. Each switch should be preceeded by a slash (/):

| Switch | Action |
|---|---|
| O | Print all Listing Addresses, etc. in Octal (Default for ALTAIR DOS) |
| H | Print all Listing Addresses, etc. in Hexadecimal (Default for non-ALTAIR versions) |
| N | Don't list Generated code |
| R | Force Generation of an Object file |
| L | Force Generation of a Listing file |
| P | Each /P allocates an extra 1ØØ bytes of stack space for use during compilation. Use /P if you get stack overflow errors during compilation. Otherwise not needed. |

Examples:

```
*,TTY:=MYPROG/N (Compile file MYPROG.FOR and list
                program on terminal but without generated code.)

*=TEST/L        (Compile TEST.FOR
                with object file TEST.REL and
                listing file TEST.LST)

*=BIGGONE/P/P   (compile file BIGGONE.FOR
                and produce object file BIGGONE.REL.
                Compiler is allocated 2ØØ extra bytes
                of stack space.)
```

## 2.2 Sample Compilation

```
A>F80

*EXAMPL,TTY:=EXAMPL

FORTRAN-80 Ver. 1.0 Copyright 1977 (C) By Microsoft
00100               PROGRAM EXAMPLE
00200               INTEGER X
00300               I = 2**8 + 2**9 + 2**10
00400               DO1J=1,5
*****   0000'       LXI     H,0700
*****   0003'       SHLD    I
00500   C           CIRCULAR SHIFT I LEFT 3 BITS -- RESULT IN X
00600               CALL    CSL3(I,X)
*****   0006'       LXI     H,0001
*****   0009'       SHLD    J
00800               WRITE(3,10) I,X
*****   000C'       LXI     D,X
*****   000F'       LXI     H,I
*****   0012'       CALL    CSL3
*****   0015'       LXI     B,0007"
*****   0018'       LXI     D,10L
*****   001B'       LXI     H,[     03      00]
*****   001E'       CALL    $WR
00850   1           I=X
*****   0021'       LXI     B,X
*****   0024'       LXI     D,I
*****   0027'       LXI     H,[     01      00]
*****   002A'       MVI     A,03
*****   002C'       CALL    $I0
*****   002F'       CALL    $ND
00900   10          FORMAT(2I15)
*****   0032'       LHLD    X
*****   0035'       SHLD    I
*****   0038'       LHLD    J
*****   003B'       INX     H
*****   003C'       MVI     A,05
*****   003E'       SUB     L
*****   003F'       MVI     A,00
*****   0041'       SBB     H
*****   0042'       JP      0009'
01000               END
*****   0045'       CALL    $EX
*****   0048'       0100
*****   004A'       0000
*****   004C'       0300

Program Unit Length=004E (78) Bytes
Data Area Length=0011 (17) Bytes

Subroutines Referenced:

$I0
CSL3
```

```
$WR
$ND
$EX

Variables:

X        0001"
I        0003"
J        0005"

LABELS:

1L       0032'
10L      000B"

*^C
A>
```

See section 4.3 for a listing of the MACRO-80 subroutine CSL3.

```
A>LINK

*EXAMPL,EXMPL1/G
[26E2   273A    39]
[BEGIN EXECUTION]

          1792          14336
         14336         -16383
        -16383             14
            14            112
           112            896
A>
```

## 1.3 FORTRAN Compiler Error Messages

The FORTRAN-80 Compiler detects two kinds of errors, Warnings and Fatal errors.

When a warning is issued, compilation continues with the next item on the source line. When a Fatal error is found, the compiler will ignore the rest of the logical line, including any continuation lines. Warning messages are preceeded by percent (%) signs, and Fatal errors by question marks (?).

For either type of error, the program should be changed so that it compiles without errors. No guarantee is made that a program which compiles with errors will execute in a sensible fashion.

The editor line number, if any, or the physical line number is printed next, followed by the error code if long error messages are not present in the compiler, or by the text of the error message if the compiler supports long error messages.

Example:

```
?Line 25: Mismatched Parentheses

%Line 16: Missing Integer Variable
```

Fatal Errors:

| Error Number | Message |
|---|---|
| 100 | Illegal Statement Number |
| 101 | Statement Unrecognizable or Misspelled |
| 102 | Illegal Statement Completion |
| 103 | Illegal DO Nesting |
| 104 | Illegal Data Constant |
| 105 | Missing Name |
| 106 | Illegal Procedure Name |
| 107 | Invalid DATA Constant or Repeat Factor |
| 108 | Incorrect Number of DATA Constants |
| 109 | Incorrect Integer Constant |
| 110 | Invalid Statement Number |
| 111 | Not a Variable Name |
| 112 | Illegal Logical Form Operator |
| 113 | Data Pool Overflow |
| 114 | Literal String is too large |
| 115 | Invalid Data List Element in I/O |
| 116 | Unbalanced DO Nest |
| 117 | Identifier Too Long |
| 118 | Illegal Operator |

| | |
|---|---|
| 119 | Mismatched Parentnesis |
| 120 | Consecutive Operators |
| 121 | Improper Subscript Syntax |
| 122 | Illegal Integer Quantity |
| 123 | Illegal Hollerith Constuction |
| 124 | Backwards DO reference |
| 125 | Illegal Statement Function Name |
| 126 | Illegal Character for Syntax |
| 127 | Statement is out of Sequence |
| 128 | Missing Integer Quantity |
| 129 | Invalid Logical Operator |
| 130 | Illegal Item following INTEGER or REAL or LOGICAL |
| 131 | Premature End Of File on input device |
| 132 | Illegal Mixed Mode Operation |
| 133 | Function Call with No Parameters |
| 134 | Stack Overflow |
| 135 | Illegal Statement Following Logical IF |

Warnings:

| | |
|---|---|
| 0 | Duplicate Statement Label |
| 1 | Illegal DO Termination |
| 2 | Block Name = Procedure Name |
| 3 | Array Name Misuse |
| 4 | COMMON Name Usage |
| 5 | Wrong Number of Subscripts |
| 6 | Array Multiply EQUIVALENCEd Within a Group |
| 7 | Multiple EQUIVALENCE of COMMON |
| 8 | COMMON Base Lowered |
| 9 | Non-COMMON Variable in BLOCK DATA |
| 10 | Empty List for Unformatted WRITE |
| 11 | Non-Integer Expression |
| 12 | Operand Mode Not Compatible with Operator |
| 13 | Mixing of Operand Modes Not Allowed |
| 14 | Missing Integer Variable |
| 15 | Missing Statement Number on FORMAT |
| 16 | Zero Repeat Factor |
| 17 | Zero Format Value |
| 18 | Format Nest Too Deep |
| 19 | Statement Number not FORMAT Associated |
| 20 | Invalid Statement Number Usage |
| 21 | No Path to this Statement |
| 22 | Missing Do Termination |
| 23 | Code Output in BLOCK DATA |
| 24 | Undefined Labels Have Occurred |
| 25 | RETURN in a Main Program |
| 26 | STATUS Error on READ |
| 27 | Invalid Operand Usage |
| 28 | Function with no Parameter |
| 29 | Hex Constant Overflow |
| 30 | Division by Zero |
| 31 | Missing RETURN in Subprogram |

# SECTION 2

# Linking FORTRAN Programs

## 2.1 The LINK-80 Command Scanner

### 2.1.1 Format of Commands

Each command to LINK-80 consists of a number of filenames and switches separated by commas:

```
objdev1:filename.ext/switch1,objdev2:filename.ext,.....
```

If the input device for a file is omitted, it defaults to the current logged disk. If the extension of a file is omitted, it defaults to .REL.

After each line is typed, LINK will load or search (see /S below) the named files. After LINK finishes this process, it will list all symbols that remained undefined followed by an asterisk.

Example:

```
A>LINK
*MAIN
 SUBR1*                   (SUBR1 is undefined)
*SUBR1
*/G                       (Starts Execution - see below)
```

Typically, to execute a FORTRAN program and subroutines, The user should type the list of filenames followed by /G (begin execution). If the FORTRAN programs require any FORTRAN Library routines, They will be satisfied automatically by searching FORLIB.REL before execution begins.

If the user wishes to first search libraries of his own, he should append the filenames followed by /S to the end of the loader command string.

### 2.1.2 LINK-80 Switches

LINK-80 has a number of switches that specify actions which affect the loading process. These switches are:

| Switch | Action |
|---|---|
| R | Reset. Put loader back in its initial state. |

| | |
|---|---|
| | Use /R if you loaded the wrong file by mistake and want to restart. /R takes effect as soon as it is encountered in a command string. |
| E | Exit from LINK-80 back to the Operating System. Use /E if you want to load a program, then save the memory image. |
| G | Start execution of the program as soon as the current command line has been interpreted. FORLIB.REL will be searched on the current disk to satisfy any undefined globals if they exist. Before execution actually begins, LINK-80 prints 3 numbers and a BEGIN EXECUTION message. The 3 numbers are the start address, the address of the next available byte, and the number of sectors used. |
| U | List all undefined globals as soon as the current command line has been interpreted. |
| M | Map. List all defined globals and their values, and all undefined globals followed by an asterisk. |
| S | Search the filename immediately preceeding the /S in the command string to satisfy any undefined globals. |

Examples:

```
*/M                (List all globals)

*MYPROG,SUBROT,MYLIB/S
                   (Load MYPROG.REL and SUBROT.REL and
                   then search MYLIB.REL
                   to satisfy any remaining
                   undefined globals.)

*/G                (Begin execution of
                   main program)
```

## 2.3 Format of LINK Compatible Object files

LINK compatible object files consist of a bit stream. Individual fields within the bit stream are not aligned on byte boundaries, except as noted below. Use of a bit stream for relocatable object files keeps the size of object files to a minimum, thereby decreasing the number of disk reads/writes.

There are two basic types of load items: Absolute and Relocatable.

The first bit of an item indicates one of the above types. If the first bit is a Ø, the following 8 bits are loaded as an absolute byte. If the first bit is a 1, then the next 2 bits are used to indicate the type of item. The relocatable items are broken down into 4 types:

| | |
|---|---|
| ØØ | Special LINK item (see below). |
| Ø1 | Program Relative. Load the following 16 bits after adding the current Program base. |
| 1Ø | Data Relative. Load the following 16 bits after adding the current Data base. |
| 11 | Common Relative. Load the following 16 bits after adding the current Common base. |

Special LINK items consist of the bit stream 1ØØ followed by a four bit control field, an optional A field wich consists of a two bit address type which is the same as the two bit field above except that ØØ specifies absolute addressing, and a B field which consists of 3 bits giving a symbol length followed by 8 bits for each character of the symbol:

```
                  A               B
1 ØØ xxxx [yy two byte value][zzz characters of symbol name]
```

| | |
|---|---|
| xxxx | Four bit field Ø-17 below |
| yy | Two bit address type field |
| zzz | Three bit symbol length field |

The following special types have a B-field only:

| | |
|---|---|
| Ø | Entry symbol (name for search) |
| 1 | Select COMMON Block |
| 2 | Program name |
| 3 | Reserved for Future Expansion |
| 4 | Reserved for Future Expansion |

The following special types have both an A-field and a B-field:

| | |
|---|---|
| 5 | Define COMMON size |
| 6 | Chain External (A is head of address chain, B is name of external symbol) |
| 7 | Define Entry point (A is address, B is name) |
| 8 | Reserved for Future expansion |
| 9 | Reserved for Future expansion |

The following special types have an A-field only:

| | |
|---|---|
| 10 | Define size of Data area (A is size) |
| 11 | Set loading location counter to A |
| 12 | Chain address. A is head of chain, replace all entries in chain with current location counter. The last entry in the chain has an address field of absolute zero. |
| 13 | Define Program size (A is size) |
| 14 | End program (forces to byte boundary) |

The following data types have neither an A nor a B field.

| | |
|---|---|
| 17 | End File |

## 2.1.2 LINK-80 Error Messages

LINK-80 has the following error messages:

| | |
|---|---|
| ?No Start Address | A /G switch was issued, but no main program had been loaded. |
| ?Loading Error | The last file given for input was not a properly formatted LINK-80 object file. |
| ?Fatal Table Collision | Not enough memory to load program. |
| ?Command Error | Unrecognizable LINK-80 Command. |
| ?File Not Found | A file in the command string did not exist. |
| %2nd COMMON Larger /XXXXXX/ | The first definition of COMMON block /XXXXXX/ was not the largest definition. Re-order module loading sequence or change COMMON block definitions. |

%Mult. Def. Global YYYYYY

More than one definition for the global (internal) symbol YYYYYY was encountered during the loading process.

## SECTION 3

## The MACRO-8Ø Assembler

### 3.1 Format of MACRO-8Ø Commands

#### 3.1.1 MACRO-8Ø Command Strings

The format of MACRO-8Ø Command strings is identical to the format of FORTRAN-8Ø command strings. See section 1.1.1.

The default extension for MACRO-8Ø source files is .MAC.

#### 3.1.2 MACRO-8Ø Switches

MACRO-8Ø Switches are the same as FORTRAN-8Ø switches except that /P, /N, and /O have no effect. See section 1.1.2.

### 3.2 Fomat of MACRO-8Ø Source Files

MACRO-8Ø is a two pass assembler that outputs a relocatable object module and produces a listing during the second pass.

In general, MACRO-8Ø accepts a source file that is almost identical to source files for INTEL compatible assemblers.

A short descrition of the features of the assembler is given below.

A. Names

All names are 1-6 characters long with the first character being A-Z or $, and the remaining characters being A-Z, Ø-9 or $.

B. Constants

1. Decimal: Numbers formed from decimal digits and not having a leading zero. The allowable range is 65535 to -65535.

2. Octal: Numbers formed from octal digits and having a leading zero. The allowable range is 0177777 to -0177777.

3. Hex: Numbers formed from 1-4 hexadecimal digits and having the form x'hnhh'. 1 or 3 digit values are treated as though zero was to the left (i.e. X'A' and x'0A' are the same). The allowable range is X'FFFF' to -X'FFFF'.

4. Character: One or two ASCII characters preceeded and followed by ". (i.e. "a" or "BC"). The character " is not allowed.

## C. Labels

A Label is a name that does not contain an imbedded space and is terminated by a colon (:).

## D. Operators

An Operator consists of an 8080 mnemonic or one of the pseudo-operations described below (i.e. MVI, RRC or EQU).

## E. Address Expressions

An address expression consists of a Names or a Constant or an address expression + or - an address expression. An Address expression uses the current assigned address of a Name or the 16 bit value of a Constant to form a 16 bit value which, after the expression is totally calculated, is truncated to the field size required by the operator. An expression is evaluated from left to right and may not contain any imbedded blanks (except those appearing inside Character Constants). An expression is terminated by a ';' or a tab which indicates the end of the operand portion of a statment. The operator MOD (i MOD j) is available for use in address expressions.

## F. Remarks

A Remark is indicated by a statement whose first character is a ; (in which case the whole statement is a remark) or by any characters following the end of an operand field. A remark is always terminated by a Carriage Return.

## G. Form

A statement consists of an optional label (if it is absent, at least one space or tab must be used in lieu of a label), followed by an operator, followed by as many address expressions as the operator requires, followed by an optional remark, and terminated by a Carriage Return character. Multiple blanks or tabs may be used to improve readability (except inside Character Constants or Character Strings).

## II. Pseudo Operations

### A. Define Byte

```
        DB      E1,E2,...,En            or DB "Character-String"
```

Each of the address expressions E1, E2, ... En is evaluated and stored in n successive bytes. The character string form allows the storing of multiple ASCII characters and may be mixed with the address expression form. Two-character Character Constants are treated as Character-Strings unless they are combined with another address expression.

### B. Define Character

```
        DC     "Character-String"
```

Each character in the character-String is stored as one byte with its high-order bit set to zero except for the last byte which has its high-order bit set.

### C. Define Space

```
        DS     E
```

The address expression E is evaluated and that many bytes of space are allocated. All names used in E must be defined prior to the DS statement.

### D. Define word

```
        DW      E1, E2, ...., En
```

Each address expression is evaluated and stored as n successive words.

E. Program Termination

```
        END     E
```

This statement is the last statement of each program. The optional address expression E gives the program exection address. If E is absent no remark may appear on the statement. If E evaluates to absolute 0, it is equivalent to no execution address.

F. Terminated Conditional Assembly

```
        ENDIF
```

Terminates Conditional assembly initiated by a previous IFF or IFT.

G. Define Entry Points

```
        ENTRY   N1, N2, ...,Nn
```

The names N1, N2, ... Nn are entry points from external programs and act as names for the program being assembled. The names must appear in an ENTRY statement prior to their appearance as a Label.

H. Define Equivalence

```
        Label   EQU     E
```

The Label of the EQU statement is assigned the address given by address expression E. The Label is required and must not have previously appeared as a Label. All names used in E must be defined prior to the EQU statement.

I. Define External

```
        EXT     N1, N2, ..., Nn
```

The names N1, N2, .. Nn are defined to be external references and may not have been used as a Label .

J. False Conditional Assembly

```
        IFF     E
```

The address expression E is evaluated and if it is False (=0) all staements down to eh next ENDIF are assembled and if it is True (not =0) they are not. No nesting of conditional assemblies is permitted.

K. True Conditional Assembly

```
        IFT     E
```

The address expression E is evaluated and if it is True (not =Ø) all statements down to the next ENDIF are assembled and if it is False (=Ø) they are not. No nesting of conditional assemblies is permitted.

L. Define Origin

```
        ORG     E
```

The address expression E is evaluated and the assembler assigns generated code starting with that value. All names used in E must be defined prior to the ORG statement and the Mode of E must not be External.

M. Page Break

```
        PAGE
```

A page break will occur on the listing. The Page statement will not list and no code is generated.

Any Pseudo-Operation may have a Label but except for EQU, the Label will be defined to be the value of the assembler location counter at the start of the Pseudo Operation.

## III. Notes

1. * indicates the value of the location counter at the start of the statement.
2. A Character-String may not contain the character ".
3. When the assembler is entered, the origin is assumed to be Relative-Ø.

4. Address expressions used in the conditional assembly pseudo-operations IFF and IFT must have all names defined prior to the use in the expression and the expression must be Absolute.

5. Address expressions whose final mode is other than Absolute must generate assembly data that is stored as two bytes.

6. The following Names are defined by the assembler to have the indicated Absolute values.

| | | | | |
|---|---|---|---|---|
| A=7 | B=0 | C=1 | D=2 | E=3 |
| H=4 | L=5 | M=6 | SP=6 | PSW=6 |

## 3.3 Sample Assembly

```
A>M80

*EXMPL1,TTY:=EXMPL1

  000000                       00100   ;       CSL3(P1,P2)
  000000                       00200   ;       SHIFT P1 LEFT CIRCULARLY 3 BITS
  000000                       00300   ;       RETURN RESULT IN P2
  000000                       00400           ENTRY   CSL3
  000000                       00450   ; GET VALUE OF FIRST PARAMETER
  000000 176                   00500   CSL3:   MOV     A,M
  000001 043                   00600           INX     H
  000002 146                   00700           MOV     H,M
  000003 157                   00800           MOV     L,A
  000004                       00850   ; SHIFT COUNT
  000004 006 003               00900           MVI     B,3
  000006 257                   01000   LOOP:   XRA     A
  000007                       01050   ; SHIFT LEFT
  000007 051                   01100           DAD     H
  000010                       01150   ; ROTATE IN CY BIT
  000010 027                   01200           RAL
  000011 205                   01300           ADD     L
  000012 157                   01400           MOV     L,A
  000013                       01450   ; DECREMENT COUNT
  000013 005                   01500           DCR     B
  000014                       01550   ; ONE MORE TIME
  000014 302 000006 '          01600           JNZ     LOOP
  000017 353                   01700           XCHG
  000020                       01750   ; SAVE RESULT IN SECOND PARAMETER
  000020 163                   01800           MOV     M,E
  000021 043                   01900           INX     H
  000022 162                   02000           MOV     M,D
  000023 311                   02050           RET
  000024                       02100           END

CSL3     000000' LOOP     000006'
*
```

## 3.4 MACRO-80 Errors

MACRO-80 errors are indicated by a one character flag in column one of the listing file. If a listing file is not being printed on the terminal, each line in error is also printed or displayed on the terminal. Below is a list of the MACRO-80 Error Codes:

| Code | Meaning |
|---|---|
| B | Block name in DATA |

|  |  |
|---|---|
| C | Too Many COMMONs |
| D | Bad Octal or Hex Digit |
| E | Expression Error |
| L | No Label in EQU |
| M | Label or Symbol defined more than once |
| N | Name too long |
| O | Bad Operator (Opcode) |
| T | Illegal Field Termination |
| U | Undefined Symbol |
| V | Value Error to MOD |
| 2 | Missing second Field for Opcode |

## SECTION 4
## Runtime Error Messages

Warning Errors

| Two Character Code | Meaning |
|---|---|
| IB | Input Buffer Limit Exceeded |
| TL | Too Many Left Parentheses in FORMAT |
| OB | Output Buffer Limit Exceeded |
| DE | Decimal Exponent Overflow (Number in input stream had an exponent larger than 99) |
| IS | Integer Size Too Large |
| BE | Binary Exponent Overflow |
| IN | Input Record Too Long |
| OV | Arithmetic Overflow |
| CN | Conversion Overflow on REAL to INTEGER Conversion |
| SN | Argument to SIN Too Large |
| A2 | Both Arguments of ATAN2 are Ø |
| IO | Illegal I/O Operation |
| BI | Buffer Size Exceeded During Binary I/O |
| RC | Negative Repeat Count in FORMAT |

Fatal Errors:

| | |
|---|---|
| ID | Illegal FORMAT Descriptor |
| FØ | FORMAT Field Width is Zero |
| MP | Missing Period in FORMAT |
| FW | FORMAT Field Width is Too Small |
| IT | I/O Transmission Error |
| ML | Missing Left Parenthesis in FORMAT |
| DZ | Division by Zero, REAL or INTEGER |
| LG | Illegal Argument to LOG Function (Negative or Zero) |
| SQ | Illegal Argument to SQRT Function (Negative) |
| DT | Data Type Doesn't Agree With FORMAT Specification |
| EF | EOF Encountered on READ |

Runtime errors are surrounded by asterisks as follows:

```
**FW**
```

Fatal errors cause execution to cease (control is returned to the operating system). Execution continues after a warning error. However, after 2Ø warnings, execution ceases as in a fatal error.

## SECTION 5
## Operating Systems

This section describes the use of FORTRAN-80 under the different disk operating systems.

### 5.1 CPM

Available devices are:

|  |  |
|---|---|
| A:, B: | (Disk Drives) |
| HSR: | (High Speed Reader) |
| LST: | (Line Printer) |
| TTY: | (Teletype or CRT) |

Disk file names are up to 8 characters long, with 3 character extensions. The standard extensions are:

|  |  |
|---|---|
| FOR | FORTRAN-80 source file |
| MAC | MACRO-80 source file |
| REL | Relocatable object file |
| LST | Listing file |
| COM | Absolute file |

CPM command lines and files are supported; i.e., a FORTRAN-80, MACRO-80, or LINK-80 command line may be placed in the same line with the CPM run command. For example, the command:

```
A>F80 =TEST
```

will cause CPM to load and run the FORTRAN-80 compiler, which will then compile the program TEST.FOR and create the file TEST.REL. This is equivalent to the following series of commands:

```
A>F80
*=TEST
*^C
A>
```

### 5.2 DTC Microfile

Available devices are:

```
    DØ:,D1:,D2:,D3:  (Disk Drives)
    TTY:             (Teletype or CRT)
```

Disk file names are up to 5 characters long with 1 character extensions. The standard extensions are:

```
    F -- FORTRAN-8Ø source file
    M -- MACRO-8Ø source file
    O -- Relocatable object file
    L -- Listing file
```

Command lines are supported in a manner similar to CPM (Section 5.1).

## 5.3 ALTAIR DOS

Available devices are:

```
    FØ:, F1:, F2:, ...  (Disk Drives)
    TTY:                (Teletype or CRT)
```

Disk file names are up to 5 characters long with 3 character extensions. The standard extensions are:

```
    FOR -- FORTRAN-8Ø source file
    MAC -- MACRO-8Ø source file
    REL -- Relocatable object file
    LST -- Listing file
```

Command lines are not supported.

FORTRAN-80 now provides the capability of disk file access via FORTRAN programs. Logical Unit Numbers 6-1Ø are preassigned to disk files. A READ or WRITE to one of these LUN's automatically OPEN's the file for input or output respectively, if it is not already open. The file remains open until closed by an ENDFILE command, or until normal program termination. A file which is OPENed by a READ or WRITE statement has a default name which depends upon the operating system:

<u>CPM, ISIS II</u>
FORTØ6.DAT, FORTØ7.DAT,..., FORT1Ø.DAT

<u>ALTAIR</u>
FORØ6DAT, FORØ7DAT,..., FOR1ØDAT

<u>DTC</u>
FORØ6D  FORØ7D,..., FOR1ØD

In each case the LUN is incorporated into the default file name.

Alternatively, a file may be OPENed using the OPEN subroutine. LUN's 1-5 may also be assigned to disk files with OPEN. Note that if LUN 3 is assigned to disk, that is where any system messages will go. The form of an OPEN call is:

CALL OPEN (LUN, Filename, Drive)

where:

LUN = a Logical Unit Number to be associated with the file (must be an integer between 1 and 1Ø).

Filename = an ASCII name which the operating system will associate with the file. The Filename should be a Hollerith or Literal constant, or a variable or array name, where the variable or array contains the ASCII name. The Filename should be blank filled to exactly the number of characters allowed by the operating system:

<u>CPM</u>

11 characters

<u>ALTAIR</u>

8 characters

DTC

6 characters

ISIS II

6 characters followed by a "." followed by a 3 character extension

Drive = the disk drive number on which the file exists or will exist (must be an integer within the range allowed by the operating system -- usually Ø or 1).

The OPEN subroutine allows the program to specify a filename and device to be associated with a LUN, whereas the default specifies a default name and uses the currently selected disk drive.

An OPEN of a non-existent file creates a null file of the appropriate name. An OPEN of an existing file (followed by an output) deletes the existing file. An OPEN of an existing file followed by an input allows access to the current contents of the file.

The ENDFILE and REWIND commands allow further program control of disk files. The form of the commands is:

```
ENDFILE(L) or REWIND(L)
```

where L is a LUN. ENDFILE(L) closes the file associated with LUN L. REWIND(L) closes the file associated with LUN L, then opens it again.

NOTE

The programmer should exercise caution when outputting to disk files. If output is done to an existing file, the existing file will be deleted and replaced with a new file of the same name.

The FORTRAN-80 library contains a number of potentially useful subroutines which may be referenced by the user from FORTRAN or Assembly programs. In the following descriptions, $AC refers to the floating accumulator; $AC is the address of the low byte of the mantissa; $AC +3 is the address of the exponent. Brackets are used in the descriptions to indicate direct or indirect addressing. For example:

(H,L) means the contents of the H and L registers.

[H,L] means the contents of the memory location(s) pointed to by the H and L registers.

The following routines are available to the programmer:

| | |
|---|---|
| $AA | $AC := $AC + FLOAT (H,L) |
| $AB | $AC := $AC + [H,L] |
| $AT | Argument transfer; see Appendix C of the FORTRAN manual |
| $BA | Backspace; (H,L) = LUN |
| $CA | $AC = FLOAT (H,L) |
| $CG | Computed GOTO processor; (H,L) = index value, other parameters<br>are passed inline: no. of labels — 1 byte<br>address of label 1 — 2 bytes<br>address of label 2 — 2 bytes<br>:<br>address of label n — 2 bytes |
| $CH | (H,L) := IFIX ($AC) |
| $D9 | (H,L) := (D,E) / (H,L) : (D,E) := remainder |
| $DA | $AC := $AC / FLOAT (H,L) |
| $DB | $AC := $AC / [H,L] |
| $E9 | (H,L) := (H,L) ** (D,E) |
| $EA | $AC := $AC ** (H,L) |
| $EB | $AC := $AC ** [H,L] |
| $EN | ENDFILE ; (H,L) = LUN |
| $EXPB | $AC := 2.Ø ** $AC |
| $IØ | Integer I/O transfer |
| $I1 | Real I/O transfer |
| $I2 | Logical I/O transfer<br>(A) = no. of parameters<br>parameter 1 = no. of elements in array<br>parameters 2 — n = address of variables to transfer |

| | |
|---|---|
| $IOERR | Prints "Illegal I/O Operation" error |
| $LI | $AC := [H,L] |
| $M9 | (H,L) := (H,L) * (D,E) |
| $MA | $AC := $AC * FLOAT (H,L) |
| $MB | $AC := $AC * [H,L] |
| $NB | $AC := -$AC |
| $ND | Terminate I/O transfer |
| $PA/$ST | PAUSE/STOP ; 6 ASCII characters are passed inline |
| $RE | REWIND ; (H,L) = LUN |
| $SA | $AC := $AC - FLOAT (H,L) |
| $SB | $AC := $AC - [H,L] |
| $TI | [H,L] := $AC |
| $RD/$WR | READ/WRITE initialize; |
| | parameter 1 = LUN |
| | parameter 2 - address of FORMAT or Ø for binary I/O |
| | parameter 3 = ERR address or Ø |
| | parameter 4 = EOF address or Ø |