

FORMAT FIELD DESCRIPTORS

r repeat specification (optional)
w field width
.d number of characters to right of decimal place
n integer count (optional)

Alphanumeric

input truncation of right characters if needed
output data is right justified
rAw Alphanumeric field descriptor
nH Hollerith descriptor

Numeric

input leading blanks are ignored
output Numeric data is right justified
rDw.d Double precision floating point descriptor
rEw.d Real floating point descriptor
rGw.d Real floating point descriptor
rlw Integer descriptor
nP Scaling factor
input: internal value =
external value / 10**n
(An exponent in the data will override the scaling factor)
output:
E & D decimal point is moved right n places, exponent is reduced by n
F external value =
internal value *10**n
G no effect unless magnitude of data causes E editing to occur, then same as E

Logical

input leading blanks are ignored characters following T or F are ignored
output data is right justified in field, blank filled on left if needed
rLw Logical descriptor

Special descriptors

/ input: skip remainder of record
output: remainder of record is blank filled and record is written
nX input: next n characters are skipped
output: n blanks are written

CARRIAGE CONTROL CHARACTERS

| CONTROL CHARACTER | ACTION TAKEN BEFORE PRINTING |
|-------------------|-------------------------------|
| 0 | Skip two (2) lines |
| 1 | Insert form feed |
| + | No advance (suppress spacing) |
| other | Skip one (1) line |

OPERATING PROCEDURE FORTRAN-80 COMPILER

To start the compiler, type:
F80 and a carriage return

The compiler will respond with:
*

It is now ready to accept commands.
The general form of the command string is:

dev:object-prgm.ext,list-dev=dev:source-prgm.ext
dev:object-prgm.ext
The name of the compiled program with default extension .REL. This file must be linked via LINK-80 before it can be executed.

list-device
The device on which the program listing is written. Can be a hardcopy device or a disk file. Default extension for the disk file is .LST.

dev:source-prgm.ext
The name of the source program which is to be used as input to the Fortran compiler. Default extension is .FOR.

Fortran-80 Compiler Switches

Each switch must be preceded with a slash

| <u>SWITCH</u> | <u>ACTION</u> |
|---------------|--|
| O | Print all listing addresses in octal |
| H | Print all listing addresses in HEX. (default) |
| N | Do not list generated op-codes. (default) |
| A | List generated op-codes. |
| R | Force generation of an object file. |
| L | Force generation of a listing file. |
| P | Allocate an extra 100 bytes of stack space for use during compilation. |
| M | Generate code suitable for ROM'S. |

LINK-80 LINKING LOADER

To start the Linker, type
L80 and a carriage return

The Linker will respond with:
*

It is now ready to accept commands.

The general form of the command string is:
dev:filename.ext/s,dev:filename.ext/s

The switches affecting the loading process are:

| <u>SWITCH</u> | <u>ACTION</u> |
|---------------|---|
| R | Reset, put loader back in initial state. |
| E | Exit and return to HDOS. |
| G | Start execution of program after linking process. |
| N | Specify filename to save object file. |
| P | Set origin for next program loaded. |
| D | Set origin for data for next program loaded. |
| U | List the origin and end of both program and data areas and all undefined globals. |
| M | List the origin end end of both program and data areas and all globals. |
| S | Search the filename immediately preceding the /S to satisfy any undefined global. |

Cross Reference Utility

To start the cross reference facility, type:
CREF80 and a carriage return

The utility will respond with:

It is now ready to accept commands.
*

The general form of the command string is:
listing-file-source-file

The default extension for the source file is .CRF. The source file must have been created with the /C switch in the MACRO-80 Assembler.

MACRO-80 ASSEMBLER

To start the Assembler, type: M80 and a carriage return

The Assembler will respond with:

*

It is now ready to accept commands.

The general form of the command string is:

dev:object-prgm.ext,list-dev=dev:source-prgm.ext
dev:object-prgm.ext
The name of the assembled program with default extension.REL . This file must be linked via LINK-80 before it can be executed.

list-device

The device on which the program listing is written. Can be a hardcopy device or a disk file. Default extension for the disk file is .LST.

dev:source-prgm.ext

The name of the source program which is to be used as input to the MACRO-80 assembler. Default extension for this file is MAC.

MACRO-80 Switches

Each switch must be preceded with a slash

| (I). <u>SWITCH</u> | <u>ACTION</u> |
|--------------------|---|
| O | Print all listing addresses in octal. |
| H | Print all listing addresses in HEX. (default) |
| R | Force generation of an object file. |
| L | Force generation of a listing file. |
| C | Force generation of a cross reference file. |
| Z | Assemble Z80 (Zilog format) mnemonics. |
| I | Assemble 8080 mnemonics. (default for HDOS) |

Microsoft FORTRAN-80

Model H8-20 REFERENCE GUIDE

for the
Heath 8-bit Digital Computer Systems

FORTRAN LINE FORMAT

1. Statement Label Columns 1-5
2. Continuation field Column 6
3. Statement field Columns 7-72
4. Identification field Columns 73-79

NOTE: Column 80 must not contain any character. This column is reserved for the carriage return character.

FORTRAN VARIABLE DATA TYPES

INTEGER - Whole numbers in the range -32768 to +32767. 2 bytes are required for storage. Negative numbers are the 2's complement of positive representations.

LOGICAL - A non-zero valued byte indicates true. A zero valued byte indicates false. 1 byte is required for storage.

REAL - Approximations of real numbers in the range 10⁻³⁸ and 10⁺³⁸. Real data are precise to 7+ significant digits. 4 bytes are required for storage.

DOUBLE PRECISION - Double Precision data are precise to 16+ significant digits in the same magnitude range as real data. 8 bytes are required for storage.

LOGICAL UNIT NUMBER ASSIGNMENT

| LUN | DEFAULT DEVICE |
|-----|----------------|
| 1 | TT: |
| 2 | LP: |
| 3 | TT: |
| 4 | TT: |
| 5 | TT: |
| 6 | SY0:FORT06.DAT |
| 7 | SY0:FORT07.DAT |
| 6 | SY0:FORT08.DAT |
| 9 | SY0:FORT09.DAT |
| 10 | SY0:FORT10.DAT |

RELATIONAL OPERATORS

- | | |
|------|--------------------------|
| .LT. | Less than |
| .LE. | Less than or equal to |
| .EQ. | Equal to |
| .NE. | Not equal to |
| .GT. | Greater than |
| .GE. | Greater than or equal to |

LOGICAL OPERATORS

- | | |
|-------|----------------------|
| .NOT. | Logical negation |
| .AND. | Logical conjunction |
| .OR. | Logical disjunction |
| .XOR. | Logical exclusive OR |

FORTRAN STATEMENTS SUMMARY

ASSIGN 1 TO b

- 1 is a statement label
- b is integer variable name

BLOCK DATA (sub)

- sub is a subprogram name

CALL s(a1,a2,...,an)

- s is the subroutine name
- ai are the arguments to be used

COMMON /cb1/list1/cb2/list2/.../cbn/listn/

- cb is the common block name
- list is the list of variables

CONTINUE

- has no effect on execution

DATA list/u1,u2,...,un/,list/u1,u2,...,on/

- list is a list of variables separated by commas
- u is the constant values to assign the variables

DECODE (a,f) k

- a is an array name
- f is a FORMAT statement number
- k is an I/O list

DIMENSION s(d),s(d),s(d),...s(d)

- a is the name of the array
- d is the array dimension declarator

DO k i = m1,m2,m3

- k is the statement label of the terminal statement
- i is the index variable
- m1 is the initial value
- m2 is the terminal value
- m3 is the incremental value (if omitted defaults to 1)

ENCODE (a,f) k

- a is an array name
- f is a FORMAT statement number
- k is an I/O list

END

- terminates program unit

ENDFILE u

- u is an Integer variable or constant

EQUIVALENCE (u1),(u2),...,(un)

- ui is a sequence of two or more variables or array elements, separated by commas.

EXTERNAL u1,u2,...,un

- ui is a subprogram name

FORMAT (s1,s2 . . . , sn)

- si is the field descriptor

t FUNCTION f(a1,a2,...,an)

- t is the data type (optional)
- f is the subprogram name
- ai are dummy argument names

GO TO k (Unconditional GO TO)

- k is the label of an executable statement

GO TO (k1,k2,...,kn)j (Computed GO TO)

- ki are labels of executable statements
- j is an integer variable

GO TO j,(k1,k2,...,kn) (Assigned GO TO)

- j is an integer variable
- ki are labels of executable statements (optional)

IF (e) m1,m2,m3 (Arithmetic IF)

- e is an arithmetic expression
- mi are labels of executable statements

IF (u) s (Logical IF)

- u is a Logical expression
- s is any executable statement except a DO statement

PAUSEc

- c is any string up to 6 characters

PROGRAM name

- name specifies the name of the main program.

READ (u,f,ERR=L1,END=L2) k

- (Formatted Sequential Read)

READ (u,ERR-L1,END=L2) k

- (Unformatted Sequential Read)

- u is the logical unit number
- f is the label of a FORMAT statement

- L1 is the label to transfer to if an error is encountered (optional)

- L2 is the label to transfer to if an EOF is reached (optional)

- k is an I/O list

READ (u,f,REC=i,ERR=L1,END=L2) k

- (Formatted Random Read)

READ (u,REC=i, ERR=L1,END=L2) k

- (Unformatted Random Read)

- u is the logical unit number

- f is the label of a FORMAT statement

- i is the record number to read (Random Access)

- L1 is the label to transfer to if an error is encountered (optional)

- L2 is the label to transfer to if an EOF is reached (optional)

- k is an I/O list

READ (u,f,ERR=L1,END=L2] (H type conversion)

- u is the logical unit number

- f is the label of FORMAT statement

- L1 is label to transfer to if an error is encountered

- L2 is the label to transfer to if EOF is reached (optional) (no I/O list is needed)

RETURN

- returns control to calling program

REWIND u

- u is an integer variable or constant

STOP c

- c is any string up to 6 characters

SUBROUTINE s (a 1,a2, . . .,an)

- s is the subroutine name

- ai are the dummy arguments (optional)

type v1,v2,v3, . . .,v4

- type is the data type specifier

- vi are variable, array or function names

WRITE (u,f,ERR=L1,END=L2) k

- (Formatted Sequential Write)

WRITE (u,ERR=L1,END=L2) k

- (Unformatted Sequential Write)

- u is the logical unit number

- f is the label of a FORMAT statement

- L1 is the label to transfer to if an error is encountered (optional)

- L2 is the label to transfer to if EOF is reached (optional)

- k is an I/O list

WRITE (u f,REC=i,ERR=L1,END=L2) k (Formatted Random Write)

WRITE (u,REC=i,ERR= L1,END=L2) k (Unformatted Random Write)

- u is the logical unit number

- f is the label of a FORMAT statement

- i is the record number to read

- L1 is the label to transfer to if an error is encountered (optional)

- L2 is the label to transfer to if an EOF is reached (optional)

- k is an I/O list

WRITE (u,f,ERR=L1,END=L2) (No variable list)

- u is the logical unit number

- f is the label of the FORMAT statement

- L1 is the label to transfer to if an error is encountered (optional)

- L2 is the label to transfer to if EOF is reached (optional)

- (no variable list is needed, the characters to be printed are contained in the FORMAT.)

FORTRAN SYSTEM

RUNTIME ERROR MESSAGE

Fatal errors cause execution to cease. Execution continues after a warning error. However, after 20 warnings, execution ceases.

Runtime errors are surrounded by asterisks:

FW

Warning Errors

| Code | Explanation |
|------|-------------------------------------|
| TL | Too many left parentheses in FORMAT |
| DE | Decimal Exponent Overflow |
| IS | Integer size too large |
| IN | Input record too long |
| OV | Arithmetic Overflow |
| CN | Conversion overflow |
| SN | Argument to SIN too large |
| A2 | Both arguments to ATAN2 are 0 |
| IO | Illegal I/O operation |
| RC | Negative repeat count in FORMAT |

Fatal Errors

| Code | Explanation |
|------|--------------------------------------|
| ID | Illegal FORMAT descriptor |
| F0 | Format field width = 0 |
| MP | Missing Period in FORMAT |
| FW | FORMAT field width too small |
| IT | I/O transmission error |
| ML | Missing left parenthesis |
| DZ | Division by zero |
| LG | Illegal argument to LOG |
| SQ | Illegal argument to SQRT |
| DT | Data type does not agree with FORMAT |
| EF | EOF encountered on READ |