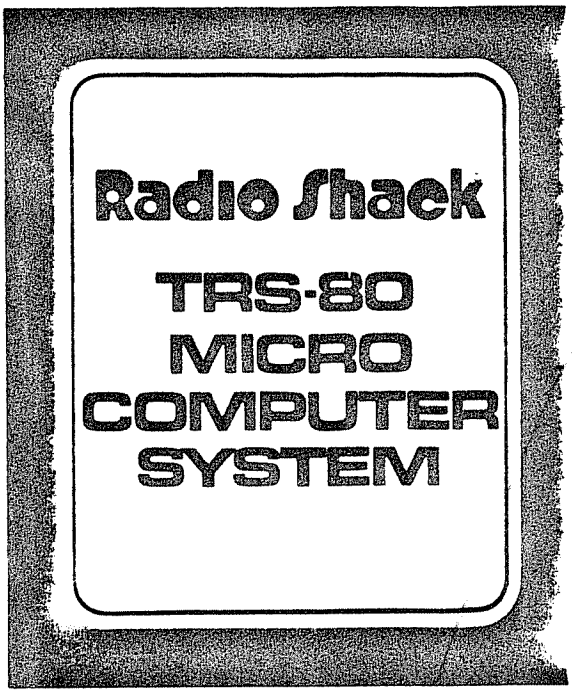
FORTRAN User's Manual



Important Note

Be sure to make BACKUP copies of both FORTRAN diskettes before you begin using the FORTRAN Package.

Contents

1. Introduction 5
   1.1 Sample Session 6
   1.2 Note on TRS-80 FORTRAN Manuals 13
2. TRS-80 FORTRAN Compiler 14
   2.1 Running the Compiler 14
   2.2 Command Format 14
   2.3 Input/Output Devices 14
3. TRS-80 FORTRAN Disk Files 14
   3.1 Default Disk Filenames 19
   3.2 CALL OPEN 19
4. Error Messages 21
   4.1 FORTRAN Compiler Error Messages 21
   4.2 FORTRAN Runtime Error Messages 23

Microsoft
TRS-80 FORTRAN Package
User's Manual

CONTENTS

SECTION 1	Introduction	5
1.1	Sample Session	6
1.2	Note on TRS-80 FORTRAN Manuals	13
SECTION 2	TRS-80 FORTRAN Compiler	14
2.1	Running the Compiler	14
2.2	Command Format	14
2.3	Input/Output Device Names	18
SECTION 3	TRS-80 FORTRAN Disk Files	19
3.1	Default Disk Filenames	19
3.2	CALL OPEN	19
SECTION 4	Error Messages	21
4.1	FORTRAN Compiler Error Messages	21
4.2	FORTRAN Runtime Error Messages	23

SECTION 1

Introduction

The TRS-80 FORTRAN Package contains the following software and documentation.

Disk	Software	Documentation
#1	TRS-80 FORTRAN Compiler	TRS-80 FORTRAN User's Manual
		FORTRAN-80 Reference Manual
#2	LINK-80 Linking Loader	LINK-80 Reference Manual
#2	FORLIB/REL FORTRAN-80 Subroutine Library	FORTRAN-80 Reference Manual Appendix E
#1	EDIT-80 Text Editor	EDIT-80 User's Guide

1.1 Sample Session

This sample session will give you a chance to exercise the FORTRAN package, so you'll see how all the parts fit together. Ideally, you should have both diskettes in the System (Drives 0 and 1) at once, so you won't have to swap diskettes. Single drive users should refer to the F80 Compiler Manual before trying this sample session, since some procedures will need to be changed.

DOS READY should be displayed.

STEP 1: Place the diskette #1 in the drive and enter the command:

```
EDIT
```

This loads the EDIT-80 text editor. EDIT-80 will respond with

```
FILE:
```

If you are using the program in Figure 1, type the filename TEMP/FOR followed by the <break> key. If you are using your own FORTRAN program, type any legal TRSDOS filename. Always follow the filename with <break> when creating a new file and with <enter> when reading in an existing file.

After EDIT-80 prints the message:

```
Creating
Version x.x
Copyright 1977,78 (c) by Microsoft
Created: xxxx
xxxx Bytes free
*
```

enter the command:

```
I
```

EDIT-80 will print 00100, which is the first line number.

STEP 2: Start entering the FORTRAN program as listed in Figure 1 (or enter your own FORTRAN program). EDIT-80 will type the next line number each time you <enter> a line.

While you're typing in your program, all of EDIT-80's editing capabilities are available to you. Read through the EDIT-80 User's Guide. You'll see how easy it is to insert and delete lines, modify

text, and search for text. This is a good chance to experiment with EDIT-80.

When writing any FORTRAN program for your TRS-80, use the Microsoft FORTRAN-80 Reference Manual to determine the correct syntax and usage of all FORTRAN statements.

STEP 3: When you are finished typing in the program, type a <break> after the next available line number to return to EDIT-80 command level. To exit the editor, enter the command:

```
E
```

The program you typed in is now saved with the name TEMP/FOR. (TEMP is the name you specified in Step 1; /FOR is a default extension supplied by the Editor.) TEMP/FOR is called the source file; it is ready to be compiled.

STEP 4: Syntax check.
Before proceeding, it is a good idea to check the program for syntax errors. Removing syntax errors now eliminates a possible recompilation later. To perform the syntax check on the source file called TEMP/FOR, place diskette #1 in the disk drive and type:

```
F80 =TEMP
```

F80 is the filename of the Compiler. =TEMP is a parameter telling the Compiler which file to compile. Since no extension is supplied, F80 uses the default extension /FOR. No object or list file is specified, so the Compiler will not output either. This is just a "dry run" to see if errors are generated.

If there are errors, KILL the file TEMP/FOR and carefully repeat Steps 1 through 4. (For this exercise, we aren't ready to use the Editor's convenient editing commands, described fully in the EDIT-80 User's Guide.)

During processing, $MAIN will be displayed. When the Compiler has finished, DOS READY will be displayed.

STEP 5: Compile the source file.
To compile the source file called TEMP/FOR and produce an object and listing file, type the following:

```
F80 TEMP,TEMP=TEMP
```

This time, in addition to specifying the target file (=TEMP with default extension /FOR), we specify output files for relocatable object code and for a listing file (showing source statements and the associated Compiler actions). The object file TEMP gets the default extension /REL, and the listing file TEMP gets the default extension /LST. For details of syntax, see Section 2 of this manual. See Figure 2 below for a copy of the listing file TEMP/LST generated by TEMP/FOR.

STEP 6: Load and execute the program.
To load the program into memory and execute it, put diskette #2 in the disk drive and type:

```
L80 TEMP-G
```

This command tells TRSDOS to load and run LINK-80, which in turn loads the object file TEMP/REL (LINK-80 provides the default extension /REL) into the correct memory locations; searches the system library to resolved any undefined references; and executes the program. In this case, LINK-80 will not create a command file. Figure 3 shows a sample run.

STEP 7: Save the object code.
The object file, once it has been loaded by LINK-80, is in a form that can be executed by the TRS-80 computer. To save a copy of this file, type:

```
L80 TEMP-N,TEMP-E
```

This command creates a command file which can be run directly under TRSDOS. TEMP-N tells LINK-80 to name the file TEMP/CMD; TEMP-E tells LINK-80 to load the object file TEMP/REL. Both /CMD and /REL are default extensions.

You can now load and run the program as a TRSDOS command file, typing:

```
DOS READY
TEMP     <ENTER>
```

FIGURE 1 FORTRAN SOURCE FILE - TEMP/FOR

```
00100    C        CONVERT FAHRENHEIT TO CENTIGRADE
00200             INTEGER F
00300             WRITE(5,5)
00400     5       FORMAT(33H         FAHRENHEIT      CENTIGRADE)
00500             DO 20 F=20,65,5
00600             C=5./9.*(F-32)
00700             WRITE(5,10)F,C
00800     10      FORMAT(12X,I2,11X,F6.3)
00900     20      CONTINUE
01000             END
01100    $
*
```

(This is the echo from the <break> key.)

FIGURE 2 LISTING FILE TEMP/LST

```
 1.  FORTRAN-80 VER. 3.2 COPYRIGHT 1978 (C) BY MICROSOFT
 2.  BYTES: 3699
 3.  CREATED: 15-FEB-79
 4.  00100     C         CONVERT FAHRENHEIT TO CENTIGRADE
 5.  00200               INTEGER F
 6.  00300               WRITE(5,5)
 7.  ******    0000'     LD        BC,$$L
 8.  ******    0003'     JP        $INIT
 9.  ******    0006'     LD        DE,5L
10.  ******    0009'     LD        HL,[      05        00]
11.  ******    000C'     CALL      $W2
12.  00400       5       FORMAT(33H          FAHRENHEIT      CENTIGRADE)
13.  ******    000F'     CALL      $ND
14.  00500               DO 20 F=20,65,5
15.  00600               C=5./9.*(F-32)
16.  ******    0012'     LD        HL,0014
17.  ******    0015'     LD        (F),HL
18.  00700               WRITE(5,10)F,C
19.  ******    0018'     LD        HL,(F)
20.  ******    001B'     LD        DE,FFE0
21.  ******    001E'     ADD       HL,DE
22.  ******    001F'     LD        (T:000000),HL
23.  ******    0022'     LD        HL,[      00        00        20        83]
24.  ******    0025'     CALL      $L1
25.  ******    0028'     LD        HL,[      00        00        10        84]
26.  ******    002B'     CALL      $DB
27.  ******    002E'     LD        HL,(T:000000)
28.  ******    0031'     CALL      $MA
29.  ******    0034'     LD        HL,C
30.  ******    0037'     CALL      $T1
31.  ******    003A'     LD        DE,10L
32.  ******    003D'     LD        HL,[      05        00]
33.  ******    0040'     CALL      $W2
34.  00800       10      FORMAT(12X,I2,11X,F6.3)
35.  ******    0043'     LD        DE,F
36.  ******    0046'     LD        HL,[      01        00]
37.  ******    0049'     LD        A,02
38.  ******    004B'     CALL      $I0
39.  ******    004E'     LD        DE,C
40.  ******    0051'     LD        HL,[      01        00]
41.  ******    0054'     LD        A,02
42.  ******    0056'     CALL      $I1
43.  ******    0059'     CALL      $ND
44   00900       20      CONTINUE
45.  01000               END
```

-continued-

```
46.  *****    005C'    LD       HL,(F)
47.  *****    005F'    LD       DE,0005
48.  *****    0062'    ADD      HL,DE
49.  *****    0063'    LD       A,41
50.  *****    0065'    SUB      L
51.  *****    0066'    LD       A,00
52.  *****    0068'    SBC      H
53.  *****    0069'    JP       P,0015'
54.  *****    006C'    CALL     $EX
55.  *****    006F'    0100
56.  *****    0071'    0500
57.  *****    0073'    00002083
58.  *****    0077'    00001084
59.
60.  PROGRAM UNIT LENGTH=007B (123) BYTES
61.  DATA AREA LENGTH=0040 (64) BYTES
62.
63.  SUBROUTINES REFERENCED:
64.
65.  $I1                          $I0                          $INIT
66.  $W2                          $ND                          $L1
67.  $DB                          $MA                          $T1
68.  $EX
69.
70.  VARIABLES:
71.
72.  F        0001"               C        0029"               T:000000
73.
74.  LABELS:
75.
76   $$L      0006'               5L       0003"               20L      005C
77   10L      002F"
78.
```

FIGURE 3 TEMP/FOR PROGRAM OUTPUT

FAHRENHEIT	CENTIGRADE
20	-6.667
25	-3.889
30	-1.111
35	1.667
40	4.444
45	7.222
50	10.000
55	12.778
60	15.556
65	18.333

The TRS-80 FORTRAN Package provides a lot more capability than is demonstrated in this short session. Keep experimenting, and you'll be pleasantly surprised at how much computing power has been added to your TRS-80.

1.2 Note on TRS-80 FORTRAN Manuals

The FORTRAN-80 Reference Manual is strictly a reference for the syntax and semantics of the TRS-80 FORTRAN language. It is not intended as a tutorial on FORTRAN programming. If you are new to FORTRAN and need help learning the language, we suggest:

1. "Guide to FORTRAN-IV Programming" by Daniel McCracken (Wiley, 1965)

2. "Ten Statement FORTRAN Plus FORTRAN IV" by Michael Kennedy and Martin B. Solomon (Prentice-Hall, 1975, Second Edition)

3. "FORTRAN" by Kenneth P. Seidel (Goodyear, 1972)

4. "FORTRAN IV, A Self-Teaching Guide" by Jehosua Friedmann, Philip Greenberg, and Alan Hoffbert (John Wiley & Sons, Inc., 1975)

5. "FORTRAN, A Structured, Disciplined Style" by Gordon B. Davis and Thomas R. Hoffman (McGraw-Hill Book Company, 1978)

The LINK-80 Manual is strictly a reference for the commands and switches available.

SECTION 2

TRS-80 FORTRAN Compiler

If you followed the sample session, you are becoming familiar with the software in your TRS-80 FORTRAN Package. Now let's look specifically at the TRS-80 FORTRAN compiler.

2.1 Running the Compiler

When you give TRSDOS the command

F80

(diskette #1 must be in the disk drive), you are running the TRS-80 FORTRAN compiler. The FORTRAN compiler takes a FORTRAN program (source file) and compiles it to generate a relocatable object file, that is, a file that is in machine code. When the compiler is ready to accept commands, it prompts the user with an asterisk. To exit the compiler, use the <break> key.

A command may also be typed on the same line as the invocation. This is called a "command line." We did this in the Sample Session when we typed the command line:

F80 =TEMP

After executing a command line, the compiler automatically exits to the operating system.

2.2 Command Format

A compiler command conveys the name of the source file you want to compile, and what options you want to use. Here is the format for a compiler command (square brackets indicate optional):

[object filename][,listing filename]=source filename[-switch...]

filename[/ext][:drive#]. If you are using the compiler's default extensions, it is not necessary to specify an extension in a compiler command.

Let's look individually at each part of the compiler command:

1. Object filename
   To create a relocatable object file, this part of the command must be included. It is simply the name that you want to call the object file. The default extension for the object filename is /REL.

2. Listing filename
   To create a listing file, this part of the command must be included. It is simply the name that you want to call the listing file. The default extension for the listing file is /LST.

3. Source filename
   A compiler command must always include a source filename -- that is how the compiler "knows" what to compile. It is simply the name of a FORTRAN program you have saved on disk. The default extension for a FORTRAN source filename is /FOR. The source filename is always preceded by an equal sign in a compiler command.

Examples (asterisk is typed by F80):

*=TEST	Compile the program TEST/FOR without creating an object file or listing file.
*TEST,TEST=TEST	Compile the program TEST/FOR. Create a relocatable object file called TEST/REL and a listing file called TEST/LST.
*,TEST.PASS=TEST.PASS	Compile the program TEST /FOR.PASS and create a listing file called TEST/LST.PASS (No object file created.)
*TESTOBJ=TEST	Compile the program TEST/FOR and create an object file called TESTOBJ/REL. (No listing file created.)

4. Switch
   A switch on the end of a command specifies a special parameter to be used during compilation. Switches are always preceded by a dash (-). More than one switch may be used in the same command. The available switches are:

Switch	Action
O	Print all listing addresses in octal.
H	Print all listing addresses in hexadecimal (default condition).
N	Do not list the object code that is generated. List only the FORTRAN source code.
P	Each -P allocates an extra 100 bytes of stack space for use during compilation. Use -P if stack overflow errors occur during compilation. Otherwise not needed.
M	Specifies to the compiler that the generated code should be in a form which can be loaded into ROMs. When a -M is specified, the generated code will differ from normal in the following ways: 1. FORMATs will be placed in the program area, with a "JMP" around them. 2. Parameter blocks (for subprogram calls with more than 3 parameters) will be initialized at runtime, rather than being initialized by the loader.

Examples:

*CT.ME,CT.ME=CT.ME-O	Compile the program CT/FOR.ME. Create a listing file called CT/LST.ME and an object file called CT/REL.ME. The addresses in the listing file will be in octal.
*CT,CT=CT-N	Compile the program CT/FOR. Create an object file called CT/REL and a listing file called CT/LST. The listing file will contain only the FORTRAN source statements, not the generated object code.

*MAX10=MAX10-P-P — Compile the program MAX10/FOR and create an object file called MAX10/REL. The compiler is allocated 200 extra bytes of stack space.

NOTE

If a FORTRAN program is intended for ROM, the programmer should be aware of the following ramifications:

1. DATA statements should not be used to initialize RAM. Such initialization is done by the loader, and will therefore not be present at execution. Variables and arrays may be initialized during execution via assignment statements, or by READing into them.

2. FORMATs should not be read into during execution.

3. If the standard library I/O routines are used, DISK files should not be OPENed on any LUNs other than 6, 7, 8, 9, 10. If other LUNs are needed for Disk I/O, $LUNTB should be recompiled with the appropriate addresses pointing to the Disk driver routine.

A library routine, $INIT, sets the stack pointer at the top of available memory (as indicated by the operating system) before execution begins.

The calling convention is:

```
LXI     B,<return address>
JMP     $INIT
```

If the generated code is intended for some other machine, this routine should probably be rewritten. The source of the standard initialize routine is provided on the disk as "INIT/.MAC". Only the portion of this routine which sets up the stack pointer should ever be modified by the user. The FORTRAN library already contains the standard initialize routine.

2.3 Input/Output Device Names

In FORTRAN I/O statements (READ and WRITE), LUNs 1, 3, 4, and 5 default to the console/keyboard, LUN 2 defaults to the line printer, and LUNs 6-10 default to the disk drives.

SECTION 3

TRS-80 FORTRAN Disk Files

SEE ALSO FORTRAN-80 REFERENCE MANUAL, SECTION 8.3.

3.1 Default Disk Filenames

TRS-80 FORTRAN may access either random or sequential disk files. Any disk file that is OPENed by a READ or WRITE statement is given a default filename that depends on the LUN:

LUN	Default Filename
6	FORT06/DAT
7	FORT07/DAT
8	FORT08/DAT
9	FORT09/DAT
10	FORT10/DAT

3.2 CALL OPEN

Instead of using READ or WRITE, a disk file may be OPENed by calling the OPEN subroutine (see the FORTRAN-80 Reference Manual, Section 8.3.2). The format of an OPEN call is:

```
CALL OPEN (LUN, Filename, Reclen)
```

where:

LUN = a Logical Unit Number to be associated with the file (must be an Integer constant or Integer variable with a value between 1 and 10).

Filename = an ASCII name which TRSDOS 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 in the form normally required by TRSDOS,

```
filename/ext.password:drive#
```

and it should be terminated with a non-alpha character, preferably a blank.

Reclen = The number of bytes you wish to specify (up to 256) as the record length. The default record length is 128 bytes. Reclen must be an Integer constant or Integer variable. If zero is

supplied for Reclen, the record length will be 256 bytes.

The following are examples of valid OPEN calls:

```
CALL OPEN (6,'TIME/DAT.JULY:1 ',256)

CALL OPEN (7,'COUNT/NUM ',200)

CALL OPEN (1,'TESTQ/MIN:2 ',100)
```

SECTION 4

Error Messages

4.1 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 ignores the rest of the logical line, including any continuation lines. Warning messages are preceded by percent signs (%), and Fatal Errors by question marks (?). The editor line number, if any, or the physical line number is printed next. It is followed by the error code or error message.

Example:

?Line 25: Mismatched Parentheses

%Line 16: Missing Integer Variable

When either type of error occurs, the program should be changed so that it compiles without errors. No guarantee is made that a program that compiles with errors will execute sensibly.

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 Too Large
115	Invalid Data List Element in I/O
116	Unbalanced DO Nest
117	Identifier Too Long
118	Illegal Operator
119	Mismatched Parenthesis

120	Consecutive Operators
121	Improper Subscript Syntax
122	Illegal Integer Quantity
123	Illegal Hollerith Construction
124	Backwards DO reference
125	Illegal Statement Function Name
126	Illegal Character for Syntax
127	Statement 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
32	Array Name Expected
33	Illegal Argument to ENCODE/DECODE

4.2 FORTRAN Runtime Error Messages

During execution of a FORTRAN program one or more of the following errors could occur. Fatal errors cause execution to cease. Execution continues after a warning error. However, execution will cease after 20 warnings. Runtime errors are surrounded by asterisks as follows

```
**FW**
```

Warning Errors:

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 0
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
F0	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 Does Not Agree With FORMAT Specification
EF	EOF Encountered on READ