INTRODUCTION
TO
MS-DOS AND WATFOR-77
(Fourth Edition)

Celal N. Kostem
Frederick W. Chapman
Edwin J. Kay
George J. Grevera

Fritz Engineering Laboratory
Department of Civil Engineering
Lehigh University
Bethlehem, Pennsylvania

January, 1991

# Table of Contents

1. INTRODUCTION ................................................................. 1  
   1.1 Supplementary Notes On WATFOR-77 for Professor Kay’s Classes  2  
   1.2 More Supplementary Notes On WATFOR-77 for Professor Kay’s Classes  6  

2. MS-DOS COMMANDS FOR WATFOR-77 USERS .............................. 9  
   2.1 Starting Your MS-DOS Computer ...................................... 9  
   2.2 Preparing Floppy Diskettes for Use ................................. 9  
   2.3 The Default Disk Drive ............................................... 10  
   2.4 Some Keyboard Pointers ............................................. 10  
   2.5 The Directory ................................................................ 11  
   2.6 Filenames .................................................................... 11  
   2.7 Copying Files .................................................................. 12  
   2.8 Wildcard Characters ..................................................... 12  
   2.9 Handling Subdirectories ................................................ 12  
   2.10 Other Important Commands ........................................... 13  
   2.11 Printing Files ............................................................. 14  

3. WATFOR-77 ON THE MICROCOMPUTER ................................... 15  
   3.1 General Considerations .................................................. 15  
   3.2 FORTRAN Programming and WATFOR-77 ......................... 16  
   3.3 Using the WATFOR-77 Text Editor .................................... 17  
   3.4 Using the WATFOR-77 Compiler ........................................ 34  
   3.5 Using the WATFOR-77 Debugger ....................................... 37  
   3.6 Large Programs & Memory Limitations in WATFOR-77 .......... 38  

4. SOME HELPFUL HINTS FOR ENGINEERING-I ............................ 43  
   4.1 MS-DOS Commands ....................................................... 43  
   4.2 Additional Scenarios ..................................................... 44  

5. An Example FORTRAN Program ............................................. 45
1. INTRODUCTION

The material contained in this document was assembled to assist the beginning users of microcomputers equipped with MS-DOS (Microsoft’s Disk Operating System), the WATFOR-77 FORTRAN compiler, and the WEDIT editor by WATCOM. The material presented herein was extracted from voluminous publications by Microsoft and WATCOM products. For advanced applications of MS-DOS and WATFOR-77, it is suggested that the readers refer to these parent publications.

This document is subject to periodic modifications, as the versions of MS-DOS and WATFOR-77 used at Lehigh University are updated. The descriptions contained in the document are based on the diskettes employed by ENGR-1: Engineering Computations course in the College of Engineering and Applied Science of the University. If there are any doubts regarding the compatibility of the document versus the ENGR-1 diskettes, it is strongly recommended that the prospective user check with the office of User Services of the Lehigh University Computing Center (LUCC) or the course leader of ENGR-1.
1.1 Supplementary Notes On WATFOR-77 for Professor Kay's Classes

The introduction of FORTRAN in April of 1957 was one of the most important milestones in computer science. In the intervening 33 years, however, a great deal has been learned about how to construct a computer language, and a great deal has been learned about software design; FORTRAN shows signs of age. In particular, FORTRAN does not promote the use of top-down design, an important principle for constructing good computer software. The designers of WATFOR-77, aware of the FORTRAN's defects, have made a number of extensions to the language, extensions which are not covered in the Engineering 1 text. I have written these notes to fill that gap in the text. Below I explain the syntax and semantics of some extensions, and I give examples the use of each.

Below, all FORTRAN statements will be in upper case. I use << >> to enclose some object to be replaced by some FORTRAN structure, e.g., <<statement>> stands for some FORTRAN statement. I enclose that part of a statement that is optional in brackets: [].

**DO ... END DO Structure**

The DO statement, in conjunction with the END DO statement, is used to repeat the execution of the block of statements appearing between the DO and END DO. The DO ... END DO has one of two (very similar) structures, depending on whether the optional <<step>> appears:

DO <<control-variable>> = <<initial-value>>, <<limit>> [, <<step>> ]

<<statement>>

<<statement>>

<<statement>>

END DO

where <<control-variable>> is a valid FORTRAN variable name, where <<initial-value>>, <<limit>> , and <<step>> are valid FOTRAN constants or valid FORTRAN variable names. When the optional <<step>> is omitted, the step is assumed to be 1.

For the moment, suppose the <<step>> is omitted (and assumed to be 1) or is a positive number. The <<control-variable>> is set equal to the <<initial-value>>. If the value of the <<control-variable>> is less than or equal to the limit, then the subsequent statements up to the END DO are executed, the control-variable is incremented by the value of the <<step>>, and the process is repeated. When the value of the <<control-variable>> is greater than the value of the <<limit>> the statements in the DO ... END DO structure are no longer executed.
In the case that the <<step>> is negative, the <<control-variable>> decreases at each repetition, and repetition continues so long as the <<control-variable>> is greater than or equal to the limit.

Example 1.

Assume that the variable COUNT is of type INTEGER. The following DO ... END DO structure prints out the sequence of numbers 1, 1, 2, 4, 3, 9, 4, 16.

```
DO COUNT = 1, 4
  PRINT *, COUNT
  PRINT *, COUNT**2
END DO
```

Example 2.

Assume that the variables COUNT and REP are of type INTEGER. The following DO ... END DO structure prints out the sequence 1, 1, 2, 1, 2, 3.

```
DO REP = 1, 3
  DO COUNT = 1, REP
    PRINT *, COUNT
  END DO
END DO
```

Example 3.

Assume that the variable COUNT is of type INTEGER. Then the following DO ... END DO structure prints out the sequence of numbers 1, 3, 5, 7

```
DO COUNT = 1, 8, 2
  PRINT *, COUNT
END DO
```

Example 4.

Assume that the variables COUNT and REP are of type INTEGER. Then the following DO ... END DO structure prints out the sequence of numbers, 10, 7, 4, 1, 6, 3, 2

```
DO COUNT = 10, 1, -4
  DO REP = COUNT, 1, -3
    PRINT *, REP
  END DO
END DO
```
Notes:

(1) Formatting (indenting) of DO ... END DO statements is optional but greatly enhances readability.

(2) Although the WATFOR-77 compiler allows the use of REAL control-variables, initial-values, limits, and steps, it is very bad practice to do so. Because of roundoff error in these cases, termination of the loop is unpredictable. You should always use INTEGER valued variables or constants in the DO ... END DO structure.

LOOP ... UNTIL Structure

The LOOP statement, in conjunction with the UNTIL statement, is used to repeat the execution of the block of statements appearing between the LOOP statement and the UNTIL statement. The LOOP ... UNTIL structure has the following syntax:

```
LOOP
  < <statement> >
  < <statement> >
  .
  .
  < <statement> >
UNTIL ( < <logical-expression> > )
```

where < <logical-expression> > is a valid logical FORTRAN expression (which evaluates to either TRUE or FALSE). The statements between the LOOP and the UNTIL statements are executed at least once. Repetition continues if the < <logical-expression> > evaluates to FALSE.

Example 5.

Assume that the variable COUNT is of type INTEGER. Then the following LOOP ... UNTIL structure prints out the numbers 1, 2, 3, 4

```
COUNT = 1
LOOP
  PRINT *, COUNT
  COUNT = COUNT + 1
UNTIL ( COUNT .GT. 4 )
```

Example 6.
Assume that the variable COUNT is of type REAL. Then the following LOOP ... UNTIL structure prints out the numbers 0.0, 0.01, ..., 0.23

\[
\text{COUNT} = 0. \\
\text{LOOP} \\
\text{PRINT *}, \text{COUNT} \\
\text{COUNT} = \text{COUNT} + 0.01 \\
\text{UNTIL} (\text{COUNT} \gt .24)
\]

Notes:

(1) Example 6 is more subtle than it appears. It looks like it is wrong, that the number 0.24 should also be printed when the statements are executed, but at least on my computer on the 24th time through the loop, the value of COUNT is .2400001, and execution of the loop ceases. This occurs because of roundoff error for REALs. You should be very careful about roundoff when using REAL variables. The above loop would have been infinite if I had written (COUNT .EQ. .24) in place of (COUNT .GT. .24).

(2) It is worth emphasizing that the LOOP ... UNTIL structure always executes the statements at least once, whereas the WHILE ... DO .... END WHILE structure (see Nyhoff and Leestma, 131-135) might not execute its statements at all.
1.2 More Supplementary Notes On WATFOR-77 for Professor Kay’s Classes

Below, all FORTRAN statements will be in upper case. I use << >> to enclose some object to be replaced by some FORTRAN structure, e.g., <<statement>> stands for some FORTRAN statement.

IOSTAT = clause for READ statements

The READ statement has the form

READ( <<control-list>> ) <<input-list>>

The <<control-list>> must specify the unit connected to the file. It may optionally have an IOSTAT = clause of the form

IOSTAT = <<variable-name>>

The <<variable-name>> must be of type INTEGER. When the READ statement is executed, the <<variable-name>> is set to a positive number if an input error occurs, is set to a negative number if the end of data is encountered ("end of file"), and is set to 0 if neither an input error occurs nor an end of data is encountered. What happens to the variables in the <<input-list>> when the end of data is encountered? They are left alone. Thus, if the variables are previously undefined and end of data is encountered, then the variables are still undefined. If they had some values from previous program actions, then these values will still be there. See example 1.

For each of the examples below, assume we have an input file named IOSTAT.DAT containing the following two lines of data.

```
2 3
1 2
```

Example 1.

Executing the code

```
INTEGER EOF,X,Y
OPEN(20,FILE='IOSTAT.DAT')
READ(20,*,IOSTAT=EOF)X,Y
PRINT *,X,EOF
READ(20,*,IOSTAT=EOF)X,Y
PRINT *,X,EOF
```

```
produces the following output:

2  0
1  0
1  -1

The output may be a bit confusing, because it appears that values have been read into X three times, but this is not so. After the second line of data is read, end of data has not been encountered yet. When the third READ is executed, the READ fails to read any input, because the end of data is encountered, and the contents of X are left alone (the contents of X remains 1).

Example 2.

Here is a sample program, where I read and print the data in the file IOSTAT.DAT. The program assumes that each line has two integers. It prints X and Y so long as EOF is 0, i.e., so long as the READ succeeds.

```fortran
PROGRAM ECHO
  INTEGER EOF, X, Y
  OPEN(20, FILE = 'IOSTAT.DAT')
  LOOP
    READ(20, *, IOSTAT = EOF) X, Y
    IF (EOF .EQ. 0) PRINT *, X, Y
    UNTIL (EOF .NE. 0)
  STOP
END
```

The output of this program is

2  3
1  2

Example 3.

Here is a sample program, where I read and print the data in the file IOSTAT.DAT. The program assumes that each line has two integers. It prints X and Y so long as EOF is 0. This is very similar to the previous example, except that it is more efficient. It tests EOF once for each time through the loop, whereas the program of example 2 tests EOF twice each time through the loop.

```fortran
PROGRAM BETTERECHO
  INTEGER EOF, X, Y
  OPEN(20, FILE = 'IOSTAT.DAT')
```
READ(20,*),IOSTAT=EOF)X,Y
WHILE (EOF .EQ. 0) DO
   PRINT *,X,Y
   READ(20,*,IOSTAT=EOF)X,Y
END WHILE
STOP
END

Note that at the end of the loop, if the READ fails, the loop is exited, if the READ succeeds, the loop is executed once again.

The output of this program is

2  3
1  2
2. MS-DOS COMMANDS FOR WATFOR-77 USERS

2.1 Starting Your MS-DOS Computer

1. You will need a "boot diskette" to start a floppy disk-based computer (a later step will explain how this diskette is created). For your convenience, the ENGR-1 WATFOR-77 diskette is also a boot diskette. Place the WATFOR-77 diskette in the uppermost drive (if the floppy drives are placed vertically) or in the leftmost drive (if the drives are arranged horizontally). This is the "A" drive, from which the computer is usually started. The other floppy drive is called the "B" drive. If there is a hard disk present, it is designated the "C" drive (whether or not there is a "B" drive present). The floppy diskette is inserted into the drive with the label side uppermost, and with the label on the trailing edge of the diskette (the side away from the computer). Be sure to close the "door" on the floppy disk drive. Turn on the power to the computer and the power to the monitor.

In order for the WATFOR-77 software to work as intended, it is ABSOLUTELY ESSENTIAL that the computer be booted with the WATFOR-77 disk in drive "A."

2. After a few brief messages, prompts for date and time will appear. You can bypass the request for date and time input by hitting the ENTER key at each prompt. However, it is strongly recommended that you enter both date and time, since a file's directory entry includes date and time fields which are useful in determining when a file was last modified or created.

3. After dealing with the date and time input, a message will appear listing various commands contained on the WATFOR-77 diskette. Below this message you will see the MS-DOS prompt "A:\>". This prompt tells you that the "A" drive is the default drive, and the ">" symbol tells you that MS-DOS is loaded and running.

2.2 Preparing Floppy Diskettes for Use

1. The WATFOR-77 diskette contains files needed to run MS-DOS and WATFOR-77 and is almost full. Therefore any FORTRAN programs which you write during the semester will have to be stored on a separate diskette. You may use any blank diskette (preferably a non-boot diskette) to store your programs, provided that the diskette has been formatted (i.e., prepared for use by MS-DOS). Format one or more blank diskettes now, using the following instructions.

2. New floppy diskettes must first be formatted before they can be used. You must decide when formatting a diskette whether you also want to make it a boot diskette. Diskettes used only for data storage should probably be formatted without a boot option, which will save you around 60,000 characters of storage space (this being the space required by the system boot programs). In order to format a diskette, you need the WATFOR-77 Diskette. The latter has, in addition to its boot capability, a utility program which formats diskettes.
3. To format an ordinary non-bootable diskette, place the WATFOR-77 diskette in drive "A" and a blank diskette to be formatted in drive "B". (Make sure that the default drive is "A.")

Type the command "FORMAT B:". You will then be prompted with a few simple instructions. Then you will be asked if you want to give the diskette a volume label. This is optional. The volume label serves as an internal diskette label and appears whenever a directory command is given.

4. To format a boot diskette, follow the instructions immediately above, using the command "FORMAT B: /S".

REMEMBER: Whenever a diskette is formatted, all the contents will be LOST!

2.3 The Default Disk Drive

To change the default drive, type the letter of the desired drive, immediately followed by a colon, e.g. "B:". Hit the "RETURN" key. (All MS-DOS commands are case insensitive so you may use upper or lower case at will. You may even mix the two.) Note the new prompt indicating the new default drive.

When using WATFOR-77, it is very important to make the "B" drive be the default drive.

With one of your newly formatted blank diskettes in drive "B:" , change the default disk drive to "B:" before continuing.

2.4 Some Keyboard Pointers

Three different keyboards are available depending upon the model of the computer. The XT and AT keyboards are basically the same in appearance and functionality and have function keys labelled F1 through F10 in two columns to the left of the main typing keys. The newer, enhanced keyboards have function keys in a single row above the main typing keys. These keys are labelled F1 through F12.

1. Viewing the keyboard from left to right, three sections can be distinguished. The first section consists of function keys (2 columns on XT/AT keyboards or 1 row on enhanced keyboards). The second section consists of all alphabetic and punctuation characters, as well as the RETURN, ESCape, CTRL, SHIFT, and ALT keys. The last section comprises (among others) the numeric keypad, the INSert, DELete, NUMber LOCK, PRT SC (print screen) and SCROLL-LOCK/BREAK keys.

2. The function keys are all programmable and WEDIT, the WATFOR-77 editor, makes use of all of them. When MS-DOS is active, only the first six function keys have assigned functions. Of these six, the F1 and F3 keys are most useful. Both keys allow you to repeat the previous MS-DOS command. Each time you hit the F1 key, a single character from the
previous command is displayed at the current prompt. Hence, you can repeat a command up to a certain point, correct an error or change a parameter, and then finish the command by using either F1 or F3 keys. Pressing the F3 key causes the entire previous command to be re-displayed. Pressing the RETURN key will re-issue the command.

3. The ESCape, CTRL, and ALT keys can take on special functions in a given applications program. In MS-DOS, the key sequence CTRL/ALT/DEL will reboot the computer. And when a WATFOR-77 program is running, the sequence CTRL/BREAK (pressed simultaneously) will terminate that program.

The notation CTRL/ALT/DEL indicates that one should simultaneously hold down the CTRL and ALT keys, and while holding down these keys, then press the DEL key. Similarly, CTRL/BREAK means that one should hold down the CTRL key and then press the BREAK key. In other words, the CTRL and ALT keys are used in the same manner as the SHIFT key.

4. The keys of the numeric keypad also double as function keys. If the SHIFT or NUMber LOCK keys are not active, then the function labeled on the lower part of each key can be executed, e.g. END, HOME, PGUP (page up) and PGDN (page down) have no functions when MS-DOS is running, though they are all employed by the WATFOR-77 editor.

5. The BACKSPACE key will rub out the character to the left of the cursor in MS-DOS and in most applications programs. It should not be confused with the left-arrow key on the numeric keypad which, in most application programs, moves the cursor left without any rub out function.

2.5 The Directory

The basic command to display a listing of files from a diskette is "DIR" (for the default drive) or "DIR d:" for a non-default drive, where "d" is the desired drive. For example, "DIR" gets a directory of the default drive and "DIR A:" will get a directory of drive "A". Note that the latter command will not change the default drive. Often, the file listing will scroll off the screen. You can compensate for this in several ways. If you hold down the "CTRL" key and press the "S" key, the display will stop scrolling. Pressing any key will cause the scrolling to resume. Type "DIR/P" to cause the listing to pause after each screen of new information. Or type "DIR/W" to display the directory in a wide format which leaves off some information so that many more entries can be displayed.

2.6 Filenames

1. Filenames in MS-DOS consist of a primary file name of up to 8 characters followed by an optional extension of up to 3 characters; the primary file name and optional extension are separated by a period (e.g., "PRIMNAME.EXT"). Virtually any character accessible from the keyboard is allowed in file names, except non printing or special function characters (mathematical symbols, the asterisk, slash, backslash, colon, etc.); the particular letters of the
alphabet and the numbers are permitted in file names. Note that embedded blanks are not allowed.

2. In addition to the primary file name and extension, a file may also be designated by the drive and directory in which it resides. This compound file designator is called the file specification. For example, "B:\WATFOR\MATRIX.FOR" designates the file "MATRIX.FOR" on drive "B", in subdirectory "WATFOR".

2.7 Copying Files

The basic format of the copy command is: "COPY d1:filename d2:filename", where "d1" is the source drive and "d2" is the target drive. For example, "COPY A:STUFF.FOR B:STUFF.FOR". This is the long form of the copy command, and it can be shortened in several ways. First, if the target filename is to remain the same as the source filename, it need not be repeated. The above example would then become "COPY A:STUFF.FOR B:". Note where there is a blank between characters and where there is no blank.

If the default drive were drive "A", then the example could be further shortened to "COPY STUFF.FOR B:". It is recommended, however, to fully specify source and target file designators until you become experienced with this command. Otherwise, there is a potential to accidentally reverse the intended direction of the copying, i.e., possibly copying an out-of-date file on top of a new file.

2.8 Wildcard Characters

1. MS-DOS employs two wildcard characters, the "*" and the "?". The latter symbol represents any single character. So, for example, the command "DIR ?BC" will list any files on the default drive which have file names of three characters, the first character being anything and the remaining characters being "BC". "?" actually represents any single character or no character! DIR ?BC would thus list BC if it existed. ???? represents 0-4 characters, for example.

The "*" stands for any number of characters up to eight. For example, "COPY ABC.* C:" will copy all files on the default drive which have a primary file name "ABC" and any characters in their extension. A common use of the "*" wildcard is the sequence "*.*", e.g., "COPY *.FOR A:" which will copy every file in the current directory of the default drive into the current directory of drive "A". "COPY *.FOR A:" IS AN EASY WAY TO BACK UP FORTRAN PROGRAMS. ALWAYS BACK UP YOUR WORK.

2.9 Handling Subdirectories

1. MS-DOS allows the use of subdirectories on both hard disks and floppy diskettes. The root directory (of the current default drive) is symbolized by the backslash "\". This directory is created when a diskette is formatted.
2. If we give the command "MKDIR directoryname", a subdirectory will be created one level below the current directory. The directory name may be up to 8 characters long, and may have an optional extension up to three characters long (the extension is usually omitted). To change the current directory to this new subdirectory, type "CD directoryname". If we give another "MKDIR" command in this directory, we will create a new subdirectory two levels below where we started.

3. The command "CD" by itself will show the current directory. If the current directory is the root directory, then a backslash will be the response to this command. Note that the current directory is automatically displayed as part of the MS-DOS prompt. (Actually this is true whenever the PROMPT $p$g command is executed. This command is usually put into the file AUTOEXEC.BAT which DOS reads when you boot the system.)

4. To move up one level in the subdirectory structure, type "CD ..". The two dots stand for the parent directory of the current subdirectory.

5. To go directly to the root directory, type "CD \" (remember the backslash is the symbol for the root directory).

6. To change to a subdirectory more than 1 level below the current directory, you must indicate each intervening subdirectory, each separated by a backslash, e.g., "CD\subdirectory1\subdirectory2".

7. To remove a subdirectory, all files in that subdirectory must be deleted and any other subdirectories contained in it must be removed. Then, the current directory must be at least one level above the subdirectory in question. Finally, the command "RMDIR subdirectoryname" can be given.

8. When referencing a file not in the current directory, the subdirectory which contains it must be explicitly specified, e.g., "COPY C:\subdirl\subdir2\filename A:". The COPY, DELete, REName, PRINT, and TYPE commands (to be treated in the next section) all come under this requirement.

2.10 Other Important Commands

1. Deleting files: "DEL filename" will remove a file from a diskette, e.g., "DEL C:STUFF.BAS" will remove the file "STUFF.BAS" from the "C" drive. The command "DEL B:*.*" will remove all files from the current directory of drive "B".

To avoid accidentally deleting files, exercise caution when using wildcards with the DEL command!!!

2. Renaming files: "REN oldfilename newfilename" will change the indicated filename. If a file with "newfilename" already exists, the command will be aborted with an error message.
3. Viewing text files: "TYPE filename" will display the contents of a text file on the screen. This is useful for a fast look at a FORTRAN source code file. Wildcard characters ("*" or "?") are not allowed with this command.

2.11 Printing Files

1. The WATFOR-77 disk prepared for Engineering I can be used to print files on either a dot-matrix printer (compatible with the IBM graphics printer) or a Hewlett-Packard LaserJet printer -- as long as the printer is directly connected to the PC. By default, the disk is set up to work with a dot matrix printer (any parallel printer, really). In order to print on a LaserJet printer, you must first issue the command "LASER" at the DOS prompt; the command does not need to be issued again until the next time you boot the computer.

In order to print on a directly connected LaserJet printer, it is essential that the computer be booted with the WATFOR-77 disk!

2. The WATFOR-77 disk contains a command called PRINT. (This command was locally written and differs somewhat from the standard MS-DOS PRINT command.) In order to print one or more files, enter "PRINT filespec filespec filespec..." at the DOS prompt. "Filespec" is short for "file specification," and as indicated earlier, may include disk drive and directory designators and even wildcard characters (e.g., "PRINT A:\STUFF\MYFILE.*"). You may enter more than one file specification at one time when using the PRINT command (e.g., "PRINT PROG1.* B:DATA.INP"). Each file printed will start on the top of a new page. If the PRINT command cannot find one of the files specified, it will say so.
3. WATFOR-77 ON THE MICROCOMPUTER

3.1 General Considerations

1. There are two versions of the WATFOR-77 compiler: WATFOR77.EXE and WATFOR87.EXE. The latter version requires the presence of an Intel 8087/80287/80387 numeric coprocessor, which is used to perform both integer and floating point arithmetic. The WATFOR77.EXE version uses software routines to implement these arithmetic functions. WATFOR77.EXE may be run with or without the numeric coprocessor; however, this chip (if it were present) would not be used by WATFOR77.EXE. WATFOR77.EXE is the version included on the ENGR-1 disk.

2. All of the WATFOR-77 software is under a campus site license. Accordingly, it may be copied by Lehigh University faculty, staff, and students for use on campus only.

3. Note that the WATFOR-77 disk prepared for Engineering I can be supplemented with two additional disks containing a graphics library and various demonstration and utility programs. These two disks are strictly optional and will not be needed in the Engineering I course. Those who would like to make copies of these two optional diskettes can do so at the following public microcomputing sites: the Central Site Users' Area, the Fairchild-Martindale and Linderman Campus Libraries, the Media Center, and the Educational Technology Center (on the Mountaintop Campus, Bldg. A). Ask for disks 2 and 3 from the set of WATFOR 77 diskettes for dual floppy systems.

4. A version of WATFOR-77 for microcomputers with hard disks is also available. This three disk set contains detailed on-line installation instructions and an automatic installation utility to help the user install WATFOR-77 onto a hard disk system. This set of disks is available for copying at the public microcomputing sites listed above. (For assistance with installing WATFOR-77 onto a hard disk system, contact User Services, room 194 of the Fairchild-Martindale Computing Center, building 8B, phone 83990.)

5. Documentation for WATFOR-77 from WATCOM publications includes:

* WATFOR-77 User’s Guide for the IBM PC with DOS (Third Edition), and Addendum. This book covers the basic operation of WATFOR-77, file and device handling, data types supported, compiler options, the WATFOR-77 debugger and special configuration options.

* WATFOR-77 Language Reference Manual (First Edition). This book gives a detailed description of WATFOR-77 implementation of FORTRAN. It is organized by topics. However, one chapter, "FORTRAN Statements," is organized alphabetically.

3.2 FORTRAN Programming and WATFOR-77

A few comments are in order concerning computer programming in general and the operation of WATFOR-77 in particular.

1. The level at which a computer functions internally is quite primitive in comparison with the level at which human beings think about and solve problems. Forcing people to work at the level of the machine would generally interfere with rather than aid the problem-solving process. In order to bridge the gap between the level at which people think about problems and the primitive level at which the machine operates, various software tools have been developed, among them "high level languages" such as FORTRAN. A language such as FORTRAN allows one to express the method to be used to solve a problem (i.e., the "algorithm") in a relatively abstract fashion, using statements which resemble a mixture of natural language (e.g., English) and mathematical notation. Given such a language for describing algorithms, two problems remain: one needs a way to enter the description of the algorithm (e.g., FORTRAN program) into the computer, and one needs to "translate" statements written in the high-level language into the more primitive statements which the machine is able to carry out (i.e., into "machine instructions"). Entering the program is accomplished by a piece of software called an "editor," and the translation of the program into machine instructions is accomplished by a piece of software called a "compiler." WATFOR-77 includes both a general-purpose text editor (WEDIT) and a FORTRAN compiler. (Note that the term "FORTRAN" is an acronym for "FORmula TRANslator" -- how appropriate!) It is noteworthy that FORTRAN is one of the few languages that supports complex numbers as an atomic data type. Common LISP and C++ support them as well.

2. At Lehigh, WATFOR-77 is set up so that the editor and compiler function as a single entity rather than as two separate pieces of software; this makes for a more convenient environment in which to write, test, and debug FORTRAN programs. Entering "WATFOR77" at the DOS prompt will cause both the FORTRAN compiler and text editor to be loaded into the computer's memory and made available for use. FORTRAN programs can then be entered and/or modified using the text editor; once a program has been entered, it can be made to execute using the "RUN" command (to be described later).

3. Getting a FORTRAN program to run actually involves three separate steps. The program must first be compiled into machine instructions. If the program makes use of any functions or subroutines contained in FORTRAN "libraries" (for example, GKS graphics subroutines), these library routines must be "linked" into the set of machine instructions which resulted from compilation. This linking results in a largely self-contained list of instructions which can be carried out by the machine. When the machine carries out these instructions, the FORTRAN program is said to "execute." This compile-link-execute sequence is automatically performed by WATFOR-77 when the "RUN" command is given.
4. The WATCOM text editor, WEDIT, can be used to edit any kind of text file; the file need not be a FORTRAN program. Since WEDIT is a general purpose text editor, it can be used apart from the WATFOR-77 compiler; in order to do so, enter "WEDIT" at the DOS prompt. The WATFOR-77 compiler (which compiles, links, and executes FORTRAN programs) can be used apart from the WATCOM text editor by entering "WATFOR77 /NOEDIT" at the DOS prompt. Consult chapter two of the "WATFOR-'77 User's Guide" for instructions on using the compiler apart from WEDIT.

NOTE THAT IN ENGINEERING-1, THE COMPILER AND EDITOR WILL BE USED TOGETHER; THUS, AS A GENERAL RULE "WATFOR77" RATHER THAN "WEDIT" OR "WATFOR77 /NOEDIT" WILL BE ENTERED AT THE DOS PROMPT.

3.3 Using the WATFOR-77 Text Editor

1. Recall that entering "WATFOR77" at the DOS prompt will load both the WATFOR-77 compiler and WATCOM text editor into the computer's memory. (Entering "WEDIT" at the DOS prompt will load only the editor.) After the editor starts you will see a screen similar to the following:

```
WATFOR-77 Editor 4.0 PC/DOS Copyright by WATCOM Systems Inc. 1981, 1989
(no_name.for) (0,1,58500)

<beginning of file>
<end of file>
```

Note the window that appears at the bottom of the screen that contains a list of function keys and their definitions. To perform the "Edit File" function for example, press ALT/E. This means that while holding down the ALT key, press the E key once. Once you become proficient at these function keys, you can hide this window by pressing ALT/Z. This will allow you to devote more of the screen to your FORTRAN code. These function keys are not a part of WEDIT but were actually programmed by Fred Chapman to help you use the editor more efficiently and quickly and to maintain compatibility with previous versions of WATFOR.
Note the "beginning of file" marker and an "end of file" marker. The cursor is initially located at the bottom of the screen on the editor's command line. To edit an existing file, press ALT/E and enter the file name (or type "Edit filename" at the editor's command line) and press the RETURN key. Entering "WATFOR77 filename" (or "WEDIT filename") at the DOS prompt will start the editor and begin editing the specified file, thereby combining the above two steps into one.

Note that WATFOR-77 compiler, a file extension of "FOR" is assumed if no extension is specified; thus, entering "WATFOR77 TEST" at the DOS prompt or ALT/E and the file name (or entering "Edit TEST" at the editor's command line) will result in the editing of a file named "TEST.FOR". It's a good idea to establish your own personal standard of using the ".FOR" extension for all of your FORTRAN source files.

2. Press ALT/X (or type "EXit") and press RETURN at the command line in order to leave the editor and return to DOS, saving any additions or modifications made to the file being edited. If the file being edited does not already have a name, a name can be specified by pressing ALT/X and then the file name (or type "EXit filename"). To leave the editor and return to DOS without saving the file, press ALT/Q (or use the "QUIT" command). ALT/S (or "Put" or "Put filename") will save the file to disk without leaving the editor. When the editor is used in conjunction with the compiler, a file extension of "FOR" is assumed if none is given when specifying the name under which the file is to be saved.

In order to obtain help on a particular editor command, enter "Help command" at the command line; thus, "Help EXit" will provide information on the "EXit" command. When "<more >" or "<hold >" appears in the lower right corner of the screen, simply press the RETURN key in order to see more information or to continue editing, respectively.

Some of the editor's functions are performed via commands entered at the command line; other functions are performed via the computer's function keys.

When in the WATFOR-77 compiler/editor, each of the ten function keys performs three different functions for a total of thirty functions. This is accomplished by using the function keys F1 through F10 in conjunction with the SHIFT and ALT keys. For example, F10, SHIFT/F10, and ALT/F10 perform three different tasks. Remember, SHIFT/F10 indicates that one should hold down the SHIFT key and then press the function key F10; similarly, ALT/F10 denotes holding down the ALT key and then pressing F10.
Press ALT/K to obtain a list of the functions which the editor assigns to the regular function keys (F1 through F10) and to the SHIFTed function keys (SHIFT/F1 through SHIFT/F10). Press ALT/F10 to obtain a list of the functions assigned to the ALTerminate function keys (ALT/F1 through ALT/F10).

4. Pressing the function key F9 will move the cursor from the editor’s command line to the editor’s full screen area, where the contents of the file being edited are displayed; pressing F9 again will move the cursor back to the command line. Every time F9 is pressed, the cursor moves alternately between these two areas of the screen.

If you press F5 (or issues the "Input" command at the command line), the cursor will move into the full screen area and a new, blank line will appear in the file being edited. The user can then enter text on this new line. Pressing RETURN will cause another blank line to appear below the line just entered. This mode of entering new lines into the file is referred to as "Input" mode. Pressing F9 will return the cursor to the command line. Should input mode be terminated accidentally, simply press F5 to once again enter input mode.

When the cursor is in the full screen area, the line on which the cursor is positioned is referred to as the "current line." When the cursor on the command line, the current line is highlighted. Pressing F5 inserts one new, blank line after the current line and positions the cursor on this new line, regardless of the cursor’s original position.

5. Note that when editing an existing file, the editor works with a copy of the file rather than the original. This copy is stored in an area of the computer’s memory which is referred to as an "editor workspace." When the file is saved to disk via ALT/X or ALT/S (the "EXit" or "Put" commands respectively), the contents of this workspace replace the version of the file stored on disk; thus, the editor does not alter the disk version of the file being edited until the file is actually saved.

It is sometimes useful to edit several different files simultaneously; the WATCOM editor has this capability. Simultaneous editing is accomplished by storing a copy of each file in a different workspace; however, only one workspace is displayed at one time. The workspace whose contents are currently displayed in the editor’s full screen area is called the "current workspace"; if only one file is being edited, the phrase "current workspace" necessarily refers to this one file.

In order to distinguish between different workspaces, each workspace is given a name, which is usually the same as the name of the file that has been loaded into the workspace. An arbitrary name can be assigned to the current workspace with the command "Name workspacename". A list of the names of all the editor’s workspaces can be obtained with ALT/W (the "SHow" command); the current workspace is indicated by the " > " character. The function key SHIFT/F10 (which is equivalent to entering "Edit" at the command line and pressing RETURN) will cause the workspace following the current workspace on the above list to become the current workspace; repeatedly pressing SHIFT/F10 will display each workspace in turn in the editor’s full screen area.
6. Pressing ALT/I and entering a file name (or entering "Get filename" at the command line) will cause the file specified to be inserted into current workspace immediately after the current line.

Note that there are fundamental differences between ALT/E (the "Edit" command) and ALT/I (the "Get" command). Suppose that during a single editing session the command ALT/E (or "Edit filename") is given several times, each time with a different filename. Each time this command is given, a new editor workspace will be created; the workspace will be given the name of the specified file and will contain a copy of that file. Suppose, on the other hand, that during a single editing session the command ALT/I (or "Get filename") is given several times, each time with a different filename. Each time this command is given, no new workspace will be created; rather, the specified file will be inserted into the current workspace, and the name of the current workspace will remain unchanged.

ALT/E (or "Edit filename") thus creates and names a new workspace, whereas ALT/I (or "Get filename") simply inserts a file into the current workspace.

7. Recall that the WATFOR-77 compiler/editor uses each function key for three distinct purposes. By and large, the regular and SHIFTed function keys pertain to editing and the ALTernate function keys pertain to compiling and editing. Some of the more useful function keys are described at the end of this section. Discussion of the ALTernate function keys that deal with compiling, executing, and debugging will be deferred until a later section concerned with the use of the WATFOR-77 compiler.

8. WEDIT automatically maintains a line numbering system. The line numbers are not displayed during editing. However, they may be easily referenced from the command line. For example, to locate line 48, and make it the current line, type "48" on the command line, and press RETURN.

9. WEDIT uses special symbols for certain lines. The current line is referenced by a "."; the last line is referenced by a "$". A "+" indicates the line after the current one and a "-" indicates a line before the current one. Numbers may be appended to these last two symbols; for example, "+10" refers to the tenth line after the current line.

10. Line number ranges are indicated by "i,j", where "i" and "j" may be numbers or special symbols. ".,$" indicates the range from the current line to the end of the current workspace. WEDIT also employs the symbol "*" to refer to all lines in the current workspace.

11. "i,j Delete" will delete all the lines in the range specified by "i,j". Omission of the range will cause the current line to be deleted. The command "*Delete" will delete all lines in the current workspace.

Be very careful when using the edit command "*DELETE" !!!
12. The command "/string/" will search the current workspace for the first line containing the specified string of characters; for example, entering "/FORTRAN Compiler/" at the command line will cause the editor to search for the first line which contains "FORTRAN Compiler". Note that the editor's string-searching is "case sensitive," meaning that whether the search string contains upper or lower case characters will effect the results of the search; thus, the command given above would not find the strings "fortran compiler" or "Fortran Compiler". The command "+/string/" causes the editor to search for the first line after the current line which contains the specified string; the command "-/string/" causes the editor to search backwards through the current workspace starting with the line immediately before the current line.

When used as a search string, the symbol "//" refers to the last search string specified; thus, one can give the command "/string/" to find the first occurrence of the specified string and then repeatedly give the command "+//" to find all subsequent occurrences. In this context, it is useful to know that the commands "/?" and "/=" will recall and re-execute, respectively, the last command given at the editor's command line.

Note that if the "/" character is itself part of the search string, it must be preceded by a "%". For example, the command "/1%/2/" will search for the first occurrence of the string "1/2". The construction "%/" is one of several "meta-characters" employed by the WATCOM editor.

13. "i,j Change/string1/string2/" will, in the designated line range, change the first occurrence of "string1" on each line to "string2". If no range is indicated, only the current line will be affected. "ChangeN/string1/string2/" will change the "Nth" occurrence of "string1" on the current line. The use of "*" in stead of a whole number will result in all occurrences of "string1" on each line in the designated range being changed. Lastly, "*Change*/string1/string2/" will change every occurrence of "string1" throughout the workspace. For example, ",,$ Change/sat/isat/" will change the first occurrence of "sat" on each line to "isat" in the range from the current line to the end of the workspace. The command ",,$ Change*/sat/isat/" will change every occurrence of "sat" to "isat" in the designated line range.

14. ALT/L (or "Directory") will list the files on the current drive. This command takes all the usual MS-DOS modifiers: wildcards, drive designations, and path specifications.

15. "ERase filename" deletes the selected file(s) from the disk. Wildcards, drive designations, and path specifications are allowed.

16. ALT/D (or the "SYStem" command) will temporarily leave the editor and return to MS-DOS; however, the WATFOR-77 compiler/editor (where applicable), and all editor workspaces are retained in the computer's memory. Any number of DOS commands may then be entered at the DOS prompt. The DOS command "EXIT" will return the user to the compiler/editor with all of the editor's workspaces intact.
Note that if only one DOS command is to be executed, it is more convenient to enter "SYStem doscommand" at the editor's command line. After the specified DOS command is executed, press the RETURN key to continue editing.

17. The WATFOR-77 compiler/editor (as well as the stand-alone editor WEDIT) is set up so that upon startup, tabs are set to columns 7, 12, 17, 22, ..., 72. Also, the editor is set up so that whenever the cursor moves from column 72 to column 73, the computer will beep. (Tabs columns and the column for the warning beep can be set or changed with the TABset and BELL commands, respectively.) These initial values were chosen to assist with the entry of FORTRAN programs and reflect that FORTRAN statements must be contained between columns 7 and 72 inclusive.

A quick reference to the function keys follows. These function keys are grouped into the following four major categories: standard function keys, alt function keys, options menu, and help menu.

**WATFOR-77/WEDIT Main Menu**

<table>
<thead>
<tr>
<th>ALT/E: Edit File</th>
<th>ALT/N: Next Wrkspc</th>
<th>ALT/X: Exit, w/Save</th>
<th>ALT/K: Std Fn Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT/S: Save File</td>
<td>ALT/W: List Wrkspc</td>
<td>ALT/Q: Quit, No Save</td>
<td>ALT/F10: Alt Fn Keys</td>
</tr>
<tr>
<td>ALT/L: List Files</td>
<td>ALT/C: Cut Text</td>
<td>ALT/D: Push to DOS</td>
<td>ALT/O: Options Menu</td>
</tr>
<tr>
<td>ALT/I: Insert File</td>
<td>ALT/P: Paste Text</td>
<td>ALT/F: Fix Screen</td>
<td>ALT/H: Help Screen</td>
</tr>
</tbody>
</table>

**Info for New Users -- ALT/1: Editing Files -- ALT/2: Running Programs**

**Standard Function Keys Menu**

<table>
<thead>
<tr>
<th>F1: Page Up</th>
<th>F6: Delete Line</th>
<th>SHIFT/F1: 1/2 PgUp</th>
<th>SHIFT/F6: Undelete Line</th>
<th>ALT/M:</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2: Page Dn</td>
<td>F7: Select Text</td>
<td>SHIFT/F2: 1/2 PgDn</td>
<td>SHIFT/F7: Command Menu</td>
<td></td>
</tr>
<tr>
<td>F3: Line Up</td>
<td>F8: DO NOT USE!</td>
<td>SHIFT/F3: Fg Color</td>
<td>SHIFT/F8: DO NOT USE!</td>
<td>Menu</td>
</tr>
<tr>
<td>F4: Line Dn</td>
<td>F9: Scrn/Command</td>
<td>SHIFT/F4: Bg Color</td>
<td>SHIFT/F9: Fill Para</td>
<td>ALT/H:</td>
</tr>
<tr>
<td>F5: Ins Line</td>
<td>F10: WEDIT Help</td>
<td>SHIFT/F5: Bd Color</td>
<td>SHIFT/F10: Edit Wrkspc</td>
<td>Help</td>
</tr>
</tbody>
</table>

**Alternate Function Keys Menu**

| ALT/F1: RUN, No Compiler Listing | ALT/F6: Locate Next Syntax Error | ALT/M: |
| ALT/F2: RUN, Listing & Output to Screen | ALT/F7: RUN, Invoke DEBUGger | Main |
| ALT/F3: RUN, Listing & Output to Disk | ALT/F8: GKS Debugging Status | Menu |
| ALT/F4: RUN, Listing & Output to Printer | ALT/F9: Print Current Workspace | ALT/H: |
| ALT/F5: RUN, Locate First Syntax Error | ALT/F10: Alt Function Keys Menu | Help |

**Options Menu**

<table>
<thead>
<tr>
<th>ALT/A: Autowrap On/Off</th>
<th>ALT/C: Prev Paragraph</th>
<th>ALT/H:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT/B: B &amp; W On/Off</td>
<td>ALT/I: First Line</td>
<td>Main</td>
</tr>
<tr>
<td>ALT/R: Mouse On/Off</td>
<td>ALT/L: Last Line</td>
<td>Menu</td>
</tr>
<tr>
<td>ALT/Z: Menu Hide/Show</td>
<td>ALT/C: Prev Paragraph</td>
<td>ALT/H:</td>
</tr>
<tr>
<td></td>
<td>ALT/A: Format Paragraph</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALT/:: Top of Paragraph</td>
<td>Help</td>
</tr>
</tbody>
</table>

**WATFOR-77/WEDIT Main Menu**

**ALT/1: Editing Files**

This function displays information for new users concerning editing files.

**ALT/2: Running Programs**
This function displays information for new users concerning running programs.

**ALT/C:** Cut Text

Remove the selected region of text from the current workspace, and place the text into the "c&p" workspace. See below for instructions on selecting text; if no region of the text has been selected, then the current line is removed from the current workspace and placed into the "c&p" workspace.

Text can be selected in either of two ways, depending on whether or not a mouse has been installed on the computer running WATFOR-77/WEDIT. To select a region of text without using a mouse, position the cursor on the first line to be selected, press the function key F7, and then use the down-arrow key to move the cursor to the last line to be selected; the selected region appears in reverse video. The selection can be cancelled by pressing F7 a second time.

Before the mouse can be used to select text, the mouse must be turned on. This is accomplished with the ALT/R (Mouse On/Off) command. (For additional information concerning mouse support, refer to the online help for ALT/R, which is a command on the Options Menu; to access this online help while in the WATFOR-77/WEDIT edit screen, first press ALT/O, and then press ALT/H.) When the mouse is first turned on with ALT/R, a second, non-flashing cursor appears in roughly the center of the screen. This is the mouse cursor, which is positioned simply by moving the mouse.

To select a region of text by using the mouse, position the mouse cursor on the first line to be selected, and "double click"; i.e., press the left mouse button two times in rapid succession. Move the mouse cursor to the last line to be selected, and "single click"; i.e., press the left mouse button only once. The selected region appears in reverse video. The selection can be cancelled by double clicking a second time.

**ALT/D:** Push to DOS

Temporarily escape from WATFOR-77/WEDIT and return to DOS. This permits DOS commands to be executed without leaving WATFOR-77/WEDIT altogether. Entering EXIT at the DOS prompt will return immediately to WATFOR-77/WEDIT, thereby resuming the edit session.

**ALT/E:** Edit File

Load the specified file into a new editor workspace; used to modify an existing file or to create a new file.
ALT/F: Fix Screen

Restore the menu to its proper size, position, and content. This command should be used when, for whatever reason (e.g., the accidental use of function key F8 or other special key combinations), the menu is no longer being displayed properly. The command fixes a variety of problems relating to the display. If the command should fail to properly restore the screen colors, press ALT/B (B & W On/Off) *twice* to restore the colors.

As a safeguard against the accidental loss of text, ALT/F saves whatever it finds in the "scrap." workspace in a workspace called "trashbin". If the former contents of the "scrap." workspace are of importance, this material can be recovered from the "trashbin" workspace; however, the material must be recovered *before* the next issuing of the ALT/F command.

ALT/F10: Alt Fn Keys

Display a brief description of what the alternate function keys do. Pressing ALT/M will leave this menu and return to the Main Menu.

ALT/H: Help Menu

Display context-sensitive help; i.e., help pertaining to the currently displayed menu. The online help provided by ALT/H was written by LUCC.

For WATCOM-supplied online help, press the function key F10. Note that the WATCOM help topic "keys" does *not* correctly describe the standard and shifted function keys. LUCC has defined these keys so as to maintain as high a degree of compatibility as possible with WEDIT Version 3.5; the WATCOM help for WEDIT Version 4.0 describes the WEDIT V4.0 key assignments, which are different from the WEDIT V3.5 key assignments. To obtain the correct descriptions of the standard and shifted function keys, switch to the Standard Function Keys Menu with ALT/K, and then invoke the LUCC online help with ALT/H.

ALT/I: Insert File

Insert a copy of the specified file into the current workspace, and place the inserted text immediately after the current line. Can be used to insert the source code for a subroutine or function into the file containing the main program, for example.

ALT/K: Std Fn Keys
Display a brief description of what the standard and shifted function keys do. Pressing ALT/M will leave this menu and return to the Main Menu.

**ALT/L:** List Files

List the names of all the files in the current directory of the default disk drive. (<F9> DIR filespec <CR> will list the names of specific files.)

**ALT/N:** Next Wrkspc

Display the next (non-scrap) workspace in the list of workspaces currently being edited; repeating this command will eventually cycle back to the currently displayed workspace.

**ALT/O:** Options Menu

Display a list of commands which can be used to customize the WATFOR-77/WEDIT environment (the menu settings) as well as a list of special cursor movement and text processing commands (available on enhanced 101 keyboards only). Pressing ALT/M will leave this menu and return to the Main Menu.

**ALT/P:** Paste Text

Copy the text contained in the "c&p" workspace into the current workspace, and insert the text immediately after the current line. (The text contained in the "c&p" workspace was presumably placed there by cutting the text from a workspace via the ALT/C command.)

**ALT/Q:** Quit, No Save

Exit from the current workspace, but *without* first saving the workspace to disk. Since the workspace is not saved to disk before exiting, the user must press the RETURN key in order to confirm the command. If other workspaces are being edited, then the next (non-scrap) workspace is displayed; if no other workspaces are being edited, then WATFOR-77/WEDIT quits altogether and returns to DOS.

**ALT/S:** Save File

Save the file in the currently displayed workspace to disk; the name of the disk file is the same as the name of the workspace. (<F9> PUT filename <CR> will save current workspace under a *different* filename.)
ALT/W: List Wrkspc

List the names of all the workspaces currently being edited; the ">" symbol designates the currently displayed workspace.

The "scrap." workspace is a temporary workspace which is used to hold the various menu screens.

The "c&p" workspace (if present) is used to hold text which has been cut from another workspace with ALT/C (Cut Text) and is to be pasted into a workspace with ALT/P (Paste Text); i.e., the "c&p" workspace serves as the cut-and-paste buffer for the ALT/C and ALT/P commands.

The "trashbin" workspace (if present) is created by the ALT/F (Fix Screen) command, and is used to hold anything that was contained in the "scrap." workspace at the time the ALT/F command was issued (such as anything that might have been accidently cut to "scrap." with the function key F8); "trashbin" helps to protect against the accidental loss of material.

ALT/X: Exit, w/Save

Save the current workspace to disk, and then exit from the current workspace. If other workspaces are being edited, then the next (non-scrap) workspace is displayed; if no other workspaces are being edited, then WATFOR-77/WEDIT quits altogether and returns to DOS.

Standard Function Keys Menu

ALT/H: Help

(Described above.)

ALT/M: Main Menu

Pressing ALT/M will leave this menu and return to the Main Menu.

F1: Page Up

Move backward one screen.

F2: Page Dn

Move forward one screen.
F3:  Line Up
Move backward one line.

F4:  Line Dn
Move forward one line.

F5:  Insert Line
The function key F5, which was mentioned earlier, can be used to create a new line immediately after the current line. The F5 key also puts the editor into insert mode and moves the cursor into the full screen area. Note that when the editor is already in insert mode and the cursor is already in the full screen area, a new line can be started simply by pressing the RETURN key; however, unlike pressing RETURN, the F5 key will not break the current line.

F6:  Delete Line
The function key F6 deletes the current line.

F7:  Select Text/Deselect Text
The function key F7 is used to select a region of text in the current workspace; this key is most often used with the ALT/C (Cut Text) and ALT/P (Paste Text) commands from the Main Menu. Press F7 to select the current line as the first line in the region; then use the down-arrow key to move to the last line in the region. The text that has been selected appears in reverse video. This text can now be removed (cut) from the workspace with ALT/C, and reinserted (pasted) into any workspace -- at any location -- with ALT/P.

F8:  DO NOT USE!
Using this function key will interfere with the menu. Use ALT/C instead.

F9:  Screen/Command
As explained earlier, the function key F9 toggles the cursor between the full screen area and the command line. You will want to remember this function key, since it is used so frequently!

F10:  WEDIT Help
Display WATCOM-supplied help topics.
SHIFT/F1: 1/2 PgUp

Move backward half a screen.

SHIFT/F2: 1/2 PgDn

Move forward half a screen.

SHIFT/F3: Fg Color

This option allows you to change the color of the foreground.

SHIFT/F4: Bg Color

This option allows you to change the color of the background.

SHIFT/F5: Bd Color

This option allows you to change the color of the border.

SHIFT/F6: Undelete Line

The *last* line deleted with F6 can be *undeleted* by using the key SHIFT/F6; the line to be restored is inserted immediately before the current line.

SHIFT/F7: Command

Position the cursor on the command line.

SHIFT/F8: DO NOT USE!

Using this function key will interfere with the menu. Use ALT/P instead.

SHIFT/F9: Fill Para

Fill Paragraph is used to perform formatting of paragraphs of text. It performs such operations as combining text to fill entire lines and inserts 2 spaces after every period. To use this function you must first use F7 (Select) or use the mouse to indicate the text that makes up the paragraph.

SHIFT/F10: Edit Wrkspc

Edit the next workspace (similar to ALT/N).
Alternate Function Keys Menu

ALT/F1: RUN, No Compiler Listing

Compile, link, and execute the program in the current workspace. Do not produce a compiler listing. Send all error messages and unit 6 output to the screen. (Note: Carriage control is *not* in effect.)

ALT/F2: RUN, Listing & Output to Screen

Compile, link, and execute the program in the current workspace. Send the compiler listing, all error messages, and unit 6 output to the screen. (Note: Carriage control is in effect.)

ALT/F3: RUN, Listing & Output to Disk

Compile, link, and execute the program in the current workspace. Send the compiler listing, all error messages, and unit 6 output to disk; the file created on disk will have the same primary filename as the workspace, but an .LST file extension. (Note: Carriage control is in effect.)

ALT/F4: RUN, Listing & Output to Printer

Compile, link, and execute the program in the current workspace. Send the compiler listing, all error messages, and unit 6 output to the printer. (Note: Carriage control is in effect.)

ALT/F5: RUN, Locate First Syntax Error

Compile the program in the current workspace. Place the cursor on the line in the program source code which contains the first syntax error, and display the corresponding error message at the bottom of the screen. This command is generally used to compile programs which are known or suspected to contain syntax errors. If the program contains no syntax errors, a message to that effect will appear at the bottom of the screen. (See ALT/F6 below for additional comments and instructions.)

ALT/F6: Locate Next Syntax Error

Locate the next syntax error in the program contained in the current workspace; place the cursor on the line in the program source code which contains the next syntax error, and display the corresponding error message at the bottom of the screen. Once the last known syntax error in the program has been located, a message to that effect will appear at the bottom of the screen the *next* time the
ALT/F6 command is issued; thereafter, ALT/F6 will have no net effect. Note that the ALT/F6 command presumes that the program has already been compiled using the ALT/F5 command.

The ALT/F5 and ALT/F6 commands are intended to be used together. The ALT/F5 command is used once to *compile* the program and locate the *first* syntax error; the ALT/F6 command is then used one or more times, generally until all *remaining* syntax errors have been located.

In the event that a small number of syntax errors in the program source code have generated a large number of error messages, it may not be desirable to review *all* the remaining error messages. After correcting the mistakes responsible for generating the bulk of the error messages, simply recompile the program by using the ALT/F5 command again. Any error messages remaining from the previous compilation will be cleared; ALT/F6 can then be used one or more times to review the new (and hopefully smaller) list of error messages. This "single ALT/F5, multiple ALT/F6" cycle can be repeated as many times as needed to rid the program of syntax errors.

**ALT/F7:**  RUN, Invoke DEBUGger

Compile and link the program in the current workspace. Do not produce a compiler listing. Do not place the program into immediate execution; instead, invoke the interactive debugger so that execution can proceed in a controlled fashion. Send all error messages and unit 6 output to the screen. (Note: Carriage control is *not* in effect.) This command is generally used with programs that are free of syntax errors, but which produce errors during the course of execution.

**ALT/F8:**  GKS Debugging Status

Display the status of GKS argument type checking. This command is equivalent to entering "GKSDEBUG" at the DOS prompt. In order to *change* the status of GKS argument type checking, you *must* first leave WATFOR-77 via ALT/X (Exit, w/Save) or ALT/Q (Quit, No Save); do *not* merely Push to DOS via ALT/D. After leaving WATFOR-77, GKS argument type checking can be enabled by entering "GKSDEBUG ON" at the DOS prompt, and can be disabled by entering "GKSDEBUG OFF" at the DOS prompt. Enter "GKSDEBUG ?" at the DOS prompt to obtain additional instructions concerning GKS argument type checking.

**ALT/F9:**  Print Current Workspace
Send the entire contents of the current workspace to the printer. A form feed will be automatically sent to the printer to cause a page eject if the auto-form-feed option was selected during the installation of WATFOR-77. (The auto-form-feed option is enabled if the WF$FF environment variable has the value 1.)

ALT/F10: Alt Function Keys Menu

Display a brief description of what the alternate function keys do. Pressing ALT/M will leave this menu and return to the Main Menu.

ALT/H: Help

(Described above.)

ALT/M: Main Menu

Pressing ALT/M will leave this menu and return to the Main Menu.

Options Menu

Menu Settings

ALT/A: Autowrap On/Off

Enable or disable automatic word wrap. By default, automatic word wrap is disabled. The first use of ALT/A enables automatic word wrap, and the second use of ALT/A disables automatic word wrap once again; subsequent uses of ALT/A alternate between these two states.

When the user is entering new text into a file, and automatic word wrap is *enabled*, the word being entered is automatically moved to the beginning of the next line as soon as that word extends beyond column 77. When the user is entering new text into a file, and automatic word wrap is *disabled*, the computer will beep when the cursor is moved from column 72 to column 73, but will not wrap text to the next line.

ALT/B: B & W On/Off

Switch the display's color scheme from color to black-and-white, or vice versa. By default, the WATFOR-77/WEDIT screen is displayed in color. The first use of ALT/B causes the screen to be displayed in black-and-white, and the second use of ALT/B causes the screen to be displayed in color once again; subsequent uses of ALT/B alternate between these two states. This command enables the
user to obtain a readable display whether the computer is equipped with a color monitor or a monochrome monitor.

When the screen is displayed in color, the current workspace appears in white letters on a blue background, and the menu is displayed in bright white letters on a red background. When the screen is displayed in black-and-white, the current workspace appears in white letters on a black background, and the menu appears in reverse video (black letters on a white background). On a monochrome monitor, text is generally easier to read when displayed in black-and-white rather than in color.

**ALT/R: Mouse On/Off**

Enable or disable mouse support. By default, mouse support in WATFOR-77/WEDIT is disabled. The first use of ALT/R enables mouse support; i.e., turns the mouse on. The second use of ALT/R disables mouse support once again; i.e., turns the mouse off. Subsequent uses of ALT/R alternate between these two states. (What does the "R" in ALT/R stand for? "Rodent", of course!)

Immediately after the mouse is turned on via ALT/R, a second, non-flashing cursor appears in roughly the center of the screen. This is the mouse cursor, which is positioned simply by moving the mouse. The mouse cursor disappears as soon as a key is pressed. To make the mouse cursor reappear, just press one of the mouse buttons, or move the mouse.

**IMPORTANT NOTE:** Before the mouse can be successfully turned on via the ALT/R command in WATFOR-77/WEDIT, the mouse must be properly installed. In particular, an appropriate mouse driver must be loaded *before* starting WATFOR-77/WEDIT. Loading a mouse driver usually involves entering the command "MOUSE" (or something similar) at the DOS prompt.

**ALT/Z: Menu Hide/Show**

Switch between a visible menu and a hidden menu. The WATFOR-77/WEDIT menu is visible by default. The first use of ALT/Z hides the menu, thereby enabling a larger portion of the current workspace to be displayed on the screen. The second use of ALT/Z makes the menu visible again. Subsequent uses of ALT/Z alternate between these two states.

For Enhanced 101 Keyboards Only

**ALT/<:** First Line
Position the cursor on the first line of the current workspace.

ALT/>: Last Line
Position the cursor on the last line of the current workspace.

ALT/{: Prev Paragraph
Position the cursor on the first line of the previous paragraph.

ALT}/{: Next Paragraph
Position the cursor on the first line of the next paragraph.

ALT/": Format Paragraph
Format the current paragraph. To select the paragraph to be formatted, simply position the cursor anywhere within the desired paragraph before issuing the ALT/" command.

Paragraph formatting includes fitting as many complete words on one line as the margins will permit. The default margins are 1 and 77, and by default, paragraphs are not indented. ( <F9> FILL leftmargin rightmargin indentation <CR> will change the margin and paragraph indentation settings.) The ALT/" command also adjusts the spacing after punctuation (so that a period at the end of a sentence is followed by two spaces, not just one, for example).

ALT/" positions the cursor immediately after the last line of the paragraph just formatted, so that the next paragraph can be formatted simply by using the ALT/" command again. This makes it easy to format several consecutive paragraphs.

NOTE: When formatting the first paragraph in a file, make sure that the paragraph is preceded by a blank line; likewise, when formatting the last paragraph in a file, make sure that the paragraph is followed by a blank line. Failing to follow these guidelines will prevent the ALT/" command from successfully formatting the first or last paragraph in the file.

ALT/:: Top of Paragraph
Position the cursor on the first line of the current paragraph. This command is intended to be used primarily after formatting a paragraph with the ALT/" command. The beginning of a paragraph just formatted
by ALT/" sometimes extends beyond the top of the screen; the ALT/: command allows the user to easily see the results of the paragraph formatting.

ALT/H: Help
(Described above.)

ALT/M: Main Menu
Pressing ALT/M will leave this menu and return to the Main Menu.

3.4 Using the WATFOR-77 Compiler

1. Suppose that 'WATFOR77' has been entered at the DOS prompt, and the WATCOM editor and WATFOR-77 compiler have been loaded into the computer's memory. Suppose also that a FORTRAN program has been newly entered during the current editing session. (Perhaps with F5 or the "input" command.) At this point it is wise to save the workspace to disk with ALT/S (or the "Put" command).

Now the FORTRAN program can be compiled, linked, and executed by pressing ALT/F1 (or by entering "RUN" at the editor's command line). (Recall that compiling, linking, and executing the FORTRAN program are accomplished by the WATFOR-77 compiler.) If the program contains no syntax errors (i.e., if there are no mistakes in FORTRAN "grammar"), the program will be executed immediately after it has been compiled and linked. Any output which is written to Unit 6 (the default unit number for output) will appear on the screen, and any input which is read from Unit 5 (the default unit number for input) may be entered at the keyboard. When the program has finished executing, "<hold>" will appear in the lower right corner of the screen; press the RETURN key to return to the editor and resume editing the FORTRAN program.

In addition to compiling, linking, and executing a FORTRAN program, the "RUN" command can create what is called a "listing file". This file is stored in the current directory of the default disk drive. It has the same primary file name as the workspace, but has an extension of "LST"; for example, if the workspace is named "TEST.FOR", the "RUN" command will create a listing file named "TEST.LST". The listing file contains a copy of the original FORTRAN program, as well as some additional information which is generated during the compilation, linking, and execution of the program. All FORTRAN statements (i.e. all non blank, non-comment, non-continuation lines) in the original program are numbered; these line numbers appear to the left of each affected line in listing file. At the end of the listing file are some statistics concerning how much time was needed to compile and execute the program, how much memory was used, etc.
If desired, the listing file can be read into the editor for examination. If the workspace containing the FORTRAN program is named "name.FOR", enter "Edit name.LST" at the editor's command line; this will create a new editor workspace containing the listing file, which can then be easily examined using the PgUp, PgDn, up-arrow, and down-arrow keys. Note that the original FORTRAN program is still in the computer's memory in a separate workspace. Repeatedly pressing ALT/N or SHIFT/F10 will alternately display the program file and the listing file in the editor's full screen area.

Be sure to "QUIT" (ALT/Q) from the workspace containing the listing file before running the program again; otherwise, the next "Edit name.LST" command will display an outdated copy of the listing file.

2. If the FORTRAN program contains syntax errors, error messages will be sent to the screen during compilation, and the program will not be executed. The compiler tries to the best of its ability to locate and describe all syntax errors contained in the program. Sometimes, however, compiler error messages can only give one a "feel" for where the true problem lies, rather than a precise description of the nature and location of the problem.

Note that the editor numbers every line in the original program, whereas the listing file numbers only lines which contain FORTRAN statements; thus, the editor's line numbers are usually different than from the statement numbers which appear in the listing file! An error message sent to the screen during compilation may refer, for example, to "line 23"; this 23 refers to the editor's line number in the original program and not to the listing file's statement number. An error message sent to the screen during execution may refer to, say, "statement 41"; this refers to the statement 41 in the listing file, not to line 41 in the editor's workspace. Remember: Compilation errors refer, when possible, to the line numbers in the editor workspace, whereas execution errors always refer to statement numbers in the program listing.

3. Pressing ALT/F1 (or by entering "RUN" at the editor's command line) is the most basic way to compile, link, and execute the FORTRAN program contained in the current workspace.

The first seven ALTERNATE function keys provide easy access to a number of variations on the "RUN" command, including, for example, the ability to direct the program listing and unit 6 output to various devices (the screen, disk, or printer), and aids for debugging both compilation and execution errors. You may grow to prefer these variations over the plain vanilla "RUN" command described earlier.

ALT/F1 will compile, link, and execute (i.e., "RUN") the program in the current workspace; unlike the ordinary "RUN" command, ALT/F1 will not produce a listing file. ALT/F1 thus saves disk space and is slightly faster than the regular "RUN" command.

ALT/F2, ALT/F3, and ALT/F4 also "RUN" the program in the current workspace; each of these three ALTERNATE function keys produces a program listing. Unlike the ordinary "RUN" command, however, these function keys send the program listing and unit 6 output to the same
device. ALT/F2 sends the program listing and unit 6 output both to the screen, ALT/F3 sends both to the current directory of the default disk drive, and ALT/F4 sends both to the printer (assuming that a printer is available).

4. ALT/F5 and ALT/F6 are useful for correcting compilation errors. ALT/F5 will attempt to compile the program in the current workspace. If the program contains no compilation errors, the program will be linked and placed into execution. (Thus, ALT/F5 is another variation on the "RUN" command.) If instead the program contains errors in FORTRAN syntax, ALT/F5 will place the cursor in the full screen area on the line which the compiler believes to be at fault; the corresponding compilation error message will be displayed at the bottom of the screen. If the compiler does not know which line is responsible for a particular error, the editor will go to the top of the file, and the error message will again be displayed at the bottom of the screen.

After correcting the first compilation error, press the function key F9 to return the cursor to the command line. Press ALT/F6 to locate the next compilation error contained in the program. Note that ALT/F6 may be used repeatedly until all subsequent compilation errors have been located; it is not necessary to recompile in order to find subsequent errors. When there are no more compilation errors to be located, a message to that effect will appear for a few moments at the bottom of the screen. After locating (and hopefully correcting) all the compilation errors, press ALT/F5 to recompile the program and see whether all errors in syntax have indeed been removed from the program.

5. For your convenience, all of the ALTernate function keys relevant to Engineering I are listed and briefly described below.

ALT/F1: RUN (i.e., compile, link, and execute) the program in the current editor workspace; do not produce a compiler listing.

ALT/F2: RUN the program in the current workspace and produce a compiler listing; direct the compiler listing and unit 6 output to the screen.

ALT/F3: RUN the program in the current workspace, directing the compiler listing and unit 6 output to the current directory of the default disk drive.

ALT/F4: RUN the program in the current workspace, directing the compiler listing and unit 6 output to the printer.

ALT/F5: RUN the program in the current workspace and locate the first compilation error. (Here, "locate" means to position the cursor in the editor workspace on the line containing the compilation error while displaying the relevant error message at the bottom of the screen.)
ALT/F6: Locate the next compilation error in the program in the current workspace, provided that this program has already been compiled using ALT/F5. ALT/F6 may be used repeatedly until all compilation errors have been located.

ALT/F7: RUN the program in the current workspace, invoking WATFOR-77's interactive DEBUGger. (This is useful for debugging execution errors. Use ALT/F10 to display a list of the fundamental DEBUGger commands.)

ALT/F9: Print the file in the current workspace.

ALT/F10: Display a help screen which briefly describes the ALTermate function keys and the fundamental DEBUGger commands.

3.5 Using the WATFOR-77 Debugger

Once a program is free of syntax errors, the program can be compiled successfully, linked, and then executed. If WATFOR-77 generates an error message during the execution of the program or if your program contains errors in logic, you may wish to recompile the program using ALT/F7. This variation on the "RUN" command will compile the program and then invoke the WATFOR-77 interactive DEBUGger rather than execute the program immediately; this facility is well suited to debugging execution errors.

Enter "Go" at the DEBUG prompt in order to place the program into execution. When the execution error is encountered, WATFOR-77 will issue the appropriate error message(s), display the FORTRAN statement in which the execution error occurred, and then return to the DEBUG prompt. Enter "Display" at the DEBUG prompt to display the 5 executable FORTRAN statements before and after the statement in which the execution error occurred (i.e., the "current statement"). Enter "Display i:j" to display statements "i" through "j", or "Display *" to display all the executable statements in the subroutine or program in which the execution error occurred (i.e., the "current program unit"). Variable values can be examined with the DEBUG command "?" or "? variblename". Variable values can be set or modified with the DEBUG command "SET variblename value". After examining the relevant executable statements and variables, and perhaps changing the values of one or more variables, you may wish to enter the "Go" command again to attempt to resume the execution of the program.

Pressing CTRL/BREAK during the execution of a program started from the DEBUGger will interrupt execution and return to the DEBUG prompt. Entering "Help" at the DEBUG prompt will briefly list all of the DEBUG commands. In order to leave the interactive DEBUGger and return to the editor, enter "Quit" at the DEBUG prompt. For more information on the WATFOR-77 interactive DEBUGger, consult Chapter 5 of the "WATFOR-77 User's Guide for the IBM PC with DOS."
3.6 Large Programs & Memory Limitations in WATFOR-77

1. When working with large FORTRAN programs, WATFOR-77 users may occasionally "bump their heads" on the upper limits of the personal computer's resources. It is important for those who process large FORTRAN programs with WATFOR-77 to understand the nature of these limitations and to know how to get around them.

There is an upper limit to the size of files which can be edited by the WATCOM editor as well as a limit on the size of files which can be compiled with the WATFOR-77 compiler. If the user tries to edit a file whose size exceeds the capabilities of the WATCOM editor, only part of the file will be loaded into the editor's workspace; some portion at the end of the file will be cut off (from the editor's copy of the disk version of the file). If the user tries to compile a file which is too large, compilation will abort with an error message such as "dynamic memory exhausted" or "object memory exhausted." The exact limits vary somewhat depending on the total amount of available memory in the computer, on how WATFOR-77 is configured, and on how WATFOR-77 is used.

2. Under no circumstances can the WATCOM editor edit files larger than 64K bytes (i.e., 65,536 characters). In actual practice, the upper limit is usually about 60,600 characters (plus or minus, say, 50 characters.) When editing a file, the exact amount of free space remaining (number of characters) is displayed on the line above the editor command line; it is the right-most of the three numbers in parentheses. (It may be necessary to press the RETURN key once in order to cause these numbers to be displayed.)

How can a FORTRAN program larger than 64K bytes be created using the WATCOM editor? The program can be split into pieces, where each piece contained in a separate file and each file is smaller than 64K bytes. How are these separate pieces combined to form a single FORTRAN program? This can be accomplished via the C$INCLUDE compiler option; that is, whenever the compiler encounters a comment line of the form

C$INCLUDE filename

the specified file is included into the source code of the program being compiled.

Consider the following simple example. Suppose that a large program is contained in three separate files called PART1.FOR, PART2.FOR, and PART3.FOR, and suppose that a fourth file called MAIN.FOR consists of these lines:

```
PROGRAM SAMPLE
C$INCLUDE PART1.FOR
```

1This section is optional reading for ENGR-1 students.
The entire program can be compiled, linked, and executed by running MAIN.FOR; the compiler will read from the files PART1.FOR, PART2.FOR, and PART3.FOR where indicated by the CSINCLUDE compiler option.

NOTE: WATCOM offers a disk-based version of WEDIT, called DEDIT, that overcomes the 64K byte limit. Interested users should direct inquiries to WATCOM. Note also that there are some good text editors for the PC available in the public domain; some of these, such as MICRO-EMACS, can edit very large text files.

3. Occasionally when WATFOR-77 tries to compile a large FORTRAN program, compilation aborts with the error message "dynamic memory exhausted." This means that for some reason, there is not enough memory available to the WATFOR-77 compiler to hold the intermediate results of compilation; when this is the case, compilation cannot be completed unless special action is taken.

Included with the complete WATFOR-77 software package is a utility program called CONFIG.COM. This utility program allows the user to tailor WATFOR-77 to suit individual needs. In particular, CONFIG.COM allows the user to set the number and size of file buffers as well as the size of the stack used by the WATFOR-77 compiler. Note that increasing the default file buffer space and/or stack size will decrease the amount of dynamic memory available by an essentially equal amount; similarly, decreasing file buffer space and/or stack size will increase the amount of dynamic memory available, and may do so enough to permit the program to compile successfully.

4. "Object memory" refers to that portion of memory which is set aside to hold the compiled program. The maximum amount of object memory permitted is limited only by the total amount of available memory in the PC; that is, the upper limit varies depending upon how much memory the machine contains and how much of that memory has already been reserved for use by various PC programs (such as the operating system, memory resident programs, virtual disks, and even WATFOR-77 itself.)

If compilation aborts with the error message "object memory exhausted," one possible solution is to use a PC which contains more memory. The MS-DOS command CHKDSK (found on MS-DOS disk I) will display the total amount of memory in a PC as well as how much of that memory is available for use (which is the more relevant quantity in this case). Note that all of the PC's at Lehigh's public microcomputing sites contain at least 512K bytes of memory.
If the computer contains a large amount of memory, but comparatively little of that memory is available for use, then it may be fruitful to free some of the memory which has been committed to other purposes. For example, a virtual disk (or "RAM" disk, i.e., a portion of memory which has been set up to behave like a very fast disk drive) occupies memory space which could be freed to provide more object memory. Similarly, unloading any memory resident programs (such as TURBO Lightning or Sidekick) which have been installed would free additional memory for use as object memory.

In the previous version of the WATFOR-77 compiler (version 2.0), the editor's workspace and the compiler's dynamic memory shared the same 64K byte portion of memory when the compiler was used in interactive mode (i.e., with the integrated editor interface); this is no longer the case. When the current release of the WATFOR-77 compiler (version 3.0) is used in interactive mode, the editor's workspace and the compiler's object memory now share the same, albeit much larger, portion of memory. This suggests that large FORTRAN programs compiled with V3.0 of WATFOR-77 are less likely to encounter the "dynamic memory exhausted" compilation error but may be more likely to encounter the "object memory exhausted" compilation error. This also suggests that emptying the editor's workspace or using the WATFOR-77 compiler in batch mode may increase the amount of object memory available, which is indeed the case.

If compilation aborts with the error message "object memory exhausted" when the compiler is used in conjunction with the integrated text editor, the following steps may well remedy the problem.

(a) Enter "Put" or "Put filename" at the editor command line in order to save the program to disk.

(b) After saving the program, clear the editor workspace by entering "* Delete" at the editor command line. This may free enough object memory to allow compilation to run to completion. (A maximum of about 60,600 bytes of object memory may be freed by this command, depending on the size of the program being edited.)

(c) In order not to confuse WATFOR-77, rename the editor workspace so that it will have a different name than the name of the program which has been saved to disk. For example, if the program has been saved to disk as "ABC.FOR", it would be safe to rename the editor workspace to "X.FOR" by entering "Name X" at the editor command line.

(d) Compile, link, and execute the program from disk by entering

\[ \text{RUN}/\text{option1}/\text{option2}/\ldots/\text{optionN} \text{ filename} \]

at the editor command line, where "/option1", ..., "/optionN" are any valid compiler options (such as /LIST and /TYPE) and "filename" is the name of the program as saved
to disk in step (a). For example, to run the program saved to disk as "ABC.FOR" and direct the listing to the screen, enter

RUN/TYPABC

at the command line.

(e) Should it be necessary to make changes to the program, enter "Edit filename" at the command line (e.g., "Edit ABC"). Use the "Edit" command rather than the "Get" command, since "Edit" gives the workspace a name (the same as the filename specified) whereas "Get" does not. After making the necessary changes, run the program by repeating steps (a) through (d) above.

32,400 bytes of additional object memory can be gained over the above method by using the compiler without the WATCOM text editor. This may be accomplished by entering

WATFOR77 /NOEDIT/option2/.../optionN filename

at the DOS prompt, where "/option2", ..., "/optionN" denote valid compiler options and "filename" is the name of the FORTRAN program to be compiled. Note that in this context it is essential to put a space between "WATFOR77" and the compiler options which are listed; the compiler options themselves must not be separated by spaces. Note that this method of using the compiler also permits one to compile FORTRAN programs which exceed the 64K limit on file size imposed by the WATCOM editor.

It may be possible to free some memory for use as object memory via the /DYNAMIC=n compiler option. This compiler option limits the maximum amount of dynamic memory available to "n" bytes, where "n" may be a whole number of bytes or a multiple of 1024 bytes (which is indicated by a suffix of "K"). For example, entering

RUN/DYNAMIC =100

at the command line of the integrated compiler/editor will limit the maximum amount of dynamic memory to 100 bytes, whereas entering

RUN/DYNAMIC =4K

will limit the maximum amount of dynamic memory to 4096 bytes. By default, the maximum amount of dynamic memory is 64K. In some cases, reducing the amount of dynamic memory allocated can free enough memory for use as object memory to allow the program to compile successfully. There is obviously a balance to be maintained here, since allocating too little dynamic memory could cause compilation to abort with the error "dynamic memory exhausted." If this should happen, refer to the previous section! (The "Object/Dynamic bytes free" statistic
found at the end of the listing file may be of some use when adjusting the \texttt{/DYNAMIC=n} compiler option.)

If none of the above eliminates the error "object memory exhausted," try reducing the size of any large arrays wherever possible. Since data as well as compiled code is stored in object memory, this may solve the problem.
4. SOME HELPFUL HINTS FOR ENGINEERING-1

The microcomputer user must be aware of a number of command statements and what they do. These are listed below according to operational categories.

4.1 MS-DOS Commands

These commands may be given when the MS-DOS prompt (e.g., "B:\>") is on the screen.

**DIR** Lists the names of the files in the directory of the disk in the default drive.

**DIR A: /P** Lists the names of the files on the disk in drive A, and pauses after every screen of output.

**DIR B:** Lists the names of the files on the disk in drive B.

**TYPE SAMPLE1.LST** Displays on the screen the contents of the file named "SAMPLE1.LST" on the disk in the default drive.

**DEL SAMPLE1.LST** Erases the file named "SAMPLE1.LST" from the disk in the default drive.

**RENAME DATA CIVIL.DTA** Renames the file named "DATA" on the disk in the default drive to the new name, "CIVIL.DTA".

**FORMAT B:** Formats the disk in your drive B. You must format a new disk before using it for the first time. (IMPORTANT: An ENGR-1 WATFOR-77 diskette or MS-DOS Disk-I must be in drive A before executing this command.)

**COPY A:PROGA.FOR B:PROGB.FOR** Copies the file called "PROGA.FOR" on the disk in drive A to a file called "PROGB.FOR" on the disk in drive B.

**PRINT B:PROG2.FOR** Gets the file called "PROG2.FOR" on the disk in drive B and prints the contents on a dot-matrix or HP Laser Jet printer. (You realize, of course, that the microcomputer used to do this must be physically wired to the printer. This command also requires ENGR-1’s WATFOR-77 disk or MS-DOS Disk-I; note that printing on your Laser Jet is best done with WATFOR-77 disk for ENGR-1, since it is configured for this purpose and contains "LASER" command.)
To copy the entire contents of one diskette onto another diskette, you can go through the following steps.

1) Get an MS-DOS (#1) disk from the consultant on duty at a public microcomputing site.

2) Place it in drive A and boot the system.

3) Type "DISKCOPY A: B:" and remove the MS-DOS disk from drive A.

4) Place the source diskette in drive A and the target diskette in drive B and depress the return key. (The diskette and directories in drive B will first be formatted and then all of the files on the disk in drive A will be copied onto the disk in drive B.

REMEMBER: When a DISKCOPY command is invoked as described above, all the contents, if any, of the diskette in drive B will be lost.

4.2 Additional Scenarios

The following situations apply after booting with the Engineering-1 WATFOR-77 Diskette in the "A" drive.

Situation: You have a program called TEST1.FOR with WRITE(*,15). Then if you do ALT/F2 at the editor's command line, compilation and output will appear on the screen. However, if you do ALT/F3 at the editor's command line, nothing will appear on the screen. Compilation and output will be put into a file called TEST1.LST.

Situation: Have a program called TEST1.FOR with the following code:

```
OPEN(6,FILE= 'RESULT.LST')
WRITE(*,16)
```

or

```
OPEN(6,FILE= 'RESULT.LST')
WRITE(6,16)
```

Then if you do ALT/F2 at the editor's command line, compilation listing will appear on the screen but output will be in a file called RESULT.LST. If you do ALT/F3 at the editor's command line, nothing will appear on the screen. Compilation listing will be in a file called TEST1.LST, and the output will be in the file called RESULT.LST.
5. An Example FORTRAN Program

```
PROGRAM MAIN
*
* This is the main program entry point.
*
* Define called functions:
COMPLEX   NEWTON
*
* Define local variables:
COMPLEX   XO, ROOT

PRINT *, 'Enter the starting point --> ',
READ *, XO
ROOT = NEWTON(XO)
```

---

This example program implements Newton's method (often called the Newton-Raphson method) for finding roots of an equation. This method states that given an initial guess, $x_0$, then

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

So if we iterate enough times over the above expression, we should get close to the root.

I've added an extra bonus/ twist. Some equations may have real, imaginary, or complex roots or all three. So I've defined the functions to be able to handle complex numbers.

Initially, you'll be prompted for a starting point, $x_0$. Since these functions can handle complex roots, you'll have to enter the starting point as a complex number. For example, you'll have to enter $(1,0)$ to start at $1$ and you'll have to enter $(-12,22)$ to start at $-12+22i$.

As an example, consider the equation $x^3 - x^2 + 1$. Since it's a third degree polynomial, we know that it has 3 roots (not necessarily distinct). Which root this method finds depends upon the initial value that we give it.

An initial value of $(-1, 0)$ finds the root $-0.7549$.
(1, 1) finds the root $0.8744+0.7449i$
(1,-1) finds the root $0.8744-0.7449i$.

Note that this method can also find the roots of equations other than polynomials!
COMPLEX FUNCTION F ( X )
* We want to determine the roots of this function. You're welcome to change this function to be able to determine the roots of other functions. But if you do, remember to change the FPRIME function to be the derivative of whatever function that you define here. Otherwise, who knows what might happen!
* Note that you are not limited to polynomials only!
* Define function arguments:
  COMPLEX X
  F = X**3 - X**2 + 1
END
***********************************************************************
COMPLEX  FUNCTION FPRIME ( X )
* Newton's method requires the first derivative of the function whose roots we are trying to determine. The first derivative of F(X) is defined here.
* Define function arguments:
  COMPLEX X
  FPRIME = 3 * X**2 - 2 * X
END
***********************************************************************
COMPLEX  FUNCTION NEWTON ( XO )
* This function implements Newton's method as described above. It uses the definitions of f and f'. Iteration occurs until the difference between Xn+1 and Xn is very small or until a constant number of iterations has been exceeded. The root that is found is displayed and returned.
* Define function arguments:
  COMPLEX XO
* Define called functions:
  COMPLEX F, FPRIME
* Define local variables:
  COMPLEX XNEXT, XLAST
  INTEGER I
* Define constants:
  PARAMETER (EPSI=1E-5, MAX=100)
  XNEXT = XO
  I = 1
  iterate until we get close enough to root or until we perform a maximum number of iterations
  LOOP
  X = X - F / FPRIME
  I = I + 1
  if |X - XNEXT| < EPSI then stop
  XNEXT = X
  end loop
XLAST = XNEXT
XNEXT = XLAST - F(XLAST) / FPRIME(XLAST)

PRINT
   '(I5,' approx=',F9.5,F9.5,'I diff=',F9.5)',
   I, XNEXT, CABS(XNEXT-XLAST)
I = I + 1
UNTIL ((CABS(XNEXT-XLAST) .LT. EPSI) .OR. (I .GT. MAX))

* report the results

NEWTON = XNEXT
IF (I .LT. MAX) THEN
   PRINT ' (''The root found near '',F9.5,F9.5,''I'')', XO
   PRINT ' (''is at '',F9.5,F9.5,''I'')', XNEXT
ELSE
   PRINT *, 'Couldn''t find a root after ', I,
   ' iterations.'
END IF

END