Radio Shack ENGINEERING DEPARTMENT 817-390-3011
A Division of Tandy Corporation 1100 ONE TANDY CENTER, FORT WORTH, TEXAS 76102

DATE: May 1979

TO: All TRSDOS Users

FROM: Computer Services

SUBJECT: TRSDOS Version 2.2 and DISK BASIC Version 2.2

Enclosed is the latest release of the TRS-80 Disk Operating System. Both TRSDOS and DISK BASIC have been revised, to provide several new features, and to improve certain operations present in Version 2.1. Highlights of 2.2 include a BASIC program renumbering command and a BASIC program recovery command for use after pressing Reset.

This diskette also includes four special utilities: a Memory Diagnostic, a Stress Test, and a pair of programs to Copy Disk Files to Tape and vice-versa.

Read the attached 2.2 Fact Sheet (21 pages), and then:

1. Make a BACKUP of this 2.2 diskette.
2. If you have only one drive, transfer all your files from 2.1 onto 2.2 System diskettes. (See Section 4 of the Fact Sheet for detailed procedures).
3. If you have two or more drives, you can continue using the 2.1 Data diskettes, even though you have a 2.2 System diskette in Drive 0.

We recommend that you discontinue using the older version. Version 2.2 will give much better results (minimum disk I/O errors, etc.). Furthermore, Radio Shack cannot provide assistance or support for older versions of TRSDOS.

NOTE

You may have used various "patches" and modifications to your 2.1 TRSDOS (for example, the KBFIX tape) None of these patches should be used under Version 2.2. They are not needed and will interfere with its operation.

Enjoy using TRSDOS Version 2.2!

CONTENTS OF THIS FACT SHEET

1 - TRSDOS VERSION 2.2

The differences between 2.2 and 2.1 fall into two categories: improvements to existing operations and new commands.

The improvements result from "fixes" of problems in the earlier version. The Functional differences include certain changes which make the system operate differently, plus features not available in the older versions of TRSDOS.

1.1 - Improvements and Internal Changes

End-of-File on Sector Boundaries

Under Version 2.1, problems occurred when a sequential file ended exactly on a sector boundary. This caused several errors, including:

- DIRectory shows file but file is not readable
- Last sector of file not readable
- COPY failed on some files
- Wrong EOF pointer in DIR (A) listing
- "INTERNAL ERROR" on BASIC programs during load

This is fixed in Version 2.2

Reduced Disk I/O Errors

Some internal changes have been made in Version 2.2 which should reduce the frequency of disk errors such as LOST DATA DURING READ, PARITY ERROR DURING READ, DATA RECORD NOT FOUND DURING READ, and system time-outs/hang-ups during lengthy disk I/O operations.

KILL Command

The operator command KILL now eliminates ALL traces of a file in the directory.

Reminder: DO NOT KILL an open file under BASIC or BASICR, as this will produce errors in the diskette directory.

System CLOSE Routine

Under some rare conditions with 2.1, the CLOSE routine could incorrectly alter the disk directory, resulting in unreadable and incorrect data files, etc. This has been fixed under Version 2.2.

Keyboard Debounce Routine

Version 2.2 incorporates an improved keyboard debounce routine to stop multiple entries from single keystrokes.

This routine is automatically loaded with the operating system--no user action is required to activate it.

VERIFY Command (Automatic Read-after-Write)

If you used this command under Version 2.1, you probably noticed that it rarely detected a write error. In fact, it was not fully implemented in Version 2.1, so that you did not get the intended benefit from using it. Under Version 2.2, this command works as described in the TRSDOS Reference Manual.

(When VERIFY is ON, TRSDOS checks to see that the sector just written is readable. If it isn't, TRSDOS returns an error message.)

It is a very good idea to issue this command each time you initialize the system. It will slow things down a bit, but in return you'll have more confidence that your programs and data are being correctly written on disk.

See the TRSDOS manual for syntax.

FORMAT for Non-Blank Diskettes

Under Version 2.1, this utility would only format a new or bulk-erased diskette. Version 2.2 will re-format a diskette without requiring that you erase it first. Of course, all data on the diskette will be lost.

When FORMAT detects a non-blank diskette, it will display a warning message:

DISKETTE CONTAINS DATA, FORMAT OR NOT?

Answer Y if you do want to re-format, N if you do not, and press <ENTER>.

Load Format Errors

When such an error occurs in 2.1, the system prints a meaningless message, and hangs up. Under 2.2, after a LOAD FORMAT error, the system indicates which file was being referenced when the error occurred, and then returns to the DOS READY mode.

Real-Time Clock

To improve the system's tolerance of disk drive performance variations, the clock interrupt procedure has been modified. As a result of this change, the clock will run slow during disk I/O. For intermittent disk accesses, the effect will usually be negligible.

1.2 - New TRSDOS Commands

APPEND

This command lets you append (add) one file onto the end of another. This is primarily useful with data files. The syntax is:

```
:-------------------------------------------------:
: APPEND file-1 TO file-2                         :
:       where file-1 and file-2 are TRSDOS file   :
:             specifications                      :
:-------------------------------------------------:
```

APPEND copies the contents of file-1 onto the end of file-2. File-1 is unaffected, while file-2 is extended to include the contents of file-1.

For example, suppose you have two mailing lists stored in text files FTWORTH/TXT and NTEXAS/TXT. You can copy the FTWORTH file onto the end of the NTEXAS file with the command:

```
APPEND FTWORTH/TXT TO NTEXAS/TXT
```

NTEXAS will now include FTWORTH/TXT at the end, while FTWORTH/TXT will be unchanged.

NOTE

If you want to APPEND BASIC programs, both files must be stored in ASCII format (SAVEd with the A option). Also, line-numbers in file-1 must be higher than those in file-2.

BASICR (Load and run enhanced BASIC)

Version 2.2 includes two BASIC interpreters, stored in files named BASIC and BASICR. The only difference between them is that BASICR has a renumber command.

NOTE

Radio Shack applications programs are intended to be run under BASIC, not BASICR.

To load and run BASICR, type:

```
BASICR <ENTER>
```

For further details, see DISK BASIC section of this Fact Sheet.

BASIC * (Execute BASIC and Recover Program in RAM)

This command lets you return to BASIC from TRSDOS and restore the program that was resident when you exited BASIC (not BASICR). Of course, if the BASIC program area has been overlaid by programs executed under TRSDOS, you won't be able to re-cover it.

For example, suppose you are using a BASIC program and the system hangs up for some reason, requiring you to press Reset. Then under TRSDOS, simply type

```
BASIC *   <ENTER>      Note the mandatory single space
                       before the asterisk *.
```

and you will return to the READY mode in BASIC. Your program should be intact.

For another example, suppose you are typing in a BASIC program and you want to return to TRSDOS to look at the diskette directory. Then type:

```
CMD"S"   <ENTER>
DIR    <ENTER>
```

After examining the directory, you can return to BASIC and recover your program by typing:

```
BASIC *  <ENTER>
```

BASIC will skip the FILES? and MEMORY SIZE? questions, and return with the prompt:

```
READY
>_
```

You can now LIST the program to be sure it was recovered.

NOTE

Do not use the BASIC * command when there is no BASIC program in RAM. If you do, the System may "hang up", requiring a Reset or Power-Off, Power-On.

BASICR * (Execute BASICR and Recover Program in RAM)

This command lets you return to BASICR (the expanded form of BASIC) from TRSDOS and restore the program that was resident when you exited BASICR (not BASIC). For further details, see BASIC * above, since the two commands are entirely similar.

Additional Information Re BASIC * and BASICR * (Page 7.1)

Both these TRSDOS commands will use the highest 64 bytes in your TRS-80 System, regardless of whether you had reserved memory or not. So if you had a machine language routine up there, you'll have to reload it. However, the MEMORY SIZE will be correctly set when you return to BASIC--no need to re-set it.

2 - DISK BASIC VERSION 2.2 (BASIC and BASICR)

2.1 - Improvements and Differences

Memory Requirements

There are two enhanced BASICs in Version 2.2, named BASIC and BASICR. The only difference between them is that BASICR offers a program-renumber command.

Both BASICs require more RAM than did BASIC Version 1.1.:

```
HOW MANY FILES? <ENTER>
MEMORY SIZE? <ENTER>
PRINT MEM <ENTER>
```

returns 37635/21351/4867, under BASICR, and 38290/21906/5522 under BASIC (numbers refer respectively to 48K, 32K, and 16K RAM systems).

To run Radio Shack applications programs, always use BASIC, not BASICR. For further important details, see Section 5.

NOTE

If you have 48K RAM and you want to reserve some high memory via MEMORY SIZE?, you must enter a number smaller than 65530. (I.E., you cannot reserve fewer than six bytes at the top of memory). However, if you do not want to reserve any memory, simply press <ENTER> to the MEMORY SIZE? question, and BASIC (or BASICR) will use all available RAM up through 65535.

256-Byte Buffers for Random Files

In DISK BASIC Version 2.1, the random file buffers were limited to 255 bytes. In other words, you could only field a total of 255 bytes. In Version 2.2, you can field 256 bytes, allowing you to access all 256 bytes of a record.

Example:

```
10 CLEAR 300
20 OPEN "R",1,"TEST/DAT"
30 FIELD 1, 128 AS A$, 128 AS B$
40 LSET A$=STRING$(128,"A")
```

```
50 LSET B$=STRING$(128,"B")
60 PRINT A$: PRINT B$
70 PUT 1
80 CLOSE
```

This program writes two 128-byte strings into the first record of the file "TEST/DAT". Under Version 2.1, you could only have written one 128-byte, and one 127-byte string.

NOTE

Because string length is limited to 255 bytes, you cannot field the entire buffer with a single variable.

```
30 FIELD 1, 256 AS A$
```

is illegal.

Don't KILL Open files under DISK BASIC

This rule is STILL true under Version 2.2. If you KILL an Open file, you will destroy the diskette directory, causing random errors to occur with increasing frequency.
Also be sure to close all files before changing diskettes or turning off power.

Additional Sequential File Access Techniques

If you are writing data in 255-byte blocks, and you want to read the data with INPUT statements, you should leave a trailing semi-colon at the end of the PRINT #n statement, so that DISK BASIC won't put a carriage return at the end of each data block.

Example:

```
10 CLEAR 300
20 OPEN"O",1,"TEST/DAT"
30 FOR I=1 TO 5
40      PRINT #1,STRING$(255,I+65);
50 NEXT I
60 CLOSE
```

This program writes five 255-byte strings with no delimiters, because of the trailing semi-colon in line 40. Since string input stops after the 255th character, no delimiter is needed. The data can be read in by the following program:

```
10 CLEAR 1000
20 OPEN"I",1,"TEST/DAT"
30 FOR I=1 TO 5
40      INPUT #1,A$
50      PRINT A$
60 NEXT I
70 CLOSE
```

If the data had been written with carriage returns, then every other INPUT would return a null string, since it would encounter a carriage return as the first character.

2.2 - New DISK BASIC Commands

CMD"I" (Exit to TRSDOS and Issue a Simple Command)

This is similar to CMD"S", except that it lets you pass an operator command for TRSDOS to execute. The syntax for the command is:

```
:-------------------------------------------------:
:  CMD "I", operator-command                      :
:     where operator-command is a string          :
:              expression defining any TRSDOS     :
:              command or any user command        :
:              file. Typically, operator-command  :
:              is a constant enclosed in quotes.  :
:           Note that parameters are not          :
:              allowed after the operator-        :
:              command; if supplied, they will    :
:              be ignored.                        :
:-------------------------------------------------:
```

Examples:

```
        CMD"I","INVADE"  <ENTER>
```

returns you to TRSDOS and executes the command file INVADE.

```
        CMD"I","DIR"  <ENTER>
```

returns you to TRSDOS and executes the Directory command.

If you use the command:

```
        CMD"I","DIR :1 (A)"
```

TRSDOS will IGNORE the parameters ":1 (A)", and show you a directory of Drive 0.

CMD"I" cannot be used with TRSDOS commands which require parameters, like KILL, LIST, PRINT and COPY.

NAME (Renumber a BASIC program - BASICR only)

This is an addition to the set of BASIC commands. It is only available under BASICR, not BASIC. NAME lets you renumber all or part of the resident BASIC program. The syntax is:

```
:-------------------------------------------------:
:  NAME  [newline] [,[startline] [,increment] ]   :
:                                                 :
:   where newline specifies the new line number   :
:           of the first line to be renumbered.   :
:           If newline is omitted, the line       :
:           number 10 is used.                    :
:         startline specifies the line number in  :
:           the original program where you want to :
:           start renumbering. If startline is    :
:           omitted, the entire program will be   :
:           renumbered.                           :
:        increment specifies the increment to be  :
:           used between each successive          :
:           renumbered line. If increment is      :
:           omitted, 10 is used.                  :
:-------------------------------------------------:
```

NAME changes all line numbers in the specifed range, as well as all line number references appearing after GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB, ON ERROR GOTO, and ERL <relational operator>.

NOTE

NAME will add trailing blanks to line number references which contain less than 5 digits. These blanks will not accumulate during subsequent renumbering operations on the same program.

Examples:

NAME

Renumbers the entire resident program, starting with new line number 10, and incrementing by 10's.

NAME 6000,5000,100

Renumbers all lines numbered from 5000 up; the first renumbered line will become 6000, and an increment of 100 will be used between subsequent lines.

NAME 10000,1000

Renumbers line 1000 (if present) and all higher-numbered lines; the first renumbered line will become line 10000, and an increment of 10 will be used between subsequent line numbers.

```
NAME 100,,100
```

Renumbers the entire program, starting with new line number 100, and incrementing by 100's.

```
NAME ,,5
```

Renumbers the entire program, starting with new line number 10 and incrementing by 5's.

Error Conditions:

1. NAME cannot be used to change the order of program lines. For example, if the original program has lines numbered 10, 20 and 30, then the command:
   ```
   NAME 15,30
   ```
   is illegal, since the result would be to move the second line of the program ahead of the first. In this case, an ILLEGAL FUNCTION CALL error will result, and the original program will be left unchanged.

2. NAME will not create new line numbers greater than 65529. Instead, an ILLEGAL FUNCTION CALL will result, and the original program will be left unchanged.

3. If an undefined line number is used inside your original program, NAME will print a warning message, UNDEFINED LINE xxxx IN yyyy, where xxxx is the original line number reference and yyyy is the original number of the line containing xxxx.

   Note that NAME will renumber the program in spite of this warning message. It will replace the number xxxx with 5 blanks, and will renumber yyyy, according to the parameters in your NAME command.

   For example, if your original program includes the line,
   ```
   110 GOTO 1000
   ```
   but does NOT include a line 1000, then NAME will print a warning,
   ```
   UNDEFINED 1000 IN 110
   ```
   and renumber the program. The text of original line 110 will be changed to:
   ```
   GOTO <five blanks here>
   ```

3 - System Checkout Utilities

Version 2.2 includes two programs to help you determine whether your disk system is functioning properly. If you are having problems with the system (disk input/output errors, apparent loss of memory, spontaneous resets, etc.), try running these tests.

If the tests show the system is okay, then check for other possible sources of errors--low-voltage "brouwnout" conditions or very short-duration transients on the power line, static electricity discharge as diskettes are handled, etc. For more information on how to eliminate external problems, see the Radio Shack booklet, "Information Guide For New Computer Owners".

If either test detects an error, repeat the test and make sure the result is repeatable. (False errors can be caused by electrical transients and other conditions noted above.) If you are sure the error is repeatable and is hardware-related, bring the unit in to your local Radio Shack for repair.

3.1 - TEST1/CMD

This program tests the TRS-80's memory (read only and random access). In the DOS READY mode, simply type:

TEST1 <ENTER>

The program automatically tests all memory locations, no matter what memory size you have. First it checks read only memory (ROM); if everything is okay, it automatically goes on to check random access memory (RAM). If all RAM checks out okay, the program says so and prompts you to press <ENTER> to return to TRSDOS.

If the program detects a ROM or RAM error, it will display a detailed message. Repeat the test to make sure it is a valid error condition. Write the message down and bring your computer in for repair.

NOTE

TEST1/CMD changes the entire contents of RAM. Before running it, be sure you have saved a disk file copy of any valuable code you may have in RAM.

3.2 - TEST2/BAS (for 32K or 48K RAM Systems Only) NOTE!

This is a stress test for the entire disk system. It takes about 10 minutes to run. The program can test any combination of drives 0 through 3.

Before running TEST2/BAS, place a system disk in drive 0, and data disks in the other drives.

NOTE

During execution, the TEST2/BAS will create several test files. If any of your files happen to have the same name, tney will be destroyed. To prevent this from happening, and also to prevent OUT OF SPACE errors, use diskettes which do not contain any of your own files.

Under TRSDOS, type:

```
BASIC   <ENTER>
HOW MANY FILES?  <ENTER>
MEMORY SIZE?  <ENTER>
RUN"TEST2/BAS"
```

The program will ask you to select which drives you want to use. Type in the appropriate number(s), with a comma after each number except the last. If you are selecting three or fewer drives, you will need to press <ENTER> twice.

For example:

```
WHICH DRIVES ARE TO BE USED?  <ENTER>
```

tells the program to test drive 0 only.

```
WHICH DRIVES ARE TO BE USED?  0,1,3 <ENTER>
?? <ENTER>
```

tells the program to test drives 0,1 and 3.

```
WHICH DRIVES ARE TO BE USED? 0,1,2,3  <ENTER>
```

tells the program to test all four drives.

While the program is running, no further keyboard input is required. The Display will be filled with meaningless messages and characters. Periodically, the program will print the message, STILL TESTING. Ignore the random messages on the display. In fact, you can go away and let the program run; it will take about 10 minutes.

If there are no errors during the test-time, the program will say so, and will end.

If an error does occur, the program will print an error message and return you to the BASIC command mode:

```
READY
>
```

Write down the message. Check your system for obvious error causes (empty drive, disk inserted incorrectly, etc.). Put a different formatted diskette in the drive. Then re-run the test. If an error occurs, and you cannot trace it to an operator problem, write down the error message and bring it with the unit to your local Radio Shack for repair.

NOTE! - REBOOT SYS AFTER USE PROG POKES LINEPRINTER OUT TO VIDIO ADDRESS

4 - Transferring Files from 2.1 to 2.2

4.1 - If you have two or more drives

Place a 2.2 system diskette in drive 0, and the 2.1 diskette you want to copy in one of the other drives. Press Reset. Type:

```
VERIFY <ENTER>
COPY filespec:d to filespec:0   <ENTER>
```

Repeat until you have copied all your files onto 2.2 system or data diskettes.

NOTE

This is only necessary if you want to have some of your own files on the system diskette. You don't need to copy files on data diskettes; Version 2.2 can read and write to 2.1 data diskettes, as long as the 2.1 data directories are "clean" - not causing random errors.

4.2 - If you have only one drive

You will need to save your old programs on tape, and then read them back under Version 2.2. With BASIC program files, this is simple:

1. Put the 2.1 diskette in drive 0, and press Reset.
2. Type:
   ```
   BASIC <ENTER>
   HOW MANY FILES?  <ENTER>
   MEMORY SIZE?  <ENTER>
   LOAD"filespec of your program"   <ENTER>
   ```
3. Prepare your recorder to record on a blank cassette.
4. Type:
   ```
   CMD"T"  <ENTER>
   CSAVE"name"  <ENTER>
   CMD"R"  <ENTER>
   ```
5. Label the cassette.
6. Repeat steps 3 through 5 until you've saved all your programs onto cassette.
7. Put a 2.2 diskette into drive 0, and press Reset.
8. Type:
   ```
   BASIC  <ENTER>
   HOW MANY FILES?  <ENTER>
   ```

MEMORY SIZE? <ENTER>

9. Prepare the recorder to play one of recorded programs.
10. Type:
    CMD"T" <ENTER>
    CLOAD <ENTER>
    CMD"R" <ENTER>
11. List the program to be sure it loaded correctly.
12. Type:
    SAVE"filepsec for your program"
13. Repeat steps 9 through 12 until you've loaded and disk-saved all your program tapes.

To copy your data files and command files in a one-disk system, you will need to use the GETDISK/BAS and GETTAPE/BAS utilities described next.

4.3 - GETDISK/BAS and GETTAPE/BAS Utilities

GETDISK/BAS copies any disk file--random or sequential data and machine-language programs--onto cassette; GETTAPE/BAS reads the cassette data and writes it into a disk file.

NOTE

These utilities duplicate the data in the original file. However, they will not set the end-of-file pointer for sequential files. Therefore, when you duplicate a sequential file, and then input from it, you may input invalid data at the END OF THE HIGHEST NUMBERED RECORD. To prevent this from occurring, don't rely on the EOF(n) function to tell you when you've reached the end.

(You can either count the data items as they are input, or add a marker at the end of the file, and then check each value input to see if you've reached the end, similar to the way you READ from a DATA statement.)

These utilities are provided to allow one-drive users to:

1) Copy 2.1 data files onto tape
2) Read them back under 2.2 and save them as 2.2 data files.

In addition to serving this purpose, the programs may come in handy whenever you want a cassette copy of a disk data file.

Don't use GETDISK and GETTAPE to tranfer BASIC programs from 2.1 to 2.2. Use CSAVE (under 2.1) and CLOAD (under 2.2); this is much faster.

To transfer 2.1 data file onto a 2.2 diskette

First you need to copy GETDISK/BAS onto your 2.1 system diskette.

1. Put the Version 2.2 system diskette into drive 0 and press RESET.
2. Type:
   ```
   BASIC  <ENTER>
   HOW MANY FILES? <ENTER>
   MEMORY SIZE?  <ENTER>
   LOAD"GETDISK/BAS"  <ENTER>
   ```
3. Get a blank cassette ready to record the program and type:

```
        CMD"T"  <ENTER>
        CSAVE"D"  <ENTER>
        CMD"R"  <ENTER>
```

4. Put a 2.1 system diskette into drive 0 and press RESET.
5. Type:
```
        BASIC  <ENTER>
        HOW MANY FILES?  <ENTER>
        MEMORY SIZE?  <ENTER>
```
6. Prepare the recorder to play back the tape you just recorded, then type:
```
        CMD"T"  <ENTER>
        CLOAD  <ENTER>
        CMD"R"  <ENTER>
```
   List the program to be sure it loaded properly.
7. Type:
```
        SAVE "GETDISK/BAS"
```

Now you are ready to copy a 2.1 data file onto tape.

NOTE

Before starting, be sure you have enough cassettes to hold the entire file. Each 8 sectors in the file will require one side of a 10-minute cassette.

1. Run GETDISK/BAS under TRSDOS Version 2.1, DISK BASIC Version 1.1. The program will ask you the name of the file you want to save on cassette, and will then copy it onto cassettes. When a cassette is filled, the program will prompt you to load another.

NOTE

For convenience, run the program repeatedly until you have copied all your data files; then go on to step 2.

2. Put Version 2.2 into drive 0 and press RESET. Type:
```
        BASIC  <ENTER>
        HOW MANY FILES?  <ENTER>
        MEMORY SIZE?  <ENTER>
        RUN"GETTAPE/BAS"  <ENTER>
```
3. Prepare the recorder to play the tape(s) you just recorded. You can insert the tapes in any order--each tape is internally labeled with the file name and the range of records that are on it. When the program has finished reading and copying one tape, simply load in another and run it again. Repeat until you've read all your tapes.

5 - Running Applications Programs Under Version 2.2

When running Radio Shack applications programs, always use a "minimal system" diskette in drive 0, containing only the following files:

```
BOOT/SYS
DIR/SYS
SYS0/SYS, SYS1/SYS, ..., SYS6/SYS
BACKUP/CMD
BASIC/CMD  (not BASICR)
The Applications Program(s)
```

To create such a minimal system, use the TRSDOS KILL command to delete the unwanted files. (Be sure you have a safe copy of all files before Killing them!) To kill protected files, you will need to use the file's password, as follows:

```
KILL FORMAT/CMD.FORMAT  <ENTER>
KILL BASICR/CMD.BASIC  <ENTER>
```

SPECIAL NOTE FOR RUNNING THE INVENTORY CONTROL SYSTEM (CATALOG NUMBER 26-1553

To run this program under Version 2.2, you must change two program lines, as follows:

In line 40, delete the statement:

```
POKE &H5C8C,0
```

The corrected line should look like this:

```
40 POKE16425,1:CLEAR500:W$="":ONERRORGOTO90:OPEN"I",1,
"TRANSFER"
```

In line 340, delete the statement:

```
POKE &H5C8C,1
```

both times it occurs in the line. The corrected line should look like this:

```
340 PRINT@472,"END OF PROGRAM":PRINT:PRINT:IFER=1 THEN CLOSE:
CLEAR50:ENDELSECLOSE:GOSUB2750:GOSUB2720:CLEAR50:END
```

Eliminate the statement, POKE &H5C8C, from lines 3570 and 3580. Line 3570 should now include the following statements only:

```
3570 IFERR/2+1=68THENCLS:CLOSE:PRINT@458,"NO DISK IN DRIVE #1 -
RUN TERMINATED":CLEAR50:PRINT:PRINT:END
```

Line 3580 should now include the following statements only:

```
3580 IFERR/2+1=70THENCLS:PRINT@465,"UNAUTHORIZED ACCESS TO
INVENTORY SYSTEM"TAB(20)"RUN TERMINATED":PRINT:CLEAR50:END
```

6 - Important Hints on Good Programming with Disk Files

Do not leave disk files Open longer than you have to. This is because the disk files are especially vulnerable to power failures and voltage transients, accidental removal of diskettes, etc.

For example, it is NOT good practice to Open a file at the beginning of a program, and leave it open until the end of the program. Instead, you should Open the file when you are ready to read or write the data, and Close the file when you've finished.

7 - Memory Usage during Loading of DISK BASIC

NOTE

This information concerns only those customers who would like to call machine language routines from DISK BASIC, or who have some other reason to keep a machine language program in high memory while using DISK BASIC.

During execution of any of the TRSDOS commands to load BASIC (BASIC, BASICR, BASIC *, BASICR *) BASIC uses the highest 64 bytes of RAM in your Computer. If you have a machine language routine stored in this area, it will be wiped out.

There is a way to protect such programs. BASIC does not search memory for the top of RAM; it simply gets the value from TRSDOS. TRSDOS generates this value during initialization, and stores it in addresses 4049-404A (hexadecimal), referred to below as TOPMEM. The least significant byte of the address is stored first (in 4049), then the most significant byte.

For example, in a machine with 32K RAM (highest address: BFFF), TRSDOS stores the following during initialization:

HEX ADDRESS	CONTENTS
4049	FF
494A	BF

To protect a machine language program in high memory, simply set TOPMEM to point to the address just below the program. This can be done with the DEBUG commmand to modify memory, "M".

For example, suppose you have a 32K RAM machine and you want to protect the top 100 bytes of RAM (BF9C-BFFF). Then you store the address BF9B in TOPMEM, as follows:

HEX ADDRESS	CONTENTS
4049	9B
404A	BF

Now when you type either BASIC, BASICR, BASIC * or BASICR *, BASIC will use the 64 bytes below BF9C. When the MEMORY SIZE question comes up, reserve memory as you normally would.

For example, in the case described above, after putting address BF9B into TOPMEM and calling BASIC, you could simply press ENTER to the MEMORY SIZE question. BASIC will only use addresses below BF9C.

```
MEMORY SIZE? <ENTER>
```