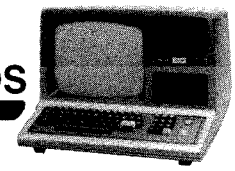


Description of TRSDOS

What Is TRSDOS?

TRSDOS (pronounced "TRISS-DOSS") stands for "TRS-80 Disk Operating System." It fulfills three roles:

1. Master Program
2. Command Interpreter
3. Program Manager

As the master program, TRSDOS enables the microprocessor and its various components to interact efficiently. The components include:

- Random Access Memory (RAM). TRSDOS reserves space for its own needs and allocates space for user programs.
- Disk Drives. TRSDOS interfaces with the disk hardware and provides a file system for storing system and user data on diskettes.
- Input/output devices. These include the keyboard, video display, printer, and RS-232-C equipment.

TRSDOS is also a command interpreter. Whenever it displays `TRSDOS READY`, you may enter commands that control how the system works. These are known as "library" commands.

In its role as program manager, TRSDOS will load and run system or user programs. During this time, the system or user program is in control of the Computer.

Figure 7 illustrates the relationships between these three roles.

Where Does BASIC Fit In?

Referring to **Figure 7**, you'll see that Disk BASIC falls under the "language package" category.

Disk BASIC consists of some general enhancements to Model III BASIC, plus the disk input/output capability. It uses Model III BASIC (stored in ROM) whenever possible. For instance, the Model III BASIC ROM includes all of the mathematical functions.

If you're used to the non-disk system, there's one difference you should understand from the beginning: In the non-disk system, BASIC is in control when you start-up. In the disk system, however, TRSDOS is in control when you start-up. You have to tell TRSDOS to load and run BASIC. Only *then* can you begin running a program written in BASIC.

How TRSDOS Uses RAM

TRSDOS is stored on the system diskette included with your Disk System. Each time the Computer is turned on or reset, the TRSDOS master program is loaded into RAM so it can take charge.

TRSDOS occupies approximately 40,000 bytes of space on the diskette; however, only a portion of that is in RAM at once. This is possible because TRSDOS is divided into several independent "modules."

The "resident" module is in memory at all times. It consists of **input/output drivers**, tables, the **command interpreter**, and other essential routines.

Additional modules are loaded as needed, and replaced (or "overlaid") by other modules when they are no longer needed.

Note: After you enter a **library** or **utility command**, you will usually hear TRSDOS accessing the system disk. It is loading an overlay module which contains the code necessary to complete the command.

The Memory Map in **Figure 8** illustrates how TRSDOS utilizes the available memory space.

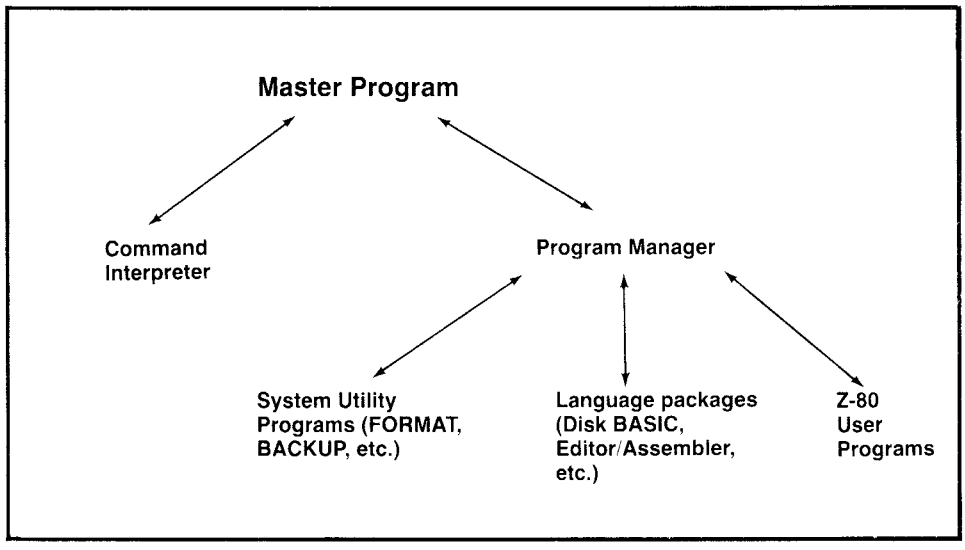


Figure 7. TRSDOS Roles.

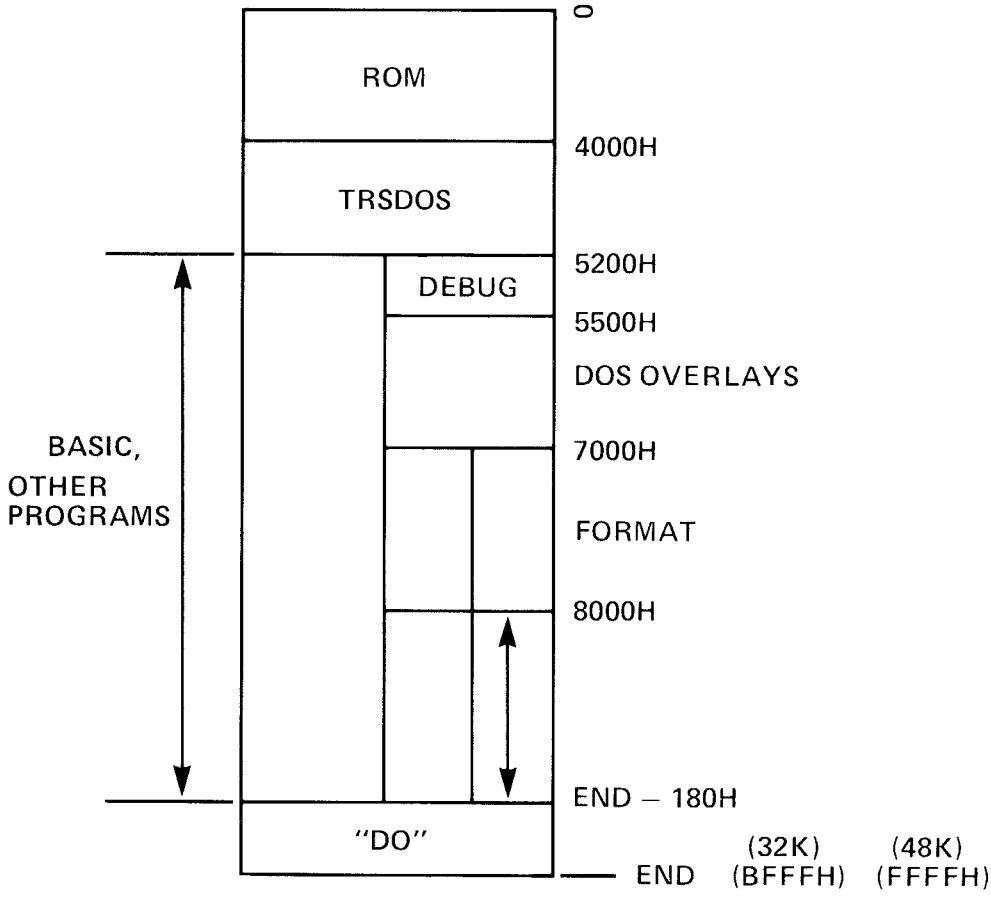


Figure 8. TRSDOS Memory Map

Using TRSDOS

Entering a Command

Whenever the TRSDOS prompt,

```
TRSDOS READY
```

is displayed, you can type in a command, which can be no more than 63 characters in length. You must press (ENTER) to end the line. TRSDOS will then "accept" the command.

For example, type: `CLS` (ENTER) TRSDOS will clear the Display and the TRSDOS READY prompt will reappear.

In general, your commands will require more than one word. For example, to kill (delete) a file named MYNAME, you have to specify the command and the filename.

`KILL MYNAME` (ENTER)

tells TRSDOS to find the file named MYNAME, eliminate it from the diskette, and release the space previously occupied by that file.

Whenever you type in a line, TRSDOS follows this procedure:

1. First it checks to see if what you've typed is the name of a TRSDOS command. If it is, TRSDOS executes it immediately.
2. If what you typed is not a TRSDOS command, then TRSDOS will check to see if it's the name of a program file on one of the drives.
3. When searching for a file, TRSDOS looks first through Drive 0, then Drive 1, etc., unless you include an particular **drive specification** with the file name —or specify the MASTER command (see **Library Commands**).

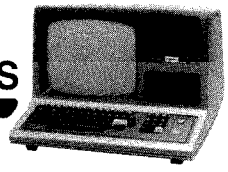
   If TRSDOS finds a specified user file, it will load and execute the file if it is a program file. Otherwise, you'll get an error message.

Command Syntax

Command syntax is a command's general form (like the grammar or structure of an English sentence). The syntax tells how to use keywords (such as DIR, LIST, CREATE, etc.) together with the necessary parameters and punctuation.

If you need help remembering the syntax form of a specific command while you're operating TRSDOS, type in:

`HELP` *command* (ENTER)



command should be the specific Library Command you're having trouble with. TRSDOS will give you the syntax format as well as a brief definition of the command.

Commands (Syntax Forms)

No-file commands

command (options) comment

options **is a list one or more parameters that may be needed by the command. Some commands have no options. The parentheses around options must be included unless the option itself is omitted.**

comment **is an optional field used to document the purpose of the command. Comments are useful inside automatic command input files (see BUILD and DO).**

One-file commands

command filename (options) comment

filename **is a standard TRSDOS file specification.**

options**— See definition above.**

comment**— See definition above.**

Two-file commands

command filename delimiter filename (options) comment

filename **is a standard TRSDOS file specification.**

delimiter **is a blank space.**

options **— See definition above.**

comment **— See definition above.**

File Specification

The only way to store information on a diskette is to put it in a disk file. Afterwards, that information can be retrieved via the file name you gave the file when you created or renamed it.

A file specification has the general form:

filename/ext.password:d

filename **consists of a letter followed by up to seven optional letters or numbers.**

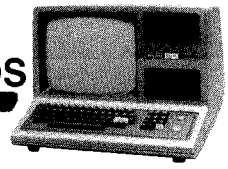
/ext **is an optional name-extension; 'ext' is a sequence of up to three letters or numbers, starting with a letter.**

.password **is an optional password; 'password' is a sequence of up to eight letters or numbers, starting with a letter.**

:d **is an optional disk-drive specification; 'd' is one of the digits 0,1,2,3.**

Note: There can be no blank spaces inside a file specification. TRSDOS terminates the file specification at the first blank space.

For example: `FILEA/TXT.MANAGER:3` references the file named FILEA/TXT with the password MANAGER, on Drive 3.



The name, extension, and drive-specification all contribute to the uniqueness of a file specification. The password does not. It simply controls access to the file.

File Names

A file name consists of a name and an optional name extension. For the name, you can choose any letter, followed by up to seven additional numbers or letters. To use a name extension, start with a diagonal slash / and add no more than three numbers or letters; however, the first character must be a letter.

For example:

MODEL3/TXT	INVENTORY	DATA11/BAS
NAMES/A12	AUGUST/A15	WAREHOUS
TEST	TEST1	TEST1/A

are all valid and distinct file names.

Although name extensions are optional, they are useful for identifying what type of data is in the file. For example, you might want to use the following set of extensions:

/BAS	for BASIC program
/TXT	for ASCII text
/CMD	for machine-language command file
/DAT	for data

One advantage of using extensions is that you can tell by just looking at the directory what kind of information a file contains.

Another advantage is that TRSDOS can recognize certain extensions. For example, if a file has the extension /CMD, the TRSDOS will load and attempt to execute that file when you type: *filename* (ENTER) omitting the extension /CMD.

Drive Specification

If you give TRSDOS a command such as: `KILL TEST/A` TRSDOS will search for the file TEST/A first in Drive 0, then in Drive 1, 2, and finally in Drive 3 until it finds the file.

Anytime you omit a drive-specification, TRSDOS will follow this sequence, unless you use the MASTER command.

It is possible to tell TRSDOS exactly which drive you want to use by specifying the drive. A drive specification consists of a colon followed by one of the digits 0,1,2, or 3, corresponding to one of the four drives you might be using.

For example: `KILL TEST/A:3` tells TRSDOS delete the file TEST/A on Drive 3 only.

Anytime TRSDOS has to open a file (e.g., to list it for you), it will follow the same sequence. When TRSDOS has to write a file, it will skip over any **write-protected** diskettes.

Password

You can protect a file from unauthorized access and use by assigning passwords to the file. That way, a person cannot gain access to a file without using the appropriate password.

It's important to realize that every file has a password, even if you didn't specify it when the file was created. In such instances, the password becomes eight blank spaces. In this case, the file becomes unprotected—anyone can gain total access simply by referring to the filename.

TRSDOS allows you to assign two passwords to a file:

- An "Update word," which grants total access to the information
- An "Access word," which grants limited access to the information (see ATTRIB)

When you create a file, the Update and Access words are both set equal to the password you specify. You can change them later with the ATTRIB command.

A password consists of a period followed by one to eight letters or numbers. If you do not assign a password to a file, TRSDOS uses a default password of eight blanks.

For example, suppose you have a file named SECRET/BAS. and the file has MYNAME as an update and access word. Then the command: `KILL SECRETS/BAS` will not cause the file to be killed. You must include the password MYNAME in the file specification.

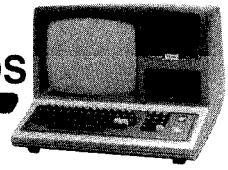
Suppose a file is named DOMAIN/BAS and has blanks for the password. Then the command: `KILL DOMAIN/BAS.GUESS` will not be obeyed, since GUESS is not the password.

A Few Important Definitions

System vs. Data Diskettes

A **system diskette** is one which contains the TRSDOS disk operating system software. Subject to space limitations, it may also contain your own files. A system diskette must always be in Drive 0 while the Computer is in use.

A **data diskette**, on the other hand, does not contain the operating system software, and therefore cannot be used in Drive 0. It may be used in Drive 1, 2 or 3. Such a diskette has a maximum of space available for storing your own programs and data.



System, Program, and Data Files

System files include the TRSDOS operating system software, the BASIC language interpreter, the FORMAT, BACKUP and CONVERT utilities, and other software which is released by Radio Shack. These files appear in the Directory with an "S" attribute. (See DIR)

Program files are stored in a special format which allows them to be loaded and executed directly from the TRSDOS READY level. For example, the BASIC interpreter is a program file.

Data files include all files that are not in the correct format to allow loading and executing from TRSDOS READY. For example, a program written in BASIC will be stored as a data file. It can be loaded and executed from BASIC, but not from TRSDOS READY.

Master Passwords

Each diskette is initially assigned a master password during FORMAT or BACKUP. (Your master password for TRSDOS is PASSWORD.) The master password allows you to use BACKUP, PURGE, and PROT on a diskette. Using a diskette's master password, you may change it (see PROT).

TRSDOS Library Commands

APPEND
Append files

APPEND *source-file destination-file*

***source-file* is the specification for the file which is to be copied onto the end of the other file.**

***destination-file* is the specification for the file which is to receive the appendage (addition).**

Note: Both *source-* and *destination-files* must be in ASCII format (data files or BASIC programs saved with the A option).

APPEND copies the contents of the source-file onto the end of the destination-file. The source-file is unaffected, while the destination-file is extended to include the source-file.

Note: The logical record lengths must match. See DIR for more information on logical record lengths.

Examples

```
APPEND WORDFILE/C WORDFILE/D
```

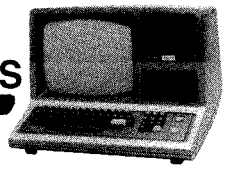
A copy of WORDFILE/C is appended to WORDFILE/D.

```
APPEND REGION1/DAT TOTAL/DAT.GUESS
```

A copy of REGION1/DAT is appended to TOTAL/DAT, which is protected with the password GUESS.

Sample Uses

Suppose you have two data files, PAYROLL/A and PAYROLL/B.



PAYROLL/A

```
Atkins, W.R.  .......
Baker, J.B.  ........
Chambers, C.P.  .....
Dodson, M.W.  .......
Kickamon, T.Y.  .....
```

PAYROLL/B

```
Lewis, G.E.  ........
Miller, L.O.  .......
Peterson, B.  .......
Rodriguez, F.  ......
```

You can combine the two files with the command:

```
APPEND PAYROLL/B PAYROLL/A
```

PAYROLL/A will now look like this:

PAYROLL/A

```
Atkins, W.R.  .......
Baker, J.B.  ........
Chambers, C.P.  .....
Dodson, M.W.  .......
Kickamon, T.Y.  .....
Lewis, G.E.  ........
Miller, L.O.  .......
Peterson, B.  .......
Rodriguez, F.  ......
```

PAYROLL/B will be unaffected. To see the APPENDed file, type `LIST PAYROLL/A (ASCII)`.

Note: Do not load a program under BASIC after an APPEND.

ATTRIB
Change a File's Password

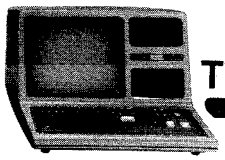
ATTRIB *file (visibility*,ACC = *name*,UPD = *name*,PROT = *level)*

***file* is the file specification.**

***visibility* must be I or N. Tells TRSDOS whether the file is Invisible (I) or Non-invisible (N) (see DIR). If omitted, *visibility* is unchanged.**

ACC = *name* tells TRSDOS the access word. If omitted, the access word is unchanged. If ACC =, is used, the access word is set to blanks.

UPD = *name* tells TRSDOS the update word. If omitted, the update word is unchanged. If UPD =, is used, the update word is set to blanks.



PROT = *level* tells TRSDOS the protection level for access. If omitted, *level* is unchanged.

Level	Degree of access granted by access word
FULL	Full access, no protection.
KILL	Kill, rename, read, execute, and write (gives total access, i.e., the least-protected).
NAME	Rename, read, execute, and write.
WRITE	Read, execute, and write.
READ	Read and execute.
EXEC	Execute only.

Note: Each level allows access to itself plus all lower levels.

ATTRIB lets you change the passwords to an existing file or makes the file invisible or non-invisible. Passwords are initially assigned when the file is created. At that time, the update and access words are set to the same value (either the password you specified or a blank password).

Examples

```
ATTRIB DATAFILE (I,ACC=JULY14,UPD=MOUSE,PROT=READ)
```

Makes the file invisible, sets the access password to JULY14 and the update password to MOUSE. Use of the access word will allow only reading and executing the file.

```
ATTRIB PAYROLL/BAS,SECRET (N,ACC=,)
```

Sets the access word to blanks. The file is made non-invisible and the protection level assigned to the update word is left unchanged.

```
ATTRIB OLD/DAT,APPLES (UPD=,)
```

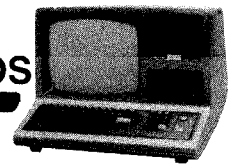
Sets the update word to blanks.

```
ATTRIB PAYROLL/BAS,PW (PROT=EXEC)
```

Leaves the access and update words unchanged, but changes the level of access.

Sample Uses

Suppose you have a data file, PAYROLL, and you want an employee to use the file in preparing paychecks. You want the employee to be able to read the file but not to change it. Then use a command like:



```
ATTRIB PAYROLL (I,ACC=PAYDAY,UPD=AVOCADO,PROT=READ)
```

Now tell the clerk to use the password PAYDAY (which allows read only); while only you know the password, AVOCADO, which grants total access to the file.

Protecting BASIC Programs

You may give a BASIC program execute-only protection using the ATTRIB command. For example, suppose the program is named TEST (no password). Under TRSDOS READY, execute this command:

```
ATTRIB TEST (ACC=,UPD=VALLEY,PROT=EXEC)
```

Now TEST has a blank access password, an update password of VALLEY, and an access level of execute only. Without using the update password, there is now only one way to execute the program:

1. Start BASIC.
2. Type: `RUN "TEST"`

   (This is the only way to access the program. If the operator attempts to LOAD it instead, BASIC will erase the program from memory before returning with READY.)

After the RUN "TEST" command, BASIC will load and execute the program. If the operator presses (BREAK) or if the program ends normally, BASIC will erase the program before returning with the READY message. This makes it impossible to obtain a listing of the program — unless the update password is used.

Of course, if you use the update password, you may gain full access to the program.

AUTO
Automatic Command after System Start-up

AUTO *command-line*

command-line **gives TRSDOS a command or the name of an executable program file created by BUILD.**

if *command-line* is given, the command will be executed on reset/power-up.

if *command-line* is omitted, the previous AUTO command is erased from the diskette.

This command lets you provide a command to be executed whenever TRSDOS is started (power-up or reset). You can use it to get a desired program running without any operator action required, except for typing in the date and time.

When you enter an AUTO command, TRSDOS writes command-line into its start-up procedure. TRSDOS does not check for valid commands; if the command line contains an error, it will be detected the next time the System is started up.

Examples

```
AUTO DIR (SYS)
```

Tells TRSDOS to execute the command DIR (SYS) after the start-up procedure. Each time the System is reset or powered up, it will automatically execute that command after you enter the date and time.

```
AUTO BASIC
```

Tells TRSDOS to load and execute BASIC each time the System is started up.

```
AUTO FORMS (WIDTH=80)
```

Tells TRSDOS to reset the printer width parameter each time the System is started up.

```
AUTO PAYROLL/CMD
```

Tells TRSDOS to load and execute PAYROLL/CMD (must be a machine-language program) after each System start-up.

```
AUTO DO STARTER
```

Tells TRSDOS to take automatic key-ins from the file named STARTER after each System start-up. See BUILD and DO.

To Erase an AUTO Command

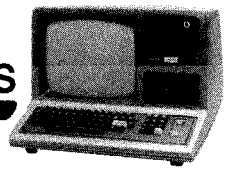
Type: `AUTO` (ENTER)

This tells TRSDOS to erase any automatic command. The command will be deleted the next time you power-up or RESET the System.

The acknowledgement: `AUTO = ""` is displayed after an erasure.

To Override an AUTO Command

You can bypass any automatic command by holding down (ENTER) while pressing RESET. You must continue holding down (ENTER) until you are prompted for the date during the initialization process.



BUILD
Create an Automatic Command Input File

BUILD *file*

***file* is a file specification which cannot include an extension.**

This command lets you create an automatic command input file which can be executed via the DO command. The file must contain data that would normally be typed in from the keyboard to the TRSDOS READY mode.

BUILD files are intended for passing command lines to TRSDOS just as if they'd been typed in at the TRSDOS READY level. **Note:** CLEAR *cannot* be used in a DO file.

When you enter the BUILD command, BUILD creates the file and immediately prompts you to begin inserting lines. Each time you complete a line, press (ENTER). (While typing in a line, you can use the usual cursor control keys for erasures and corrections.)

To end the BUILD file, simply press (BREAK) at the beginning of a line.

First type: BUILD *filename*

You will then be prompted to type in the command text. You then type in up to 63 characters, then press (ENTER). You may enter as many lines as necessary. Press (BREAK) to quit and return to TRSDOS READY.

A Sample BUILD-File

Here's a hypothetical BUILD-file that initializes the serial interface and the printer driver:

```
SETCOM (BAUD=1200,WAIT)
FORMS (WIDTH=80)
PAUSE SERIAL INTERFACE & PRINTER INITIALIZED
```

CLEAR
Clear User Memory

CLEAR (START = *aaaa*, END = *bbbb*, MEM = *cccc*)

START = *aaaa* tells TRSDOS where to start clearing user memory. *aaaa* is a four-digit hexadecimal number from 6000 to the end of user memory. If this option is omitted, 6000 is used. If this option is used, END = *bbbb* must also be used.

END = *bbbb* tells TRSDOS to clear user memory to a specified end. *bbbb* is a four-digit hexadecimal number no less than the START number and no greater than the top of memory. If this option is used, START = *aaaa* must also be used.

MEM = *cccc* sets the memory protect address. *cccc* is a four-digit hexadecimal number from 0000 to FFFF. If this option is omitted, the memory protect address is reset to end of user RAM.

If all options are omitted, all available RAM is cleared, memory-protect is reset to end of memory, the Display is cleared, all I/O drivers are reset (see Memory Requirements of TRSDOS).

This command gets you off to a fresh start.

Depending on the options you select, this command will:

- Zero user memory (load binary zero into each memory address above 6000)
- Clear the Display
- Un-protect all memory

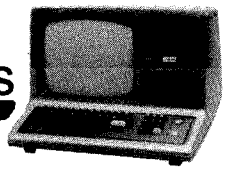
See **Memory Requirements of TRSDOS** for more information on the memory-protect address. **Note:** CLEAR *cannot* be used in a DO file.

Example

```
CLEAR (START=9000,END=0A000)
```

Note: Hexadecimal numbers which begin with a letter must be prefaced by zero (see above example).

```
CLEAR (MEM=7000)
```



CLOCK
Turn On Clock Display

CLOCK (*switch*)
***switch* gives TRSDOS one of two options, ON or OFF.**
If option is omitted, TRSDOS uses ON.

This command controls the real-time clock display in the upper right corner of the Video Display. When it is on, the 24-hour time will be displayed and updated once each second, regardless of what program is executing.

Clock display is OFF at TRSDOS start-up.

Note: Except during cassette and disk I/O, the real-time clock is always running, *regardless* of whether the clock display is on.

Examples

```
CLOCK
```

Turns on the clock display.

```
CLOCK (OFF)
```

Turns the clock display off.

See TIME and DATE.

CLS
Clear the Screen

CLS

This command clears the Display and puts it in the 64 character/line mode.

Example

```
CLS
```

COPY
Copy a File or Files

Three forms:

A) COPY ***source-file destination-file***

***source-file* is a file specification for the file to be copied.**

***destination-file* is a file specification for the name and drive of the duplicate file.**

B) COPY ***source-file :d***

***source-file* is defined above.**

***:d* tells TRSDOS to copy the file onto drive *d*, using the same file name.**

C) COPY ***/ext :d***

***/ext* is a "wild-card" file specification in which the file name is omitted and the extension is given. TRSDOS will copy all files which have a matching extension, regardless of the file name.**

***:d* is defined above.**

This command copies source-file into the new file defined by destination-file. This allows you to copy a file from one disk to another, using a single drive if necessary. (In the latter case, you must include drive specifications in *both* file specifications.) For single-drive systems (Drive 0), both diskettes must contain TRSDOS. (i.e., Data diskettes aren't allowed in Drive 0.)

Examples

```
COPY OLDFILE/BAS NEWFILE/BAS
```

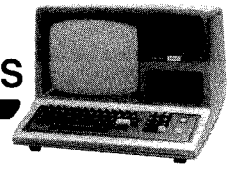
Copies OLDFILE/BAS into a new file named NEWFILE/BAS. TRSDOS will search through all drives for OLDFILE/BAS, and will copy it onto the first disk which is not write-protected.

```
COPY NAMEFILE/TXT :1
```

This command specifies a file named NAMEFILE/TXT to another disk.

```
COPY FILE/EXT:0 :1
```

This command copies FILE/EXT from Drive 0 to Drive 1.



```
COPY /BAS:0 :1
```

tells TRSDOS to copy all Drive 0 files which have the extension /BAS. The files will be copied onto Drive 1, using their present file names and extensions.

Sample Use

Whenever a file is updated, use COPY to make a backup file on another disk. You can also use COPY to restructure a file for faster access. Be sure the destination disk is already less segmented than the source disk; otherwise the new file could be more segmented than the old one. (See FREE for information on file segmentation.)

To rename a file on the same disk, use RENAME, not COPY.

CREATE
Create a Pre-allocated File

CREATE *filename* (LRL = *aaa*,REC = *bbb*)

***filename* is the file specification.**

LRL = *aaa* is the logical record length. *aaa* is a decimal number between zero and 255. If omitted, 256 is assumed.

REC = *bbb* is the number of records to allow for. *bbb* is the number of records desired. If omitted, no records are allocated.

This command lets you create a file and pre-allocate (set aside) space for its future contents. This is different from the default (normal) TRSDOS procedure in which space is allocated to a file dynamically, i.e., as necessary when data is written into the file.

If you open the file for sequential writes, TRSDOS will de-allocate (recover) any unused granules when the file is closed. If you open the file for random access, TRSDOS will not de-allocate space when the file is closed.

You may want to use CREATE to prepare a file which will contain a known amount of data. This will usually speed up file write operations. File reading will also be faster, since pre-allocated files are less segmented or dispersed on the disk — requiring less motion of the read/write mechanism to locate the records.

Examples

```
CREATE DATAFILE/BAS (REC=300, LRL=0)
```

Creates a file named DATAFILE/BAS, and allocates space for 300 256-byte records.

```
CREATE NAMES/TXT.IRIS (LRL=64,REC=50)
```

Creates a file named NAMES/TXT protected by the password IRIS. The file will be large enough to contain 50 records, each 64 bytes long.

```
CREATE PAYROLL/BAS
```

Creates a file named PAYROLL/BAS but allocates no space to it.

Sample Use

Suppose you are going to store personnel information on no more than 250 employees, and each data record will look like this:

Name (Up to 25 letters)
Social Security Number (11 characters)
Job Description (Up to 92 characters)

Then your records would need to be 25 + 11 + 92 = 128 bytes long.

You could create an appropriate file with this command:

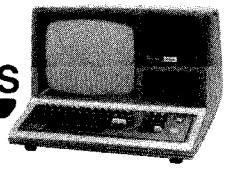
```
CREATE PERSONNL/TXT (REC=250,LRL=128)
```

Once created, this pre-allocated file would allow faster writing than would a dynamically allocated file, since TRSDOS would not have to stop writing periodically to allocate more space (unless you exceed the pre-allocated amount).

DATE
Reset or Get Today's Date

DATE *mm/dd/yy*

***mm/dd/yy* is the specification for the month (*mm*), day (*dd*) and year (*yy*).**



Each must be a two-digit decimal number between the following ranges:

mm	01-12
dd	00-31
yy	00-99

The specifications are an option; however, if one specification is used, they all must be used.

If *mm/dd/yy* is omitted, TRSDOS displays the current date.

If *mm/dd/yy* is given, TRSDOS resets the date.

This command lets you reset the date or display the date.

You initially set the date when TRSDOS is started up. After that, TRSDOS updates the date automatically, using its built-in calendar. You can enter any two-digit year after 1900.

When you request the date, TRSDOS displays it in the format: `07/25/80` for July 25, 1980.

Examples

```
DATE
```

Displays the current date.

```
DATE 07/18/80
```

Resets the date to July 18, 1980.

DEBUG
Start Debug Monitor

DEBUG

This command starts the debug monitor, which allows you to enter, test, and debug machine-language programs.

Its features include:

- Full- or half-screen displays of memory contents
- Commands for modifications to RAM and register contents
- Single-step execution of programs
- Breakpoint interruption of program execution
- Transfer of control (Jump)
- "Editing" of disk-files

DEBUG uses the memory area from X'4E00' to X'54FF' (see **TRSDOS Memory Map**). DEBUG can only be used on programs in the user area X'5500' to TOP.

Examples

```
DEBUG
```

Turns DEBUG ON. Press (Q) to quit debugging and return to TRSDOS.

```
Q
```

Turns DEBUG OFF.

Command Description

Debug commands are usually entered by pressing a single key. In most cases, you do not have to press (ENTER) after the command has been typed in. Either a prompt will immediately be displayed or DEBUG will execute the operation without further instruction.

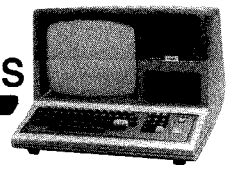
In some cases, you will have to enter a specific hexadecimal value or address (see R and J commands, for instance). Instead of pressing (ENTER) after the address is typed in, you will have to press (SPACEBAR).

Once you have entered the DEBUG program, you may use any of the following special commands:

D (Display Memory Contents)

Press (D) to display the contents of memory. TRSDOS will respond with the prompt: `D ADDRESS =` You should type in the hexadecimal address of the memory location you wish to see.

The display will be either half- or full-screen, depending on the format you are currently using (see below).



X (Half-Screen Display)

Press (X) to put the Display in the half-screen format. A 128-byte block of memory will be displayed starting with the next lowest address which is a factor of 16.

Figure 9 shows a typical half-screen format.

S (Full-Screen Display)

Press (S) to display the contents of a 256-byte block of memory starting with the next lowest address which is a factor of 256.

Note: The last 16 bytes on the Display will be overlaid by any command line typed in after the full-screen display is updated.

M (Modify RAM)

Press (M) to change to the disk utility display format (see the F command). TRSDOS will respond with the prompt: `M ADDRESS =` You should type in the four-digit hexadecimal address of the memory location you wish to modify, followed by a blank space (anything other than a space will abort the command).

The display will change to the memory edit format. The cursor will appear as a blinking character at the specified location.

To exit the modify mode, press (ENTER) to keep all changes made.

R (Change Register Contents)

Type:

R*aa*,*bbbb* (SPACEBAR)

***aa* is the name of one of the register pairs AF, BC, DE, HC, or PC.**

***bbbb* is the four-digit hexadecimal value which will be loaded into *aa*. If fewer than four digits are typed in before pressing (SPACEBAR), leading zeros are assumed.**

I (Instruction Single-Step)

Pressing (I) will allow the Computer to execute a single Z-80 instruction. The display will then be updated.

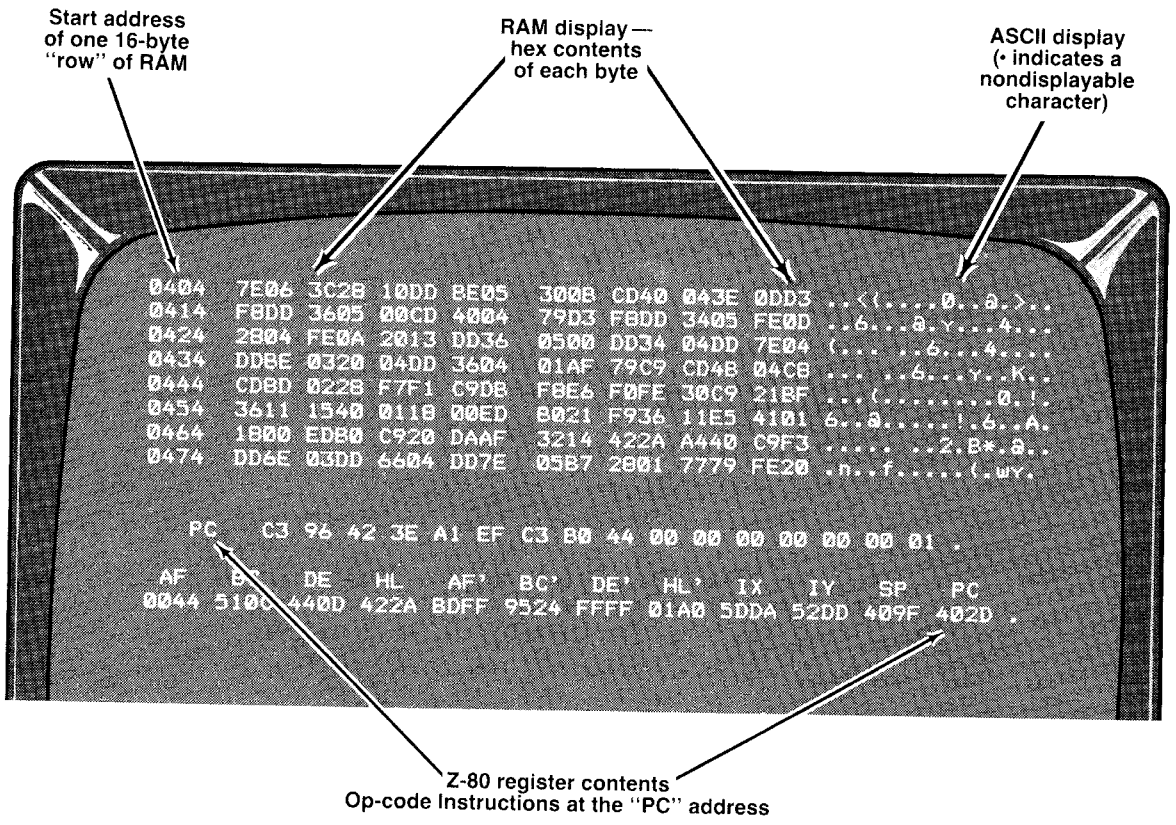


Figure 9. Half-Screen Format.

The instruction in the memory contents referenced by the program counter is executed. The program counter is increased by the appropriate value, and the control is returned to DEBUG.

DEBUG will not, however, step through a call or jump into a ROM address.

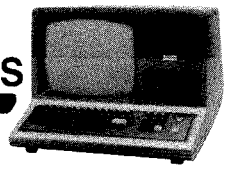
C (Call Single-Step)

If you wish to complete an entire call/return sequence, press Ⓒ. The call is then executed and control is returned to DEBUG when the subroutine returns. Otherwise, this instruction acts just like the T command.

You will not be able to step through a call or jump into a ROM address.

U (Update)

Pressing Ⓤ causes the Display to be updated repeatedly. Press any key to exit from this mode.



; (Increment Display Address)

If the Display is half-screen, the first location shown is incremented by 16 when you press (;). If the full-screen format is displayed, the starting address will be incremented by 256.

– (Decrement Display Address)

If the Display is half-screen, the first location is decremented by 16 when you press (-). If the full-screen format is displayed, the starting address will be decremented by 256.

J (Jump)

Press (J) to transfer control to a machine-language program, setting optional breakpoints.

Debug will respond with the prompt: `J ADDRESS? =`

You may type in a jump address and a breakpoint address. The command is terminated when you press (ENTER). Type in the addresses in one of three formats:

J ADDRESS? = *aaaa,bbbb* (ENTER)
J ADDRESS? = *aaaa* (ENTER)
J ADDRESS? = ,*bbbb* (ENTER)

***aaaa* is a four-digit hexadecimal address specifying the jump destination. If omitted, the address in the PC register is used.**

***bbbb* is a four-digit hexadecimal address specifying a breakpoint. Before the Computer executes an instruction at this address, it will return control to DEBUG. If this address is omitted, control will not return to DEBUG.**

Notes: Breakpoints must be set at the *beginning* of Z-80 instructions. You may not set breakpoints in ROM addresses. The breakpointed address will contain an X'F7' until the breakpoint is encountered. Then the original contents will be restored and DEBUG will take control again.

Q (Quit)

Pressing (Q) turns off DEBUG and returns control to TRSDOS.

F (File Patch Utility)

This command lets you load and modify the contents of a diskette file.

When you press (F), DEBUG will respond with the prompt: FILESPEC?. Enter the name of the file to be patched.

DEBUG will set up a full-screen display showing the first 256 bytes in the file. You can "page" through the file using the (↓) and (↑) keys.

Figure 10 gives a typical display.

In this file-display mode, both hexadecimal and ASCII are given for each byte. If a code has no displayable character, a period is shown in the ASCII area.

The display control commands are like those for the normal file-display mode:

(↓) Next page

(↑) Previous page

To change the file contents, press (M). This puts you in a modify-memory mode like the one previously described. Use the arrow keys to position the cursor (a blinking character), then type in the correct contents as a hexadecimal value. When you are through changing a page on the display, press (ENTER). The diskette file will be updated and you will be returned to the file-display mode.

To cancel changes made, do not press (ENTER), press (BREAK). This will put you back in the file-display mode without updating the diskette file. You may press (↓) then (↑) to restore the page display to its actual contents.

To quit patching a file, press (BREAK) while in the file display mode. DEBUG will prompt you for a new file specification. Press (BREAK) again and you will be returned to TRSDOS READY.

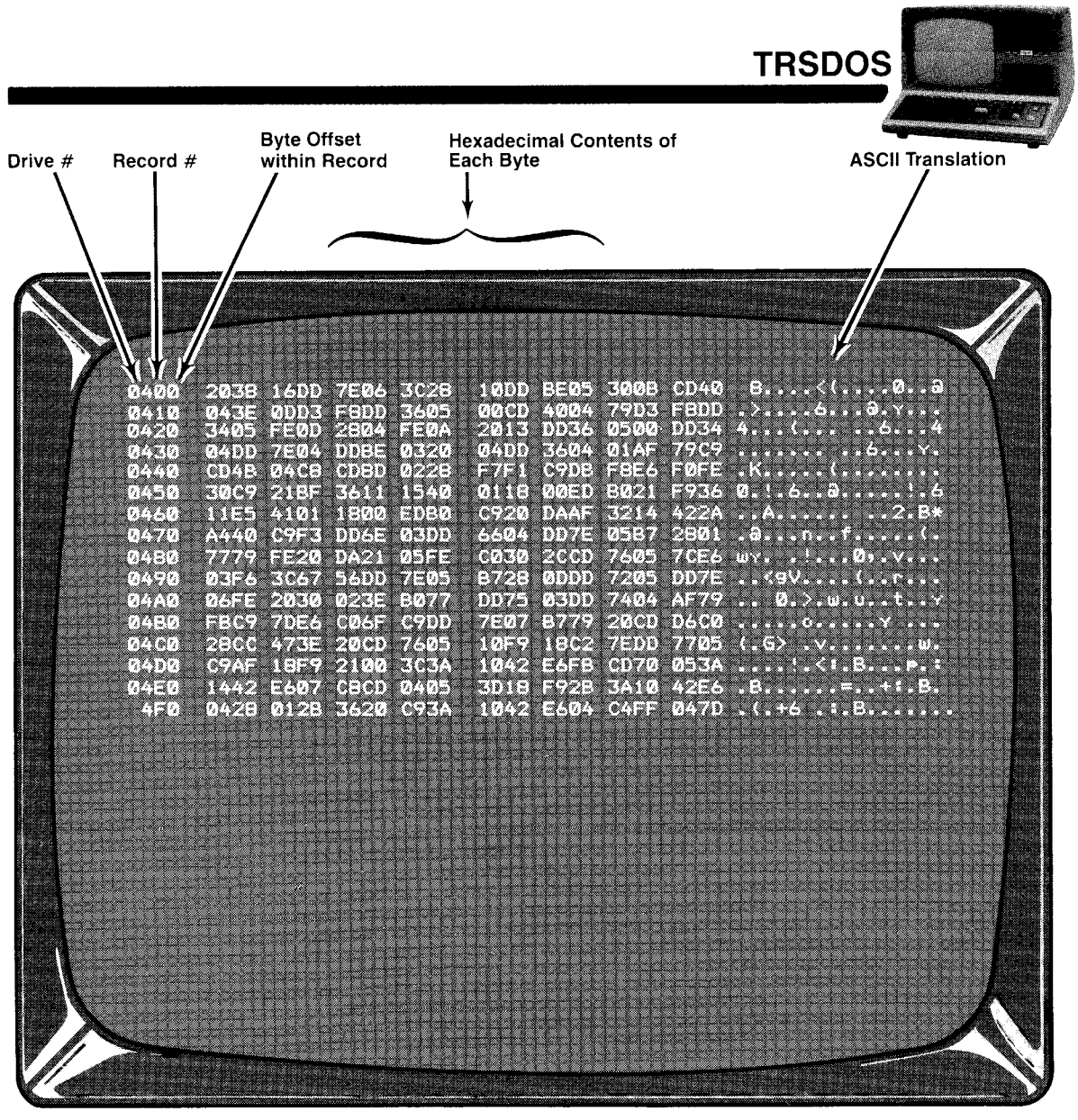


Figure 10. Full-Screen Format

DIR
List the Diskette Directory

DIR :*d* (INV,SYS,PRT)

:*d* is the desired drive directory. If omitted, Drive 0 is assumed.

INV lists the invisible user files. If omitted, non-invisible user files are listed.

SYS lists system and user files. If omitted, only non-invisible user files are listed.

PRT lists the directory to the Printer. If omitted, the directory will be listed on the Video Display only.

If no option is given, TRSDOS lists non-invisible user files in Drive 0.

This command gives you information about a disk and the files it contains.

To pause the listing, press @. To continue, press ENTER. To terminate the listing, press BREAK.

Examples

```
DIR
```

Displays the directory of non-invisible user files in Drive 0.

```
DIR :1 (PRT)
```

Lists the directory of the user files in Drive 1 to the Printer.

Sample Directory Listing

(See **Figure 11**.)
Definition of column headings

① File Name — The name and extension assigned to a file when it was created. The password (if any) is not shown.

② Attributes — A four-character field.

The first character is either I (Invisible) or N (Non-invisible).

The second character is S (System) or * (User) file.

The third character gives the password protection status:

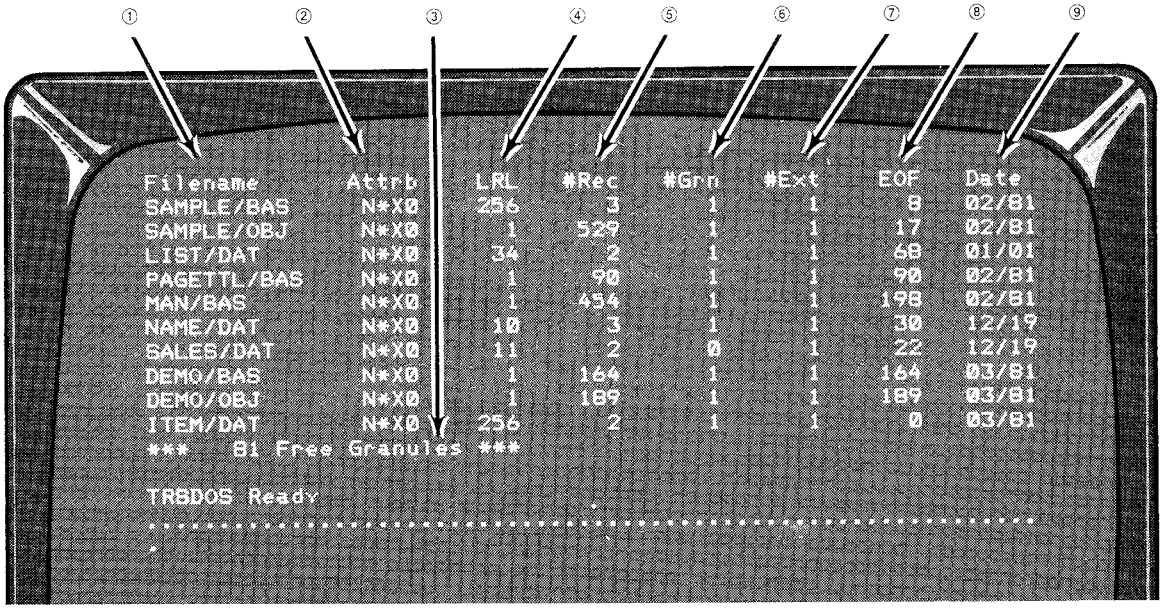


Figure 11. Directory Listing.

X The file is unprotected (no password).

A The file has an access word but no update word.

U The file has an update word but no access word.

B The file has both update and access words.

The fourth character specifies the level of access assigned to the access word:

0 Total access

1 Kill file and everything listed below.

2 Rename file and everything listed below.

3 This designation is not used.

4 Write and everything listed below.

5 Read and everything listed below.

6 Execute only.

7 No access.

③ Number of Free Granules — How many free granules remain on the diskette.

④ Logical Record Length — Assigned when the file was created.

⑤ Number of Records — How many logical records have been written.

⑥ Number of Granules — How many granules have been used in that particular file.

⑦ Number of Extents — How many segments (contiguous blocks of up to 32 granules) of disk space are allocated to the file.

⑧ End of File (EOF) — Shows the last byte number of the file.

⑨ Creation Date — When the file was created.

DO
Begin Auto Command Input from a BUILD-File

DO *command-line*

***command-line* is the name of file created with BUILD. No extension should be specified. The file will automatically be given the extension /BLD.**

This command reads and executes the lines stored in a special-format file created with the BUILD command. The System executes the commands just as if they had been typed in from the Keyboard.

Command lines in a BUILD file may include library commands or file specifications for user programs.

When DO reaches the end of the automatic command input file, it returns control to TRSDOS.

The DEBUG and CLEAR command *cannot* be included in a BUILD file.

In addition to executing TRSDOS library commands, you can load and execute user programs from a DO-file. You will probably want to make your program name be the last line in the DO-file.

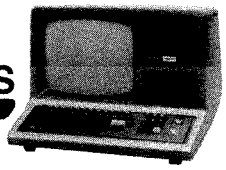
Examples

```
DO STARTER
```

TRSDOS will begin automatic command input from STARTER, after the operator answers the Date and Time prompts.

```
AUTO DO STARTER
```

Whenever you start TRSDOS, it will begin automatic command input from STARTER.



Sample Uses

Suppose you want to set up the following TRSDOS functions automatically on start-up:

```
FORMS (WIDTH=80)
CLOCK (ON)
```

Then use BUILD to create such a file. If you called it BEGIN, then use the command: `AUTO DO BEGIN` to perform the commands each time TRSDOS starts up.

DUAL
Duplicate Output to Video and Printer

DUAL (*switch*)

***switch* is one of two options, ON or OFF. If omitted, TRSDOS uses ON.**

This command duplicates all video output to the Printer, and vice versa. It takes effect immediately.

Notes:

1. Video and printer output may be different because of intrinsic differences between output devices and output software.
2. Using the DUAL command will slow down the video output process.
3. The DUAL command cannot be used during ROUTE and vice versa.
4. The printer should be ready when you execute the command.

Sample Use

For a printed copy of all system/operator dialog, type: `DUAL` (ENTER)

To turn off the DUAL process, type: `DUAL (OFF)` (ENTER)

DUMP
Store a Program Into a Disk File

DUMP *file* (START = *aaaa*, END = *bbbb*, TRA = *cccc*, RELO = *dddd*)

***file* is the file specification**

START = *aaaa* is the start address of memory block. *aaaa* must be a four-digit hexadecimal number greater than or equal to X'7000.'

END = *bbbb* is the end address of the memory block. *bbbb* must be a four-digit hexadecimal number.

TRA = *cccc* is the transfer address where execution starts when the program is loaded. *cccc* must be a four-digit hexadecimal number. If this option is omitted, the command will default to TRSDOS re-entry.

RELO = *dddd* is the start address for relocating or loading the program back into memory. *dddd* must be a four-digit hexadecimal number. If this option is omitted, no relocation will take place.

Note: Addresses must be hexadecimal form, without the x' ' notation. You must add the prefix "0" to any hex number which begins with a letter.

This command copies a machine-language program from memory into a program file. You can then load and execute the program at any time by entering the file name in the TRSDOS READY mode.

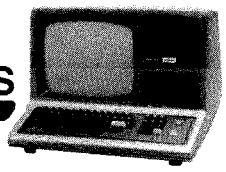
Examples

```
DUMP LISTER (START=7000,END=7100,TRA=7004)
```

Creates a program file named LISTER/CMD containing the program in memory locations X'7000' to X'7100'. When loaded, LISTER/CMD will occupy the same addresses, and TRSDOS will protect memory beginning at X'7000'. The program is executable for the TRSDOS READY mode.

```
DUMP PROG2 (START=7000,END=7F00,TRA=8010,RELO=8000)
```

Creates a program file named PROG2/CMD containing the program in addresses X'7000' to X'7F00'. When loaded, PROG2/CMD will reside from X'8000' to X'8F00'. Execution will start at X'8010'.



ERROR
Display Error Message

ERROR *number*

***number* is a decimal number for a TRSDOS error code.**

This command displays a descriptive error message. For example, after receiving the message, `* * ERROR 47 * *` you may respond with the command: `ERROR 47` (ENTER) and TRSDOS will display the full error message.

For a complete list of error codes and messages, see the **Technical Information** section of this manual.

FORMS
Set Printer Parameters

FORMS (WIDTH = *w*, LINES = *l*)

WIDTH = *w* is the maximum number of characters per output line. If a line reaches this length, TRSDOS will insert a carriage return to force a new line. If this option is omitted, the current maximum width will be used. To disable the maximum line width feature, use WIDTH = 255. TRSDOS will not force new lines.

LINES = *l* is the number of lines per page. TRSDOS does not use this value. However, BASIC will use it in computing the necessary page displacement for execution or if LPRINT CHR$(12) is executed. If LINES = *l* is omitted, the current value is used.

This command lets you modify the printer forms control features of TRSDOS. The default values are:

Maximum line width: 132

Lines/page: 60

FORMS also sets the line count to 0.

Examples

If you are using 8½″-wide forms, you will probably want to set WIDTH = 80:

```
FORMS (WIDTH=80)
```

If you are using 14″-long forms, you may want to set LINES = 78.

```
FORMS (LINES=78)
```

This change will allow the BASIC statement, LPRINT CHR$(12), to advance a page by the correct number of lines.

Notes:

1. The WIDTH you specify is stored in RAM location 16427. The LINES you specify is stored in RAM location 16424.
2. The Printer *must be ready* when you execute this command.

FREE
Display Disk Allocation Map

FREE :*d* (PRT)

:*d* is the drive specification. If omitted, Drive 0 is used.

(PRT) tells TRSDOS to send the map to the Printer.
If omitted, TRSDOS sends the map to the Video Display only.

This command gives you a map of granule allocation on a diskette. (A granule, 1280 bytes, is the unit of space allocation.) It also shows the location of the directory and any flawed sectors.

When a diskette has been used extensively (file updates, files killed, extended, etc.), files often become segmented (dispersed or fragmented). This slows the access time, since the disk read/write mechanism must move back and forth across the diskette to read and write to a file.

FREE helps you determine just how segmented your disk files are. If you decide you'd like to re-organize a particular file to allow faster access, you can then COPY it onto a relatively "clean" diskette.

Examples

```
FREE
```

Displays a free space map of the diskette in Drive 0.

```
FREE (PRT)
```

Lists the free space for Drive 0 to the Printer.

```
FREE :1 (PRT)
```

Lists the Drive 1 map to the Printer.

A Typical FREE Display

Four special symbols are used in the FREE map.

•	Unused Granule
Direct	Directory Information
X	Allocated Granule
Flawed	Granule Contains a Flawed Sector (Unusable)

A typical free map display is shown in **Figure 12**.

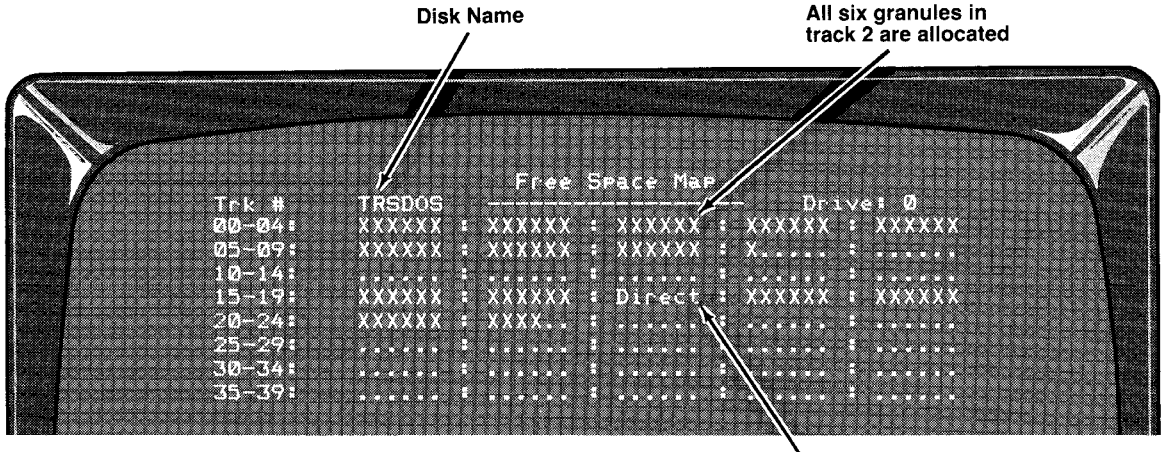


Figure 12. Free Map.

HELP
Explanation of TRSDOS Command

HELP *command*

***command* is the specific TRSDOS command or subject on which you need help. If omitted or if an invalid subject is given, TRSDOS will list all available subjects.**

Example

If you type in the following: HELP BACKUP (ENTER) TRSDOS will respond with the syntax format, a definition of the command, and an explanation of the abbreviation.

HELP SYNTAX tells TRSDOS to explain the HELP descriptions.

KILL
Delete a File or Group of Files

Two syntaxes:

A) KILL *file*
***file* is a file specification**

B) KILL */ext:d*
***/ext* is a file extension that *must* contain three characters.**
***:d* is a drive specification. It *must* be provided.**

This command deletes one file or a group of files, depending on which form is used. Form A deletes the specified file. If no drive specification is given, TRSDOS deletes the file from the first diskette that contains it.

Form B deletes all files with a specified extension, regardless of the file name of each file. If no drive specification is given, the files will be deleted from the first drive that contains a matching file specification.

Examples

```
KILL TESTPROG/BAS
```

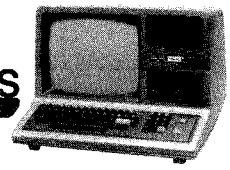
Deletes the named file from the first drive that contains it.

```
KILL JOBFILE/IDY,PASSWORD:1
```

Deletes the named file from Drive 1. The file has a password of PASSWORD.

```
KILL /BAS:0
```

Deletes from Drive 0 all files having the extension /BAS.



LIB
Display Library Commands

LIB

This command lists to the Display all the library commands. For help with a command, use HELP.

Example

```
LIB
```

LIST
List Contents of a File

LIST *file* (PRT,SLOW,ASCII)

***file* is the file specification.**

PRT tells TRSDOS to list to the Printer. If omitted, only the Video Display is used.

SLOW tells TRSDOS to pause briefly after each record. If omitted, the listing is continuous.

ASCII tells TRSDOS to list the file in ASCII format. If omitted, hexadecimal format is used.

This routine lists the contents of a file. The listing shows both the hexadecimal contents and the ASCII characters corresponding to each value. For values outside the range (X'20', X'7F'), a period is displayed.

Use the ASCII option for text files and BASIC programs saved with the A option.

Note: Only ASCII codes X'00'-X'7F' are sent to the Printer. Bit 7 is always set to 0.

During the listing, press @ to pause, ENTER to continue, or BREAK to exit.

Examples

```
LIST DATA/TXT (ASCII)
```

Lists the contents of DATA/TXT in ASCII format.

```
LIST FILE/A (SLOW)
```

Lists the contents of FILE/A, pausing after each record.

```
LIST PROGRAM/CMD (PRT)
```

Lists the file PROGRAM/CMD to the Printer.

LOAD
Load a Program File

LOAD *file*

***file* is a file specification for a file created by the DUMP command.**

This command loads a machine-language program file into memory. After the file is loaded, TRSDOS returns to the TRSDOS READY mode.

You cannot use this command to load a BASIC program or any file created by BASIC. See the BASIC Reference Manual for instructions on loading BASIC programs.

Note: The file must load into the user area (X'7000'-TOP).

Examples

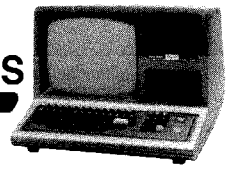
```
LOAD PAYROLL/PT1
```

Sample Use

Often several program modules must be loaded into memory for use by a master program. For example, suppose PAYROLL/PT1 and PAYROLL/PT2 are modules, and MENU is the master program. Then you could use the commands:

```
LOAD PAYROLL/PT1
LOAD PAYROLL/PT2
```

to get modules into memory, and then type: `MENU` to load and execute MENU.



MASTER
Set Master Read/Write Drive

MASTER (DRIVE = *a*)

***a* is the drive specification. If omitted Drive 0 is set as the master drive.**

This command allows you to assign a specified drive as the Master Read or Write drive in the system. When searching for a file, TRSDOS will start with the master drive.

If the file is not found on the specified drive, TRSDOS will continue searching on the next higher-numbered drive.

Example

After you enter the command: `MASTER (DRIVE=1)` Drive 1 becomes the master drive.

PATCH
Change the Contents of a Disk File

PATCH *file* (ADD = *aaaa*,FIND = *bb*,CHG = *cc*)

***file* is the file specification**

ADD = *aaaa* specifies the address at which the data is found. *aaaa* is a four-digit hexadecimal number.

FIND = *bb* specifies the string you wish to find (or compare to). *bb* is a hexadecimal sequence.

CHG = *cc* specifies the new contents for the byte(s). *cc* is a hexadecimal sequence.

Note: This utility is for machine language programs *only*.

This command lets you make minor corrections in any disk file, provided that:

1. You know the existing contents and location of the data you want to change.
2. You want to replace one string of code or data with another string of the same length.

You can use PATCH to make minor changes to your own machine-language programs; you won't have to change the source code, re-assemble it, and re-create the file.

Another application for PATCH is to allow you to implement any modifications to TRSDOS that may be supplied by Radio Shack. That way, you do not have to wait for a later release of the operating system.

Sample Use

Suppose you want to change seven bytes in a machine-language program file. First determine where the seven-byte sequence resides in RAM when the program is loaded. Then make sure your replacement string is the same length as that of the original string. For example, you might write down the information as follows:

File to be changed: VREAD

Start address: X'5280'

Sequence of code to be changed: X'CD2D25E5'

Replacement code: X'00000009'

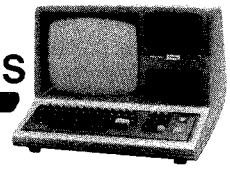
Then you could use the following command:

```
PATCH VREAD (ADD=5280,FIND=0CD2D25E5,CHG=00000009)
```

PAUSE
Pause Execution for Operator Action

PAUSE *message*

***message* is the message to be displayed during the pause execution. This is optional. If omitted, PAUSE will be displayed by itself.**



This command is intended for use inside a DO file so TRSDOS can print a message or reminder.

To continue after the pause, TRSDOS prompts you with the message:

```
PRESS <ENTER> TO CONTINUE
```

Example

```
PAUSE INSERT DISKETTE #21
```

TRSDOS displays PAUSE, next the message and then prompts you to press (ENTER) to continue execution.

```
PAUSE
PRESS <ENTER> TO CONTINUE
```

TRSDOS displays PAUSE and then next prompts you to press (ENTER) to continue. See BUILD and DO for sample uses.

PROT
Use or Change a Diskette's Master Password

PROT :*d* (PW,LOCK)

:*d* is an optional drive specification. If omitted, Drive 0 is used.

PW tells TRSDOS you want to change the master password.

LOCK tells TRSDOS to assign the master password to all unprotected user files. If omitted, the unprotected files remain unprotected.

PROT lets you use the master password to protect all unprotected files at once, or to change the master password.

The master password will be needed to BACKUP the diskette, so be sure to remember it!

Note: The master password on the TRSDOS factory-release diskette is PASSWORD.

Examples

```
PROT :Ø (PW)
```

Tells TRSDOS to change the master password on the Drive 0 diskette. TRSDOS will prompt you first for the old master password, then for the new master password.

```
PROT :1 (LOCK)
```

Tells TRSDOS to assign the master password to all unprotected user files. TRSDOS will first prompt you for the master password.

PURGE
Delete Files

PURGE :*d* (file-type)

:*d* is the drive which contains the disk to be purged.

***file-type* must be one of the following:**

SYS All System and User files (no Invisible)
INV All Invisible and User files (No System)
ALL All files on disk (User, System, Invisible)

If *file-type* is omitted, TRSDOS defaults to User files.

This command allows quick deletion of files from a particular diskette. To use PURGE, you must know the diskette's master password. (TRSDOS System diskettes are supplied with the password PASSWORD.)

When the command is entered, TRSDOS will ask for the diskette's password. Type in up to eight characters. Press (ENTER) if you typed fewer than eight characters. The System will then display user filenames one at a time, prompting you to KILL or leave each file.

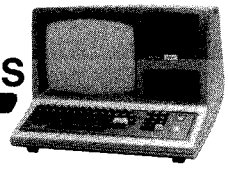
Example

```
PURGE :1
```

TRSDOS will purge user files from Drive 1. This would include BASIC programs.

```
PURGE :1 (INV)
```

TRSDOS will purge all invisible files in Drive 1.



Note: System diskettes contain some files which are not shown in any of the directory listings. You may delete these files with a special form of PURGE:

PURGE * *:d (file-type)*

The asterisk tells TRSDOS to ask you if you want to delete the System files. If you do delete them, the diskette becomes a data diskette and may only be used in Drive 1, 2 or 3.

The other parts of this command are as explained previously. However, be sure to do the PURGE using Drive 1, 2 or 3, since the diskette will become "non-system" *during* the PURGE.

RELO
Change Where Program Loads into Memory

RELO *file* (ADD = *aaaa*)

***file* is the file specification.**

ADD = *aaaa* specifies the new load address. *aaaa* is a four-digit hexadecimal number referring to an address in the user memory. *aaaa* must be in the user area of RAM.

This command allows you to change the address at which the program loads into memory. It does not change the program itself.

Note: This command may be useful in conjunction with DUMP.

Example

```
RELO PROGRAM/CMD (ADD=6578)
```

TRSDOS will load the program PROGRAM/CMD at the new memory address of 6578.

RENAME
Rename a File

RENAME *oldname newname*

***oldname* is the old file name.**

***newname* is the new file name.**

The file name may include a drive specification and or password.

The new file name should not include a drive specification or password.

This command lets you rename a file or program. Only the name/extension is changed; the data in the file and its physical location on the diskette are unaffected.

RENAME cannot be used to change a file's password protection. Use ATTRIB to do that.

RENAME also checks to see that the intended new name does not duplicate a filename currently on the same diskette. If it does, the command is cancelled and an error message is displayed.

Examples

```
RENAME MATHPAK MATHPAK/BAS
```

Tells TRSDOS to add the extension to the filename.

```
RENAME ABCDE/DAT ABCDEF/DAT
```

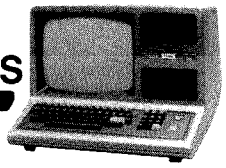
Tells TRSDOS to change the filename only.

```
RENAME PAYROLL1/TXT.GSR PAYROLL2/TXT
```

Tells TRSDOS to change the filename; the password is retained automatically.

```
RENAME FILE1:3 FILE2
```

Tells TRSDOS to change the filename of the file on Drive 3.



ROUTE
Routing I/O Devices

ROUTE (SOURCE = *aa*,DESTIN = *bb*)

SOURCE = *aa* specifies the source I/O device.

DESTIN = *bb* specifies the destination I/O device.

***aa* and *bb* may be any meaningful combination of the following two-letter abbreviations:**

DO	**(Display)**
PR	**(Printer)**
KB	**(Keyboard)**
RI	**(RS-232 Input)**
RO	**(RS-232 Output)**

If the SOURCE and DESTINATION options are omitted, TRSDOS resets I/O Drivers to their original I/O routes. The SOURCE and DESTINATION devices must both be output or both be input.

This command allows you to route I/O devices automatically. For example, TRSDOS can route information from the Printer (PR) to the Display (DO).

Note: ROUTE cannot be used in conjunction with the DUAL command.

Examples

```
ROUTE (SOURCE=PR,DESTIN=DO)
```

TRSDOS will route your Printer output to the Display.

```
ROUTE
```

I/O drivers are returned to their original state.

For further details on routing I/O see "Routing Input/Output" in the Model III Manual.

SETCOM
Set Up RS-232-C Communications

SETCOM (OFF,WORD = *a*,BAUD = *b*,STOP = *c*,PARITY = *d*,*mode*)

OFF turns RS-232-C off.

WORD = *a* is the number of bit/byte desired. *a* must be either 5, 6, 7, or 8, depending on your needs. If omitted, the word length is not changed.

BAUD = *b* specifies the baud rate. *b* must be a decimal number between 50 and 9600. If omitted, the baud rate is not changed.

STOP = *c* specifies the number of stop bits. *c* must be either 1 or 2. If omitted, stop bits are not changed.

PARITY = *d* determines whether the parity is odd, even, or none. *d* must be 3 (none), 1 (odd), or 2 (even). If omitted, parity is not changed.

***mode* type either WAIT or NOWAIT**

Options must be entered in the order shown.

If all the options are omitted, TRSDOS displays the current RS-232-C settings.

This command initializes RS-232-C communications via the serial channel. Before executing it, you should connect the communications device to the Model III.

See the Model III Operation Manual for a description of RS-232-C signals used.

See **Using the RS-232-C Interface** in the Model III Manual for further details.

Examples

```
SETCOM (WORD=7,BAUD=300,STOP=1,PARITY=3,WAIT)
```

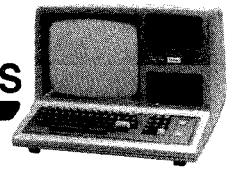
This would set the RS-232-C to seven bit words, 300 baud, one stop bit, no parity, and place it in the WAIT mode.

```
SETCOM
```

The command without specifications will display the current settings.

The following program will allow you to use your Computer as a terminal. For further information, refer to the Operation section of your Model III Operation Manual.

Note: This program executes at 300 Baud.



```
5   DEFINT A-Z              'INTEGER VARIABLE FOR SPEED
10  POKE 16890, 0           'DON'T WAIT FOR SERIAL I/O
15  POKE 16888, (5*16)+5    'TX/RCV AT BAUD RATE 300
20  DEFUSR0 = &H005A: REM SET UP CALL TO $RSINIT
40  X = USR0(0)
60  DEFUSR1 = &H0050
65  DEFUSR2 = &H0055
70  CI = 16872              'CHARACTER INPUT BUFFER
80  CO = 16880              'CHARACTER OUTPUT BUFFER
90  ' CHECK FOR SERIAL INPUT
110 X = USR1(0)             'CALL $RSRCV
120 C$ = CHR$(PEEK(CI))     'LOOK AT INPUT BUFFER
130 PRINT C$                'IF C = 0, NOTHING HAPPENS
140 ' CHECK FOR KEYBOARD INPUT
150 C$ = INKEY$
160 IF C$ = "" THEN 110     'NO KEY, SO GO CHECK SERIAL
165 PRINT C$;               'SELF ECHO
170 POKE CO, ASC(C$)        'PUT CHARACTER INTO OUTPUT BUFFER
190 X = USR2(0)             'CALL $RSTX
200 GOTO 110                'GO CHECK SERIAL INPUT
```

TAPE
Tape/Disk Transfer

TAPE (S=*source*,D=*destination*)

***source* and *destination* are abbreviations for the storage devices to be used:**

T Tape

D Disk

R Random access memory

Note: TAPE can only be used with machine-language programs. BASIC programs must be CLOADed and CSAVEed.

This command transfers Z-80 machine-language programs from one storage device to another. The following transfers are possible:

- Tape to disk

- Disk to tape
- Tape to RAM

Examples

```
TAPE (S=T,D=D)
```

Starts a tape-to-disk transfer. TRSDOS will prompt you CASS?. Select the desired baud rate (H for high, L for low). TRSDOS will then prompt you to press (ENTER) when the recorder is ready to play to the Computer. When you press (ENTER), the tape will begin loading.

Note: If no asterisks flash, the recorder volume may need adjustment or the baud rate setting may be incorrect.

TRSDOS will read the file name from the tape and use that name for the disk file. It will copy the program to the first write-enabled diskette, starting with the master drive (see **MASTER**).

```
TAPE (S=D,D=T)
```

Starts a disk-to-tape transfer. TRSDOS will prompt you for the desired cassette baud rate, then for the diskette file specification. Then it will tell you to press (ENTER) when the cassette recorder is ready to record from the Computer.

```
TAPE (S=T,D=R)
```

Starts a tape-to-RAM transfer. TRSDOS will prompt you for the cassette baud rate, and will tell you to press (ENTER) when the recorder is ready to play to the Computer. After loading the program, TRSDOS will begin execution at the transfer address specified on the tape.

TIME
Reset or Get the Time

TIME *hh:mm:ss*

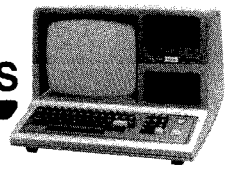
***hh:mm:ss* specifies the hour *hh*, minute *mm*, and second *ss*.**

Each must be a two-digit decimal number between the following ranges:

hh	**0-23**
mm	**0-59**
ss	**0-59**

If *hh:mm:ss* is given, TRSDOS resets the time.

If *hh:mm:ss* is not given, TRSDOS displays the current time.



This command lets you reset or display the time.

Time uses a 24-hour clock. For example, 1:00 P.M. is displayed as 13:00.

You initially set the time when TRSDOS is started up. After that, TRSDOS updates the time automatically, using its built-in clock.

When you request the time, TRSDOS displays it in this format: `14:15:31` for 2:15:31 P.M.

Examples

`TIME`

Displays the current time.

`TIME 13:20:00`

Resets the time to 1:20:00 P.M.

Note: If the clock is allowed to run past 23:59:59, it will re-cycle to zero, the date will be incremented, and the clock will continue to run.

WP
Write-Protect Via Software

WP (DRIVE = *d*)

***d* specifies the disk drive to be protected. If omitted, all drives will be unprotected.**

Diskettes can be protected from being overwritten by this command. It is a software write-protect rather than a hardware write-protect (such as the write-protect tab on the diskette).

Only one drive may be protected at a time.

To unprotect a drive, making it accessible to writing, simply enter the command WP without options or with a different drive number specified. The WP command will not override a write-protect tab.

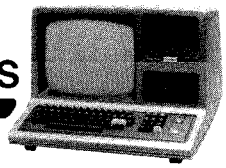
Examples

```
WP (DRIVE=1)
```

TRSDOS will write-protect the disk in Drive 1.

```
WP
```

TRSDOS will eliminate write-protection on all drives.



TRSDOS Utility Commands

BACKUP
Create an Exact Copy of an Original Disk

BACKUP *:source :destination*

***:source* specifies the drive containing the original diskette. If omitted, TRSDOS will prompt you for this information.**

***:destination* specifies the drive containing the diskette to receive the copy. If omitted, TRSDOS will prompt for it.**

***:source* and *:destination* may reference the same drive.**

BACKUP copies the contents of the source disk to the destination disk. This gives you a "safe" copy of the disk. Always keep an extra copy of data or programs you have stored on your disks.

Note: Both source and destination diskettes must be write-enabled.

TRSDOS will prompt you at each step after you type: `BACKUP`

If you omitted the source/destination-drive numbers, TRSDOS will begin with the prompts: `SOURCE DRIVE NUMBER.`

Type in the number of the drive that contains the source diskette and press (ENTER).

`DESTINATION DRIVE NUMBER?`

Type in the number of the drive that will contain the destination diskette and press (ENTER).

`SOURCE DISK MASTER PASSWORD?`

Type in the password assigned to your source diskette.

`DISK CONTAINS DATA, USE DISK OR NOT?`

Type in `Y` (Yes) or `N` (No).

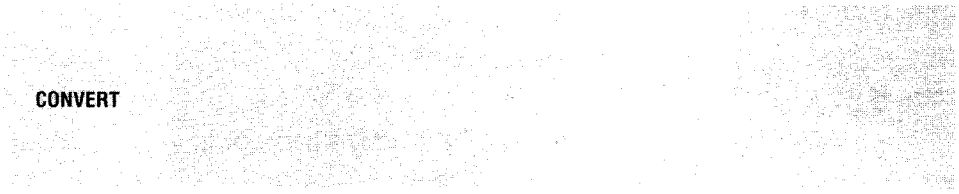
`DO YOU WISH TO RE-FORMAT THE DISK?`

Type in `Y` (Yes) or `N` (No).

If you specified the source/destination drives, TRSDOS will request the PASSWORD, skipping the first two steps.

TRSDOS will then take charge of formatting and verifying the destination disk as well as letting you know if there are any errors or flawed tracks.

CONVERT
Model I to Model III File Conversion Utility



Model I formatted diskettes cannot be used in the Model III Disk System. However, the CONVERT utility can read a Model I diskette and copy its non-system files onto a Model III TRSDOS diskette. This diskette may then be used normally in the Model III Disk System. The original Model I diskette may still be used in a Model I Disk System, since it is unchanged by CONVERT.

CONVERT does not convert or change data; it converts the *file storage format*. For this reason, Model I Disk BASIC programs may require slight changes before they will run properly in the Model III Disk System. Model I machine-language programs may require major or minor changes before they will run in the Model III Disk System. You may make these changes on the Model I diskette before using CONVERT, or on the Model III diskette containing the converted files.

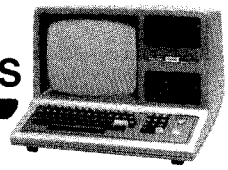
For hints on program conversion, see:

- **Technical Information** in this manual
- Technical Information in the Model III Manual
- The manual, *Instructions for Converting Specified Model I Programs for use on TRS-80 Model III.*

Drive Usage

In two-drive systems, the files must be copied onto a Model III system diskette in Drive 0; in three- or four-drive systems, the files may be copied onto a data diskette in Drive 1, 2 or 3.

During the conversion process, the Model I diskette is referred to as the "source"; the Model III diskette, the "destination." The source diskette cannot be in the same drive as that of the destination diskette.



Password Protection

CONVERT is designed to preserve the password security of each file that it transfers. To accomplish this and still allow the copying of protected files, CONVERT follows different procedures depending on the access and update passwords on each file.

In the simplest case, a file has blank access and update passwords. The copied file will be given blank passwords. (If you have a Model I Disk System with TRSDOS 2.3, you may use the PROT command to remove all passwords from all files. This will simplify the CONVERT process. Do this on the Model I system before you attempt to convert to Model III.)

In another case, the access and passwords are different. If the access password is blank and the update is not, then TRSDOS will prompt you for the update password. If you know the update password, type it in. The file will be copied with access and update passwords set to the old update password. If you don't know the update password, simply press (ENTER). The file will be copied with the access password set to blanks and the update password set to an unknown value.

If the access and update passwords are not blank and they are not the same, TRSDOS will not copy the file, but will print the message, `FILE SKIPPED`, and continue with the next file in the source directory.

Sample Use

Get the Model I diskette ready. If you have a Model I Disk System with TRSDOS 2.3, try to remove all passwords from all your files. This will prevent any problems with passwords. The password protection may be restored with the Model III ATTRIB or PROT commands after the conversion is complete.

Using the Model III Disk System, you must always have a Model III TRSDOS diskette in Drive 0. TRSDOS READY should be displayed. Type: `CONVERT` (ENTER).

The program will ask, `SOURCE DRIVE?`. Type in the number of the drive containing the Model I diskette, and press (ENTER). Then the program will ask, `DESTINATION DRIVE?`. Type in the drive number and press (ENTER). In two-drive systems, you must use Drive 0 as the destination.

During the conversion process, the name of each file will be displayed as it is copied. If password information is needed, TRSDOS will prompt you for it. If you know the update password, type it in and press (ENTER). The file will be copied and given the same update password. If you do not know the update password, simply press (ENTER), in which case the file will be copied and given an unknown update password.

If a file name on the source diskette is already used on the destination diskette, TRSDOS will print this message: `FILE EXISTS, USE IT?`. If you type `Y`, TRSDOS will copy the file. The previous contents of the Model III file will be lost. If you type `N`, TRSDOS will skip the file, and get the next one from the Model I diskette.

FORMAT
Prepare a Data Diskette

FORMAT *:d*

***:d* specifies the disk drive which contains the diskette to be formatted. If *:d* is omitted, TRSDOS will prompt you for this information.**

This command lets you prepare data diskettes (either new or disks which contain undesired data or programs), leaving a maximum amount of space for your program and data files.

Note: Data diskettes may only be used in Drives 1, 2, and 3 except during a BACKUP or FORMAT.

FORMAT takes a blank (new or magnetically erased) diskette, records track/sector boundaries on it, then initializes it with and creates a directory.

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

```
DISK CONTAINS DATA, USE DISK OR NOT?
```

Type Y (Yes) and press (ENTER) if you do want to reformat, N (No) and press (ENTER) if you want to save the disk information.

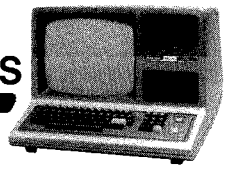
FORMAT will lock out any defective tracks to prevent data from being lost in these areas.

If you begin to get READ errors during access, reformat the disk.

Example

```
FORMAT :1
```

After you are prompted for DISKETTE NAME? and MASTER PASSWORD?, TRSDOS will format Drive 1.



HERZ50
Set Up for 50 Hz AC power (non-USA users)

DO HERZ50

starts the utility to change the system for 50 Hz operation. HERZ50 is a DO-file.

This utility is provided for customers in areas where the AC power is 50 rather than 60 Hz. It should not be used by any other customers. HERZ50 simply places a patch on the diskette that changes the clock speed for 50 Hz users.

HERZ50 is a DO-file that makes a change in the software of TRSDOS. Only the Drive 0 diskette is changed. Be sure it is write-enabled before you start the DO-file. Once the HERZ50 change is done, it will remain in effect for that diskette.

To perform the change, type:

```
DO HERZ50
```

Once the change has been made, you will need to reset the system to put the change into effect. This loads the new software into RAM.

LPC
Line Printer Control

LPC

The LPC utility program allows TRSDOS to ignore multiple carriage return commands. Without LPC, a top-of-form (LPRINT CHR$(12)) command will add an extra carriage return/line feed each time it is executed. Also, LPC masks the high bit of each data byte, allowing you to send certain intercepted codes to the printer. For instance, the BASIC statement LPRINT CHR$(140) will send code 140-128 = 12 (LPRINT CHR$(12)) to the Printer.

The printers that require LPC are:

Line Printer III (26-1156)
Line Printer VI (26-1166)
Daisy Wheel WP50 (26-1157)
Qume Daisy Wheel (26-1157A)
Daisy Wheel II (26-1158)

and all future printers.

Printers that do not require LPC:

26-1150, 1152, 1153, 1154, 1159, and the A version of LPIII (26-1156A).

You must load the LPC program before you load an application program. The easiest way to do this is to copy LPC onto your data/program diskette and then use the AUTO command to load LPC automatically each time you use the system. For instance, type:

```
COPY LPC:1 :0 (ENTER)
```

Then, to make LPC an AUTO command on the diskette, type:

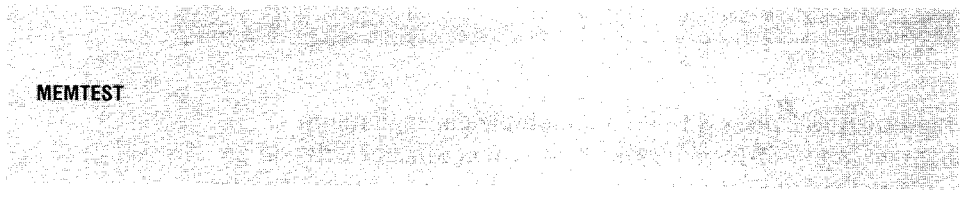
```
AUTO LPC/CMD (ENTER)
```

Whenever you use your program diskette, LPC will automatically load into memory and you can use the program as usual.

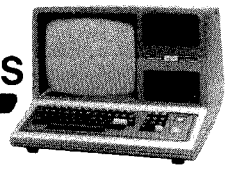
LPC locates into the highest available memory. There is no need to set MEMORY SIZE to protect LPC. It "hides" itself. However, you still need to set memory if required by your application program. LPC will be killed if the CLEAR command is used.

Warning: Once the LPC utility program is loaded and installed, you should not reload it except after a reset. Reloading re-installs the program and uses up more space each time! LPC will not execute if the Printer has been routed elsewhere. Also, if LPC has been executed and then the Printer is routed elsewhere, the original printer driver will regain control after the routing.

MEMTEST
Test Memory



This program tests your Model III's memory (read only and random access). In TRSDOS READY, just type MEMTEST and press (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 and checks random access memory (RAM). If all RAM checks out okay, the program continues.

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 contact your nearest Radio Shack for assistance.

Note: MEMTEST changes the entire contents of RAM. Before running it, be sure you have saved any valuable code you may have in RAM.

XFERSYS
Transfer System Files

XFERSYS

XFERSYS lets you upgrade your version of Model III TRSDOS by copying all system files from a new release diskette (source) onto a previously released diskette (destination) (i.e., version 1.2 to version 1.3, etc.).

System files which already exist on the destination diskette are replaced by those from the source diskette. Files which do not exist on the destination diskette are added. User files (program and data) are unaffected.

Steps to upgrade a diskette

Make backup copies of all diskettes to be upgraded. This is an important precautionary step. These backup copies should be kept until the upgrading process is complete and confirmed.

Note: Both source and destination diskettes must be write-enabled.

Insert the new release of TRSDOS into Drive 0 and press the RESET button. Then type `XFERSYS` (ENTER).

After the program heading appears, TRSDOS will prompt you with `DISKETTE TO CONVERT READY IN DRIVE 1 (Y/Q)?`. Type `Y` (yes) or `Q` (quit) and press (ENTER).

The upgrading process will then take place. When the process is complete, TRSDOS will tell you so and take you back to TRSDOS READY.

Note: If an error occurs, including your trying to upgrade a non-system diskette, the operation will be cancelled and take you back to TRSDOS READY.

Technical Information

Contents of This Section:

Disk Organization
File Structure
System Routines for Assembly I/O
    Data/Device Control Blocks
    Physical and Logical Records
    Fundamental TRSDOS I/O Calls
    Additional Routines and Storage Addresses
TRSDOS Error Codes/Messages

Disk Organization

Each TRSDOS system diskette contains a TRSDOS system, a utility command library, and a file directory.

Each diskette is single-sided and has 40 tracks of information. Each track contains 18 sectors of 256 bytes each.

Normally, data read/write operations may be initiated only at sector boundaries, and must consist of exactly 256 bytes. However, TRSDOS allows the user to have maximum flexibility with minimal effort by automatically blocking and de-blocking all file accesses to user-specified logical record lengths, even if this requires "spanning" of two sectors.

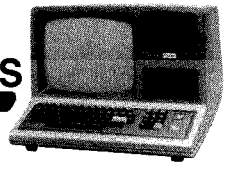
The system disk file structure allows maximum use of disk file space by automatically segmenting files across a diskette in several small pieces. These pieces are correlated into one logically contiguous file by the system without your needing to know the physical file location. This structure eliminates time-consuming disk-packing operations.

File Structure

A TRSDOS file is composed of one or more segments of storage space. Each segment consists of from one to 32 physically contiguous granules of storage. A granule is the minimum allocatable unit of storage, and consists of three sectors (768 bytes).

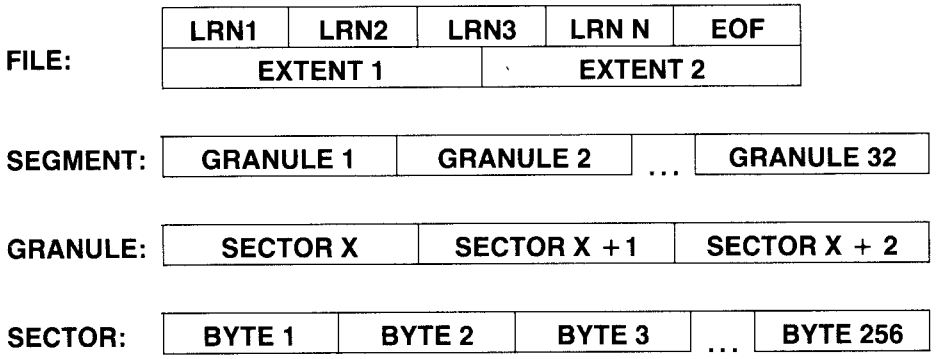
Since a file is always lengthened by granules, a small amount of free storage is generally present at the end of every file. This free storage allows minor file additions to be made in space which is physically contiguous to the file.

Every time a disk file is extended (either initialized or lengthened), extra granules may be allocated to that file, depending on the file's accumulated length, diskette space, saturation, etc. These extra granules, along with all



granules after the one containing the file's EOF mark, are recovered and returned to the system when the file is closed.

A TRSDOS file



LRN: Logical Record Number, used to specify an individual, user-defined logical record. Such a logical record is the smallest unit of information which can be addressed during disk input/output (a physical record is the unit which is actually read from or written to disk).

File: A group of logical records; the largest unit of information which can be addressed by a TRSDOS command.

Sector: A physical record, composed of 256 contiguous bytes.

Granule: The minimum allocatable unit of storage for any file.

Extent: One contiguous allocation of granules.

System Routines for Assembly-Language I/O

This information is provided for customers who wish to write their own assembly level I/O routines. An explanation of the calling sequence and parameters for each necessary I/O routine is given. A knowledge of Z-80 machine code is assumed.

The following notations are standard in this section:

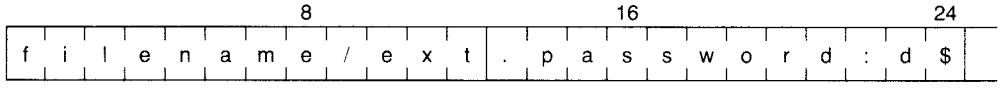
(HL) = *xxxx* Registers HL contain the address of (point to) *xxxx* in machine format. (If address of *xxxx* = 34B2H then the values in the registers are: H = 34; L = B2). Other register pairs will also be indicated this way.

A = *xx* Register A contains the numeric value of *xx* in binary form. Register A is used to return the TRSDOS error code for I/O calls. A complete list of error codes and their meanings appears at the end of this chapter. Other registers will also be indicated this way.

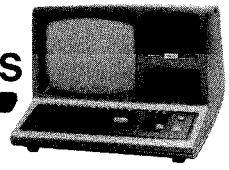
Z=OK	Zero flag is set (OK) if successful return from the system routines.
X'*nnnn*' or *nnnn*H	Hard RAM address in hex notation (e.g., 402D is X'402D').
LRL	Logical Record Length, 1-255 bytes only. You can define records any length you wish up to 255 bytes maximum. A length of zero is a special case for physical records only, and indicates the LRL=256 bytes.
BUFFER	256 user-designated bytes in RAM for TRSDOS to read sectors from or write sectors into. If LRL=0, this area is the responsibility of the user to manage before and after I/O. TRSDOS manages this area if LRL is between 1 and 255 bytes. Do not alter this area when using logical record processing.
UREC	(User Record) The address of the contiguous RAM byte-string assigned by the user as his logical record area. Its length must be equal to LRL. It is a different area from BUFFER.
LSB/MSB	Least-significant byte followed by most significant byte. This is the standard Z-80 format for addresses.
$*name*	The "$" is prefixed to all system locations and call routines, so they will not be confused with TRSDOS commands or utilities. For example, $OPEN.

DCB before $OPEN and after $CLOSE

The DCB (device control block) is defined as 50 contiguous bytes of RAM designated by the user. Before $OPEN and after $CLOSE, it is a left-justified, compressed (no spaces) ASCII string, as in a standard TRSDOS filespec. The string is terminated with a carriage return.



Notes: /ext, .password, and :d are optional.
$ stands for a carriage return (X'OD')



DCB while $OPEN

Address	Length	Explanation
DCB+0	3	Reserved
+3	2	Physical Buffer address (LSB/MSB)
+5	1	Offset to delimiter at end of current record
+6	1	File drive number residence
+7	1	Reserved
+8	1	EOF offset of last delimiter in last physical record
+9	1	LRL (logical record length)
+10	2	NRN (next record # — $OPEN sets = X'0000' — LSB/MSB)
+12	2	ERN (ending record # — (last in file) LSB/MSB)
+14	50	Reserved

NRN Next Record Number defines which record is to be read or written by the next system call for $READ or $WRITE. It is automatically incremented by one after each system call. In order to process random files, use the $POSN call to direct TRSDOS to the record you wish to transfer next.

ERN Ending Record Number is the last record number currently in the file. It is put into the directory at $CLOSE time, so if it is expected to be correct, the user *must* close his files after adding records to a file. This value may also be used to position to end of file so that new records may be added to the end of the file. To position to the end of file use a call to $POSN with a record number of ERN + 1. $POSN is described later.

Physical and Logical Records in TRSDOS

A physical record is defined as one sector of disk. One sector of disk contains 256 user data bytes. The artificial term "granule" is defined to be 3 sectors of disk space. There are 6 granules on each of the 40 tracks on the disk. A granule is the least amount of space allocated by TRSDOS. For programming purposes, the physical records in a file are numbered from 0 to N. The largest record number (N) in a file will then be 3 times the number of granules allocated minus one ((3*G) − 1). All TRSDOS granule allocations are made as needed *at the time of write, not when the file is created.*

Bytes	Sectors	Granules	Tracks	Disk
256	1	—	—	—
768	3	1	—	—
4608	18	6	1	—
184320	720	240	40	1

Disk Space Table: For each 5¼" Disk Drive

A logical record is defined by the user of TRSDOS. It may be anywhere from 1 to 255 bytes in length. Once a file is opened with a specific LRL (Logical Record

Length), the length is fixed until the file is closed. To change a file's LRL, you must CLOSE it and re-OPEN it with the new LRL.

Each opening of the file sets a single, fixed record-length. TRSDOS will "block" logical records into (or from) one physical record for maximum space utilization on the disk.

Blocking is putting more than one logical record into one physical record. For instance, four 64-byte logical records will fit into one 256-byte physical record. A logical record may be broken into two parts by TRSDOS in order to fill the last portion of one physical record entirely before beginning to use the next physical record (i.e. records are spanned). This occurs when the physical record length is not an even multiple of the logical record length.

If the user wishes to do his own blocking, he may specify a logical record length of 0 bytes at the time of INIT/OPEN and must himself manage the contents of the physical record buffer area of 256 bytes. TRSDOS *will not move* a logical record for the user if LRL = 0; in this particular case it will only read/write the physical record to/from the buffer. Once control is shifted to your program, you will have about 20 bytes of stack size left.

Fundamental TRSDOS I/O Calls

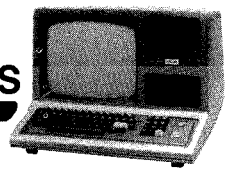
There are 17 fundamental TRSDOS routines involved in handling file I/O. These are:

$BACKSPACE	$POSN
$CLOSE	$PUTEXT
$DIVIDE	$RAMDIR
$DMULT	$READ
$FILPTR	$REWIND
$INIT	$SYNTAX
$KILL	$VERF
$OPEN	$WRITE
$POSEOF	

The detailed calling sequences and discussions for each of these routines follow. Note that *all* of these system calls use register F and do not restore its value before return. In order to apply this data properly, you should read through all of these descriptions and clear up all of the points that are not obvious to you by using other reference materials. If you are successful in doing this you will find that TRSDOS is a workable tool for your programming ideas.

$INIT — 17440/X'4420'

$INIT is provided as an entry point to TRSDOS which will create a new file entry in the directory and open the DCB for this file. $INIT scans the directory for the filespec name given in the DCB. If the filespec name is found, $INIT simply opens



the file for use. If the name is not found, a new file is created with the filespec name.

Entry Conditions

(HL) = BUFFER (see beginning of this section for notation)
(DE) = DCB
B = LRL
CALL $INIT

Exit Conditions

IY = changed
Z = OK
C carry flag is ON if a new file was created
A = TRSDOS error code. (Error codes listed at end of this chapter)

$OPEN—17444/X'4424'

$OPEN provides a way to open the DCB of a file which already exists in the directory. The DCB *must* contain the filespec of the file to be opened *before* entry to $OPEN.

Entry Conditions

(HL) = BUFFER
(DE) = DCB
B = LRL
CALL $OPEN

Exit Conditions

Z = OK
Z = 0 if file does not exist.
A = TRSDOS error code.
IY = changed

$POSN—17474/X'4442'

$POSN positions a file to read or write a randomly selected logical record. Since it deals with logical records, the proper computation is done to locate which physical record(s) contain the data. Following a $POSN with a $READ or $WRITE will transfer the record to/from RAM.

Note that positioning to logical record zero sets the file to read the first logical record in the file. To position to end of file in order to add new records onto the end, use the record number ERN + 1.

Entry Conditions

(DE) = DCB (must have been opened previously)
BC = Logical record number to position for.
CALL $POSN

Exit Conditions

Z = OK
A = TRSDOS error code.

$READ—17462/X'4436'

If LRL0, $READ transfers the logical record whose number is in the DCB as NRN into the RAM area addressed as UREC for the length LRL as defined at open time. The record comes from "BUFFER" defined at open time. If TRSDOS must read a new physical record to satisfy the request, it will do so. "Spanned" logical records will be re-assembled as necessary. $READ automatically increments NRN by 1 in the DCB after the transfer is completed. $INIT/OPEN set NRN = X'0000' in order to read the first record with the first READ.

If LRL = 0, $READ transfers one physical record into BUFFER, defined at open time, from the disk file. Registers HL are ignored. $READ increments NRN as above.

Entry Conditions

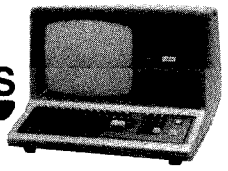
(HL) = UREC if LRL is not zero. Unused if LRL = 0.
(DE) = DCB
CALL $READ

Exit Conditions

Z = OK
A = TRSDOS error code. (EOF = X'1C' or X'1D')
(see errors 28,29 for EOF or NRF)

$WRITE—17465/X'4439'

If LRL0, $WRITE transfers the one logical record from the RAM area addressed as UREC for the length LRL as defined at open time. The record goes into the BUFFER which was defined at open time. If TRSDOS must write a physical record



in order to satisfy the request, it will do so. "Spanning" will be handled by TRSDOS as necessary. At $INIT $OPEN time the DCB value of NRN is set to X'0000' so that the first record will be written. After each logical record is transferred, the NRN value in the DCB will be incremented by 1.

If LRL = 0, $WRITE transfers one physical record from BUFFER into the disk file using the NRN in the DCB. BUFFER is defined at $INIT/OPEN time only. The DCB value NRN is updated as above, after the WRITE.

Entry Conditions

(HL) = UREC if LRL is not zero. Unused if LRL = 0
DE = DCB
CALL $WRITE

Exit Conditions

Z = OK
A = TRSDOS error code.

$VERF — 17468/X'443C'

The only difference between $VERF and $WRITE is that $VERF writes one physical record to disk and then reads it back into a special TRSDOS RAM area not defined by the user. This special area and the original write buffer are then compared byte by byte to assure that the record was successfully written.

Entry Conditions

(HL) = Same as $WRITE above.
(DE) = DCB
CALL $VERF

Exit Conditions

Z = OK
A = TRSDOS error code.

$PUTEXT — 17483/X'444B'

This routine will add an extension to a filename if an extension does not already exist. An extension to a filename may be useful for identifying the type of data in the file.

Entry Conditions

(DE) = DCB
(HL) = The extension to be added to the file
CALL $PUTEXT

Exit Conditions

None

$BKSPC—17477/X'4445'

This routine positions the file record pointer to the previous record.

Entry Conditions

(DE) = DCB
CALL $BKSPC

Exit Conditions

Z = Valid position
NZ = Invalid position in file

$REWIND—17471/X'443F'

Point to the beginning of the file. This routine positions the file pointer to the first record in the file. This is useful when the same file must be processed more than once.

Entry Conditions

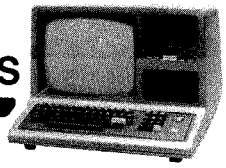
(DE) = DCB

Exit Conditions

Z = Good file specification
NZ = Bad file specification
CALL $REWIND

$POSEOF—17480/X'4448'

Point to the end of file. This routine positions the file pointer to the last record in the file. This may be used to extend a sequential access file.



Entry Conditions

(DE) = DCB
CALL $POSEOF

Exit Conditions

Z = Good file specification
NZ = Bad file specification

$SYNTAX—17436/X'441C'

Move a file specification to DCB. This routine takes a file specification and checks it for validity and moves it to a DCB so that the file may be opened.

Entry Conditions

(HL) = Filename
(DE) = DCB
CALL $SYNTAX

Exit Conditions

Z = Good file specification
NZ = Bad file specification

$DIVIDE—17489/X'4451'

The divide routine takes a 16-bit dividend and an eight-bit divisor. After division, the quotient replaces the 16-bit dividend and the remainder replaces the eight-bit divisor.

Entry Conditions

HL = Dividend
A = Divisor
CALL $DIVIDE

Exit Conditions

HL = Quotient
A = Remainder (0 indicates no remainder).

$DMULT—17486/X'444E'

The multiply routine uses a 16-bit multiplicand and an eight-bit multiplier. After multiplication takes place, the product replaces the 16-bit multiplicand.

Entry Conditions

HL = Multiplicand
A = Multiplier
CALL $DMULT

Exit Conditions

H = High order byte
L = Middle order byte
A = Low order byte

H	L	A
High	Middle	Low

$RAMDIR—17040/X'4290'

This routine allows you to examine a diskette directory (one entry or the entire directory) or the diskette's free space. The information is written into a user specified RAM buffer.

Only non-system files will be included in the RAM directory.

Entry Conditions

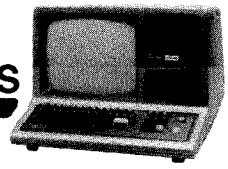
HL = RAM Buffer. If C=0, size = 1761 [max #*22+1]. If C=1 to 96, size =22. If C=255, size = 64.
B = Specified drive number
C = Function switch:

Contents of C	Results
0	Gets entire directory into RAM. (See RAM Directory Format).
1-96	Gets one specified directory record into RAM, if it exists. (See RAM Directory Format).
255	Gets free-space information (See RAM Directory Format).

CALL $RAMDIR

Exit Conditions

NZ = Error occurred.
Z = No error. (HL) = directory or free-space information.



RAM Directory Format

The directory is made up of records, one per file. All values are hexadecimal. Each record placed in user RAM is in the following format:

Byte Number	Contents
0-14	*filename/ext:d* (left-justified followed by spaces)
15	Protection Level, binary 0-6
16	Byte EOF, binary 0-255
17	Logical record length, binary 0-255
18-19	Last sector number in file, binary LSB, MSB
20-21	Number of Granules allocated (LSB,MSB) binary
22	"+" (marks the end of directory list after entire directory.)

Free Space Message Format

****nnnnn* Free Granules***

Where *nnnnn* is a decimal number. The entire message is ASCII-coded.

$FILPTR—17037/X'428D'

This routine provides information on any user file that is currently open. It enables you to obtain the drive number and the logical file number for any file and should be used in conjunction with $RAMDIR.

Entry Condition

(DE) = Data Control Block (DCB) defined when file was opened.
CALL $FILPTR

Exit Conditions

NZ = Error occurred.
Z = No error. The following registers are set up:
B = Which drive contains the file (0,1,2, or 3).
C = Logical file number (1-96)

Note: This operates with user files only.

$CLOSE—17448/X'4428'

$CLOSE closes a file from the last processing done. *It is very important to do a $CLOSE on every file opened before the program ends.* If you do not close a file, the directory entry for this file is incorrect if any new records have been

written into the file. Other cases are not given here, but it is very important to TRSDOS that all of the "housekeeping" be complete for file management.

Entry Conditions

(DE) = DCB
CALL $CLOSE

Exit Conditions

Z = OK
A = TRSDOS error code.

$KILL—17452/X'442C'

$KILL deletes the directory entry for a file and releases the disk storage. The file may be open or closed; $KILL will operate in either case.

Entry Conditions

(DE) = DCB
CALL $KILL

Exit Conditions

Z = OK
A = TRSDOS error code.

Additional Routines and Storage Addresses

$JP2DOS—16429/X'402D'

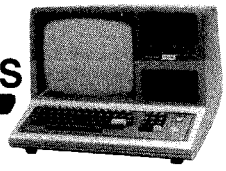
This routine transfers control to TRSDOS READY.

Entry Conditions

JP $JP2DOS

Exit Conditions

None



$DATE—12339/X'3033'
$TIME—12342/X'3036'

These routines return the date and time in ASCII format:

Date: MM/DD/YY
Time: HH/MM/SS

Entry Conditions

(HL) = Eight-byte buffer to receive the date/time text
CALL $DATE
CALL $TIME

Exit Conditions

(HL) = Date or time text

$DATLOC—16922/X'421A'
$TIMLOC—16919/X'4217'

These locations store the date and time in binary format:

$DATLOC (Three bytes): YY DD YY
$TIMLOC (Three bytes): SS MM HH

$ERRDSP—17417/X'4409'

This routine displays a TRSDOS error message determined by the contents of the accumulator (A). This register contains an error code (0 = no error) after completion of any system routine.

Entry Conditions

A = TRSDOS error code (see Table at the end of this section). In a TRSDOS error code, bits 6 and 7 are normally reset (off). So $ERRDSP interprets them as controls.

Bit #	Set	Not Set (Normal Condition)
7	Return to caller upon completion	Return to TRSDOS upon completion
6	Give detailed error message	Give error number only

CALL $ERRDSP

Exit Conditions

None

Sample Use

```
     CALL    $SYSRTN         ; ANY SYSTEM ROUTINE
     JR      Z,OKGO          ; CHECK FOR ERROR
; THE FOLLOWING INSTRUCTION SETS BIT 6 (DETAILED
; ERROR MESSAGE) AND BIT 7 (RETURN TO CALLER AFTER
; DISPLAYING MESSAGE
     OR      C0H             ; BINARY 11000000
     CALL    $ERRDSP         ; NOW CALL ERROR DISPLAY
; CONTINUE HERE UPON RETURN FROM ERROR DISPLAY
; YOUR ERROR HANDLER MAY GO HERE
;            :
;            :
;            :
OKGO  ;CONTINUATION AFTER $SYSRTN WITH NO ERROR
```

$DSPDIR — 17433/X'4419'

This command displays the directory listing of all non-protected user files in a specified drive.

Entry Conditions

(X'4271') = ASCII-coded drive number "0," "1," "2," or "3"
CALL $DSPDIR

Exit Conditions

All registers are changed.

$COMDOS — 17049/X'4299'

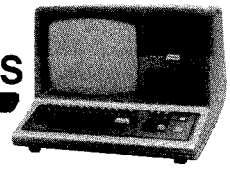
This routine executes a TRSDOS command and returns to TRSDOS READY.

Entry Conditions

(HL) = Text of TRSDOS command, terminated by X'0D.'
JP $COMDOS

Exit Conditions

None



$CMDDOS—17052/X'429C'

This routine executes a TRSDOS command and returns to the caller.

Entry Conditions

(HL) = Text of TRSDOS command, terminated by X'0D.'

Exit Conditions

All registers are changed.

Caution: TRSDOS commands will overlay RAM up to X'6FFF.'

$CMDTXT—16933/X'4225'

This is the start address of a buffer containing the last command line entered under TRSDOS READY. Using this buffer, your program may recover parameters that were included in the last command line.

For example, given a program named EDITOR/CMD, we want the operator to select an input file name when the program is loaded and executed from TRSDOS READY:

```
TRSDOS READY
EDITOR MYFILE
```

The program, EDITOR, can recover the name of the file in the $CMDTXT buffer.

Note: On entry to a program, (HL) = First non-blank character following the program name.

$MEMEND—17425/X'4411'

This storage location contains the highest address available. It is normally the same as the physical end of RAM, but you may change it for special purposes.

The address is in LSB, MSB sequence.

TRSDOS Error Codes/Messages

0 No Error Found
1 CRC Error During Disk I/O
2 Disk Drive Not In System
3 Lost Data During Disk I/O
4 CRC Error During Disk I/O
5 Disk Sector Not Found
6 Disk Drive Hardware Fault
7 **Undefined Error Code**
8 Disk Drive Not Ready
9 Illegal I/O Attempt
10 Required Command Parameter Not Found
11 Illegal Command Parameter
12 Time Out On Disk Drive
13 I/O Attempt To Non-System Disk
14 Write Fault On Disk I/O
15 Write Protected Disk
16 Illegal Logical File Number
17 Directory Read Error
18 Directory Write Error
19 Invalid File Name
20 GAT Read Error
21 GAT Write Error
22 HIT Read Error
23 HIT Write Error
24 File Not Found
25 File Access Denied Due to Password Protection
26 Directory Space Full
27 Disk Space Full
28 Attempt to Read Past EOF
29 Attempt to Read Outside of File Limits
30 No More Extents Available
31 Program Not Found
32 Invalid Drive Number
33 **Undefined Error Code**
34 Attempt to Use Non-program File as a Program
35 Memory Fault During Program Load
36 **Undefined Error Code**
37 File Access Denied Due to Password Protection
38 I/O Attempt to Unopen File
39 Invalid Command Parameter
40 File Already In Directory
41 Attempt to Open File Already Open