TRSDOS™-II
Reference Manual

CUSTOM MANUFACTURED IN U.S.A. BY RADIO SHACK, A DIVISION OF TANDY CORPORATION

TRS-80®

TRS-8Ø®

TRSDOS™-II REFERENCE MANUAL

Radio Shack®

About This Manual

This manual shows how you can use TRSDOS-II to:

- Store, retrieve, or manipulate information on disk
- Use TRSDOS-II system routines within your own machine-language programs

Terms

Below is a list of terms that we use frequently in this manual. The underlined words represent variable information which you must supply.

command	represents the TRSDOS-II command you want to execute.
comment	is an optional field used to document the purpose of the command line.
{options}	is a list of one or more parameters that may be needed by the command. Some commands have no options. If you don't use a comment at the end of the command line, you can usually omit the braces { } around options.
parameter	is a variable item of information which customizes a command.
filespec	is a standard TRSDOS-II file specification having the general form: filename/ext.password:drive(disk name)
TO	is a delimiter (separator) which you usually can replace with a space. For example, BACKUP :Ø TO :1 is the same as BACKUP :Ø :1.
hard disk	refers only to a hard disk, Drives 4-7.
diskette	refers only to a floppy diskette, Drives Ø-3.

disk	refers to a disk that may be either a hard disk or a floppy diskette. Drives Ø-7.
primary drive	refers to the disk drive that contains the operating system information, Drive Ø or 4.
RAM	(random access memory) is memory storage that can be changed (written to) as well as read.
RAM buffer	is an area in RAM for temporary data storage.

Notations

For clarity and brevity, we use some special notations and type styles in this manual.

CAPITALS and punctuation
indicate material that you must enter exactly as it appears or material that you see on your computer's video display.

<KEYBOARD CHARACTER>
indicates the key you press.

lower-case underlined
represent words, letters, characters or values you supply.

X'nnnn'
specifies nnnn as a hexadecimal (base 16) number. All other numbers in the text of this manual are in decimal (base 1Ø) form, unless otherwise noted.

About TRSDOS-II

TRSDOS-II (pronounced Triss-Doss Two) is a powerful and easy-to-use disk operating system which provides you with a full set of library commands and utilities. In addition, many of the most useful system routines can be called directly by your machine-language programs.

Contents

Contents (continued)

CHANGES AND CORRECTIONS

TO THE

TRSDOS-II REFERENCE MANUAL

Make the following corrections to your TRSDOS-II Reference Manual:

page

2/7 ATTRIB -- the WRITE option no longer exists. If you specify WRITE, it defaults to RENAME.

2/6Ø FORMS -- two new options:

O specifies that a DMP or DWP series printer is attached, such as a DWP-41Ø or a DMP-4ØØ.

F specifies that an older printer is attached, such as a Line Printer V or a Daisy Wheel II.

To use DMP type printer with TRSDOS-II type:

FORMS O

at TRSDOS-II Ready. You may wish to put this command in an automatic command file (see AUTO) so that it is executed every time you start up TRSDOS-II (TRSDOS-II starts up with FORMS F).

2/57 FORMAT -- SIZ can be any number from 1 - 122Ø (**not** 222Ø).

2/93 SAVE -- the DC/DM options should be used with wildcards only.

2/1Ø3 SPOOL -- using background printing, you can print **from** a spool file while you use the system.

3/1Ø6 SOUND SVC -- the sound duration must be stored in the L register. The H register is ignored.

87592Ø9

Radio Shack®
A DIVISION OF TANDY CORPORATION
FORT WORTH, TEXAS 76102

INTRODUCTION

Loading TRSDOS-II

When you install and power up your computer, you see the TRSDOS-II startup logo. This means you are in the TRSDOS-II operating system. You are then prompted to enter the date. Enter the date in the form mm/dd/yyyy. For example, type:

```
Ø8/25/1982
```

for August 25, 1982. You are then prompted to enter the time. You can skip this question by simply pressing <ENTER>. The time starts at ØØ.ØØ.ØØ.

If you want to set the time, type the time in the 24-hour format -- hh.mm.ss. The seconds are optional. For example, type:

```
14.3Ø <ENTER>
```

for 2:3Ø p.m.

After you enter the date and time, the following prompt is displayed.

```
TRSDOS-II Ready
.......................................
```

This means that you are at the TRSDOS-II command level. At this level you can execute a program, utility, or library command.

> Note: To perform any other operation, your system must be under the control of an application program.

If an error occurs while the system is under the control of TRSDOS-II, you receive one of the error messages listed in Appendix B.

If you get an error message not listed, it came from an application program. See the application program manual for an explanation of the error message.

Entering a Command

You may enter a command whenever the TRSDOS-II Ready prompt is displayed. The command can have up to 8Ø characters.

You must capitalize all letters in a command. Therefore, you may find it convenient to operate the keyboard in the caps mode. The keyboard is in the caps mode when the <CAPS> light is on. Press <CAPS> to turn the light on or off.

When in the caps mode, all alphabet keys are interpreted as capital letters, regardless of whether you press <SHIFT>. (Numeral and symbol keys remain the same.)

End each command by pressing <ENTER>.

For example, type:

```
CLS <ENTER>
```

and TRSDOS-II clears the display.

> Note: TRSDOS-II allows key-aheads of up to 8Ø characters. This means you can type in commands while previous ones are being executed. (The key-ahead is not displayed until TRSDOS-II or the application program is ready to interpret it.)

Executing a Program

You can also execute a program (such as SCRIPSIT™ or PROFILE™) at the TRSDOS-II Ready prompt. If what you enter is not a recognized command, TRSDOS-II checks to see if it is the name of a program. It checks for the program file on all the drives, beginning with the primary drive (unless you specify a drive number).

If TRSDOS-II finds a matching program file, it loads and executes the file. Otherwise, you get an error message.

Disk Files

You can keep a record of information you type into your computer by storing it on a disk in a "disk file." A disk file can contain a program, a collection of data, a project report you intend to make, or almost anything you want.

When your computer stores the file, it indexes the file's name and disk location in a special place on the disk called the disk's directory. Whenever you want to access the file, the computer can immediately find its location by using this directory.

Filespec

When you create a disk file, you need to give it a name. The name is just one part of a file specification -- filespec for short.

The filespec is the standard TRSDOS-II format. It contains the following information:

filename/ext.password:drive(disk name)

filename

The name of your file can be any you like; however, do not use the name of a library command or utility. The filename may be typed in lower case. It may be no longer than eight alphanumeric characters; the first character must be a letter. For example, if you want to save a file containing an inventory list, you could name it simply:

INVNTRY

/ext -- extension

If you want to further identify your file, you can give it a second name by adding an extension. An extension is preceded by a slash and has one to three alphanumeric characters. The extension, too, may be typed in lower case.

You can use an extension to provide additional information on a file. For example, using extensions such as /NEW, /IRS, and /PAY, you could distinguish files that have the same name or divide files into categories.

You can also use an extension to indicate the type of file you have. For example, you may wish to use some of the following:

/BAS	for	BASIC programs
/TXT	for	ASCII text
/DAT	for	Data files
/OBJ	for	Object code
/REL	for	Relocatable code
/SRC	for	Source code

If the extension /DAT is added to the inventory name, the filespec becomes:

```
INVNTRY/DAT
```

.password

Some files let you protect them. You can give this protection via a password either when creating the file or when using the ATTRIB command.

A password is a sequence of up to eight alphanumeric characters, the first of which must be a letter. A period (.) precedes it as a delimiter. The password may be typed in lower case.

There are two levels of passwords and the protection they provide -- access password and update password. These passwords can inhibit entry to a file. In addition, the access password can provide various levels of protection.

When you create a file and assign a password, the access and update passwords are the same. (If you do not assign a password, eight blanks are used.) Later, if you choose, you can change these passwords by using the ATTRIB command. This provides the additional protection to your files. (See the ATTRIB command for details.)

With the password SESAME, the new filespec of the inventory file is:

```
INVNTRY/DAT.SESAME
```

:drive

Often when you use your computer, you'll use more than one disk at a time. To speed the file access, you can specify the number of the drive that contains the file. The drive number is a number from Ø-7. It is preceded by a colon.

If you omit the drive number from your filespec, your computer automatically starts looking for the file on all available drives, beginning with the primary drive.

To indicate that your inventory program is located in Drive 2, use the filespec:

```
INVNTRY/DAT.SESAME:2
```

Refer to your hardware's owner's manual for information on your drive numbers.

(disk name)

You may want to indicate the name of the disk on which the file is stored. The disk name was assigned when you formatted or backed up the disk.

The disk name is a field of up to eight alphanumeric characters, the first of which is a letter. Parentheses ( ) surround the name. If you specify the disk name, you must also specify the drive number.

By adding the disk name WREHSE to the inventory program, you form the filespec:

```
INVNTRY/DAT.SESAME:2(WREHSE)
```

Of course, every filespec you enter won't include all of these optional specifications. However, you can use any combination of the fields as long as you follow the guidelines described above.

Here are more examples of valid TRSDOS-II filespecs:

```
DOPROG.OPEN
CLR/BAS:1
COMPTR:Ø(OPRSYS)
DEPT69/TXT.BOSS:4(PAYROLL)
GAMES:1
THESIS/OLD:2
TEST/CMD
```

Wildcard

Certain commands and supervisor calls (SVCs) let you specify a collection of files by using a "wildcard" mask. An asterisk (*) in a file specification represents a wildcard field and means "any sequence of zero or more characters." For example:

```
*/BAS:1
```

represents all the files stored on the diskette in Drive 1 that have the extension /BAS.

```
D*
```

represents all the files stored on the disk in the primary drive that begin with D and do not have extensions. For example, if you want a directory of all the files that begin with the letter D and have an extension, type:

```
DIR D*/* <ENTER>
```

TRSDOS-II returns a listing of all the files beginning with D and having extensions:

```
Disk Name:TRSDOS          Drive:4          Ø9/3Ø/82    ØØ.27.18
 File Name     Created   Updated   Atrb  Fil   Rec   # of  ----Sectors----
               MM/DD/YY  MM/DD/YY        Typ   Len   Records  Alloc  Used
DECRPT/BAS     Ø4/28/82  Ø4/28/82  D*XØ   F    256       55      55     55
DATASALE/BAS   Ø4/28/82  Ø4/28/82  D*XØ   F     64      1ØØ      26     25
DANTE/BAS      Ø4/28/82  Ø4/28/82  D*XØ   F    256       39      39     39
DEF/DTA        1Ø/15/8Ø  Ø4/28/82  D*XØ   F    255       12      12     12
DISBRK/CMD     1Ø/3Ø/8Ø  Ø3/Ø3/82  P*XØ   F    256        1       1      1
DIR/EFC        Ø8/14/8Ø  Ø1/29/82  P*XØ   F    256        4       4      4
6 Files Displayed
```

Super Wildcard

Besides the wildcard (*), TRSDOS-II has a super wildcard (!). You can use it to specify all files, with and without extensions.

For example, you may want to FCOPY from a diskette to hard disk all files, regardless of whether they have extensions. Using the wildcard, you must give two FCOPY commands:

```
FCOPY */*:1 TO 4 <ENTER>
FCOPY *:1 TO 4 <ENTER>
```

The first command copies files with extensions. The second then copies files without extensions.

If, however, you use the super wildcard and type:

```
FCOPY !:1 TO 4 <ENTER>
```

you can do the entire FCOPY in one step. TRSDOS-II copies **all** Drive 1 files to the hard disk.

You can use the wildcard and super wildcard with these commands:

DIR	FCOPY
FILES	KILL
MOVE	

COMMANDS & UTILITIES

How to Use This Section

This section contains an alphabetical listing of all TRSDOS-II commands and utilities.

Commands

Commands are system operations that do not use user memory. You can use commands from within programs.

To see a list of all commands, use the LIB command. Type:

```
LIB <ENTER>
```

and the following list is displayed:

```
TRSDOS-II Ready
LIB

AGAIN  ATTRIB  AUTO   CLEAR  CLICK  CLS    DATE   DEBUG  DIR    DO
DUAL   ECHO    ERROR  FC     FLOPPY FORMS  FREE   HOST   I      KILL
LIB    LOAD    PAUSE  PROT   PURGE  RENAME RESET  SCREEN SETCOM SPOOL
STATUS TIME    T      VERIFY
```

Utilities

Utilities use some or all of user memory. They return to TRSDOS-II Ready; you cannot use them effectively within programs.

The utilities are:

```
APPEND   BACKUP   BUILD   COPY    CREATE   DRIVE
DUMP     FCOPY    FILES   FORMAT  HELP     LIST
MEMTEST  MOVE     PATCH   PRINT   RECEIVE  RESTORE
SAVE     TERMINAL
```

Entry Organization

Each entry in this section is identified as either a command or a utility.

The "syntax" is the first line you see after the keyword. Use it as your guide to type in a command. (See "Syntax" below for details.)

A definition of the command or utility follows the syntax. This definition tells you exactly what the command or utility does.

Next, the entry includes additional information on the parameters of the command. A command may require you to supply some values. It also may offer several "options" that customize the command to your needs. Values and options are discussed in the additional parameter information.

Further explanation of the command follows the parameter information. This explanation includes special instructions on the command and switches and tells how best to use the command for your purposes.

Finally, each entry gives examples of the command's use.

Syntax

The command's syntax tells you what format to use when you type the command.

For example, the syntax for the CLS (Clear Screen) command is simply:

```
CLS
```

CLS <ENTER> is all you type to execute this command.

The syntax for the KILL command includes an additional parameter (a value you supply):

```
KILL filespec
```

The value is indicated by lower-case underlined. In this case, it is a TRSDOS-II filespec. For example, if you want to kill the disk file named SAMPLE in Drive 1, you type:

```
KILL SAMPLE:1 <ENTER>
```

The COPY command has even more parameters:

COPY source filespec to destination {option}

Here, you must supply the name of the source filespec you wish to copy and the destination to which you want it copied. For example:

COPY NEW/DAT:1 TO NEWDAT/1:2 <ENTER>

copies the Drive 1 file NEW/DAT onto the diskette in Drive 2; names the new file NEWDAT/1.

Sometimes the additional information is required, as are the filespecs in the above examples. Sometimes a command offers several options. These are indicated in braces and may be either optional or required. The text tells you which are required. The COPY example above has one option:

{ABS}

When typing the COPY command, you must decide if you need this option. (ABS tells TRSDOS-II to overwrite any existing file if it has the same name as the destination.) If you need it, type:

COPY NEW/DAT:1 TO NEWDAT/1:2 ABS <ENTER>

Unless you include a comment or omit a second, optional filespec, you usually can omit the braces.

Although the variable comment is not included in every syntax statement, you may add one at any time. Comments are for your information only. For example:

COPY NEW/DAT:1 TO NEWDAT/1:2 {ABS} Latest version

documents the purpose of the COPY command.

You may want to use the comment if you are calling the command from a DO file (see the DO command) or a program.

Every command uses some variation of the syntax forms discussed above. The command entry will help you decide which values and options to use.

AGAIN **Library Command**

AGAIN

Re-executes the most recently entered command.

You cannot use AGAIN after certain library commands, utilities, and user programs. For example, you cannot use it after FCOPY {DIR}.

Example

If you just executed the command:

TIME <ENTER>

type:

AGAIN <ENTER>

to re-execute it.

APPEND **Utility Program**

APPEND source filespec TO destination

Copies the contents of the source filespec onto the end of the contents of the destination. (The contents of the source file remain the same.)

The destination can be a filespec or drive number.

If the destination is a drive number, TRSDOS-II appends only if the drive contains a disk file with the same name as the source filespec.

The types of the two files must match. Both must contain variable-length records (VLRs) or both must contain fixed-length records (FLRs). (See "Record Length" in the "Technical Information" section.)

You cannot use the APPEND command with ISAM files (indexed access files used by some compilers, such as the COBOL compiler) or TRSDOS-II DO files.

Examples

```
APPEND WORDFILE/2 TO WORDFILE/1 <ENTER>
```

copies the contents of WORDFILE/2 onto the end of WORDFILE/1.

```
APPEND REGION1/DAT TO TOTAL/DAT.GUESS <ENTER>
```

appends REGION1/DATA to TOTAL/DAT, which is protected by the password GUESS.

ATTRIB **Library Command**

ATTRIB filespec {options}

Assigns or changes the password and protection level of a filespec.

When you create a file, you may specify one password. TRSDOS-II assigns this as both the **access password** and the **update password.**

The access password limits your file access to operations such as read and execute.

The update password gives you total file access -- the authority to do all file operations, including change, rename, and kill.

If the access and update passwords are the same, you have total access. (See "Filespec" in the "Introduction" for further explanation of passwords.)

ATTRIB divides file access. It lets you assign two passwords; it also lets you assign different levels of protection to the access password.

The options are:

ACC=password sets the access password. If you omit this option, the access password stays the same.

UPD=password sets the update password. If you omit this option, the update password stays the same.

PROT=level sets the access protection level. If you omit this option, the level stays the same. The optional protection levels for access to a file are:

NONE	No access
EXEC	Execute only
READ	Read and execute
WRITE	Read, execute, and write
RENAME	Rename, read, execute, and write
KILL	Kill, rename, read, execute, and write (gives the access password total access)

The access password, which protects the file's contents at the level set by PROT, could be for the operator. For example, if the protection level is READ, the operator can only read and execute the file.

Similarly, the update password could be for the programmer. When the update password is used, TRSDOS-II ignores the protection level and gives the programmer total access.

Examples

```
ATTRIB DATAFILE ACC=JULY14, UPD=MOUSE <ENTER>
```

sets the access password to JULY14 and the update password to MOUSE. The protection level remains at the previous level.

```
ATTRIB PAYROLL/BAS.PW PROT=EXEC <ENTER>
```

leaves the access and update words unchanged, but changes the protection level to EXEC (execute).

```
ATTRIB DATAFILE/1.PRN UPD=OPEN PROT=READ <ENTER>
```

sets the update password to OPEN and the protection level to READ.

AUTO **Library Command**

AUTO command line

Stores the command line, which executes automatically whenever you start up TRSDOS-II. (After you enter the date and time, TRSDOS-II loads and executes the command.)

The command line is optional. It can be a TRSDOS-II command or the name of an executable program file. If you omit the command line, TRSDOS-II simply deletes the auto command line currently stored.

When you enter the auto command line, TRSDOS-II does not check it for errors. Errors are detected when the command executes.

To override a set auto command line, press <HOLD> before or after you enter the date when you start up the system.

Examples

```
AUTO DIR {SYS} <ENTER>
```

executes the command DIR {SYS} at startup and then displays TRSDOS-II Ready.

```
AUTO BASIC <ENTER>
```

executes BASIC.

```
AUTO DO DRIVESET <ENTER>
```

executes the command file named DRIVESET. (See the BUILD and DO commands for details on creating and executing command files.)

```
AUTO <ENTER>
```

turns off the auto command currently stored.

BACKUP **Utility Program**

BACKUP drivel **TO** drive2 {options}

(FOR FLOPPY DISKETTE USE ONLY)
Makes an exact, mirror-image copy of the source diskette in floppy drivel to the destination diskette in floppy drive2.

If the destination diskette is unformatted, the BACKUP utility formats it before copying.

To use BACKUP, you must have at least two floppy drives (drivel and drive2 cannot be the same drive number). If you don't have at least two floppy drives, use the COPY command.

The options are:

ID=diskette-name assigns the name of the new diskette. If you omit this option, TRSDOS-II gives the new diskette the same name as the source diskette.

PW=source-password indicates the master password of the source diskette. TRSDOS-II duplicates a diskette only if you give the correct password or if PW=PASSWORD. (All disks distributed by Radio Shack use PASSWORD as the default master password.) If PW does not equal PASSWORD and you do not give the correct password, backup aborts with an error.

NEW=destination-password assigns the password of the new diskette. If you omit this option, TRSDOS-II uses the password of the source diskette.

ABS overwrites, without prompting, any data already on the destination diskette. If you omit this option, TRSDOS-II prompts you before overwriting.

By using BACKUP to duplicate your TRSDOS-II diskette, you can create another operating system diskette.

We suggest that you make copies of all your diskettes -- especially your system diskette -- and store your originals

in a safe place. This reduces the possibility of losing important information should something happen to the diskette you are using.

If you try to back up a single- to a double-sided diskette, TRSDOS-II performs the backup automatically. If you try to back up a double- to a single-sided diskette, however, an error occurs.

Examples

BACKUP Ø TO 1 <ENTER>

makes an exact, mirror-image copy of the diskette in Drive Ø to the diskette in Drive 1.

BACKUP Ø TO 3 {ID=ACCTDISK NEW=MAY24} <ENTER>

copies the diskette in Drive Ø to the diskette in Drive 3; names the new diskette ACCTDISK and assigns it the master password MAY24.

BACKUP 1 TO Ø <ENTER>

copies a data diskette in Drive 1 to a diskette in Drive Ø. After entering this command, you are prompted to remove the system diskette and place your destination diskette in Drive Ø.

BUILD **Utility Program**

BUILD filespec

Creates or lets you edit a DO file, an automatic command input file which contains one or more library commands, utilities, or programs. You can execute the DO file with the DO command.

The filespec may not include an extension.

Creating a Do File

When you enter the BUILD command with a non-existing filespec, BUILD creates the file and then prompts you to begin inserting lines:

```
Enter Command Line (1-80)
.............................
```

At this prompt, you can enter a command line of up to 80 characters.

Or, to end the file, simply press <ENTER> at the prompt.

Pressing <BREAK> also ends the file, but it omits the last line in the file, unless you end that line with a carriage return.

Example

To create a new file named COPYSCR, type:

```
BUILD COPYSCR <ENTER>
```

TRSDOS-II displays the command line prompt:

```
Enter Command Line (1-80)
.............................
```

Answer the prompt by typing:

```
COPY NEWFILE/LST:4 TO 0 <ENTER>
```

to command TRSDOS-II to copy NEWFILE/LST from Drive 4 to Drive Ø whenever you type DO COPYSCR.

TRSDOS-II then asks if you want to:

```
Store Line? (cr/esc)
```

If you wish to store the line and continue building the file, press <ENTER>. If you do not wish to store the line, press <ESC>. In this case, press <ENTER>.

Now, type and store these command lines as you did the first:

```
COPY DIRECTRY/NME:4 TO Ø
COPY TRANSFER/ACT:4 TO Ø
PAUSE "INSERT DISK Ø"
<ENTER>
```

The last line tells TRSDOS-II that you have finished entering command lines. TRSDOS-II displays the prompt:

```
******** Edit Complete ********
```

and returns to TRSDOS-II Ready.

Now, whenever you type:

```
DO COPYSCR <ENTER>
```

TRSDOS-II executes the file by copying the files from the diskette in Drive 4 to the diskette in Drive Ø. Then, it pauses, displaying the message:

```
PAUSE "INSERT DISK Ø"
Press any Key to Continue
```

Editing a Do File

When you enter the BUILD command with an existing filespec, TRSDOS-II copies the file into a new file with the same name and the extension /OLD. Then, you can edit the file.

Example

To edit the file you created, type:

```
BUILD COPYSCR <ENTER>
```

TRSDOS-II displays the first command line and the edit prompt:

```
COPY NEWFILE/LST:4 TO Ø
Keep, Delete, Fix, Replace, Insert, or Quit?
Enter (K,D,F,R,I,Q)....
```

This prompt allows the following options:

KEEP	Copies the line, as is, into the new file.
DELETE	Deletes the line by not copying it into the new file.
FIX	Lets you edit the line. (For details on FIX, see the FC command.)
REPLACE	Deletes the displayed line and lets you insert replacement lines. Press <ENTER> at the beginning of a line to stop replacing.
INSERT	Lets you insert lines ahead of the line being displayed. Press <ESC> to erase errors in the insert line.
QUIT	Ends the editing session.

Type K <ENTER> to keep the first line, as is, and display the second line for editing.

```
COPY DIRECTRY/NME:4 TO Ø
```

To delete this line, type D <ENTER>. The message **Line Deleted from File** appears. TRSDOS-II displays the next line for editing:

```
COPY TRANSFER/ACT:4 TO Ø
```

To "fix" the line, type F <ENTER>. TRSDOS-II displays the line, with the cursor positioned over the first character.

Press <right arrow> to move the cursor to where you want to make the change. Change the extension to LGR, by typing LGR over ACT. After making the change, press <ENTER> once to move the cursor to the beginning of the line.

Press <ENTER> again to store the new line in the file.

Type Q <ENTER> to quit the session. TRSDOS-II prompts:

END OF FILE -- Add to EOF (Y/N)?

Type Y <ENTER> if you want to add lines to the end of the BUILD file. If you do not want to add lines, type N <ENTER>; TRSDOS-II Ready is displayed.

CLEAR — Library Command

CLEAR

Clears user memory and returns to TRSDOS-II. It does not re-initialize the input/output (I/O) drivers or protected high memory.

Example

```
CLEAR <ENTER>
```

CLICK — Library Command

CLICK {switch}

Turns the key entry sound on or off for those computers that support sound.

The switch options are:
ON turns on CLICK
OFF turns off CLICK

Example

```
CLICK ON <ENTER>
```

causes the computer to generate sound whenever you press certain keys. <CTRL>, <CAPS>, <LOCK>, <SHIFT>, and <REPEAT> do not generate sound.

CLS — Library Command

CLS

Clears the display and positions the cursor at the upper left corner of the display.

Example

```
CLS <ENTER>
```

COPY Utility Program

COPY source filespec TO destination {ABS}

Copies the source filespec to the destination.

The destination can be a filespec or drive number.

If you specify a filespec as the destination, TRSDOS-II checks to see if the destination already exists. If the file does not exist, TRSDOS-II creates it (on the specified or first available drive).

If you specify only the drive number as the destination, TRSDOS-II copies the source filespec to the disk in that drive. It gives the destination file the same name as the source filespec.

ABS is optional. If you include ABS, TRSDOS-II overwrites without prompting any non-protected, existing file that has the same name as the destination. If you omit ABS, TRSDOS-II prompts you before overwriting.

Normally, COPY uses memory below X'3ØØØ'. However, when copying from one diskette to another in a **single** drive, it uses memory up to the beginning of unprotected memory (see "Memory Requirements" in the "Technical Information" section).

Examples

```
COPY OLDFILE/BAS:3 TO NEWFILE/BAS:2 <ENTER>
```

copies OLDFILE/BAS from the diskette in Drive 3 to the diskette in Drive 2 and names the new file NEWFILE/BAS.

```
COPY NEW/DAT TO DEFUNCT/DAT:2 ABS <ENTER>
```

searches for NEW/DAT and copies the file (from the first drive that contains it) to DEFUNCT/DAT in Drive 2. If you already have a disk file named DEFUNCT/DAT in Drive 2, TRSDOS-II overwrites it with the new file.

COPY FILE/A:4 TO FILE/B:1(DOUBLE) <ENTER>

copies FILE/A from the diskette in Drive 4 to the diskette in Drive 1 and names the new file FILE/B.

COPY TESTPROG TO 3 <ENTER>

searches for TESTPROG and copies the file from the first drive that contains it to the diskette in Drive 3. It names the new file TESTPROG.

CREATE Utility Program

CREATE filespec {options}

Creates a file named filespec and pre-allocates space for its contents. If you use CREATE, TRSDOS-II does not de-allocate (recover) unused space when you close the file.

If you do not use CREATE, TRSDOS-II allocates space for your file dynamically as you write to it. Then, when you close the file, TRSDOS-II de-allocates the unused space.

The options are:

NGRANS=number allocates number * 5 sectors to the file. For example, if you want to allocate 1ØØ sectors to the file, set NGRANS to 2Ø.

NRECS=number assigns the specified number of fixed-length records to the file. If you use this option, you must also use the LRL= option.

LRL=number (used only with the NRECS= option) assigns number as the logical record length. The number can be from 1-256. If you omit this option, the record length defaults to 256.

TYPE=V specifies the record type as variable-length.

TYPE=F specifies the record type as fixed-length. If you omit both TYPE= options, TRSDOS-II uses TYPE=F. (The extended mode does not apply to CREATE.)

NGRANS and NRECS are mutually exclusive. If you use NGRANS, do not use NRECS. If you use NRECS, do not use NGRANS.

(For more information about record lengths and types, see "Disk Files" in the "Technical Information" section.)

Examples

```
CREATE DATAFILE/BAS  NRECS=3ØØ,LRL=256 <ENTER>
```

creates a file named DATAFILE/BAS and allocates space for 3ØØ 256-byte fixed-length records (FLRs).

```
CREATE TEXT/1  NGRANS=1ØØ,TYPE=V <ENTER>
```

creates a variable-length record file (VLR file) named TEXT/1 and allocates 5ØØ sectors to it.

```
CREATE NAMES/TXT.IRIS  NRECS=5ØØ,LRL=3Ø <ENTER>
```

creates an FLR file named NAMES/TXT, which is protected by the password IRIS. The file can hold 5ØØ 3Ø-byte records.

DATE **Library Command**

DATE {mm/dd/yyyy}

Resets the date or displays the date and time string.

You first set the date when you start up your computer. DATE lets you change it without resetting your computer.

mm/dd/yyyy is optional. Use it to reset the date. mm is a 2-digit month specification; dd is a 2-digit day of the month specification; yyyy is a 4-digit year specification. If you omit this option, TRSDOS-II displays the current date and time.

Examples

DATE <ENTER>

displays the current date and time as:

Thu Jul 1 1982 182 -- 14.55.Ø5

for Thursday, July 1, 1982, the 182nd day of the year, 2:55:Ø5 p.m. Note that leading zeroes are not displayed.

DATE Ø8/21/1982 <ENTER>

resets the date to August 21, 1982, and displays the new information.

DEBUG **Library Command**

DEBUG {switch}

Sets up the debug monitor which lets you enter, test, and debug machine-language programs.

The switch options are:
ON turns on DEBUG
OFF turns off DEBUG

The switch is optional. If you omit it when DEBUG is off, TRSDOS-II returns the status DEBUG. If you omit it when DEBUG is on, TRSDOS-II enters DEBUG.

While DEBUG is on, you automatically enter the debug monitor whenever you load and execute a user program. In this mode, you can enter any of a special set of single-key commands to study how your program is working.

DEBUG loads into the high-memory area sometimes reserved by TRSDOS-II for special programming. (See "Memory Requirements" in the "Technical Information" section.) While DEBUG is on, TRSDOS-II automatically protects this area from being overlaid by BASIC or other user programs.

You can use DEBUG only on programs in the user area X'28ØØ' to TOP. (See the STATUS command to locate TOP.)

To use DEBUG from BASIC, you must turn on DEBUG before you start BASIC.

Examples

```
DEBUG ON <ENTER>
```

turns on DEBUG (loads it into high memory, protects high memory, and sets up a "scroll window" -- a block of lines that is to be scrolled). The lower 11 lines of the display make up the scroll window. The upper 13 lines contain the debug monitor display.

```
DEBUG OFF <ENTER>
```

turns off DEBUG and removes protection of high memory.

DEBUG <ENTER>

enters DEBUG if it is on; returns the status of DEBUG if it is off.

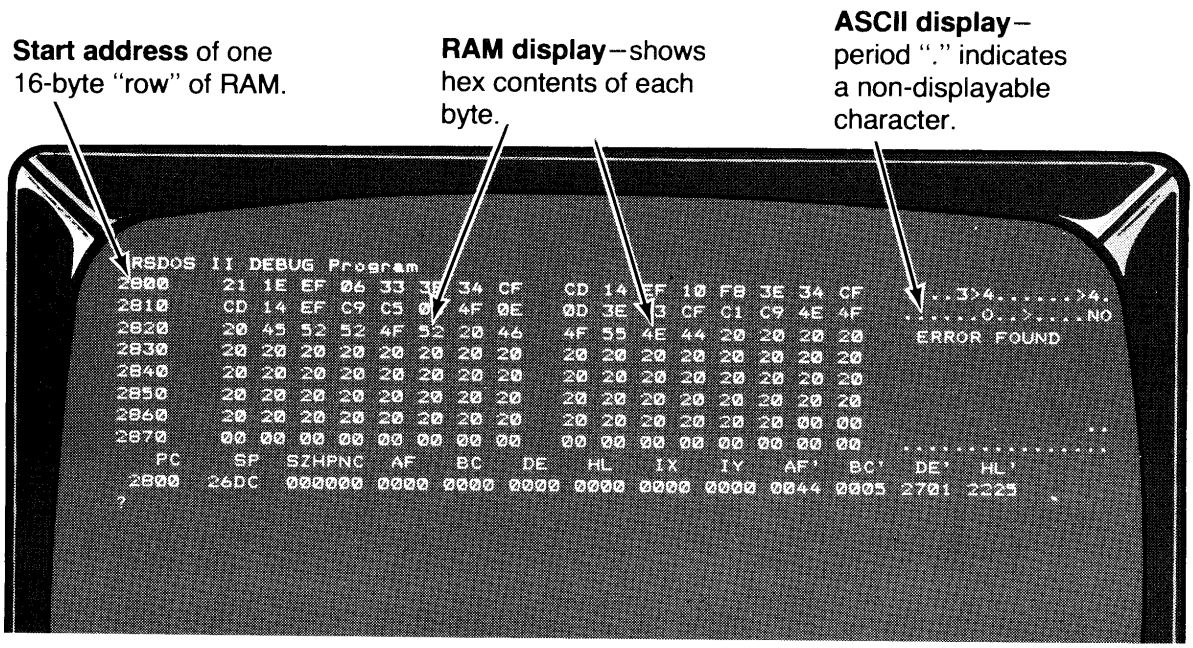
To Enter the Debug Monitor

To enter the monitor when DEBUG is off, type:

DEBUG ON <ENTER>
DEBUG <ENTER>

To enter the monitor when DEBUG is on, do one of the following:

- Type DEBUG. TRSDOS-II enters the debug monitor.
- Load a user program by typing its filespec. TRSDOS-II loads the program and enters the debug monitor. The transfer address for the program is in Register PC's display.



Z-8ØA Register Contents & RAM Display. SZHPNC are the flag bits in Register F.

The ? is the command prompt; it indicates that you can enter one of the single-key commands. Press <H> (for "help") to display a "menu," a list of DEBUG commands. To enter one of the commands, press the letter you see capitalized in the command menu. For example, to enter the memory command "raM," press <M>.

Most commands prompt you to enter additional information or subcommands. While entering commands and subcommands, you'll find the following keys useful:

<ESC>	Returns to the ? prompt and cancels the current command
<BACKSPACE>	Backspaces the cursor and erases the previous character
<left arrow>	Backspaces the cursor without erasing characters
<right arrow>	Moves the cursor forward without erasing characters
<F1>	Positions the cursor at the starting memory address when used with certain subcommands
<TAB>	Tabs the cursor when used with certain subcommands

Command Description

B (Breakpoint)

When execution reaches a breakpoint which you have set in a program, control returns to the debug monitor, and the program counter points to the breakpoint address.

Press <B> to set a breakpoint. Place the breakpoint at the beginning byte of an opcode -- **never** in the middle of an instruction.

TRSDOS-II prompts you to enter the breakpoint number. You can enter up to eight breakpoints, so type in a number from 1 to 8.

Next, TRSDOS-II prompts you to enter the new address for that breakpoint. If the breakpoint is already set, TRSDOS-II displays the old breakpoint address, and the original instruction that goes in that address.

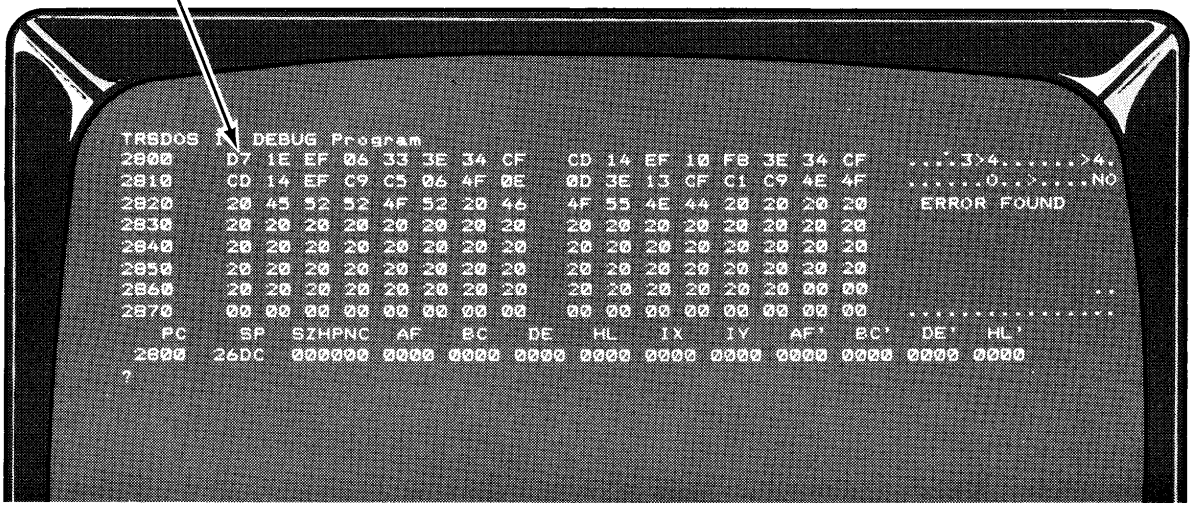
While a breakpoint is in place, the memory display for the breakpoint address shows X'D7'.

For example:

```
? B #=1 A=28ØØ
```

puts a breakpoint (#1) at address X'28ØØ'. The memory display for X'28ØØ' shows X'D7'.

X'D7' indicates a breakpoint has been set at this address.



To delete a single breakpoint without affecting any others, press <ENTER> instead of providing a new address for the breakpoint.

C (Continue)

After the debug monitor has stopped at a breakpoint, press <C> to resume execution of your program at the address pointed to by PC. The original instruction at the breakpoint address executes, but the breakpoint remains in place.

D (Decimal Format)

Press <D> to display all addresses in decimal form. (TRSDOS-II still displays the contents of all registers and memory addresses in hexadecimal.) In the decimal display format, you must enter all addresses as 5-digit decimal numbers.

E (Empty Breakpoint Table)

Press <E> to delete all breakpoints. TRSDOS-II restores all breakpointed instructions.

F (Find Hex String)

Press <F> to search in memory for a string of up to 2Ø bytes. DEBUG prompts you to type in the start (S=) and end (E=) addresses of the area to be searched. It also prompts you for the data (D=) for which to search.

Type in the search string in hexadecimal format, then press <ENTER>. The debug monitor displays the first occurrence of the string. If the string is not in the search area, the current memory display stays the same.

Example

```
? F S=28ØØ  E=4ØØØ  D=C3ØØ7Ø  <ENTER>
```

searches memory from X'28ØØ' through X'4ØØØ' for the 3-byte hexadecimal string X'C3ØØ7Ø'.

X (Hex Format)

Press <X> to return the display to hexadecimal format. In this mode, you must enter all addresses as 4-digit hexadecimal numbers.

J (Jump)

Press <J> to load the "jump" address into Register PC.

This is a 2-step process:

1. Press <J> and type in the jump address. This loads the address into Register PC.

2. Press <C> to continue execution at that point.

For example, type:

```
? J A=28ØØ
```

then press <C> to start execution at X'28ØØ'.

L (Load or Copy Memory to Memory)

Press <L> to copy a block of memory. DEBUG prompts you to type in the start (S=) and end (E=) addresses of the block to be copied and the destination address (T=) for the first byte to be copied.

The copy is incremental. DEBUG copies the first byte to the first destination address, then the second byte to the second destination address, and so on.

Examples

```
? L S=28ØØ  E=28FØ  T=3ØØØ
```

copies from addresses X'28ØØ' through X'28FØ' into memory at X'3ØØØ' through X'3ØFØ'.

You can use this command to fill memory with a specific value. Put the desired value in address nnnn and use a command such as this:

```
? L S=nnnn E=xxxx T=nnnn + 1
```

This copies the value in nnnn into every location from nnnn + 1 to xxxx. For example, if X'28ØØ' contains X'2Ø', then:

```
? L S=28ØØ E=3ØØØ T=28Ø1
```

fills memory from X'28Ø1' to X'3ØØØ' with X'2Ø'.

O (Debug Off)

Press <O> to exit the debug monitor and turn off DEBUG. All breakpoints are removed from your program, and DEBUG returns to TRSDOS-II. If you want only to exit DEBUG, press <S>; DEBUG remains "on."

P (Print Display)

Press <P> to send a copy of the display to the printer.

M (Examine and Change Memory)

Press <M> to enter this command. DEBUG prompts you to type in the starting address of memory to be examined. As soon as you type in the complete address, the memory display shows the 128-byte area that starts at that address.

While the A= prompt is present, you can scroll through memory 64 bytes at a time by pressing <ENTER>. The up and down arrows move the cursor in 128-byte increments.

To modify any memory in the display area, press <F1> while the A= prompt is displayed. The cursor moves up into the memory display area.

When the cursor is in the hexadecimal area, enter hexadecimal values. DEBUG updates the memory display as you type in each nibble (hexadecimal character, half a byte).

When the cursor is in the ASCII area, enter ASCII characters. Press </> to return to hexadecimal entry.

To cancel all changes in memory, press <ESC>. **To effect all changes,** press <F2>.

R (Modify Registers)

Press <R> to change a register pair. The R=> prompt appears. Type in a letter indicating which register pair you want to change:

A for	AF	B for	BC	D for	DE	H for	HL
X for	IX	Y for	IY				
F for	AF'	C for	BC'	E for	DE'	L for	HL'

The cursor moves to the first byte of the register pair. While in the register modify mode, use the cursor control keys <right arrow> and <left arrow> to move over one nibble at a time. Use <TAB> to advance to the next register pair.

To cancel changes in register contents, press <ESC>. **To effect changes** made, press <F2>.

S (System)

Press <S> to return to the TRSDOS-II Ready mode. DEBUG is still on; when you load and execute a program, you again enter DEBUG.

While the cursor is in the memory display area, the cursor control keys are:

<F1>	Homes the cursor
<TAB>	Tabs the cursor
<ENTER>	Moves the cursor to the start of the next row
</>	Moves the cursor in and out of the ASCII area
	Moves the cursor:
<up arrow>	up
<down arrow>	down
<left arrow>	left
<right arrow>	right

DIR **Library Command**

DIR source {options}

Displays the specified disk's directory.

The source can be a standard TRSDOS-II filespec, a wildcard, or a drive number (Ø-7). If you omit the drive number, TRSDOS-II goes to the primary drive.

The options are:

PRT sends the directory listing to the printer. If you omit this option, TRSDOS-II displays the directory on the screen.

SYS lists system files. If you omit this option, TRSDOS-II displays only user files.

Disk Name:TRSDOS Drive:4 Ø9/3Ø/82 ØØ.27.18

File Name	Created	Updated	Atrb	Fil	Rec	# of	----Sectors----	
	MM/DD/YY	MM/DD/YY		Typ	Len	Records	Alloc	Used
DESIGN2	Ø2/18/82	Ø2/18/82	P*XØ	F	256	2	2	2
SAMPPROG	Ø1/Ø1/81	12/28/81	D*XØ	V	+++	+++	4	4
SAMPPROG/BAS	Ø1/Ø1/81	12/28/81	D*XØ	V	+++	+++	7	7
OLDFILE	Ø1/Ø1/81	Ø1/Ø4/82	D*XØ	F	1ØØ	2Ø	8	8
NEWFILE	Ø1/Ø1/81	12/28/81	D*XØ	F	1Ø6	2Ø	9	9
HOLD	1Ø/29/8Ø	Ø4/28/82	D*XØ	F	256	6	6	6
ABC/DTA	11/Ø3/81	Ø4/28/82	D*XØ	F	255	+++	Ø	+++
RDTEST4	11/Ø3/81	Ø4/28/82	D*XØ	F	256	6	6	6

8 Files Displayed

What the Column Headings Mean

- Disk Name -- The name assigned to the diskette when it was formatted or backed up.
- File Name -- The name and extension assigned to a file when it was created.
- Creation Date -- The date on which the file was created.

- Update Date -- The date on which the file was last updated.

- Attributes -- A 4-character field which lists the file's attributes:

  The first character is either P for "program file" or D for "data file."

  The second character is either S for "system file" or * for "user file."

  The third character gives the password protection status.

X	The file is unprotected (has no passwords).
A	The file has an access word only.
U	The file has an update word only.
B	The file has both update and access words.

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

Ø,1	Kill, rename, read, execute, and write
2	Rename, read, execute, and write
3	Not used
4	Read, execute, and write
5	Read and execute
6	Execute
7	No access

  See the ATTRIB command for an explanation of how to change the access password, update password, and protection level.

- File Type -- Indicates the record type for the file:

F	Fixed-length records
V	Variable-length records

- Record Length -- The length assigned when the file was created (applies only to FLR files). Plus signs (+) indicate a VLR file.

- Number of Records -- The number of logical records that have been written. Plus signs (+) indicate that none has been written or that the file has variable-length records, so the number cannot be calculated.

- Sectors Allocated -- The number of sectors (256-byte blocks) allocated to the file.
- Sectors Used -- The number of sectors to which data has been written. Plus signs (+) indicate the file has no data.
- Files Displayed -- The number of files on the directory listing.

Examples

```
DIR <ENTER>
```

displays the directory of the user files (that are on the disk) in the primary drive.

```
DIR 1 {SYS,PRT} <ENTER>
```

lists, to the printer, the directory of the system files in Drive 1.

```
DIR */BAS:1 <ENTER>
```

displays the directory of the user files in Drive 1 that have the extension /BAS.

```
DIR B*/* <ENTER>
```

displays the directory of the files that are in the primary drive, start with the letter B, and have extensions.

```
DIR B* <ENTER>
```

displays the directory of the files that are in the primary drive, start with the letter B, and do not have extensions.

```
DIR ! <ENTER>
```

displays the directory of all files, with or without extensions, that are in the primary drive.

DO Library Command

DO filespec

Executes a DO file, an automatic command input file that contains one or more library commands or programs. TRSDOS-II executes the commands as if you had typed them in from the keyboard.

You can create a DO file with the BUILD command. Command lines in this file may include library commands or file specifications for user programs.

You cannot include the DEBUG command in a DO file.

Also, you cannot execute a DO file and the SPOOL command at the same time. The first one executed has priority over the second.

Running User Programs from a DO file

Besides executing TRSDOS-II library commands, you can load and execute user programs from a DO file. When you do so, enter your program name as the last line in the DO file.

For example, to perform some library commands and then run a BASIC program called MENU, enter this as the last line in your program:

```
BASIC MENU <ENTER>
```

You can "chain" DO files by putting another DO command at the end of a DO file.

In addition, you can run user programs from the middle of a DO file. Your program runs normally. Your program must return to TRSDOS-II for the execution of the DO file to continue. If, however, you press <BREAK> during program execution, your program ends and control returns to TRSDOS-II.

Examples

```
DO STARTER <ENTER>
```

begins automatic command input from STARTER.

```
AUTO DO STARTER <ENTER>
```

sets DO STARTER as the auto command file. Whenever you start up, TRSDOS-II first executes the command file STARTER.

DRIVE Utility Program

DRIVE drive {options}

Lets you:

- Gain the best use of a floppy disk drive by changing the following disk drive settings:
  a) seek rate (the rate at which the computer is able to access the diskette
  b) diskette swap detection
  c) wait (for a drive ready condition)

- Turn secondary floppy or hard disk drives off-line

If you omit all options, DRIVE returns the current settings for the specified drive.

The following information offers a thorough explanation of the DRIVE command and all its options. Please read it before using this command.

The options are:

RATE=number (used for floppy drives only) Sets the seek rate of the floppy disk drive. The number is from Ø-3 where:
Ø = 3 milliseconds
1 = 6 milliseconds
2 = 1Ø milliseconds
3 = 15 milliseconds
RATE is optional. If you omit RATE, the setting stays the same.

DETECT (used for floppy drives only) sets the diskette swap detection. This causes TRSDOS-II to check the drive hardware for a "door opened" condition. Set Push-Button and Thinline drives to DETECT.

NODETECT (used for floppy drives only) sets the diskette swap to "no detection." This causes TRSDOS-II to ignore any "door opened" conditions received from the drive hardware. Set Latch drives to NODETECT.

WAIT (used for floppy drives only) sets TRSDOS-II to wait for the drive to gain proper motor speed, if a "Drive Not Ready" error occurs. TRSDOS-II tries again. If the error occurs again, the drive is considered not ready, and an error code is generated. Set Thinline drives to WAIT.

NOWAIT (used for floppy drives only) sets TRSDOS-II to not wait if a "Drive Not Ready" error occurs. Immediately generates an error code. Set Push-Button and Latch drives to NOWAIT.

OFFLINE (used for secondary drives only) sets a drive off-line. TRSDOS-II ignores that drive.

ONLINE (used for secondary drives only) sets a drive on-line. ONLINE is the default.

Gaining the Best Use of Floppy Disk Drives

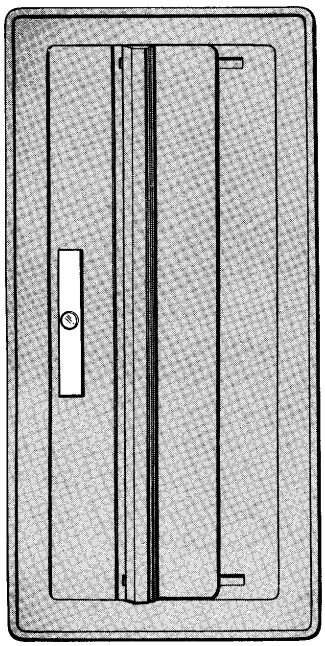
When TRSDOS-II starts up, it initializes your drives to the following settings:

Drive	Seek Rate	Swap Detect	Wait/Nowait Status
Ø	1Ø ms	DETECT	WAIT
1 - 3	15 ms	NODETECT	WAIT

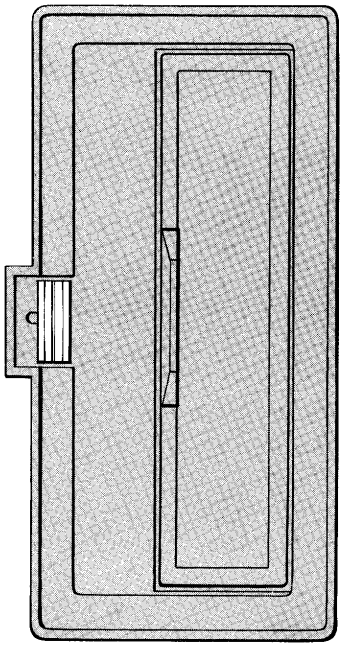
Any type of floppy drive can operate under these settings. However, to get the best use of your particular drive, try different settings.

There are three types of drives that could be on your computer. Each type has its own specifications that determine how it can be set up.

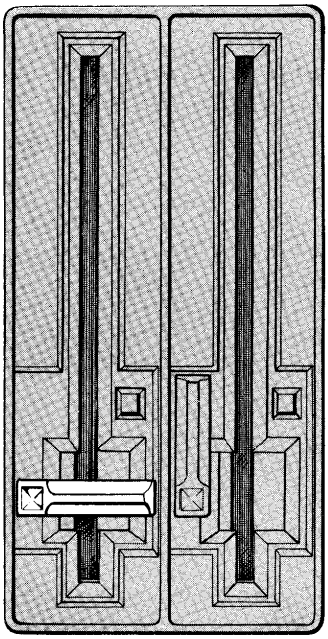
You can determine the type of drive you have by looking at the pictures on the next page.



Push-Button



Latch



Thinline

The Three Types of Drives.

We suggest you try the following settings for each of these drives.

Drive	Seek Rate	Swap Detect	Wait/Nowait Status
Push-Button	1Ø ms	DETECT	NOWAIT
Latch	15 ms*	NODETECT*	NOWAIT
Thinline	3 ms	DETECT	WAIT*

* These settings are required for these particular drives and are set this way at startup.

When using DRIVE with the seek rate, swap detect, and wait options, be sure to note the following:

- When reset, TRSDOS-II always returns to the startup settings. Once you have tested the settings on your computer, you can use the AUTO command (or a DO file) to start the DRIVE utility automatically upon power-up or reset.

- If you receive many I/O errors on disk reads/writes after changing the seek rate, you probably set the rate too fast for that particular drive. To solve this, either issue the DRIVE command again with the proper seek rate or reset the computer.

- **Latch drives** cannot detect whether a drive door has been opened since the last disk access. Always set Latch drives to the NODETECT option.

- **Thinline drives** have a built-in feature to reduce the wear on the floppy diskette. If a Thinline drive is not accessed for 2Ø seconds or more, the drive motor shuts off until the next drive access. At the next disk access, it takes about 8/1Ø of a second for the motor to reach proper speed.

- Always set **Thinline drives** to the WAIT option. If you run a Thinline drive with the NOWAIT option, a "Drive Not Ready" error occurs because the motor could not reach the proper speed before the access.

Examples

If your Drive Ø is a Thinline, this command:

```
DRIVE Ø {RATE=Ø,DETECT,WAIT} <ENTER>
```

lets you get the best use of Drive Ø.

If your Drive 1 is a Push-Button, this command:

```
Drive 1 {RATE=2,DETECT,NOWAIT} <ENTER>
```

lets you get the best use of Drive 1.

If your Drive 1 is a Latch, this command:

```
DRIVE 1 {RATE=3,NODETECT,NOWAIT} <ENTER>
```

lets you get the best use of Drive 1.

Turning Drives Off-Line

The OFFLINE option turns a secondary disk drive off-line; ONLINE turns it back on-line. You can use both options with hard or floppy secondary disk drives:

- Hard Disk Drives -- If you have more than one hard disk drive, you can move or copy files to your secondary drives and turn these drives off-line. Thus, you protect your files from access and/or change.
- Floppy Drives -- When you turn a non-existing or unused secondary floppy drive off-line, TRSDOS-II accesses your disks faster.

The default is ONLINE. When you turn a floppy disk drive back on-line after it is off-line, you must also use the I command to re-initialize the drive.

Examples

```
DRIVE 5 {OFFLINE} <ENTER>
```

protects files on Drive 5 from access.

```
DRIVE 3 {OFFLINE} <ENTER>
```

tells TRSDOS-II not to access Drive 3; this speeds up access time.

```
DRIVE 2 {ONLINE} <ENTER>
```

tells TRSDOS-II to try to access Drive 2.

If you have a hard disk, you might find it helpful to create a command file (see the BUILD command for details) of these commands:

```
DRIVE 6 {ONLINE}
MOVE !:5 TO 6 {ABS}
DRIVE 6 {OFFLINE}
```

This causes TRSDOS-II to turn Drive 6 on-line, move all the files on Drive 5 to Drive 6, and then turn Drive 6 back off-line.

Using AUTO to Set Drives

You also can build a special command file that automatically sets your floppy disk drives for maximum efficiency and turns any non-existent drive off-line as you start up.

For example, if your computer has two Thinline floppy drives and two hard disk drives, you can use the following as your automatic command file. (See the AUTO, BUILD, and DO commands for more information.)

First get into the BUILD command mode by typing:

```
BUILD DRIVESET <ENTER>
```

Now you can enter program lines such as:

```
DRIVE Ø {RATE=Ø,DETECT,WAIT} <ENTER>
DRIVE 1 {RATE=Ø,DETECT,WAIT} <ENTER>
DRIVE 2 {OFFLINE} <ENTER>
DRIVE 3 {OFFLINE} <ENTER>
DRIVE 6 {OFFLINE} <ENTER>
DRIVE 7 {OFFLINE} <ENTER>
<ENTER>
```

The last line saves the program lines.

To set this command file as the auto command, type:

```
AUTO DO DRIVESET <ENTER>
```

Now, whenever you start up your computer, the DRIVESET command file executes automatically and sets all your drives to their maximum efficiency.

When you set a command file such as DRIVESET as the auto command, TRSDOS-II overwrites any existing auto. To set a command file and keep the existing auto, use an example such as the following.

The SCRIPSIT system diskette has the auto command STARTUP set. When you enter the date and time, this auto command initializes the SCRIPSIT system diskette and goes directly to the SCRIPSIT menu.

If you want to set an auto file that uses this auto startup and sets the drives to their maximum efficiency, you can use the following command file.

If you have a computer with four Thinline floppy disk drives and one hard disk drive, get into the BUILD command mode and enter lines such as:

```
DRIVE Ø {RATE=Ø,DETECT,WAIT} <ENTER>
DRIVE 1 {RATE=Ø,DETECT,WAIT} <ENTER>
DRIVE 2 {RATE=Ø,DETECT,WAIT} <ENTER>
DRIVE 3 {RATE=Ø,DETECT,WAIT} <ENTER>
DRIVE 5 {OFFLINE} <ENTER>
DRIVE 6 {OFFLINE} <ENTER>
DRIVE 7 {OFFLINE} <ENTER>
STARTUP <ENTER>
<ENTER>
```

Now when you enter the command:

```
AUTO DO DRIVESET <ENTER>
```

TRSDOS-II automatically sets the drives to their maximum efficiency, turns non-existing drives off-line, and then starts up SCRIPSIT.

DUAL **Library Command**

DUAL {switch}

Sends the video display output to the printer and vice versa.

You must use one of the switch options. They are:
ON Turns dual routing on
OFF Turns dual routing off

Because there are intrinsic differences between the output devices and the output software, the printer and display outputs may differ.

When dual routing is on, the video output process slows.

Be sure your printer is on-line and powered up while dual is on. If it is not, the video display appears to output characters at the rate of 1 every 3 seconds (3Ø seconds, if you have run the LPII patch file). To solve this, turn DUAL off or ready your printer.

Examples

DUAL ON <ENTER>

turns dual routing on, so all output is sent to the line printer and video display at the same time.

DUAL OFF <ENTER>

turns dual routing off.

DUMP Utility Program

DUMP filespec {options}

Copies the filespec, a machine-language program, from memory to disk. You can then load or execute the program at any time.

hhhh is a 4-digit hexadecimal number without the X' ' notation.

You can use some or all of the following options:

START=hhhh sets the program's starting address. You must include this option.

END=hhhh sets the program's ending address. You must include this option.

TRA=hhhh sets the address at which your program begins executing after you load it. If you omit this option, DUMP uses the address set by RELO. The transfer address must be less than the ending address.

RELO=hhhh sets the address at which your program begins loading back into memory. If you omit this option, TRSDOS-II uses START.

RORT=R (Return) loads filespec, but does not execute it.

RORT=T (Transfer) loads and executes filespec from TRSDOS-II Ready. If you omit the RORT= options, TRSDOS-II uses RORT=T.

Examples

DUMP TEST/FIL START=64FØ,END=6AFØ,TRA=67F2,RORT=R <ENTER>

creates filespec TEST/FIL, which contains the program in memory location X'64FØ' to X'6AFØ'. When loaded, it occupies the same memory location. Since RORT=R, you cannot execute the program from TRSDOS-II Ready mode.

DUMP INTCOM/DMA START=6ØØØ END=67FF TRA=31Ø8 RELO=3ØØØ <ENTER>

creates filespec INTCOM/DMA, which contains the program in memory range X'6ØØØ' to X'67FF'. When loaded, the program

resides from X'3ØØØ' to X'37FF' and execution starts at X'31Ø8'. You can execute the program from the TRSDOS-II Ready mode by typing:

INTCOM/DMA <ENTER>

ECHO Library Command

ECHO

Lets you type information to the display, without TRSDOS-II interpreting it as a command. Press <BREAK> to stop ECHO and return to TRSDOS-II Ready.

Example

```
DUAL ON <ENTER>
ECHO <ENTER>
```

Now, what you type is output to the printer, as well as to the display. (Most printers do not print the line until they receive a carriage return or the input buffer is filled).

ERROR Library Command

ERROR number

Displays a message that describes error number. When TRSDOS-II gives you a reverse (black-on-green/white) message, such as:

* * ERROR 47 * *

You can find out what ERROR 47 is by typing:

ERROR 47 <ENTER>

For a complete list of error codes, messages, and explanations, see Appendix B of this manual.

Example

ERROR 3 <ENTER>

returns the error message:

PARAMETER ERROR ON CALL

FC Library Command

FC

Edits and repeats the last command entered.

When you specify FC ("fix command"), TRSDOS-II displays the last command entered and lets you edit that command. Then, it executes the edited command.

After "fixing" the command, you can execute it only when the cursor is positioned as far left as possible on the line. Press <ENTER> to move the cursor to this position. Press <ENTER> again to execute the "fixed" command.

Use the following subcommands to edit command lines:

Key	Function
<F1>	Inserts blank spaces. (Note: Blank spaces are the only insertions that FC allows. If you need to insert characters, insert spaces and then type over the spaces with the desired characters.)
<F2>	Deletes a character.
<CTRL> <E>	Moves the cursor to the end of the line.
<ESC>	Starts over. TRSDOS-II disregards all previous changes.
<BREAK>	Aborts and returns to TRSDOS-II Ready.
<ENTER>	Moves the cursor to the left-most position. If cursor is already there, TRSDOS-II executes the command.
<left arrow>	Moves the cursor to the left. Does not wrap around.

Key	Function
<right arrow>	Moves the cursor to the right. When the cursor reaches the last position on the line, it returns to the left-most position.
<TAB>	Tabs (skips over) to the next tab stop, without erasing characters. Tabs are set at 8-space intervals (8, 16, 24, and so on).
<CTRL> <T>	Back-tabs without erasing characters.
<CTRL> <W>	Deletes to the end of the line, starting at the current cursor position.

Example

Suppose you have just executed the following command:

COPY PAYROLL/DAT:1 :4 <ENTER>, type:

To re-display the command for editing, type:

FC <ENTER>

Move the cursor so it is on top of the number 4 -- by pressing <TAB> or <right arrow> -- and type the number 3. Press <ENTER> to return the cursor to the beginning of the line. The new command is:

COPY PAYROLL/DAT:1 :3

Press <ENTER> again to execute the command.

FCOPY Utility Program

FCOPY source TO destination {options}

Copies files from floppy diskette TRSDOS 1.2, 1.2a, 2.Ø, 2.Øa, or 2.Øb to a disk formatted by TRSDOS-II (and vice versa except to TRSDOS 1.2 or 1.2a).

If you are going to use TRSDOS files under TRSDOS-II, you must FCOPY the files to disks formatted by TRSDOS-II. This is because TRSDOS formats 26 sectors per diskette track and TRSDOS-II formats 32 sectors per diskette track and 34 sectors per hard disk track. If you try to use a TRSDOS diskette while operating under TRSDOS-II, you get the error "Illegal I/O Attempt."

You can also use FCOPY to get a directory listing while operating under TRSDOS-II. The directory can be for files formatted by either TRSDOS or TRSDOS-II. The syntax for this special use of FCOPY is given later in this entry.

The source is one of the following:

filespec that must include a drive number
wildcard:drive
drive

If you specify the source as a drive number and omit ALL, TRSDOS-II prompts you to enter a wildcard mask.

The destination is one of the following:

filespec, only if the source is also a filespec. The filespec must include a drive number.
drive

drive is a drive number from Ø-7.

You cannot use FCOPY with FLOPPY {OFF}. (See FLOPPY command.)

The options are:

ABS overwrites any existing data without prompting. If you omit this option, TRSDOS-II prompts you before overwriting.

PROMPT tells TRSDOS-II to prompt you before it copies a file. Press <Y>, <N>, <Q>, or <S> (for "yes," "no," "quit," or "stop prompting").

ALL copies all user files.

SYS copies language and application programs. If you use SYS, the destination must be the primary drive. If you omit this option, only user files are copied.

The syntax for displaying a directory listing is:

FCOPY drive {DIR,SYS,PRT}

The braces are required with this form of FCOPY.

When you use the FCOPY command with DIR and SYS, TRSDOS-II returns the directory according to the way the diskette in the specified drive is formatted. For example, if you enter the commmand:

FCOPY 1 {DIR,SYS} <ENTER>

and the diskette in Drive 1 is formatted by TRSDOS 2.Øa, the computer returns a directory of both system and user files.

On the other hand, if you use the same command on a disk formatted by TRSDOS-II, it returns a directory of system files only.

Examples

FCOPY NEWFILE/TXT TO :3 <ENTER>

copies NEWFILE/TXT, which is on a TRSDOS-formatted diskette, to the Drive 3 diskette, formatted by TRSDOS-II.

FCOPY 1 TO Ø {ALL} <ENTER>

copies all user files from the TRSDOS-formatted diskette in Drive 1 to the Drive Ø diskette, formatted by TRSDOS-II.

```
FCOPY !:4 TO 1 <ENTER>
```

copies all user files, with and without extensions, from the Drive 4 to Drive 1. (See "Super Wildcard" in the "Introduction.")

```
FCOPY TRN/TXT:1 TO TRNTXT/OLD:4 <ENTER>
```

copies the file TRN/TXT from the diskette in Drive 1 to Drive 4. It names the new file TRNTXT/OLD.

```
FCOPY 2 {DIR} <ENTER>
```

lists the directory of user files for the TRSDOS diskette in Drive 2.

FILES Utility Program

FILES source {options}

Lists alphabetically the filenames that are stored on the specified source.

The source can be a filespec, wildcard, or drive number (Ø-7). If you omit the drive number, TRSDOS-II goes to the primary drive.

The options are:

SYS lists all system files. SYS is optional; if you omit it, TRSODS-II lists only the user files.

PRT prints the files. PRT is optional; if you omit it, TRSDOS-II lists the files on the screen.

Unlike the DIR command, FILES lists only filenames. It lists them in five columns (from left to right) across the screen.

FILES allows full wildcarding. For details, see "Wildcards" in the "Introduction."

Examples

FILES */BAS:4 {PRT} <ENTER>

lists to the printer all user files in Drive 4 that have the extension /BAS.

FILES Ø {SYS} <ENTER>

lists on the display all Drive Ø system files. The system directory is in Drive Ø on a floppy drive system.

FLOPPY Library Command

FLOPPY {switch}

Ignores all references to floppy drive numbers within filespecs. This is useful when a program or library command includes a reference to a filespec in which a drive is specified.

The switch options are:

ON sets FLOPPY on. TRSDOS-II goes to the drive(s) specified in the filespec.

OFF sets FLOPPY off. TRSDOS-II ignores the drive number(s) in the filespecs.

If you omit both options, TRSDOS-II displays the current status of FLOPPY.

TRSDOS-II powers up using FLOPPY {ON}.

Only those commands that require filespecs are affected by FLOPPY. These are:

ATTRIB	DO	MOVE
APPEND	DUMP	OPEN
BUILD	KILL	PATCH
COPY	LIST	RENAME
CREATE	LOAD	

When any of these commands is used with FLOPPY {OFF}, TRSDOS-II ignores the drive numbers. The drive search begins with the primary drive.

Examples

```
FLOPPY {OFF} <ENTER>
```

TRSDOS-II begins the search with the primary drive.

```
FLOPPY {ON} <ENTER>
```

uses the drive numbers included in a filespec.

```
FLOPPY <ENTER>
```

displays the status of FLOPPY.

FORMAT Utility Program

FORMAT drive {options}

Prepares a blank disk for use by defining the tracks and sectors and writing system information onto it. (For more information, see "Diskette Organization" in the "Technical Information" section.)

With a hard disk, FORMAT initializes the system, as well as formats it. Formatting Drive 4 (the primary hard disk drive) transfers TRSDOS-II to that drive. Then, whenever the computer is turned on or reset, it automatically loads TRSDOS-II from Drive 4.

You can format either a blank or already formatted disk. If the disk already is formatted, you lose all information when you reformat it.

You can format a single- or double-sided disk, without spedifying the number of sides. TRSDOS-II automatically formats the correct type.

drive specifies the drive in which the blank disk is to be formatted. It can be one of the following numbers: 1, 2, 3, 4, 5, 6, or 7. You cannot format your primary drive (the drive that contains the operating system). Floppy drive users cannot format Drive Ø. Hard disk users cannot format Drive 4, except to initialize.

If you omit the drive, TRSDOS-II prompts you for it.

The options are:

ABS overwrites any existing data without prompting. If you omit this option, TRSDOS-II prompts you before overwriting.

ID=disk-name assigns a name to the disk being formatted. If you omit this option, TRSDOS is used.

PW=password assigns the master password to the disk. The master password allows access to all user files (via the PROT command). It also allows full BACKUP privileges. If you omit this option, PASSWORD is used.

DIR=number places the primary directory on Cylinder number. If you omit this option, TRSDOS-II uses Cylinder 44 for floppy diskettes or Cylinder 13Ø for hard disks. You can put the primary directory on any cylinder from 1-71 for floppy diskettes or from 1-maximum cylinder number minus 3 for hard disks.

ALT=number places the alternate directory on Cylinder number. If ALT=ØØ, no alternate directory is created. If you omit the ALT= option, TRSDOS-II uses the formula directory + 3 to compute the placement of the alternate directory. For floppy diskettes, 3 represents three tracks; for hard disks, it represents three cylinders. You can put the alternate directory on any cylinder from 1-71 for floppy diskettes or from 1-maximum cylinder number minus 3 for hard disks.

SIZ=number puts the specified number of filenames in the initial directory. For hard disks and floppy diskettes, the number can be from 1-222Ø. If you omit this option, the number defaults to 18Ø (single- or double-sided floppy) or 336 (hard disks).

FULL/NONE is the verification level. FULL reads each sector and compares it to the value written during initialization. NONE prevents verification. FULL is the default.

ILV=number sets the interleave factor (ratio of number:1), which determines the order in which TRSDOS-II is to access disk sectors. Between disk accesses, TRSDOS-II must do a certain amount of processing. (The amount depends upon your application.) The proper ILV factor can reduce file work time by minimizing disk rotation between accesses. If you omit the option, the number defaults to 1Ø.

HDS=number (for hard disk systems only) specifies the number of heads on a drive. You can obtain the number from your Hard Disk Owner's Manual. If the drive number is from 4-7, HDS= is required. If you omit the option, TRSDOS-II prompts you for the number.

CYL=number (for hard disk systems only) specifies the number of cylinders on a drive. You can obtain the number from your Hard Disk Owner's Manual. If the drive number is from 4-7, CYL= is required. If you omit the option, TRSDOS-II prompts you for the number.

PRE=number (for hard disk systems only) sets the write precompensation start cylinder for a hard disk. If you omit the option, it defaults to 128. If you change the PRE= default, the DIR= default changes to PRE + 2 and the ALT= default changes to DIR + 4.

Note: If you are not familiar with interleave factors and write precompensation, we recommend that you do not use the ILV= and PRE= options. The default values for these options are suitable for most applications.

Examples

```
FORMAT 1 {ID=ACCOUNTS,PW=IRS} <ENTER>
```

formats the diskette in Drive 1, names it ACCOUNTS, and gives it the password IRS.

```
FORMAT <ENTER>
```

prompts you for the drive to use and then formats the disk in that drive. Since no options are used, the disk is named TRSDOS, the password is PASSWORD, and all the other options are defaults.

```
FORMAT 2 {DIR=Ø1,ALT=Ø4,SIZ=36Ø} <ENTER>
```

formats the diskette in Drive 2, puts the primary directory on Cylinder 1 and the alternate directory on Cylinder 4, and sets the number of directory records to 36Ø. (The leading zeroes in the cylinder specifications are optional.)

```
FORMAT 4 {HDS=6,CYL=23Ø} <ENTER>
```

formats Drive 4 for 6 read/write heads and 23Ø cylinders and automatically transfers TRSDOS-II to Drive 4.

When to Format

To prepare a new disk

Before you can use a new disk, you must format it (unless it is a floppy diskette and you use BACKUP). After formatting, record the disk name, date of creation, and password. Store this information in a safe place. It helps you estimate how long a diskette has been in use. And, if you forget the master password, it ensures continued access.

To erase all items of data from a disk

To "start over" with a disk, you can reformat it. This erases all old information and locks out all flawed sectors which have developed. It puts the system information back on the disk and leaves the "good" sectors available for information storage. Use the FULL verification option with this application.

To initialize a hard disk system

Before you can use your hard disk system, you must format the hard disks. Formatting Drive 4 transfers TRSDOS-II to that drive. You must use HDS= and CYL= when formatting hard disks.

FORMS Library Command

FORMS {format options}
FORMS {switch options}

Sets up printer parameters.

The format options are:

P=number sets the number of lines per page. The number can be from Ø-255. If omitted, P=66.

L=number sets the maximum number of lines to print on a page before issuing an automatic top of form. The number can be from Ø-255. If omitted, L=6Ø. The number of lines must be less than or equal to the page length. If either is Ø, both must be Ø. If L=Ø, TRSDOS-II issues no automatic top of form.

W=number sets the maximum number of characters to print on a line before issuing an automatic carriage return. The number can be from Ø-255. If omitted, W=132. If W=Ø, TRSDOS-II issues no automatic carriage returns.

C=byte sets the output to the specified 1-byte hexadecimal control code and sends it to the printer on completion of the FORMS command.

The default parameters are P=66, L=6Ø, W=132, and C=Ø. If you want to use the default parameters, you don't need to issue the FORMS command.

To determine the parameters to set for the format options, use the following formulas:

P (lines per page) =
lines per inch * inches per form length

L (printed lines per page) =
lines per page - lines per top and bottom margins

W (characters per line) =
characters per form width - characters per side margins
If W is greater than the form width, TRSDOS-II automatically breaks the line at the maximum length and continues printing on the next line.

C (control codes) are required for some printers (for example, to set up for the double-space character). TRSDOS-II sends the specified code to the printer or print file during execution of the FORMS command.

The switch options are:

- **A** outputs a line feed after a carriage return (auto line-feed mode), even if the transparent mode is in effect. A updates the line count by carriage returns, not by line feeds.
- **D** ignores all printer output ("dummy" mode).
- **N** returns to the normal (non-transparent, non-dummy) mode. This is the default mode.
- **Q** cancels the auto line-feed mode.
- **R** returns to the parallel printer driver.
- **T** sends the top-of-form character, X'ØC', to the printer.
- **S** switches to the Serial Channel B printer driver. (You must do SETCOM before data is sent to the printer.)
- **X** sends all items of data to the printer or printer file without any translation (transparent mode).

See the PRCTRL SVC in the "Technical Information" section for information on the transparent and dummy modes.

Examples

```
FORMS <ENTER>
```

resets all FORMS parameters to the default values.

```
FORMS P=56, L=51, W=92 <ENTER>
```

sets the page length to 56, the printed lines per page to 51, and the characters per line to 92.

```
FORMS D <ENTER>
```

invokes the dummy mode. TRSDOS- II ignores all printer commands.

```
FORMS S <ENTER>
```

sets up the serial printer driver.

FREE Library Command

FREE drive {PRT}

Lists the number of free sectors on a disk, the number of free sectors that are grouped (in areas called extents), and the number of extents.

The drive is optional. If omitted, it defaults to the primary drive.

PRT, also, is optional. If you use PRT, TRSDOS-II sends the list to the printer. If not, it displays the list on the video. If you omit the drive, you must enclose PRT in braces.

FREE lists the extents that are available. The format is:

```
nnnn nnnn nnnn nnnn
nnnnn Free Sectors in n Extents
```

nnnn is the number of sectors per extent. nnnnn is the total number of sectors on the disk. n is the number of extents on the disk. Here is a sample FREE list:

```
FREE LIST for Drive:4   Disk Name:TRSDOS
6       33      16587   34      16592
33252 Free Sectors in 5 Extents
```

Examples

```
FREE <ENTER>
```

displays the amount of free space on the disk in the primary drive.

```
FREE {PRT} <ENTER>
```

lists to the printer the amount of free space on the disk in the primary drive. Because no drive is specified in this example, you must use the braces, { }.

```
FREE 2 PRT <ENTER>
```

lists to the printer the amount of free space on the disk in Drive 2.

HELP Utility Program

HELP command

Displays the syntax of a TRSDOS-II command.

The command is optional. If you omit the command or type an unrecognized command, TRSDOS-II displays the TRSDOS-II commands and general subjects for which HELP is available.

The HELP command cannot be used from TERMINAL or BASIC.

Example

HELP MOVE <ENTER>

displays the syntax for the MOVE command.

HELP SYNTAX <ENTER>

returns an explanation of the format of the HELP messages.

HOST Library Command

HOST {switch}

Lets your computer act as host to a remote terminal or a computer acting as a terminal.

The switch options are:
ON turns HOST on
OFF turns HOST off

The switch is optional. If you omit it, TRSDOS-II displays the switch on/off status.

As the host, your computer communicates via Serial Channel A. It accepts keyboard input from the RS-232C interface and transmits all display output to the same interface.

While HOST is on, the keyboard and the remote terminal both can provide keyboard input. Output is duplicated to the video display and Channel A. Remote characters have a higher priority than local characters.

To turn the HOST program on, follow this procedure:

1. Initialize Serial Channel A with the SETCOM command. Use the appropriate parameters for your computer. (See the SETCOM command.) Do not turn off Channel A while HOST is active.

2. Turn the HOST on by typing:

   HOST ON <ENTER>

3. As TRSDOS-II starts HOST, it asks you if you want to disable the remote <BREAK> key. Answer accordingly.

Examples

If your computer is connected, via a modem, to another computer running the TERMINAL program, use this command:

HOST ON <ENTER>

to instruct the computer to accept "keyboard" input from the remote terminal and to echo all display output to the remote terminal.

To stop the HOST operation, type:

HOST OFF <ENTER>

I — Library Command

I :drive

Reads the diskette IDs from the specified floppy drive. If you omit the drive number, TRSDOS-II reads the IDs from all floppy drives.

Whenever you swap diskettes, you must execute this command immediately **after** you swap diskettes.

Use the I command after:

- Using FORMAT
- Inserting a different diskette into Drive Ø, 1, 2, or 3
- Using FCOPY, BACKUP, SAVE, or RESTORE.

If the diskette you insert after the swap is formatted by TRSDOS (instead of TRSDOS-II), TRSDOS-II displays Error 39 (Illegal I/O Attempt).

> Note: When you swap from a single-sided diskette to a double-sided diskette (or vice versa) in Drive Ø, you must press the RESET switch after swapping.

Example

```
I <ENTER>
```

reads the diskette IDs from all floppy drives in the system.

```
I :3 <ENTER>
```

reads the diskette ID from Drive 3.

KILL Library Command

KILL filespec

Deletes filespec from the directory and frees the space allocated to it. The filespec can be a wildcard with drive number.

Before it deletes the file, TRSDOS-II displays the complete filespec and gives you one more chance to change your mind. It asks "Delete? (Y/N/Q).." -- "yes," "no," or "quit."

Example

KILL DATAFILE/OLD <ENTER>

deletes DATAFILE/OLD from the directory and frees all space allocated to it.

KILL */BAS:4 <ENTER>

deletes all files with the extension /BAS. TRSDOS-II prompts you before deleting each file. You can end the process by pressing <Q> to quit.

LIB — Library Command

LIB

Displays a list of all library commands.

The LIB command list includes only those commands located below X'28ØØ'. Therefore, you can execute library commands from BASIC, TERMINAL, and so on. The library commands CLEAR, DEBUG, and DO return to TRSDOS-II Ready.

Example

```
LIB <ENTER>
```

returns a list of the TRSDOS-II library commands.

LIST Utility Program

LIST filespec {options}

Lists the contents of filespec.

This list shows the hexadecimal contents and the ASCII characters corresponding to each value. For values outside the range of X'2Ø' to X'7F', TRSDOS-II displays a period (.).

The options are:

PRT lists the contents of filespec to the printer. If you omit PRT, TRSDOS-II sends the list to the video display.

SLOW pauses between records. If you omit SLOW, TRSDOS-II does not pause.

R=number sets the starting record to the number, which can be from 1-65535. If you omit the R= option, TRSDOS-II uses 1. (See the "Technical Information" section for details.)

A lists only the ASCII characters.

Examples

```
LIST DATA/BAS <ENTER>
```

lists the contents of DATA/BAS.

```
LIST TEXTFILE/1 SLOW <ENTER>
```

lists the contents of TEXTFILE/1, pausing between records.

```
LIST TEXTFILE/1 R=1ØØ,A <ENTER>
```

lists TEXTFILE/1 starting with the 1ØØth record. TRSDOS-II displays only ASCII characters.

```
LIST PROGRAM/CMD PRT <ENTER>
```

lists PROGRAM/CMD to the printer.

LOAD Library Command

LOAD filespec

Loads a machine-language program named filespec and then returns to the TRSDOS-II mode.

Example

LOAD MARKET/OBJ <ENTER>

loads the machine-language program file named MARKET/OBJ into memory.

MEMTEST **Utility Program**

MEMTEST

Tests the random access memory (RAM) in your computer. You may select either a full memory test (X'ØØØØ' to end) or a user memory test (X'3ØØØ' to the top of user memory).

Note: See the STATUS command to locate the top of user memory.

To execute the test, type the following at TRSDOS-II Ready:

MEMTEST <ENTER>

After running the full memory test, you must reset the system.

After running the user memory test, control returns to TRSDOS-II Ready. During the operation, all user memory is cleared. No high-memory routines should be active while MEMTEST is running. Use the STATUS command to find out if any are active.

MOVE **Utility Program**

MOVE source **TO** destination {options}

Copies one or more user files to the destination disk.

The source is one of the following:

filespec

wildcard:drive You can use either a wildcard (*) or super wildcard (!).

drive

If you specify drive and omit the ALL option, TRSDOS-II prompts you to enter a filespec or wildcard mask.

The destination is the drive number of the disk to which you want your files moved. Your moved files retain the name of the source filespec.

drive is a drive number from Ø-7.

The options are:

ABS overwrites, without prompting, any existing files on the destination disk that have the same name as the source.

PROMPT displays each file before moving it and gives you a set of options for that file. The PROMPT options are: Y/N/S/Q (for "Yes -- Copy," "No -- Don't copy," "Stop prompts and proceed with all copies," "Quit this command -- Make no more copies"). If you omit PROMPT, TRSDOS-II moves all files that match the file specifications.

ALL moves all user files.

Note: You cannot move password-protected files.

MOVE is useful when you want to copy all of your TRSDOS-II files from floppy diskette to hard disk, or vice versa.

For example, type:

```
MOVE Ø TO 4 {ALL} <ENTER>
```

to move all your user files from the diskette in Drive Ø to the hard disk.

Examples

```
MOVE DAT/FLE:1 TO 3 <ENTER>
```

moves the file DAT/FLE from the diskette in Drive 1 to the diskette in Drive 3, keeping the filename DAT/FLE.

```
MOVE */PAY:4 TO 2 <ENTER>
```

copies all user files with the extension /PAY from Drive 4 to the diskette in Drive 2.

```
MOVE !:4 TO 3 <ENTER>
```

moves all user files, with and without extensions, from Drive 4 to the diskette in Drive 3.

PATCH Utility Program

PATCH filespec {options}

Lets you make minor corrections in any disk file, provided that:

- You know the existing contents and location of the data you want to change
- You want to replace one string of code or data with another string of the same length
- The file is a fixed-length record file (FLR file)

The filespec is the file you want to change. If it is a system file, you need not include a password. If it is a protected user file, however, you must include the password.

The options are:

A=hhhh sets the starting address of the data to be changed. This is where the data resides in memory when the program is loaded. hhhh is a 4-digit hexadecimal value without the X' ' notation.

F=findstring indicates the string that is now in the patch area.

C=changestring indicates the data that is to replace the findstring. The changestring must contain the same number of bytes as the findstring.

R=record tells which record contains the data to be changed. It is a decimal number from 1-65536.

B=starting byte specifies the position of the first byte to be changed. It is a decimal number from 1-256.

Use either the A= option alone or the R= and B= options together.

You can use PATCH to make minor changes in your own machine-language programs. You can use PATCH to make minor replacement changes in data files, also.

PATCH lets you implement any changes to TRSDOS-II that may be supplied by Radio Shack. This way, you do not have to wait for a new release of TRSDOS-II.

Note: If you press <BREAK> during a PATCH operation, before any changes have been made in the file, PATCH closes the file and returns to TRSDOS-II. The file is unchanged. Once the PATCH process begins, however, <BREAK> has no effect.

Using PATCH on a TRSDOS-II System File

When Radio Shack releases a modification to TRSDOS-II, you receive a printout of the exact PATCH commands that you must enter to make the change.

To make such a change, follow these steps:

1. Make a backup copy of the TRSDOS-II diskette to be patched.
2. Insert the TRSDOS-II diskette to be changed into Drive Ø. (Make sure the diskette is not write-protected.)
3. At TRSDOS-II Ready, type in the PATCH command specified in the printout.
4. After the patch is complete, reset the computer to see that the diskette is operating as a TRSDOS-II system diskette.

Using PATCH on a Z-8Ø Program File

In this context, "program files" refers strictly to those files stored with the P attribute. Use the DIR command to learn the attribute. (See instructions for changing data files.)

If you want to change four bytes in a machine-language program file, you must first determine where the 4-byte sequence resides in RAM when the program is loaded. Next,

make sure your replacement string is the same length as the original string. For example, you might write down the information as follows:

```
File to change:  VDREAD
Start address:  X'528Ø'
Sequence of the code to change:  X'CD2C25E5'
Replacement code:  X'ØØØØØØC9'
```

Then, you could use the following command:

```
PATCH VDREAD A=528Ø,F=CD2C25E5,C=ØØØØØØC9
```

Using PATCH on Data Files (Including BASIC Programs)

If you have a file stored with the D attribute (data file), you must specify the patch area in terms of the logical record that contains the data and the starting byte of the data. (The LIST command gives this information.)

For example, if you want to change a 12-byte sequence in a file called NAMEFILE, use the LIST command to find the location of the sequence. If it is in Record 128, starting at Byte 14, write down the information this way:

```
File to change:  NAMEFILE
Record number:  128
Starting byte:  14
Sequence of text to change:  "JOHN'S DINER"
Replacement text:  "JACK'S PLACE"
```

Then, use the following command to patch a data file:

```
PATCH NAMEFILE R=128,B=14,F="JOHN'S DINER",C="JACK'S PLACE"
```

For data files, notice that either string can include single quotation marks, as long as the string is enclosed within double quotation marks (or vice versa).

Error Conditions

If a TRSDOS-II error occurs during the patch operation, you receive the appropriate error message, and the patch ends without changing the file.

PATCH can produce these messages, also:

PATCH STRING TOO LONG -- ABORT This occurs when you patch a data file and the findstring spans two records. You must perform the patch in two steps, one for each record that contains a part of the findstring.

FILE CONTAINS VARIABLE-LENGTH RECORDS -- ABORT You can patch FLR files only.

STRING NOT FOUND The findstring was not found at the patch location you specified. Before patching a file, you must know the exact patch location and the contents at the location.

ADDRESS OUT OF PROGRAM-LOAD RANGE -- ABORT This occurs when you attempt to patch a program file and some or all of the patch string (either the findstring or the changestring) is outside the RAM area where the program resides when it is loaded. Check the A= option. Also, be sure the findstring and changestring are not longer than you intend them to be.

PAUSE **Library Command**

PAUSE prompt message

Displays PAUSE and the prompt message and prompts you to press any key to continue.

The prompt message is optional. If you omit it, TRSDOS-II displays PAUSE.

This command is useful in a DO file. (See the DO command for more information.)

Example

Type:

```
PAUSE Insert Diskette #21
```

to display:

```
PAUSE Insert Diskette #21
Press any Key to Continue
```

Press <ENTER> to continue.

Type:

```
PAUSE <ENTER>
```

to display:

```
PAUSE
Press any Key to Continue
```

Press <ENTER> to continue.

PRINT **Utility Program**

PRINT filespec {A,V}

Prints out the contents of the filespec, omitting the record numbers and hexadecimal codes (LIST gives those).

The filespec must be a text file.

The options are:

A treats the first byte in each record as a FORMS control character. The meaning of the character in the first byte is:

"1" does a form feed before printing (top of form).

"b" does a carriage return before printing (single-space).

"Ø" does two carriage returns before printing (double-space).

"+" does a carriage return without a line feed, if the printer has the ability. The characters that follow the code are printed over the current line.

V outputs the filespec to the video display, as well as to the printer.

Use the A option when the filespec contains the control codes listed ("1", "b", "Ø", "+").

Example

PRINT PROGRAM/TXT V <ENTER>

outputs PROGRAM/TXT to the video display, as well as to the printer.

PROT Library Command

PROT drive {options}

Changes the password protection of the disk in the specified drive by using the master password.

The options are:

OLD=password specifies the disk's current master password. This option must be specified when using any option below.

NEW=password specifies the disk's new password.

LOCK assigns all user files the new master password. TRSDOS-II sets both the update and access words to this value. (See the ATTRIB command for information on access and update passwords.)

UNLOCK removes the passwords from all user files.

You first assign a disk's master password during format or backup. Your TRSDOS-II system disk has the master password PASSWORD.

Examples

```
PROT 1 OLD=PASSWORD, NEW=H2O <ENTER>
```

changes the master password of the diskette in Drive 1 from PASSWORD to H2O.

```
PROT Ø OLD=H2O, UNLOCK <ENTER>
```

removes passwords from every user file on the diskette in Drive Ø.

```
PROT Ø OLD=H2O, NEW=ELEPHANT, LOCK <ENTER>
```

changes the master password from H2O to ELEPHANT and assigns the new password to every user file.

PURGE **Library Command**

PURGE {drive} {options}

Deletes files from the disk in the specified drive.

The drive is optional. If you omit it, TRSDOS-II uses the primary drive.

The options are:

SYS System files (program and data)
PROG User machine-language program files
DATA User data files
ALL User program and user data files

If you omit the options, TRSDOS-II lets you purge data files only.

Once you enter the PURGE command, TRSDOS-II prompts you for the disk's password. Type in up to eight characters and press <ENTER>. (All disks distributed by Radio Shack use PASSWORD as the password.)

The system then displays the filenames, one at a time. It prompts you to kill the file, keep it, or quit the operation.

Examples

PURGE 1 <ENTER>

deletes data files from Drive 1.

PURGE 2 {PROG} <ENTER>

deletes user machine-language program files from the diskette in Drive 2.

RECEIVE Utility Program

RECEIVE {CH=channel, offset}

Lets you receive object code from another computer via RS-232C. The data must be sent in Intel Hex Format as described below in "Required Data Format."

The channel, which is required, can be either Serial Channel A or B.

The offset is optional. It tells TRSDOS-II to offset each load address that is specified in the incoming data (see "Required Data Format").

The offset options are:

ADD=hhhh increments the data load address by the hexadecimal value hhhh

SUB=hhhh decrements the data load address by the hexadecimal value hhhh

If you omit the offset, the data loads at the address specified in the incoming data.

Before using RECEIVE, you must initialize one of the serial channels with SETCOM. Select the appropriate parameters, depending upon the requirements of the transmitting device.

Loading Address

The data is loaded into memory according to the load information contained within the data. It must load above X'2FFF' and below the top of user memory. (Execute the SETCOM command, then use the STATUS command to get this latter value.)

If the data load address is out of this range, you can use the offset option to bring the load address into the desired range. Then, when the data is dumped into a program file, use the RELO= option (of the DUMP command) to specify the original load address.

Required Data Format

The transmitting program must send the data in "Intel Hex Format."

Each byte of data is sent as a pair of hexadecimal ASCII-coded characters:

1. High nibble (most significant four bits), sent as the first byte of pair

2. Low nibble (least significant four bits), sent as the second byte of pair

For example, the value X'F7' is sent as two bytes, "F" (X'46') followed by "7" (X'37').

Because only "." and ASCII-coded hexadecimal numbers are sent, the data is always in the range [X'3Ø', X'3A'] or [X'41',X'46']. Values outside this range stop reception and produce an error message.

Record Format

You must send records as follows:

In the chart below, MSB represents "most significant bits." LSB represents "least significant bits."

Character Number	Contents	Comments
1	":"	Sync-character to indicate the beginning of the record.
2	High nibble of record length (N)	2-byte sequence that gives the number of byte **pairs** in the record. Zero means 256 byte pairs follow.
3	Low nibble of record length (N)	

Character Number	Contents	Comments
4	High nibble of MSB of load address	4-byte sequence that gives the address where the data is to start loading. The address specified must be in the user area [X'28ØØ',TOP]. (See the STATUS command to locate TOP.)
5	Low nibble of MSB	
6	High nibble of LSB of load address	
7	Low nibble of LSB of load address	
8	High nibble of LSB of EOF(end of file) code	Byte pair that gives the EOF code. Any non-zero value means EOF (no records follow). A value of zero means records follow.
9	Low nibble of LSB of EOF code	
1Ø	First byte of the first data pair	First byte is ASCII code for the first hex digit; second byte is ASCII code for the second hex digit.
11	Second byte of the first data pair	
8 + (N*2)	First byte of last data pair	Last pair of data characters.
9 + (N*2)	Second byte of last data pair	
1Ø + (N*2)	First byte of data checksum (high nibble)	Byte pair that represents 2s complement of the data (all byte pairs after the "." up to, but not including, the checksum). Note that each byte pair is changed back to the original byte of data before it is summed.
11 + (N*2)	Second byte of data checksum (low nibble)	

Sample Record:

Character Number	Sample Data ASCII	Hex Value
1	":"	3A
2	"Ø"	3Ø
3	"2"	32
4	"2"	32
5	"8"	38
6	"Ø"	3Ø
7	"Ø"	3Ø
8	"Ø"	3Ø
9	"Ø"	3Ø
1Ø	"3"	33
11	"7"	37
12	"7"	37
13	"Ø"	3Ø
14	"A"	41
15	"3"	33

This record contains two byte pairs of data:

"3" "7" representing the value X'37'
"7" "Ø" representing the value X'7Ø'

and starts loading at X'28ØØ'. The 1-byte sum of the original bytes (represented in pairs by characters 2 through 13) is X'5D'. The 2s complement of X'5D' is X'A3' -- which is represented by Bytes 14 and 15.

Examples

First, initialize Serial Channel A with the desired settings and establish a connection with the sending device. You know that the data load address is above X'2FFF' and below the top of user memory. (See the STATUS command to locate the top of user memory.)

To begin the reception, type:

```
RECEIVE {CH=A} <ENTER>
```

If the connection is "good," your computer displays:

```
Ready to receive          #=
```

The sending device now begins to transmit. The number of the current record is displayed after #=. When the transmission is complete, you should see:

```
START ADDRESS=hhhh        LAST ADDRESS=hhhh
TRA ADDRESS=hhhh
TRSDOS-II Ready
```

In each case, hhhh is the corresponding 4-digit hexadecimal number.

The code is in the memory area specified in the data itself.

Now, you can use the DUMP command to create a program file on diskette.

When you initialize Serial Channel B with the correct parameters, you can establish a connection with the sending device. If the load address is X'28ØØ' and the end address is X'37FF', you must add X'Ø8ØØ' to the load address so the receive program is not overlaid. To do this, type:

```
RECEIVE CH=B ADD=Ø8ØØ <ENTER>
```

This command causes the data to be loaded, starting at X'28ØØ' + X'Ø8ØØ' = X'3ØØØ'.

After receiving the data, you want to dump it into a program file called PROGRAM1. Type:

```
DUMP PROGRAM1 START=3ØØØ END=3FFF RELO=28ØØ <ENTER>
```

Notice that the RELO= option resets the load address to its original value (its value before the data was transmitted).

RENAME **Library Command**

RENAME filespec1 **TO** filespec2

Changes a file's name and extension from filespec1 to filespec2.

RENAME does not change the file's password, contents or position on the disk. (See the ATTRIB command to change the password.)

Examples

RENAME BACK/LST TO NEWLIST/1 <ENTER>

renames BACK/LST to NEWLIST/1.

RENAME REPORT/AUG:3 TO REPORT/SEP <ENTER>

renames REPORT/AUG on Drive 3 to REPORT/SEP.

RESET **Library Command**

RESET

Resets and restarts TRSDOS-II.

This command is almost identical to using the RESET switch. If your computer is capable of generating sound, RESET turns off the sound.

Example

RESET <ENTER>

RESTORE Utility Program

RESTORE source TO destination {options}

Retrieves from floppy diskettes any files stored in compressed form by the SAVE command. Because SAVE stores files in a special format, RESTORE is the only way to return these files to the hard disk drive.

Note: For your convenience, please read Appendix F before continuing with the RESTORE entry.

The source specifies a floppy diskette and is one of the following:

filespec that must include a drive number
wildcard:drive
drive

In source specifications, drive is a drive number from 1-3 for floppy drive systems or Ø-3 for hard disk systems.

The destination is optional. It can be one of the following:

drive
filespec:drive Use if {options} is {IND}.

In destination specifications, drive is a drive number from Ø-7, but it cannot be the same as the source drive number.

If you specify DIR, you cannot specify a destination.

If you omit the destination, TRSDOS-II uses the first available disk drive.

If both the source and destination are drive, all user files are restored.

The options are:

ABS retrieves the specified files. If you use this option, TRSDOS-II overwrites any existing file of the same name. If you omit the option, TRSDOS-II prompts you to ready the source and destination diskettes and it prompts you before overwriting.

DIR displays the dataset directory and identifier, if Volume Ø is in the source drive. If Volume Ø is not in the source drive, TRSDOS-II displays only the dataset identifier.

IND (indirect) specifies the destination as an indirect file, the contents of which are used as a list of destination filespecs.

KILL deletes the destination file before it opens the file for restoring.

PROMPT asks for verification of each file for restoring. Press <Y>, <N>, <Q>, or <S> (for "yes," " no," "quit," or "stop prompting and continue").

PRT sends the directory listing to the printer. Use only with the DIR option.

SYS retrieves all system files, including system (language) and applications programs. If used with DIR, SYS lists the directory of system files.

RESTORE reads information from a dataset created by SAVE. If you enter a volume of this dataset out of sequence, TRSDOS-II informs you of the mistake. The system also informs you if you accidentally enter a volume from a different dataset during a RESTORE.

Note that the TRSDOS-II diskette must remain in Drive Ø on floppy drive systems. Also, single-drive saves and restores are not allowed. For example, RESTORE :1 :1 is illegal.

When you're restoring files in a dataset, TRSDOS-II prompts you with:

Mount NEXT Diskette in Drive n -- Press ANY Key to continue.

which instructs you to enter the next volume of the dataset.

Examples

RESTORE Ø TO 4 <ENTER>

retrieves all saved non-system files on Drive Ø and puts them in Drive 4.

RESTORE !:2 TO 4 <ENTER>

retrieves all saved non-system files, with and without extensions, from the floppy diskette in Drive 2 and puts them on the hard disk in Drive 4.

RESTORE 1 PROGRAMS {IND} <ENTER>

where PROGRAMS is an indirect file containing the files:

MAILIST/PRG:4
MAILDAT/TXT:4
CHANGES/TXT:4

retrieves the files from the floppy diskette in Drive 1 and puts them in the filespecs, defined in PROGRAMS, on hard disk Drive 4. Note that "TO" is optional.

RESTORE */SRC:Ø 4 <ENTER>

retrieves all Drive Ø user files that have the extension /SRC and puts them on hard disk Drive 4. The filenames stay the same.

RESTORE :1 {DIR,PRT}

sends the directory of the floppy diskette in Drive 1 to the printer.

SAVE **Utility Program**

SAVE source **TO** destination {options}

Creates a file-by-file backup of the source onto the destination.

You must have at least two drives to use the SAVE command; the source and destination drives cannot be the same.

Note: For your convenience, please read Appendix F before continuing with the SAVE entry.

The source is one of the following:
drive (You must specify ALL if you use drive.)
filespec if {options} is {IND}. The filespec must include a drive number.
wildcard:drive

In all source specifications, drive is a drive number from Ø-7.

The destination specifies a floppy drive and is:
drive specifies a drive number from 1-3 for floppy disk systems and Ø-3 for hard disk systems.

The options are:
ABS overwrites, without prompting, any data already on the destination diskette. If you omit this option, TRSDOS-II prompts: Destination Diskette Ready?
DC value date compares the creation date of each specified source file against the date specified and saves the file, if all other criteria are met.
DM value date compares the last modification date the same way DC compares the creation date.
IND (indirect) specifies the source as an indirect file, the contents of which are used as a list of source filespecs.
PROMPT asks for a file verification before saving. You may respond by pressing <Y>, <N>, <Q>, or <S> (for "yes," "no," "quit," or "stop prompting -- continue").

ALL saves all user files. If you use drive as the source, you must use ALL.
SYS saves all system files, including language and application programs.

Note: value is one of the following:
< where < means less than or equal to
> where > means greater than or equal to
= where = means equal to

The date must be in the form: mmddyy.

Examples

There is a variety of ways to use SAVE. The simplest is:

SAVE 1 TO 2 {ALL} <ENTER>

This copies all the files from Drive 1 into a compact form on the diskette in Drive 2.

Wildcarding

Wildcards also offer an easy way to save several files or an entire disk. For example:

SAVE */CBL:4 TO Ø <ENTER>

saves all Drive 4 files with the extension /CBL and puts them on the diskette in Drive Ø.

Using the IND Option

The indirect option lets you save groups of files by creating an indirect file, a file consisting of one or more filespecs (similar to a DO file). You can use the BUILD command to create this list of filespecs.

At TRSDOS-II READY, type:

BUILD PROGRAMS:Ø <ENTER>

This creates an indirect file called PROGRAMS.

After TRSDOS-II prompts you with:

```
Enter command line (1-8Ø)
.....................................
```

Enter your list of file specifications including drive numbers, for example:

```
ORDERS:5 <ENTER>
REPORTS/*:6 <ENTER>
```

To exit the BUILD and return to TRSDOS-II READY, press <BREAK>.

You are now ready to save the files (specified by the indirect file) to the specially formatted floppy diskette. Type:

```
SAVE PROGRAMS:Ø TO 1 {IND} <ENTER>
```

Both ORDERS and REPORTS are now found in the file named PROGRAMS on the diskette in Drive Ø and saved to the diskette in Drive 1.

Note: The IND option lets you save more than one file from each hard disk; it also lets you save from more than one hard disk. As a result, you might save multiple files that have the same name. Because the save and restore directory does not specify drive numbers for files, you could lose duplicate filenames.

For example, if you created an indirect file that has these files:

```
*/FOR:4
*/CBL:4
*/FOR:5
```

Drives 4 and 5 may have duplicate filenames with the /FOR extension. Before you use indirect, examine all the files to be saved. Rename any duplicate filenames or before saving or create different datasets.

Using the DC and DM Options

Another way to save files is to do so with respect to their creation or modification (update) dates. For example, suppose your directory showed these creation and update dates for your files:

Filename	Created	Updated
MENU/PRG	6/1/81	9/2/81
PRGONE/PRG	6/1/81	8/16/81
PRGTWO/PRG	6/1/81	7/3Ø/81
PRGTHR/PRG	6/1/81	6/16/81
PAYROLL/DAT	9/15/81	1Ø/15/81
CHECKS/DAT	9/15/81	1Ø/15/81
TEST/PRG	1Ø/29/81	1Ø/29/81

If you want to save only those files created on June 1, 1981, use the following command:

```
SAVE */*:5 TO Ø {DC=Ø6Ø181} <ENTER>
```

The first four files are saved to the floppy diskette in Drive Ø.

In the same sense, the first four files were updated on or before September 2, 1981 (9/2/81). Type:

```
SAVE */PRG:5 TO Ø {DM<Ø9Ø281} <ENTER>
```

and all files updated on or before the specified date are saved.

SCREEN Library Command

SCREEN

Copies all of the current screen display to the printer.

Graphics characters are printed as periods and reverse (green/white-on-black) alphanumeric characters are printed as normal (black-on-green/white) characters.

Example

To copy the current command line to the printer, type:

```
SCREEN <ENTER>
```

SETCOM Library Command

SETCOM {options}

Sets up Serial Channels A and B (on the back panel) for communicating with a remote device, via a modem or hardwire connection.

If you are not a machine-language programmer and want to communicate with a remote device, you can either use the HOST program or buy a communications program. The manual that comes with the communications program explains how to use it. See your Radio Shack store for information.

The options are:

A=OFF turns off Channel A's RS-232C Communication settings

B=OFF turns off Channel B's RS-232C Communication settings

A=(baud rate,word length,parity,stop bits) sets up Channel A for RS-232C Communication

B=(baud rate,word length,parity,stop bits) sets up Channel B for RS-232C Communication

If you omit all options, TRSDOS-II displays the status of both channels.

The RS-232C settings can be the following:

baud rate	11Ø, 15Ø, 3ØØ, 6ØØ, 12ØØ, 24ØØ, 48ØØ, 96ØØ. If you omit the setting, 3ØØ is used. (Some programs do not run correctly at speeds higher than 24ØØ baud.)
word length	5, 6, 7, 8. If you omit the setting, 7 is used.
parity	E, O , or N (for "even," "odd," or "none"). If you do not specify the setting, even is used.

stop bits	1, 2. If you do not specify the setting, 1 is used.

You must put a comma after every option but the last. The options are positional. For example, the third item in an option list must always specify parity. To use a default value, omit the option and insert only the comma.

To change the settings on a currently active channel, first turn off the channel.

Before executing this command, connect the remote device to Channel A or B.

Then, after executing it, you can begin sending and receiving data, using one of these TRSDOS-II supervisor calls. (See the "Technical Information" section for details.)

Name	SVC Number	Function
ARCV	96	Channel A receive
ATX	97	Channel A transmit
ACTRL	100	Channel A control
BRCV	98	Channel B receive
BTX	99	Channel B transmit
BCTRL	101	Channel B control

These system routines are available only when the respective channel has been initialized. (See "Technical Information" for details.)

Examples

```
SETCOM A=( ) <ENTER>
```

sets up Channel A for serial communications, using all the default parameters. TRSDOS-II function calls 96 and 97 (ARCV and ATX) are available for serial input/output. The status of Channel B remains the same.

```
SETCOM B=(6ØØ,8,,2), A=OFF <ENTER>
```

sets up Channel B:

baud rate	6ØØ
word length	8 bits
parity	even (default)
stop bits	2

and turns off Channel A.

```
SETCOM A=(12ØØ,8,O,), B=(,,,2) <ENTER>
```

sets up Channels A and B.

Setting	Channel A	Channel B
baud rate	12ØØ	3ØØ (default)
word length	8	7 (default)
parity	odd	even (default)
stop bits	1 (default)	2

```
SETCOM <ENTER>
```

displays the status of both channels.

```
SETCOM A=OFF, A=( ) <ENTER>
```

resets Channel A to the default parameters.

SPOOL Library Command

SPOOL {options}

Captures printer output or prints a spool file.

SPOOL increases the efficiency of the system by letting you use the system while a print operation is in progress.

The options control the spool function. They are:

ON turns on SPOOL. You must set this switch before you can use the other options.

OFF turns off SPOOL and closes the capture file and print file.

N,F=filespec creates a capture file named filespec.

P,F=filespec begins background printing of filespec.

K keeps the file after printing it. If you omit this option, TRSDOS-II deletes the file after printing it. (TRSDOS-II won't delete a print file if the file is closed by a SPOOL S or if a disk error occurs in the print file.)

C=number specifies the number of copies you want. The number can be from 1-255. If you omit this option, TRSDOS-II makes one copy.

L=line specifies the line number at which printing starts. The line may be any number from 1-65535. If you omit this option, printing begins at Line 1.

H halts background printing but saves the current position for later resumption (R option).

R,L=line resumes background printing after a halt (H option), or displays the current line number if the spooler has not been halted. If you use R,L=line, printing resumes at the specified line. If you use only R, printing resumes at the point where it stopped.

S stops printing. It closes but doesn't kill the print file and leaves the capture file open.

If you omit all options, TRSDOS-II displays the SPOOL status.

SPOOL performs two functions which you can use at the same time or one at a time:

- It saves or "captures" the data that normally goes to the printer. The spooler then can either throw away this captured data or save it in a capture file for later use.
- It prints data from a disk file while other operations are in progress. That is, you can use the system -- everything except the printer -- while printing the file. While the spool file is printing, your system captures the real-time printer output for later use.

Example 1

Capture File

In this example, you can run a program that outputs to the printer. Instead of waiting (until printing is complete) to use your system, you can capture the program in a disk file to print out later.

To do this, call the capture file SPOOL1 and type:

```
SPOOL ON <ENTER>
SPOOL N,F=SPOOL1 <ENTER>
```

This saves all printer output in SPOOL1. To stop capturing the printer output in SPOOL1, type:

```
SPOOL OFF <ENTER>
```

Now SPOOL1 is a text file which you can list or print in the normal means, at your convenience.

Example 2

Background Printing

Here you can print to a file created by SPOOL while you use the system. Using the SPOOL1 file from the first example, type:

```
SPOOL ON <ENTER>
SPOOL P,F=SPOOL1 <ENTER>
```

TRSDOS-II begins printing the file as a "background task." Printing takes place only when the system isn't busy with some higher priority operation, such as interpreting and executing your keyboard commands. Because this example doesn't include the K or C= option, TRSDOS-II prints only one copy and then deletes SPOOL1.

After completing the print file, the spooler doesn't turn itself off. Type:

```
SPOOL OFF <ENTER>
```

Example 3

Simultaneous Capture File and Background Printing

To save real-time printer output while SPOOL prints a file, you can use this example.

You need one capture file (SPOOL1) and one print file (SPOOL2).

To turn on SPOOL, create the capture file (SPOOL1), and begin capturing printer output in it, type:

```
SPOOL ON <ENTER>
SPOOL N,F=SPOOL1 <ENTER>
```

You can now use the computer normally. Then, when you're ready to print out SPOOL1, close the file and make SPOOL2 the new capture file. Do this by typing:

```
SPOOL  N,F=SPOOL2 <ENTER>
```

To print SPOOL1 and save any real-time printing in SPOOL2, type:

```
SPOOL  P,F=SPOOL1 <ENTER>
```

If you want to halt the print-file operation, type:

```
SPOOL H <ENTER>
```

This doesn't affect the capture-file operation. To resume printing, type:

```
SPOOL R <ENTER>
```

STATUS Library Command

STATUS

Displays system status information.

This command tells you the first address of protected high memory (non-user memory) and lists the on/off status of active TRSDOS-II functions.

Example

To locate the top of user memory, type:

STATUS <ENTER>

and subtract 1 from the value displayed.

T Library Command

T

Moves the printer to the next page (top of form). This command does the same as FORMS with the T option.

If you are using SPOOL and it is capturing, T sends top-of-form character X'ØC' to the capture file.

Example

T <ENTER>

advances printer to the next top of form.

TERMINAL

Transforms your computer into a "terminal" to another computer's "host" program. You can use TERMINAL for transmission and reception of ASCII text only. You cannot use it for machine-language object code.

Communication takes place through Serial Channel A. In most applications, you can hook up the computers through telephone lines via a modem.

TERMINAL's three modes of operation, all of which are described in detail in "Modes of Operations" are:

- **Menu** -- Lets you select TERMINAL's menu commands, switch command options (such as "on" and "off"), and execute TRSDOS-II library commands.
- **Interactive Terminal** -- Transmits your keyboard input and displays incoming data.
- **Transmit from RAM** -- Transmits prepared data at high speeds. Incoming data is displayed.

Setting Up

For communication through telephone lines, you need a modem such as the Telephone Interface II (26-1171), Modem I (26-1172), or Modem II (26-1173), and the RS-232C Cable (26-44Ø3).

1. Set up the modem according to its instructions and connect it to Serial Channel A on the back panel of the computer display console.
2. Set the modem to either the originate or the answer mode -- whichever is the opposite of the host program with which you are going to communicate. Set it to full or half duplex, again depending upon the requirements of the the host program.

3. Turn on the modem and the computer system.

4. Find out which RS-232C parameters (SETCOM) are required by the host program you plan to use:
   Baud rate
   Word length
   Parity
   Number of stop bits

   Initialize Serial Channel A accordingly (see "Running Terminal" below).

In some of the examples showing uses of TERMINAL, charts are used. These show the prompts you see on the display and what you should type in response.

Running Terminal

1. From TRSDOS-II Ready, start TERMINAL by typing:

   TERMINAL <ENTER>

   The program starts up in the menu mode with the prompt:

   -- Enter Menu Selection ..

2. Initialize Serial Channel A according to the requirements (baud rate, word length, parity, and number of stop bits) of the host program with which you are going to communicate. Type:

   S <ENTER>

   When the program prompts you to type in a TRSDOS-II command, type in the SETCOM command just as you would in the TRSDOS-II Ready mode. For example:

   SETCOM A=(3ØØ,7,N,2) <ENTER>

   enables Serial Channel A with 3ØØ baud, 7-bit words, no parity and 2 stop bits. After TRSDOS-II executes the command, control returns to TERMINAL's menu mode.

3. If you plan to use the TERMINAL printer command (P), which is described later, initialize the printer now with the FORMS command. Type:

   S <ENTER>

   and enter the appropriate FORMS command at the prompt.

4. To select another menu command, type in the letter specified in the menu. (See the "Menu Commands" section below for a list of options.) To redisplay the entire menu, type:

   M <ENTER>

Modes of Operation

Menu Mode

The only mode from which you can select menu commands. It also lets you enter one of TERMINAL's other modes -- transmit from RAM or interactive terminal. The menu mode is an off-line mode (you cannot transmit characters to the host program, and if characters are sent to you, they are lost).

Interactive Terminal Mode

Sends to the host program the characters you type and displays incoming characters as they are received. If the host program echoes your transmissions, they appear on the display. If not, you can select the self-echo option to display your keyboard input.

Using the R command, you can save incoming characters temporarily in the RAM buffer. Then, you can use the P command to output them to the printer.

If a transmission error occurs, TERMINAL displays an error message and waits for you to correct the error condition. (See "Error Conditions" at the end of the TERMINAL entry.) After you do, normal input/output resumes. There are three ways to enter the interactive terminal mode:

- With the T command from the menu mode
- With the O command, upon completion of an auto sign-on (menu mode)
- After transmission from the RAM buffer

To return to the menu mode, press <BREAK>.

> Note: Certain hosts prompt you to use a break character or sequence to initialize transmissions. Since the <BREAK> key sends the program from the interactive mode to the menu mode, TRSDOS-II uses <ESC> as the break character. You can also set your own break character or sequence with the B command.

Transmit from RAM (and Auto Sign-On)

Sends the contents of the RAM buffer to the host program and passes control to the interactive mode.

Auto sign-on (O command) works the same as transmit from RAM. (The following applies to both operations.)

To load the RAM buffer with prepared text from a disk file, use the G command. (If you are using auto sign-on, your auto sign-on message is sent.) You can send the data in the RAM buffer one line at a time when the host program prompts you that it is ready (W command), or you can send it in a continuous stream.

During transmission, your computer displays incoming text. If the host program echoes your transmissions, you can verify that the data was sent accurately.

During transmission, adjust the delay between characters by repeatedly pressing the <up arrow> (faster) and <down arrow> (slower) keys. If echoed data appears garbled, slow down the transmissions. If it is coming too slowly, speed it up.

If TERMINAL receives a break character or sequence in this mode, it pauses until it receives the next character. If it receives an X'13', TERMINAL pauses until it receives an

X'11'. (By convention, X'13' is called the DC3 signal and means pause. X'11' is called the DC1 signal and means resume).

Use the X command to enter this mode.

To exit this mode and return to the menu mode, press <BREAK>.

Menu Commands

A Build Auto Sign-On Message -- Lets you prepare an automatic sign-on message that contains your responses to the standard sign-on questions asked when you first call a host. By sending this automatic message to the host via the O command, you avoid repeating your responses each time you call the host.

The message can be up to 6Ø keyboard characters, including control characters. All control characters are displayed as +, but the true control code is sent. (When you display a message, no control codes are shown.) To imbed a carriage return in the message, press <down arrow>.

For example, if the host asks these sign-on questions:

```
User ID?
User Password?
Program Name?
```

and you want to store these responses in the auto sign-on buffer:

```
STL-314 <X'ØD'>
SHOWME <X'ØD'>
MENU <X'ØD'>
```

answer the prompts as follows:

Prompt	Type
-- Enter Menu Selection ..	A <ENTER>
The Current Auto Sign-On is	
Change? (Y/N) ..	Y <ENTER>
Enter Auto Sign-On Message (1-6Ø)	STL-314 ∨SHOWME∨ MENU <ENTER>
The Current Auto Sign-On is	
STL-314	
SHOWME	
MENU	

On the display, there is a blank line below the prompt "The Current Auto Sign-On is." This indicates that the original auto sign-on was blank or contained non-display characters.

B Set/Change Break Character or Sequence -- Lets you select the incoming code that is to be interpreted as a "break." It also lets you define a key to send the same break character or sequence.

You can use any code from Ø-255 as the break character and any duration from 1-451 milliseconds for the break sequence. (This is determined by the host program.) For the user-defined break key, you can use any key except <BREAK> or <CTRL> <C>.

The following example shows how to set up X'ØA' as the break character and <CTRL> <D> as the break key:

Prompt	Type
-- Enter Menu Selection ..	B <ENTER>
Break Key is Now 1B Hex	
Change? (Y/N) ..	Y <ENTER>
Enter New Key (1)	<CTRL> <D> <ENTER>
Break Key is Now Ø4 Hex	
Type of Break is Now CHR	
Change? (Y/N) ..	N <ENTER>
Break Char is Now Ø3 Hex	
Change? (Y/N) ..	Y <ENTER>
Enter New CHAR Value in Hex (2)	ØA <ENTER>
Break Char is Now ØA Hex	

C **Copy RAM Buffer to Disk** -- Creates a disk file copy of the text in the RAM buffer. The new file has a record length of 1.

Use this command to save data received into the RAM buffer in the interactive terminal mode. To minimize hookup time, do this after ending the connection to the host program. Or, if the RAM buffer is full, save the data in a disk file, then reset it and reopen it for more data.

For example, to save a report (that you receive in the interactive terminal mode) as a disk file named REPORT, answer the prompts as follows:

Prompt	Type
— Enter Menu Selection ..	C <ENTER>
Enter Filespec (1-34)	REPORT <ENTER>

This creates the new file but leaves the RAM buffer contents and status unchanged. If REPORT exists, the new data overwrites it.

To stop the copy process, press <BREAK>, which closes the disk file and returns to the menu.

D **Display RAM Buffer** -- Displays the contents of the RAM buffer. To pause the display, press <HOLD>. To continue, press <HOLD> again. If the printer command is on when you issue this command, the text is output to the printer, also. To enter the command, type (at the Enter Menu Selection prompt):

D <ENTER>

To stop the display function, press <BREAK>. The menu returns.

E **Self-Echo** -- Lets you display the characters you send via TERMINAL.

Some hosts echo the text you send. As the host receives each character, it sends it right back to you and displays

what you sent. When communicating with this kind of host, set your modem to full duplex.

If the host does not echo your text, you must use the self-echo command to display the text you send. With such a host, set your modem to half duplex.

To switch the echo option (on or off), simply type E <ENTER>. The new state of the option is displayed, and the menu prompt returns.

F Set/Change <F1> and <F2> Keys -- Lets you set <F1> and <F2> to output any code from Ø-255. This is useful if you use a particular code often.

For example, if the host recognizes X'13' (<CTRL> <S>) as pause control and X'11' (<CTRL> <Q>) as resume control, you may want to change these to <F1> and <F2> for convenience. To do this, answer the prompts as follows:

Prompt	Type
-- Enter Menu Selection ..	F <ENTER>
F1 Key Will Send a Ø1 Hex Code	
Change? (Y/N) ..	Y <ENTER>
Enter New Char Value in Hex (2)	11 <ENTER>
F1 Key Will Send a 11 Hex Code	
F2 Key Will Send a Ø2 Hex Code	
Change? (Y/N) ..	Y <ENTER>
Enter New Char Value in Hex (2)	13 <ENTER>
F2 Key Will Send a 13 Hex Code	

Now when you press <F1> while in the interactive terminal mode, TERMINAL transmits the resume control X'11'. When you press <F2>, it transmits the pause control X'13'.

G Get Disk File into RAM Buffer -- Lets you load (into the RAM buffer) text that is stored in a disk file. Then, you can send the text to the host via the X command (transmit from RAM). The previous contents of the RAM buffer are lost.

The disk file can contain fixed-length or variable-length records of any length. However, you should load and send only ASCII files. You can send any programs, if you saved them in ASCII format.

For example, to send a document stored in the file DOCUMENT/TXT, answer the prompts as follows:

Prompt	Type
-- Enter Menu Selection ..	G <ENTER>
Enter Filespec (1-34)	DOCUMENT/TXT <ENTER>

TERMINAL loads the file and returns to the menu. It closes the RAM buffer.

Now, if the host program is ready to accept data, you can use the X command to send the data. After transmission, TERMINAL goes to the interactive terminal mode.

L **Line Feed** -- Tells TERMINAL how to handle an incoming line feed (X'ØA'). If the option is on, TERMINAL ignores all line feeds; if it is off, TERMINAL does not ignore them.

This is useful if the host sends a line feed after each carriage return. Since the TRSDOS-II display and printer drivers automatically perform a line feed after a carriage return, the incoming line feed is redundant. The line feed option should be on.

To switch the option (on or off), type L <ENTER>. The new state of the option is displayed, and the menu prompt returns.

M **Display Menu** -- Clears the display and redisplays the menu. Use this when you have entered so many commands that not all of the menu commands are visible.

O **Enter Terminal Mode with Auto Sign-On** -- Starts transmission of the current auto sign-on message. After it sends the message, TERMINAL enters the interactive terminal mode.

To stop transmitting the auto sign-on, press <BREAK>. This returns control to the menu.

For details, see "Transmitting from RAM."

Note: Most host programs cannot receive anything until they send the first prompting message. Because of this, you should:

1. Go to the interactive terminal mode (T commmand) when connection is first made and wait for the host to send its first prompt character.

2. Press <BREAK> to return to the menu.

3. Start the auto sign-on (O command).

P Printer -- Switches the printer option (on or off). When the option is on, incoming text is copied to the printer. Initialize the printer with the FORMS command before you use the P command. When you use the D command while the P option is on, the RAM buffer text is copied to the printer.

TERMINAL uses a circular buffer for efficient output to the printer. If characters overflow the buffer, they may not be printed. They are displayed, however, and saved in RAM if the buffer is open. (Check your printer's specifications for the maximum character input rate. At 3ØØ baud, 7-bit characters may come in as fast as 3Ø per second.)

To minimize hookup time, keep the P option off while on-line with the host. Save the incoming text in RAM, and upon completion of the hookup, turn on the P option. Then use the D command to get a hard copy of the data.

To switch the printer option, type P <ENTER>. The new state of the option (on or off) is displayed and the menu prompt returns.

Q Quit -- Returns control to TRSDOS-II. If there is data in the RAM buffer, it is lost. (You cannot restart TERMINAL and recover lost data.)

R RAM Buffer -- (interactive terminal mode only) Lets you save in RAM some or all the data received by "opening" and "closing" the RAM buffer. Later, you can use the D command to examine the data. Or, you can use the C command to save the data in a disk file.

When you open this area of memory known as the RAM buffer, you can either reset it or retain its current contents. If you retain the contents, new incoming text is loaded after existing text.

To switch the RAM buffer option (open or closed), type R <ENTER>. The new state of the option is displayed. If you have just opened the buffer, you receive the following prompt:

```
RAM Buffer Now Open
Reset RAM Buffer? (Y/N) ..
```

If you type Y <ENTER>, the buffer resets and existing contents are lost. For more information, see "Using the RAM Buffer."

S Perform Library Command -- Enters TRSDOS-II and lets you enter a library command. After the command executes, control returns to TERMINAL's Menu. A few TRSDOS-II commands and programs automatically return to TRSDOS-II Ready. If you execute any of these commands while in TERMINAL, control returns to TRSDOS-II Ready, not to TERMINAL.

T Enter Terminal Mode -- Directly enters the interactive terminal mode. When in this mode, press <BREAK> to return to the menu.

V Video Filter -- Filters out data characters that produce undesirable results when output to the display.

When a character such as ESC (X'1B') is output, it causes TERMINAL to clear the screen and home the cursor. With the video filter on, you can prevent this by "filtering" these characters from the display. If the RAM buffer is open, they are saved in RAM, regardless of whether the V option is on or off.

The codes (given in hexadecimal) that this option filters are:

```
Ø1,Ø2,Ø3,Ø4,Ø5,Ø6,Ø7,ØB,ØC,ØE,ØF,
1Ø,11,12,13,14,15,16,1E,1F
```

If TERMINAL receives any of these characters while the video filter is on, it displays a "+" in the character's place. To switch the option (on or off), type V <ENTER>. The new state of the option is displayed, and the menu prompt returns.

W Set/Change Prompt-Wait Character -- Sets a special character as the prompt-wait character to cue the terminal to continue the transmission.

This lets you use the high-speed transmit from RAM mode, even when the host program can accept only one line at time. (It does not affect operation in the interactive terminal mode.)

Normally, the host program sends a prompt, such as a question mark, when it is ready for the next line. (A line is defined as a string of characters ended by a carriage return X'ØD'.) In the interactive terminal mode, you simply wait until this prompt is displayed. The prompt-wait feature makes TERMINAL do the same thing while in the transmit from RAM mode.

You can define the prompt-wait character as any keyboard character from X'2Ø' to X'7F'.

Leave the prompt-wait feature off when the host program is storing characters as received and is not sending a ready-for-next-line prompt. TERMINAL transmits text from RAM in a continuous stream.

Note: When you start the transmit from RAM (X command) or auto sign-on (O command), the first line is sent immediately, without waiting for a prompt. After that, each line is sent after the prompt is received.

To turn off the prompt-wait option, press <HOLD> when the program asks for a new character.

X Transmit RAM Buffer and Enter Terminal Mode -- Enters the transmit from RAM mode where it sends the current contents of the RAM buffer to the host program. When the entire buffer has been sent, TERMINAL goes into the interactive terminal mode. For details, see "Transmitting from RAM."

To stop transmitting from RAM, press <BREAK>. Control returns to the menu.

Using the RAM Buffer

You can use the RAM buffer to store incoming text (R command) and prepared text from a disk file (G command) so TERMINAL can send it quickly. The RAM buffer reduces costly hookup time by letting you perform time-consuming operations -- such as preparing data or printing it out -- while TERMINAL is off-line.

If the buffer is filled during a load from disk (G command) or while receiving data in the interactive terminal mode, a warning message is displayed and the buffer is closed. If you are loading a disk file, control returns to the menu mode and the buffer is filled with the data that was loaded.

If you are in the interactive terminal mode, normal I/O continues, but it is no longer saved in the buffer.

Saving the RAM Buffer

When the buffer is filled in the interactive terminal mode -- or when you think it is almost full:

1. Transmit a pause or break control character to the host program.
2. Press <BREAK> to return to the menu.
3. Use the C command to copy the contents of the RAM buffer to a disk file.
4. Reset the RAM buffer with the R command.
5. Use the T command to return to the interactive terminal mode.

Opening and Closing the RAM Buffer

To save portions of the text during I/O of the interactive terminal mode, use the R command. Prior to receiving the data you want to save:

1. Transmit a pause or break control character to the host program.

2. Press <BREAK> to return to the menu.

3. Use the R option to switch the RAM buffer status. If it is closed, open it. If it is open, you may reset it or leave it. To add new data onto the end of old data, do not reset it. To delete old data, reset it.

4. Use the T command to return to the interactive terminal mode.

5. Direct the host program to resume transmission. The data is saved in the RAM buffer as it is received.

Saving the Options You Have Selected

You can save these commands in a customized version of TERMINAL:

- Prompt wait and definition of prompting character.
- Definition of break character or sequence from host program and assignment of a break key on your computer.
- <F1> and <F2> characters.
- Line feed.
- Printer.
- Self echo.
- Video filter.

- Auto sign-on.
- Speed of transmit from RAM and auto sign-on (as set by the "up"and "down" keys). Once you learn the maximum rate of transmission the host program can handle, you can set that as the default rate.

After you select the options for your customized version, use the DUMP command to create a new program file. TERMINAL resides from X'3ØØØ'. Give this customized program a name other than TERMINAL, and leave the TERMINAL program in its original form.

For example, to call your customized version MINE, answer the prompts as follows:

Prompt	Type
— Enter Menu Selection ..	S <ENTER>
Enter TRSDOS Command (1-79)	DUMP MINE START=3ØØØ, END=3FFF <ENTER>

Now you have a customized version of TERMINAL that starts up when you type (at TRSDOS-II Ready):

MINE <ENTER>

Sample Uses

To Send a Program -- If you intend to send a program (or any data) via TERMINAL, you must first store it in an ASCII-formatted disk file. When you have done this, set up the modem and initialize Serial Channel A as explained previously. See your modem manual for the appropriate procedure for getting on-line.

For example, to send a disk file named SORTDATA on Drive 1, get on-line and load the TERMINAL program. Enter the interactive terminal mode (at the Enter Menu Selection prompt) by typing:

T <ENTER>

Go through the necessary sign-on, and when you want to send the program, press <BREAK> to return to the menu. (If you want to use the prompt wait option, select it now.) Then answer the prompts as follows:

Prompt	Type
-- Enter Menu Selection ..	G <ENTER>
Enter Filespec (1-34)	SORTDATA:1 <ENTER>

TERMINAL loads the program into RAM. Make sure the host is ready to receive the program, then (at the Enter Menu Selection prompt) type:

```
X  <ENTER>
```

The terminal now sends the program to the host. Press <BREAK> if you want to stop transmission for any reason, control returns to the menu. Otherwise, upon completion of the program transmission, control goes into the interactive terminal mode.

To Receive a Program -- If the host program with which you are communicating is ready to send you an ASCII-formatted program, you must first go to the menu. At the Enter Menu Selection prompt, type:

```
R <ENTER>
```

If the buffer is closed, repeat this command and TERMINAL displays the message:

```
RAM Buffer Now Open
Reset RAM Buffer (Y/N) ..
```

Type:

```
Y  <ENTER>
```

This opens and clears the buffer. Use the T command to return to the interactive terminal mode. Then, tell the host to send the program.

After you receive the program, press <BREAK> to return to the menu; then answer the prompts as follows:

Prompt	Type
-- Enter Menu Selection ..	C <ENTER>
Enter Filespec (1-34)	NEWPROG <ENTER>

This copies the program in RAM into a disk file named NEWPROG.

Error Conditions

In the interactive terminal mode, transmit from RAM mode, or during auto sign-on, TERMINAL may detect errors related to the serial transmission. In such cases, it displays an error message in reverse video and, if possible, continues normal I/O.

The error messages that can occur while you are in the interactive terminal mode are:

P Parity error -- The received character is displayed after the P.

O Over-run -- At least one character has been received but not picked up by TERMINAL. This occurs if you are in the menu mode while the host program is sending characters.

F Framing error -- The received character is displayed after the F. Check your SETCOM parameters to see that they match the requirements host program.

The following errors can occur in any mode except the menu:

DATA CARRIER LOST -- Check the telephone/modem connection. TERMINAL pauses until the carrier is restored.

DATA CARRIER RESTORED -- If TERMINAL was transmitting from RAM or sending an auto sign-on, it starts over at the beginning of the text after data carrier is restored.

BREAK SEQUENCE RECEIVED -- If the host program sends a break sequence, or sends TERMINAL'S own break character, this message is displayed. If TERMINAL is in the transmit from RAM or auto sign-on mode, it pauses until the next character is received from the host.

TIME **Library Command**

TIME {hh.mm.ss}

Lets you reset the time or display the date and time. hh is a 2-digit hour specification; mm is a 2-digit minute specification; ss is a 2-digit second specification. If you omit .ss, TRSDOS-II uses .ØØ.

If you type TIME <ENTER>, TRSDOS-II displays the current date and time. The following is a sample display:

```
Thu Mar 25 1982 84 -- 14.15.31
```

for Thursday, March 25, 1982, the 84th day of the year, 2:15:31 p.m.

You first set the time when you start up TRSDOS-II. Once you set the time, TRSDOS-II updates the time and date automatically, using its built-in clock and calendar.

If the time passes 23.59.59, TRSDOS-II does not start over at ØØ.ØØ.ØØ. Instead, it continues with 24.ØØ.ØØ. However, the next time you use the TIME or DATE command, TRSDOS-II converts the time to its correct 24-hour value and updates the date. If you let the clock run past 59.59.59, it re-cycles to zero, and TRSDOS-II does not update the date to include the 6Ø-hour period.

Examples

```
TIME <ENTER>
```

displays the current date and time.

```
TIME 13.2Ø.ØØ <ENTER>
```

resets the time to 1:2Ø:ØØ p.m.

```
TIME 18.24 <ENTER>
```

resets the time to 6:24:ØØ p.m.

VERIFY **Library Command**

VERIFY {switch}

Reads after each write operation to verify that the data is readable. If it is not readable, TRSDOS-II again tries to write and then read it. If it still is not readable, TRSDOS-II returns an error message, telling you the operation is not successful.

The switch options are:
ON sets VERIFY on
OFF sets VERIFY off

The switch is optional. If you omit it, TRSDOS-II returns the current status of VERIFY.

TRSDOS-II starts up with VERIFY on. For most applications, you should leave it on.

TRSDOS-II always verifies directory writes. It verifies user writes (writes of data into a file), however, only when VERIFY is on.

Examples

VERIFY ON <ENTER>

reads after each write to verify that the data is readable.

VERIFY OFF <ENTER>

turns off the verify function.

VERIFY <ENTER>

displays the status of the verify switch.

TECHNICAL INFORMATION

Introduction

This section explains TRSDOS-II on a technical level. It teaches you how to use TRSDOS-II system routines within your own Z-8Ø assembly language programs. You also may find the information useful in programming with BASIC or other languages.

Memory Requirements

TRSDOS-II resides on your primary drive; it occupies only a small portion of memory at any one time. The supervisor program, input/output drivers, and other essentials are always in memory. Auxiliary code is loaded as needed into an "overlay area."

Memory addresses Ø-1Ø239 (X'ØØØØ'-X'27FF') are reserved for TRSDOS-II.

All TRSDOS-II utilities use memory in the range of X'28ØØ' -X'2FFF'.

The following utilities use all of user memory: FORMAT, BACKUP, FCOPY {SYS} and single-drive COPY.

You must locate user programs above X'27FF'. You may want to locate them above X'2FFF', so you can use the utility overlays without losing your program.

Memory Requirements of TRSDOS-II

Decimal Address	Area	Hex Address
Ø-1Ø239	TRSDOS-II Resident Area	ØØØØ-27FF
1Ø24Ø-12287	Utility Overlay Area	28ØØ-2FFF

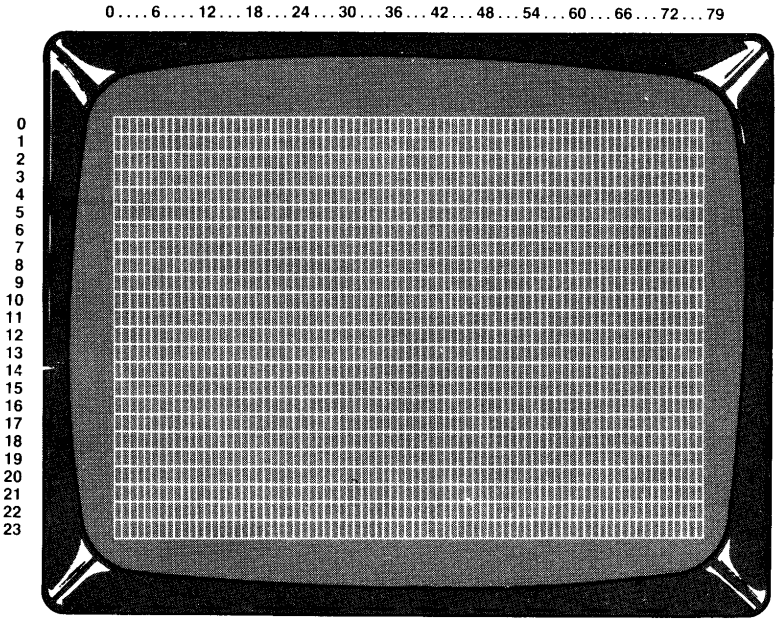
Decimal Address	Area	Hex Address
12288-61439	User Area	3ØØØ-EFFF
61440-65535	TRSDOS-II Demand Resident Area	FØØØ-FFFF

Video Display

The display has two modes of operation -- scroll and graphics. Cursor motion and allowable input characters are different in the two modes.

Graphics Mode

In the graphics mode, the display can be thought of as an 8Ø by 24 matrix, as illustrated below:



DISPLAY POSITIONS, GRAPHICS MODE

Note: The display has two character sizes: 8Ø characters per line and 4Ø characters per line. The above illustration shows the 8Ø characters-per-line mode.

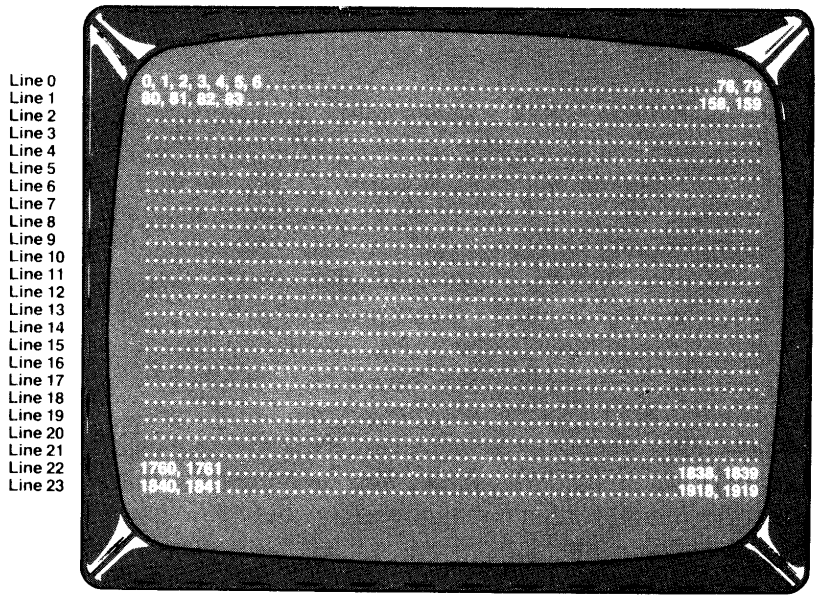
Each time an acceptable character is received, it is displayed at the current cursor position, which is set upon entry to the graphics-mode routines. After the character is displayed, the cursor position is advanced, as follows:

> If the cursor is to the left of Column 79, it advances to the next column position on the same row.
>
> If the cursor is at Column 79, it wraps around to Column Ø on the next row.

Cursor movement works the same way in all directions. For example, if the cursor is at Row 23, Column 4Ø, and TRSDOS-II receives the X'FF' (graphics-down) code, the cursor wraps around to Row Ø in the same column.

Scroll Mode

In the scroll mode, the display can be thought of as a sequence of 192Ø display positions, as illustrated below:



DISPLAY POSITIONS, SCROLL MODE

Note: The display has two character sizes: 8Ø characters per line and 4Ø characters per line. The illustration above shows the 8Ø characters-per-line mode.

Each time an acceptable character is received, it is displayed at the current cursor position. Then, the cursor advances to the next higher numbered position.

When the cursor is on the bottom line, and a line feed or carriage return is received, or the bottom line is filled, the entire display is "scrolled." TRSDOS-II:

1. Deletes Line Ø

2. Moves Lines 1-23 up one line

3. Blanks Line 23

4. Sets the cursor at the beginning of Line 23

Note: You can use the SCROLL SVC to protect Lines Ø to 22 from scrolling.

Diskette Organization

TRSDOS-II can use either single- or double-sided, double-density diskettes. It automatically formats each diskette according to its type.

Each side of the double-sided diskette contains 77 tracks, numbered Ø-76. Therefore, you can think of the double-sided diskette as one diskette with 154 tracks.

The single-sided diskette has 77 tracks on one side only.

The sector is the basic unit of space allocation in TRSDOS-II. Except for Track Ø, each track contains 32 sectors, numbered 1-32. Each sector contains 256 bytes.

TRSDOS-II reserves Track Ø for itself and formats it single-density. Track Ø contains 26 sectors and 128 bytes per sector. On double-sided diskettes, only one side has a single-density Track Ø.

The capacity of a double-sided diskette is:

(153 * 32 * 256) + (1 * 26 * 128) = 1,256,7Ø4 bytes

The capacity of a single-sided diskette is:

(76 * 32 * 256) + (1 * 26 * 128) = 625,92Ø bytes

Single-sided

Diskette	Tracks	Sectors	Bytes
1	76	2,432	622,592
---	1	32	8,192
---		1	256

Double-sided

Diskette	Cylinders	Tracks	Sectors	Bytes
1	77	153	4,896	1,253,376
---	1	2	64	16,384
---	---	1	32	8,192
---	---	---	1	256

Disk Files

Methods of File Allocation

TRSDOS-II gives you two ways to allocate disk space for files: dynamic allocation and pre-allocation.

Dynamic Allocation

With dynamic allocation, TRSDOS-II allocates sectors only when the file is written to. For example, the first time you open a file for output, no space is allocated. The first space allocation is done at the first write. TRSDOS-II allocates more space as required by later writes.

With dynamically allocated files, TRSDOS-II de-allocates (recovers) unused sectors when you close the file.

Pre-Allocation

With pre-allocation, you specify the number of sectors to be allocated. The only way to do this is by using the CREATE command when you create the file.

When the file becomes too large for the space allocated to it, TRSDOS-II dynamically extends (enlarges) it as needed.

It does not, however, de-allocate unused sectors when you close a pre-allocated file. To reduce the size of a pre-allocated file you must copy it to a dynamically allocated file. The COPY command does this automatically when the destination is a dynamically allocated file.

Record Length

TRSDOS-II transfers data to or from a diskette one sector (256 bytes) at a time. These data blocks are called "physical records."

TRSDOS-II transfers data to or from a file in buffers which are from 1 to 256 bytes long. These buffers are called "user records" or "logical records."

TRSDOS-II automatically "blocks" your logical records into physical records to be transferred to the disk. It also automatically "de-blocks" the physical records into logical records, which are used by your program.

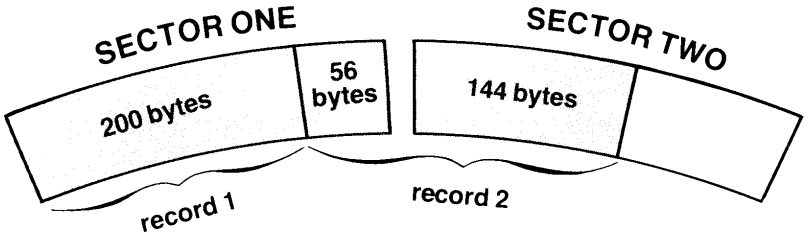
Therefore, your only concern during file access is with logical records. You never need to worry about physical records, sectors, tracks, and so on. This is to your benefit, since physical record lengths and features may change in later TRSDOS-II versions, while the concept of logical records will not.

From this point on, the term "record" refers to a "logical record."

Spanning

If the record length is not an even divisor of 256, the records automatically are spanned across sectors.

For example, if the record length is 2ØØ, Sectors 1 and 2 look like this:



Fixed-Length and Variable-Length Records

TRSDOS-II files can have either fixed-length or variable-length records. Files with fixed-length records are called FLR files; files with variable-length records, VLR files.

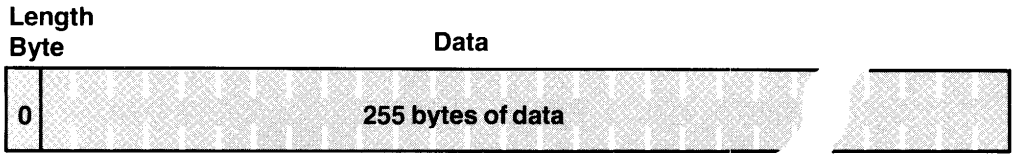
For an FLR file, you set the record length when you open the file for the first time. This length can be from 1-256 bytes. You cannot change the set record length unless you overwrite the file with new data.

For a VLR file, you specify the record length in a 1-byte length field at the start of each record. The lengths of records in a VLR file can vary. For example, the first record in a file might have a length of 32; the second record, 17; the third record, 25Ø; and so on.

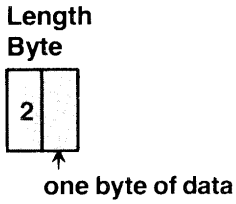
The length byte for a variable-length record indicates the entire length of the record, including the length byte. It can be from Ø-255. You can use 1, but it has no meaning because no record is 1-byte long.

Examples

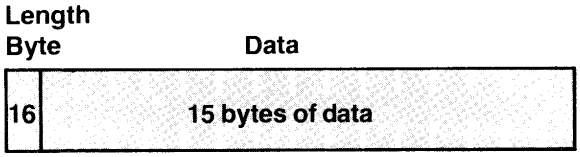
A length byte of Ø indicates that the record has 255 bytes of data:



A length byte of 2 indicates that the record has 1 byte of data:



A length byte of 16 indicates that the record has 15 bytes of data:



Record Numbers

Records are numbered from Ø (beginning of file) to 16,777,214.

A file may contain up to 16,777,216 bytes of storage. Therefore, to determine the number of records a file can hold, use the following formula:

16,777,216 / logical record length = number of records

For example:

16,777,216 / 38 = 441,5Ø5 records.

For a file with a record length of 1, the number of records cannot exceed 16,777,214.

The record number 16777215 (X'FFFFFF') positions to the "end of file" (EOF), regardless of the actual number of records in the file.

Record Processing Capabilities

TRSDOS-II allows both direct and sequential file access.

Direct access, sometimes called "random access," lets you process records in any order you specify.

To get direct access to Records Ø-65533, you simply specify the record number (when using one of the direct access SVCs).

To get direct access to Records 65534-16777216, however, you must choose the extended file access mode when you open the file. This mode specifies the record number in three bytes, letting you use the large number.

The direct access SVCs are DIRRD (direct-read) and DIRWR (direct-write).

Sequential access lets you process records in the order in which they are stored in the file. TRSDOS-II automatically accesses the record that follows the last one processed. When you first open the file, it accesses Record Ø.

FLR files can be opened by either direct access or sequential access.

VLR files can be opened only by sequential access. You cannot position to a specific record, since the varying record lengths make it impossible to calculate the record's position.

The sequential access SVCs are READNX (read-next) and WRITNX (write-next). When you first open the file, sequential processing starts with Record Ø. However, you can use DIRWR and DIRRD to position to the EOF.

Examples with FLRs

Assume you have an open FLR file. Here are some typical sequences you can do via the file processing routines:

- **Read and/or write records in the file -- in any order.**
  Using DIRRD and DIRWR, you can: read Record 5, write at the EOF, read Record 3, write Record 3, and so on.

- **Read or write sequentially, starting anywhere in the file.**
  Do a DIRRD to the record at which you want to start. Then, do READNX or WRITNX until done.

- **Write sequentially, starting at EOF.**
  Do a DIRWR to the EOF. Then, do WRITNX until done.

- **Learn the number of records in a file.**
  Do a DIRRD to the EOF. Then, use the LOCATE routine to get the current record number. This equals the number of records + 1.

Examples with VLRs

Assume you have an open VLR file. Here are some typical sequences you can do via file processing routines:

- **Read or write sequentially, starting at the first record.**
  Do READNX and WRITNX until done.

- **Write sequentially, starting at the EOF.**
  Do a DIRWR to the EOF. Then, do WRITNX until done.

Note: Whenever you write to a VLR, the last record you write becomes the end of file (EOF), automatically.

You cannot update a VLR file directly. You must read in the file and output the updated information to a new VLR file.

Supervisor Calls

Supervisor calls (SVCs) are operating system routines available to any user program. They alter certain system functions and conditions; let you access files; do input/output to the keyboard, video display, and printer; and do various computations.

The available TRSDOS-II SVCs are:

KEYBOARD SVCs
KBCHAR
KBINIT
KBLINE
KBPUT
VIDKEY

VIDEO SVCs	
CURSOR	VDINIT
SCROLL	VDLINE
VDCHAR	VDREAD
VDGRAF	VIDRAM

PRINTER SVCs
PCTRL
PRCHAR
PRINIT
PRLINE

COMMUNICATIONS SVCs	
ACTRL	BCTRL
ARCV	BRCV
ATX	BTX
	RS232C

SYSTEM CONTROL SVCs	
CLRXIT	HLDKEY
DATE	INITIO
DELAY	JP2DOS
DOSCMD	RETCMD
ERRMSG	SETBRK
ERROR	SETUSR
	TIMER

FILE ACCESS SVCs	
CLOSE	LOCATE
DIRRD	OPEN
DIRSET	RAMDIR
DIRWR	RDDIR
DISKID	READNX
FILPTR	RENAME
KILL	REWIND
	WRITNX

COMPUTATIONAL SVCs
BINDEC
BINHEX
MPYDIV

MISCELLANEOUS SVCs	
LOOKUP	SORT
PARSER	STCMP
RANDOM	STSCAN
SOUND	WILD

Note: Appendices C and D, the SVC quick reference lists, summarize each SVC's function.

Each SVC has a function code that you use to call it. These codes range from Ø through 127. The first 96 codes (Ø-95) are defined by TRSDOS-II. Codes 96-127 are available for you to define (see the SETUSR SVC).

SVC Notation

In this section we use the following notations:

Notation	Meaning
RP = data	Register Pair RP contains the data
n1 < R < n2	The register pair contains a value greater than n1 and less than n2
(RP) = data	Register Pair RP contains the address of ("points to") the data
NZ = Error	If the Z flag is not set, an error occurred

When a range is not given in the description of the SVC, you can use any representable number. For example, a register can contain any value from Ø-255.

Calling Procedure

All SVCs are executed via the **RST 8** instruction.

1. Load the function code of the desired SVC into Register A. Also load any other registers that the SVC needs. See "Supervisor Calls."

2. Execute an RST 8 instruction.

3. If the function is successful, the Z flag is set upon return from the SVC. If the Z flag is not set, there is an error. Register A has the appropriate error code (except after certain computational SVCs, which use Register A to return other information).

During the execution of an SVC, the SVC processor does not alter any memory above X'2FFF'. Only those Z-8Ø registers used to pass parameters from the SVC are changed. However, all the prime registers are used by TRSDOS-II; they are not restored.

Examples

Time-Delay

```
      LD    BC,TIMCNT            ; LENGTH OF DELAY
      LD    A,6                  ; FUNCTION CODE 6 = DELAY-SVC
      RST   8                    ; JUMP TO SVC
;   DELAY OVER-PROGRAM CONTINUES HERE
```

Output a Line to the Video Display

```
      LD    HL,MSG               ; POINT TO THE MESSAGE
      LD    B,1Ø                 ; B = CHARACTER COUNT
      LD    C,ØDH                ; C=CTRL CHAR. TO ADD AT END
      LD    A,9                  ; CODE 9 = DISPLAY LINE-SVC
      RST   8                    ; JUMP TO SVC
      JR    NZ, GOTERR           ; JUMP IF I/O ERROR
;   IF NO ERROR THEN PROGRAM CONTINUES HERE
```

Get a Character from the Keyboard

```
GETCHAR  LD     A,4              ; CODE 4 = GET CHARACTER-SVC
         RST    8                ; JUMP TO SVC
         JR     NZ,GETCHAR       ; DO AGAIN IF NO CHARACTER
;   CHARACTER IS IN REGISTER B
```

Using the Serial Communications SVCs

Initialization

Before doing any serial I/O, you must initialize the channel you want. Use either the SETCOM library command or the RS232C SVC.

Input/Output

All serial I/O is character-oriented, similar to keyboard input and video output. The main difference is that your program must check the communications status at various times. This is done to ensure that the communications link is in place and that data is not being lost.

After any attempted serial I/O, TRSDOS-II returns the communications status to your program in Z-8Ø Register A. The bits in the register are used individually to show the on/off status of various conditions. Certain bits apply to transmit operations; others apply to all serial I/O.

Programming with TRSDOS-II

This section tells you how to execute your own machine-language programs under TRSDOS-II. It has two sub-sections:

- **Program Entry Conditions** -- How control is transferred to your program after the program is loaded from the disk
- **Handling Programmed Interrupts** -- How to write an interrupt service routine for <BREAK> key processing and TIMER interrupts

To create and use a program, follow this procedure:

1. Enter the program into memory, using DEBUG, or via the serial interface channel from another device, or using an assembler and linker.
2. If you use DEBUG or an I/O channel to load the program, use the DUMP command to save the program as an executable disk file, setting the load and transfer addresses.
3. To run the program, input the filename to the TRSDOS-II command interpreter (TRSDOS-II Ready mode).

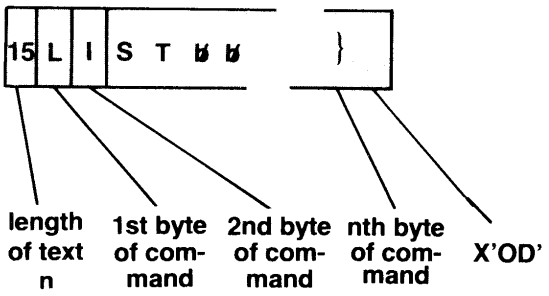
Program Entry Conditions

Upon entry to your program, TRSDOS-II sets up the following registers:

BC = address of the first byte following your program (the first free byte for use by your program).
DE = highest unprotected memory address (the end of memory that can be used by your program).

HL = address of the buffer containing the last command entered to the TRSDOS-II command interpreter.

The first byte of the buffer contains the command line length, excluding the carriage return. The text of the command follows this length byte. For example:



Handling Programmed Interrupts

TRSDOS-II allows two user-programmed interrupts as described under SETBRK and TIMER. When either kind of interrupt is received (you press the <BREAK> key or the TIMER counts to zero), control transfers to your interrupt handling routine.

Note: System routines called by your program or called by interrupt handlers can be interrupted, also.

Upon entry to your interrupt processing routine, TRSDOS-II sets up the registers as follows:

(SP) = address of the next instruction to be executed when the interrupt was received.

Other registers:
Contents are the same as they were when the interrupt was processed.

Before doing any processing, you should save all registers. When you finish processing, restore all registers and execute a return to continue with the interrupted program.

Keep the interrupt handling routines short. Ideally, the routine simply flags the main program that an interrupt has

occurred, and then returns. The main program then can respond to the interrupt flag when convenient.

Always end your interrupt handler with the RET instructions and with all registers intact.

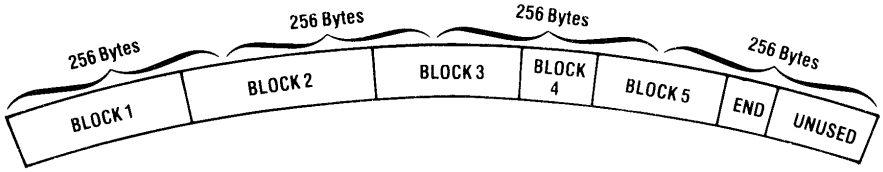
TRSDOS-II is serially reusable but not always re-entrant. Specifically, your interrupt routine should not call TRSDOS-II SVCs, since under some conditions this causes unpredictable results.

Program Files

This section describes the required format and structure of program files. In it you'll also find a summary of the procedure for writing a program file using TRSDOS-II file-access SVCs.

Program File Format

A program file is stored on the diskette in blocks, which might look like this:



The lengths of the blocks vary, depending on the type of block and the amount of information in it. The blocks must border each other. There can be no unused bytes between them.

There are three major types of blocks:

- **Program data blocks** -- These contain the actual program data, prefixed by four bytes of header information.
- **Comment blocks** -- These contain documentation for the programmer. Comment blocks are not loaded or examined by the loader. Comment blocks are prefixed by two bytes of header information.

- **Trailer blocks.** Each program file ends with a trailer block. It marks the end of file (EOF) and tells TRSDOS-II what to do after the file has been loaded -- either to transfer to a specified address or to return to the caller. Trailer blocks are four bytes long.

The first byte in a block identifies the block type, as follows:

Contents (Hex)	Block Type
Ø1	Program data
Ø5	Comment block
Ø2	Trailer block; jump after loading
Ø3	Trailer block; return to the caller after loading
ØØ, Ø4, Ø6 - FF	Reserved for TRSDOS-II use; the loader fails if one of these codes is used

Structure of Program Data Blocks

Program data blocks consist of the following:

Byte #	Contents
1	Block identifier (equals binary 1)
2	Length (the number of bytes of program data plus two for the load address)
3-4	Load address (where the following program data starts loading into RAM); it must be in the LSB-MSB format
5-end	Program data

The length byte gives the number of bytes in the rest of the block -- following the 2-byte load address. This sum may range from 3 (the 2-byte address plus 1 byte of data) to 258 (the 2-byte address plus 256 bytes of data). You must translate the sum into the range Ø-255.

To do this, take the number of **program** bytes and increment by two. Note that values greater than 255 "wrap around" to Ø, 1, and 2. Here is a table:

Number of Bytes after Address (program data only)	Value for Length Byte
1	3
2	4
3	5
...	...
253	255
254	Ø
255	1
256	2

The load address tells TRSDOS-II where in RAM to load the data in the block. TRSDOS-II loads the bytes serially. The load address must allow the entire block of program data to load in the "user" area of RAM.

Structure of Comment Blocks

Comment blocks consist of the following:

Byte #	Contents
1	Block identifier (equals binary 5)
2	Length (the number of bytes in the comment)
3-end	Comment

The length byte gives the number of bytes in the comment (the number of bytes after the length itself). This sum ranges from Ø-255.

Length Byte	Comment Length
Ø	Ø
1	1
2	2
...	...
255	255

Structure of Trailer Blocks

Each program file ends with a single trailer block that tells TRSDOS-II what to do next:

Jump to a specified "transfer" address
or
Return to the caller

Trailer blocks consist of the following:

Byte #	Contents
1	Block identifier (equals binary "2" or "3")
2	Length (equals 2)
3-4	Transfer address; it must be in the LSB-MSB sequence and must be in the user area of RAM

If the block identifier equals 2, TRSDOS-II jumps to the transfer address after loading the program. If it equals 3, TRSDOS-II does not jump to the transfer address. Instead, it returns to the caller after loading the program, with the following prime registers set:

HL' = transfer address taken from the trailer block
DE' = address of the last usable byte
BC' = first byte following the program just loaded

Procedure for Writing a Program File

Program files must have fixed-length records of length 256 and must have the P (program) access type. You set both of these when you create the file, using a creation code of 1 or 2 (see the OPEN SVC).

The TRSDOS-II program loader treats the program file as a serial byte stream, independent of the record boundaries. The loader assumes the program blocks are "back-to-back" (there are no unused bytes between blocks). As programmer, you are responsible for:

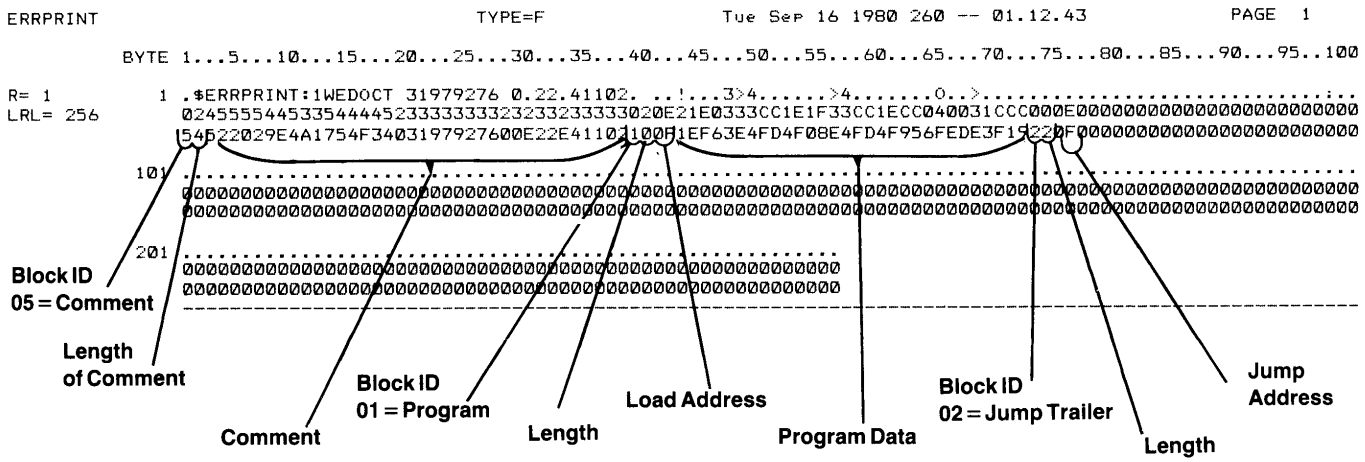
1. Storing the correct byte sequence in the logical record area.

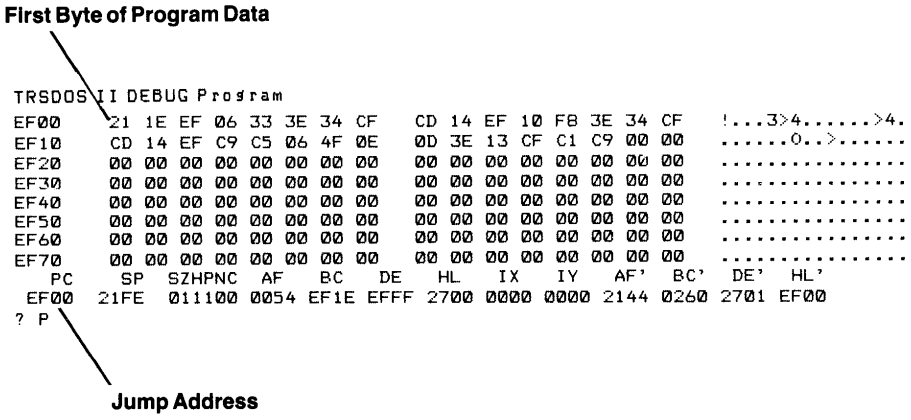
2. Calling the disk write SVC (DIRWR) when each 256-byte record is filled. Blocks can and must "span" sectors. The only unused bytes in the file are after the trailer block. See the illustration below.

TRSDOS-II does not load user program below "user RAM." You also should make sure blocks do not load above "user RAM."

Note: The TRSDOS-II DUMP command always writes a comment as the first block in the file. The comment contains the filespec, followed by the current TRSDOS-II time/date text.

Illustrated Program File Listing





High-Memory Modules: Their Addresses and Uses

As mentioned earlier, there are five high-memory modules that are conditionally loaded. They are DO, SPOOL, HOST, DEBUG and the communications drivers. They reside in high memory (above X'FØØØ'). The approximate memory map is as follows:

Module	Address
Comm drivers	X'FØØØ'
DEBUG	X'F4ØØ'
HOST*	X'F4ØØ'
SPOOL*	X'F64Ø'
DO*	X'FC8Ø'

* Cannot be used with DEBUG

Addresses may change in future releases.

If your application requires a large amount of RAM, you may need to use high memory. For example, you may have a large BASIC program or a machine-language subroutine that you must link to BASIC. In either case, you might need a full RAM for the BASIC program itself. You must use high memory. If you **do** use high memory, keep in mind the following:

- **If you use BASIC, the use of high memory gives you slightly less RAM for your program and variable storage than you had with TRSDOS.**

  Upon program-load, TRSDOS-II gives X'EFFF' in Register DE as the high-memory address.

  If you must, you may override this limit by specifying this command line:

  BASIC - M:upper limit of high-RAM address

  The address to use as the upper limit must never exceed decimal 63487. The same rule applies to an applications program.

- **The use of high RAM prevents the use of certain high-memory modules.**

Assume that you invoke a system function that loads high RAM with a module. Then, your application loads something on top of it. (TRSDOS-II cannot prevent you from overlaying an already loaded high-memory module.) You are going to have problems. (See the memory map above for details.)

Be careful about overlaying the addresses of the following modules:

- **Communications drivers** -- If you use a serial printer, do not overlay X'FØØØ'-X'F3FF'.

  The code in that area must be intact if your application has printed output. That is when the comm drivers are active. Overlaying in this area can cause serious problems.

- **SPOOL** -- This is another module that can be used by almost any application that has printed output. To give operator flexibility, you should not use the RAM area of X'F64Ø'-X'F8ØØ' in your application software.

  If you must use this RAM area, make it clear to the operator that he may not use SPOOL. If any of the spool code space is overlayed after TRSDOS-II has loaded the spool code in that area, serious problems will result.

- **HOST** -- Lets a remote terminal execute your applications and programs as if the remote were local (the local keyboard and video). Again, using this space limits the operator's flexibility and causes serious problems.

You **can** use the high-memory space without causing problems, **if** you plan carefully. Know what you need and when you need it. Then, tell the operator what to do (or not to do) when running your application.

For example, assume that your application program does not create printed output. In this case, you do not need the spool or comm drivers areas (if the comm drivers are needed just for serial printer support).

To use this space, make sure SPOOL and the comm drivers are not active before or during the run of the application. Do this by starting execution of the application with a DO file (which resides in the top of RAM).

In this DO file, you may turn off SPOOL and the comm drivers before the application is executed. Remember that you must turn off both Serial Channels A & B for the high-memory comm drivers to be inactive.

For full operational flexibility, you should stay away from the high-memory areas. Then, you can use all of the high-memory routines during program execution.

Future releases of TRSDOS-II may provide more features or routines that will use high memory. Keep this in mind when you design your applications. To use these features, you may have to update your high-memory subroutines. Consider the efforts required to make these program changes. Your program may be hard to update, or it may be hard to get revised copies out to the users.

ACTRL **Function Code 1ØØ**
Control Channel A

Performs control functions on Serial Channel A.

TRSDOS-II sets up ARCV, ATX, and ACTRL when you initialize Channel A with RS232C. If you call any of these routines while the channel is not initialized (active), you get an error return code of 1 (no function code exists).

Entry Conditions

A = 1ØØ
B = option switch

Exit Conditions

NZ = Error
B = status code

The option switch settings are:

option switch	Result
Ø	Gets the current status of the serial I/O (SIO) channel into Register B.
1	Gets the current character count in the serial receive buffer into B.
2	Turns on Request to Send (RTS). RTS is on when the channel is initialized.
3	Turns off RTS.
4*	Starts transmitting the break sequence.
5	Stops transmitting the break sequence.
6	Resets the serial receive buffer count to zero.
7	Resets the SIO error condition.

* Before transmitting a break sequence, make sure the sure the transmit buffer is empty. As the programmer, you are responsible for sending the break sequence for the appropriate time period, as required by the application or host computer.

The 8-bit grouping of flags returned in Register B consists of:

Bit	Meaning
Ø	Clear to Send (CTS) is not present
1	Unused
2	Transmitter is busy
3	Modem data carrier is not present
4	Parity error is occurring in the byte that is being received
5	Data overflow is occurring because of a byte that is being received
6	Framing error is occurring in the byte that is being received
7	Break sequence (extended null character) is being received

ARCV **Function Code 96**
Channel A Receive

Returns a single character from Serial Channel A.

TRSDOS-II sets up ARCV, ATX, and ACTRL when you initialize Channel A with RS232C. If you call any of these SVCs while the channel is not initialized, you get an error return code of 1 (no function code exists).

Input Buffer

Each channel (A and B) has an internal receive buffer to reduce overruns when receiving data at high speeds. This buffer is a first-in, first-out buffer (FIFO) which is established when the channel is initialized.

When a character is received, the character and the hardware status code are placed in the Channel A buffer. If Channel A has not been initialized, no characters are received.

An overrun occurs when a character is received and there is no more room in the buffer. The character that caused the overrun replaces the last character received. Bit 5 of the hardware status code is set; this indicates that an overrun occurred.

When the ARCV SVC is executed, the oldest character in the buffer is returned to Register B; the hardware status at the time the character was received is returned to Register A.

> Note: When a character that caused an error is returned to Register B, the error is indicated in Register A.

If there are no characters "waiting" in the buffer, the communications status returned in Register A reflects the current status of the serial interface.

Entry Conditions

A = 96

Exit Conditions

A = status code
B = character returned, if any
NZ = Character was not received, check the status code
C flag = Modem carrier was not present when you entered the SVC

The 8-bit grouping of flags returned in Register A (after serial input) consists of:

Bit	Meaning
Ø	Not used
1	Not used
2	Not used
3	Modem carrier was not present
4	Parity error occurred on the last character received in Register B
5	Data is lost because of an overflow; Register B contains the last character
6	Framing error occurred on the last character received
7	Break sequence (extended null character) was received

ATX **Function Code 97**
Channel A Transmit

Outputs a single character to Serial Channel A.

TRSDOS-II sets up ATX, ARCV, and ACTRL when you initialize Channel A with RS232C. If you call any of these SVCs while the channel is not initialized, you get an error return code of 1 (no function code exists).

TRSDOS-II transmits data bytes even if no carrier is present. You must check the status flags and Register A for error conditions. "Carry flag set" (C flag) means that no carrier is present during transmission. You may ignore this flag if your application does not need the carrier to be present.

Register A contains status information after serial output (upon return from ATX).

Entry Conditions

A	=	97
B	=	character to be sent, (ASCII code)

Exit Conditions

A	=	status code
NZ	=	Character is not transmitted, check the status code
C flag	=	Modem carrier was not present when you entered the SVC

The 8-bit grouping of flags returned in Register A (after serial output) consists of:

Bit	Meaning
Ø	Clear to Send (CTS) was not present
1	Not used
2	Transmitter busy
3	Modem carrier was not present
4*	Parity error on the last character received
5*	Data is lost because of a hardware-level overflow

Bit	Meaning
6*	Framing error occurred on the last character received
7*	Break sequence (extended null character) was received

* TRSDOS-II returns bits 4 through 7 only when it does not send the character.

BCTRL — Function Code 1Ø1
Control Channel B

Performs control functions on Serial Channel B.

TRSDOS-II sets up BRCV, BTX, and BCTRL when you initialize Channel B with RS232C. If you call any of these SVCs while the channel is not initialized, you get an error return code of 1 (no function code exists).

Entry Conditions

A = 1Ø1
B = options switch

Exit Conditions

NZ = Error
B = status code

Valid option switch settings are:

option switch	Result
Ø	Gets the current status of the serial I/O channel into Register B.
1	Gets the current character count in the serial receive buffer into B.
2	Turns on Request To Send. RTS is on when channel is initialized.
3	Turns off RTS.
4*	Starts transmitting the break sequence.
5	Stops transmitting the break sequence.
6	Resets the serial receive buffer count to zero.
7	Resets the SIO error condition.

* Before transmitting a break sequence, make sure the sure the transmit buffer is empty. As the programmer, you are responsible for sending the break sequence for the appropriate time period, as required by the application or host computer.

The 8-bit grouping of flags returned in Register B consists of:

Bit	Meaning
Ø	Clear to Send (CTS) is not present
1	Unused
2	Transmitter is busy
3	Modem data carrier is not present
4	Parity error is occurring in the byte that is being received
5	Data is lost because of an overflow caused by a byte that is being received
6	Framing error occurred in the byte that is being received
7	Break sequence (extended null character) was received

BINDEC **Function Code 21**
Binary/Decimal

Converts a 2-byte binary number to an ASCII-coded decimal, and vice versa. The decimal range is from Ø to 65535.

Entry Conditions

A = 21
B = function switch
B = Ø: convert binary to ASCII decimal
B ≠ Ø: convert ASCII decimal to binary

If B = Ø (binary to decimal):
DE = 2-byte binary number to convert
(HL) = 5-byte area to contain the ASCII decimal value
The field contains decimal digits in the range [X'3Ø',X'39'], with leading zeroes on the left as needed to fill the field. For example, ØØØ21 represents 21.

If B ≠ Ø (decimal to binary):
(HL) = 5-byte area containing the ASCII decimal value to convert

Exit Conditions

(HL) = decimal value
DE = binary value

BINHEX **Function Code 24**
Binary/Hexadecimal

Converts a 2-byte binary number to an ASCII-coded hexadecimal, and vice versa. The hexadecimal range is from X'ØØØØ' to X'FFFF'.

Entry Conditions

A = 24
B = function switch
B = Ø: convert binary to ASCII hexadecimal
B ≠ Ø: convert hexadecimal to binary

If B = Ø (binary to hexadecimal):
DE = 2-byte binary number to convert
(HL) = 4-byte area to contain the ASCII hexadecimal value
The field contains hexadecimal digits with leading zeroes on the left, as needed to fill the field. For example, ØØFF represents the number X'FF'.

If B ≠ Ø (hexadecimal to binary):
(HL) = 5-byte area containing the ASCII hexadecimal value to convert

Exit Conditions

(HL) = hexadecimal value
DE = binary value

BRCV Function Code 98
Channel B Receive

Returns a single character from Serial Channel B.

TRSDOS-II sets up BRCV, BTX, and BCTRL when you initialize Channel B with RS232C. If you call any of these SVCs while the channel is not initialized, you get an error return code of 1 (no function code exists).

Input Buffer

Each channel (A and B) has an internal receive buffer to reduce overruns when receiving data at high speeds. This buffer is a first-in, first-out buffer (FIFO) which is established when the channel is initialized.

When a character is received, the character and the hardware status code are placed in the Channel B buffer. If Channel B has not been initialized, no characters are received.

An overrun occurs when a character is received and there is no more room in the buffer. The character that caused the overrun replaces the last character received. Bit 5 of the hardware status code is set; this indicates that an overrun occurred.

When the BRCV SVC is executed, the oldest character in the buffer is returned to Register B; the hardware status at the time the character was received is returned to Register A.

> Note: When a character that caused an error is returned to Register B, the error is indicated in Register A.

If there are no characters "waiting" in the buffer, the communications status returned in Register A reflects the current status of the serial interface.

Entry Conditions

A = 98

Exit Conditions

A = status code
B = character returned, if any
NZ = Character not received; check the status code
C flag = Modem carrier was not present when you entered the SVC

The 8-bit grouping of flags returned in Register A (after serial input) consists of:

Bit	Meaning
Ø	Not used
1	Not used
2	Not used
3	Modem carrier was not present
4	Parity error occurred on the last character received in Register B
5	Data is lost because of an overflow; Register B contains the last character
6	Framing error occurred on the last character received
7	Break sequence (extended null character) was received

BTX **Function code 99**
Channel B Transmit

Outputs a single character to Serial Channel B.

TRSDOS-II sets up BTX, BRCV, and BCTRL when you initialize Channel B with RS232C. If you call any of these routines while the channel is not initialized, you get an error return code of 1 (no function code exists).

TRSDOS-II transmits data bytes even if no carrier is present. You must check the status flags and Register A for error conditions. "Carry flag set" (C flag) means that no carrier is present during transmission. You may ignore this flag if your application does not "care" whether the carrier is present.

Register A contains status information upon return from BTX.

Entry Conditions

A	=	99
B	=	character to be sent (ASCII code)

Exit Conditions

A	=	status code
NZ	=	Character not transmitted; check the status code
C flag	=	Modem carrier was not present when you entered the SVC

The 8-bit grouping of flags returned in Register A (after serial output) consists of:

Bit	Meaning
Ø	Clear to Send (CTS) was not present
1	Not used
2	Transmitter is busy
3	Modem carrier was not present
4*	Parity error occurred on the last character received
5*	Data is lost because of a hardware-level overflow

Bit	Meaning
6*	Framing error occurred on the last character received
7*	Break sequence (extended null character) was received

* TRSDOS-II uses Bits 4-7 only when it does not send the character.

CLOSE Function Code 42
Close Disk Files

Ends access to the file you specify. This SVC first writes all unsaved data to the disk, then updates the directory before closing the file.

Entry Conditions

A = 42
(DE) = data control block of the currently open file

Exit Conditions

NZ = Error
A = error code

Upon return, the filespec (except for the password) automatically is put back into the DCB.

CLRXIT **Function Code 57**

Clear User Memory and Jump to TRSDOS

Clears (writes binary zeroes to) user memory and gives control to TRSDOS-II.

Entry Conditions

A = 57

Exit Conditions

None

CURSOR **Function Code 26**
Blinking Cursor

Turns the blinking cursor on or off. TRSDOS-II keeps track of the current cursor position, whether the cursor is on or off.

Entry Conditions

A = 26
B = function switch
B = Ø: blinking cursor off
B ≠ Ø: blinking cursor on

Exit Conditions

None

DATE — Function Code 45
Return Date String

Sets or returns the time and date. TRSDOS-II returns the data as a 26-byte ASCII string with 8 fields.

Contents of Time/Date string:

| Fri | Oct | 1| 1982 | 274 | 13.2Ø.42 | 1Ø | 4 |

Name of Day	Mon.	Day of Mon.	Year	Day of Year	Time	Mon. of Year	Day of Week

Represents the data "Friday, October 1, 1982, the 274th day of the year, 1:2Ø:42 p.m., the tenth month of the year, the fourth day of the week."

Monday is considered day Ø. The date calculations are based on the Julian Calendar.

Entry Conditions

A = 45
B = function switch
  B = Ø: get time/date
  B = 1: set date
  B = 2: set time
(HL) = a dependent of the contents of Register B

If B = 1:
(HL) = 1Ø-byte buffer containing the date (The date is in the form mm/dd/yyyy.)

If B = 2:
(HL) = 8-byte buffer containing the time (The time is in the form hh.mm.ss.)

Exit Conditions

NZ = Error
If B = Ø:
(HL) = 26-byte buffer where the time/date are to be stored

DELAY **Function Code 6**
Delay Loop

Provides a delay routine and returns control to the calling program after the time you specify has elapsed.

Background processing, such as the SPOOL printing function, uses this delay time.

Entry Conditions

A = 6
BC = delay multiplier switch
BC = Ø: the delay time is 426 milliseconds
BC ≠ Ø: the delay time is:
6.5 * (BC - 1) + 22 microseconds.

Exit Conditions

None

DIRRD Function Code 35
Direct-Read

Reads the specified record of an FLR file.

Also, you can use DIRRD to read the first record or the EOF (X'FFFF') of a VLR file.

Upon return, your record is in the record area pointed to by RECADR in the parameter list. Or, if the record length is 256, and you are reading a fixed-length record, it is in the area pointed to by BUFADR (see the OPEN SVC for more information).

Entry Conditions

A = 35
(DE) = data control block of the currently open file
BC = record number
BC = Ø: position to beginning of file
BC = X'FFFF': position to end of file
HL = Reserved

If you chose the E mode when you opened the file, then:
BC = address of 3 bytes in RAM The format is:

upper byte	middle byte	lower byte

where ØØ ØØ ØØ = first record of file
and FF FF FF = end of file (EOF)

Exit Conditions

NZ = Error
A = error code

DIRSET **Function Code 59**
Directory Information

Gets directory information on any open file.

DIRSET sets up the 8-byte block of memory that is one of the entry parameters for the RDDIR SVC. After you open the file, you can:

1. Find out which drive contains the file (See the RDDIR SVC).

2. Set up the 8-byte block in user memory (BC) for RDDIR. This lets RDDIR get directory information about the open file.

Entry Conditions

A = 59
DE = address of the open data control block
BC = address of the 8-byte block of user memory

Exit Conditions

All registers return unchanged, except Register A.

BC = address of the 8-byte block of user memory set for RDDIR call to get the directory information of the open data control block (pointed to by DE)
Z = No error
NZ = I/O error
A = error code

Note: Be sure to load Register Pair DE with zero (LD DE,Ø) after executing DIRSET and before executing RDDIR. If DE is non-zero, RDDIR expects a wildcard mask and, in most cases, cannot find a match.

DIRWR **Function Code 44**
Direct-Write

Writes a specified record to a specified position in an FLR file. Also, you can use DIRWR to write the first record (Ø) or the EOF (X'FFFF') record in a VLR file.

Before calling DIRWR, put your record into the record area pointed to by RECADR in the parameter list. Or, if the record length is 256 and the record type is fixed, put the record in the area pointed to by BUFADR (see the OPEN SVC for more information).

Entry Conditions

A = 44
(DE) = data control block of the currently open file
BC = record number
BC = Ø: position to beginning of file
BC = X'FFFF': position to end of file
(HL) = Reserved

If you chose the E mode when you opened the file, then:
BC = address of 3 bytes in RAM The format is:

upper byte	middle byte	lower byte

where ØØ ØØ ØØ = first record of file
and FF FF FF = record after current EOF (EOF +1)

Exit Conditions

NZ = Error
A = error code

DISKID **Function Code 15**
Read Disk ID

Reads the diskette ID(s) from any or all of Drives Ø-7. This is useful when the program must ensure that the operator has inserted the proper diskette.

TRSDOS-II places the diskette ID(s) in the buffer pointed to by Register Pair HL. If a drive is not ready, TRSDOS-II places blanks in the buffer. If you specify 255, TRSDOS-II reads the IDs from all drives. If you have do not have hard disks, the last 32 bytes of the buffer are blank.

Entry Conditions

```
A    = 15
B    = drive select code
       B = Ø-7:  read from specified drive (Ø-7)
       B = 255:  read all disks' IDs
(HL) = disk ID buffer
```

If B = Ø-7:
The buffer must be 8 bytes long.

If B = 255:
The buffer must be 64 bytes long.

Exit Conditions

```
(HL) = disk ID(s)
NZ   = Error
A    = error code
```

DOSCMD Function Code 37

Execute TRSDOS-II Command

Passes a command string to TRSDOS-II Ready mode for execution. After the command is executed, control returns to TRSDOS-II. All open files are closed automatically.

Entry Conditions

A = 37
(HL) = command string
B = length of string

Exit Conditions

None

ERRMSG **Function Code 52**
Error Message

Returns an 8Ø-byte error description to the specified buffer area, in response to the specified error number.

Entry Conditions

A = 52
B = error number
(HL) = 8Ø-byte buffer area above X'27FF'

Exit Conditions

NZ = Error
A = error code

ERROR **Function Code 39**

Display Error Number

Displays the message ERROR, followed by the specified error code, at the current cursor position.

Entry Conditions

A = 39
B = error number

Exit Conditions

NZ = Error
A = error code

FILPTR **Function Code 58**
Get Pointers of a Open File

Gives information on an open user file. If it does not make a match, the SVC returns an error.

Note: We recommend restricting your use of the FILPTR since this SVC is restricted to 96 files. If you are familiar with FILPTR, you should change your existing programs to use RDDIR instead.

Entry Conditions

A = 58
(DE) = data control block of the currently open file

Exit Conditions

Z flag = No error
NZ = Error
B = drive that contains the file (binary Ø-7)
C = position of the file in the diskette directory (binary 1-96)

Notes

This SVC is intended for use with the RAMDIR SVC. After opening a file, you may:

1. Use FILPTR to find out which drive contains the file and which directory record contains the file's information.

2. Use RAMDIR to obtain information about that file (current file space allocated/used, protection level, and so on).

HLDKEY **Function Code 29**

Enable the <HOLD> Key

Suspends and restarts terminal output whenever you press <HOLD>.

This SVC must first be enabled; call the SVC with B = 1. You must periodically call the SVC to see if the <HOLD> key has been pressed; call the SVC with B > 1. This places the program in control of the pause checking. If <HOLD> has been pressed, the program pauses until <HOLD> is pressed again.

Returning to the TRSDOS-II READY command level turns off the <HOLD> processor. However, you can execute a command without turning off the <HOLD> processor. See the JP2DOS and DOSCMD SVCs for ways to execute a command.

Some library commands and utilities, including DIR and LIST, reset the <HOLD> processor.

No matter how many times you press <HOLD> before you call HLDKEY, TRSDOS-II interprets it as one keystroke, thus starting the pause.

Entry Conditions

A = 29
B = function code

The HLDKEY function codes are:

Contents of B	Meaning
Ø	Turns off the <HOLD> processor. Pressing <HOLD> generates X'ØØ'.
1	Turns on the <HOLD> processor. Pressing <HOLD> does not generate keyboard data. TRSDOS-II intercepts it.
>1	Checks for the <HOLD> key. If you have pressed it, TRSDOS-II pauses until you press it again.

Exit Conditions

NZ = <HOLD> processor is off

INITIO **Function Code Ø**
Initialize I/O

Initializes all input/output drivers.

This SVC is already done by TRSDOS-II. You do not need to call it, except in extreme error conditions.

Entry Conditions
A = Ø

Exit Conditions
NZ = Error
A = error code

JP2DOS Function Code 36
Jump to TRSDOS-II

Returns control to TRSDOS-II, after closing all open files and doing system housekeeping.

Another way to do this is to execute an RST Ø instruction. There are no entry conditions for this method.

Entry Conditions
A = 36

Exit Conditions
None

KBCHAR **Function Code 4**
Keyboard Character

Gets one character from the keyboard. If characters are present in the key-ahead buffer, the first character in the buffer is returned to Register B.

The <BREAK> key is masked from you. It is never returned, since it is intercepted by TRSDOS-II. If you enable a SETBRK routine, control passes to the processing program whenever <BREAK> is pressed (see the SETBRK SVC). Otherwise, control passes to TRSDOS-II READY.

Entry Conditions

A = 4

Exit Conditions

B = character found (if any)
If no character is returned, B is unchanged.
NZ = No character is present
A = error code

KBINIT **Function Code 1**
Keyboard Initialize

Initializes the keyboard input driver. You might want to call KBINIT before starting keyboard input, to clear previous keystrokes and the key-ahead buffer. TRSDOS-II automatically does this at startup.

Entry Conditions
A = 1

Exit Conditions
NZ = Error
A = error code

KBLINE **Function Code 5**
Keyboard Line

Inputs a line from the keyboard to a buffer and echoes the line to the display, starting at the current cursor position. As it receives and displays each character, KBLINE advances the cursor to the next position.

When you enter KBLINE, the input buffer contains a string of periods, which it sends to the display. This string is for your convenience. It indicates the length of the input field.

The keyboard line ends when you press <ENTER> or when the input buffer is filled. When you end the line input, KBLINE sends a carriage return and an "erase to end of screen" command to the display. It stores a carriage return as a character input only if you press <ENTER>.

Entry Conditions

A = 5
(HL) = input buffer
B = maximum number of characters to receive (1-255)

Exit Conditions

B = actual number of characters input (including carriage return)
C = terminating character of input
C = X'ØØ': input buffer is filled without a carriage return
C = X'ØD': line ended with a carriage return

If you type a control code not listed below, KBLINE places it in the buffer and represents it on the display with the ± symbol.

Key	Hex Code	Function
<BACKSPACE>	Ø8	Backspaces the cursor and erases a character.
<ENTER>	ØD	Ends the line. Clears trailing periods on the display but not in the buffer.

Key	Hex Code	Function
<CTRL> <W>	17	Fills the remainder of the input buffer with blanks. Blanks the remainder of the display line.
<CTRL> <X>	18	Fills the remainder of the input buffer with blanks. Blanks to the end of the display.
<ESC>	1B	Reinitializes the input function by filling the input buffer with periods and restoring the cursor to its original position.
<left arrow>	1C	Backspaces the cursor to allow editing of the line. Does not erase characters.
<right arrow>	1D	Advances the cursor to allow editing of the line. Does not erase characters.

KBPUT **Function Code 3Ø**
Put Character into Key-Ahead Buffer

Puts one character into the keyboard key-ahead buffer, in the same manner that pressing that key on the keyboard does.

The character is placed after any characters already in the buffer.

If the character is X'ØØ', TRSDOS-II puts it into the buffer if HLDKEY is off. If HLDKEY is on, X'ØØ' triggers hold processing. (See the HLDKEY SVC.)

If the character is X'Ø3', it invokes the <BREAK> key processing. (See the SETBRK SVC.)

Entry Conditions

A	=	3Ø
B	=	character to be put into the buffer

Exit Conditions

Z	=	Character is put into the buffer
NZ	=	Buffer is full; the character is not in the buffer
A	=	error code

KILL **Function Code 41**
Delete File From Directory

Deletes the specified file from the directory. You must close a file before you can KILL it.

Entry Conditions

A = 41
(DE) = data control block

Exit Conditions

NZ = Error
A = error code

LOCATE **Function Code 33**
Locate Record

Returns the number of the current record (the last record accessed). You can use LOCATE only with FLR files.

Entry Conditions

A = 33
(DE) = data control block of the currently open file
HL = Reserved

Exit Conditions

BC = current record number
NZ = Error
A = error code

If you chose the E mode when you opened the file, then:

BC = address of 3 bytes in RAM where the record number is to be stored, in the following format:

upper byte	middle byte	lower byte

where ØØ ØØ ØØ = first record of file

LOOKUP **Function Code 28**
Look Up in a Table

Does a lookup function on a table of 3-byte entries. Each entry looks like this:

Byte 1	Byte 2	Byte 3
Search Key	Data, e.g., address in LSB, MSB sequence	

At the end of the table, you must place a 1-byte X'FF'. Since X'FF' is the table terminator, your search keys must be between X'ØØ' and X'FE'.

Given a 1-byte search argument, LOOKUP locates the first matching entry in the table. It uses a sequential search algorithm.

If the table contains search keys followed by addresses, then (upon return from the routine with the Z flag set) you can JP (HL) to transfer control to the desired address.

Entry Conditions

A = 28
(HL) = byte 1 of the table
B = search argument

Exit Conditions

NZ = Search argument is not found
HL = data
H contains byte 3
L contains byte 2

MPYDIV **Function Code 23**
Multiply Divide

Multiplies or divides with one 2-byte value and one 1-byte value.

Entry Conditions

```
     A       = 23
     B       = function switch

If B = Ø:  multiply
     HL      = multiplicand
     C       = multiplier

If B ≠ Ø: divide
     HL      = dividend
     C       = divisor
```

Exit Conditions

```
If B = Ø:
     HL      = product (HL * C)
     C flag  = Overflow occurred
     A       = overflow byte
     Z       = Product is Ø

If B ≠ Ø:
     HL      = quotient (HL/C)
     C       = remainder
     C flag  = Dividing by Ø; division is not attempted
     Z       = Quotient is Ø
```

OPEN Function Code 4Ø
Open File

Creates new files and opens existing files.

A given file can be open under only one data control block (DCB) at a time. Because of the versatile file processing routines, this one DCB is enough to handle the various I/O applications.

Entry Conditions

A = 4Ø
(DE) = data control block
(HL) = parameter list

Exit Conditions

NZ = Error
A = error code

Before calling OPEN, you must reserve space for the data control block, parameter list, buffer area, and record area.

Data Control Block (6Ø bytes)

The data control block (DCB) is used by TRSDOS-II for file access bookkeeping. Use it specify the filespec when opening a file.

Before calling OPEN, place the filespec at the beginning of the DCB, followed by a carriage return. (See "File Specification" in the "Introduction.")

For example:

Contents of First Bytes of DCB Before Open
($ signifies a carriage return)

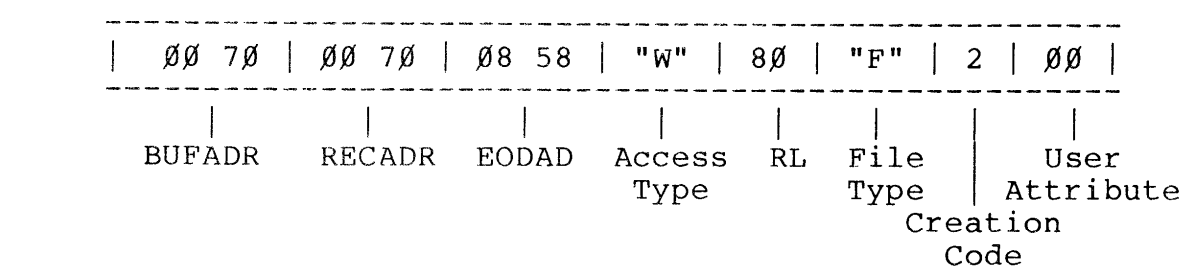
```
|f i l e n a m e/e x t . p a s s w o r d :d (disk name) $ |
```

While a file is open, TRSDOS-II replaces the filespec with information TRSDOS-II uses for bookkeeping. When you close the file, TRSDOS-II returns the original filespec (except the password) to the DCB.

Never modify any part of the DCB while the file is open. The results are unpredictable.

Parameter List (11 bytes)

Contents of Parameter List (sample)



BUFADR (Buffer Address)

Two-byte field that points to the beginning of the buffer area.

The buffer area is the space TRSDOS-II uses to process all file accesses. When spanned records are required, you must reserve 512 bytes. When spanning is not required, reserve only 256 bytes.

With FLR files, spanning is required only when the record length is not an even divisor of 256. For example, if the record length is 64, then each physical record contains four records, and no spanning is required. In this case, reserve only 256 bytes for processing.

However, if the record length is 24 (not an even divisor of 256), then some records must be spanned. In this case, reserve 512 bytes.

With VLR files, you must reserve 512 bytes for processing. Spanning may be required, depending on the lengths of the records.

RECADR (Record Address)

Two-byte field that points to the beginning of the record area.

This is where TRSDOS-II places the record after a disk read and where you would put the record to be written to disk.

For FLR files with a record length of 256, RECADR is not used. Your record is in the buffer area pointed to by BUFADR.

If you have an FLR file with a record length not equal to 256, make this buffer the same length as the record.

If you have VLR files, be sure to make the buffer long enough to contain the longest record in the file (including the length byte). If you are not sure of the length, reserve 256 bytes.

EODAD (End of Data Address)

Two-byte field which can be used to give TRSDOS-II a transfer address to use in case TRSDOS-II reaches the EOF during an attempted read.

Control transfers to the EODAD if this happens. If EODAD = Ø and the EOF is reached, the SVC returns with the EOF error code in Register A.

Access Type

Specifies the type of access you desire. It can be any of the following:

ASCII Character	Meaning
R	Read only
W	Read/Write (data files)
P	Read/Write access to Z8Ø program files

If OPEN finds a program file, TRSDOS-II returns a P in the read/write field in the parameter list. For example, a call to OPEN with a parameter list of:

ØØ7Ø ØØ7Ø Ø858 R 8Ø F Ø ØØ

returns the following, if the file found is a program file:

ØØ7Ø ØØ7Ø Ø858 P 8Ø F Ø ØØ

RL (Record Length)

One-byte field that specifies the record length to be used.

Zero indicates a record length of 256. For VLR files, ignore this field. If the file exists, and the creation code is Ø, TRSDOS-II supplies the correct RL value, regardless of what is stored there.

File Type

One-byte field that contains an ASCII "F," "V," or "E" (for "fixed-length," "variable-length," or "extended mode").

Once a file exists, this attribute cannot be changed. If the file exists, and the creation code is Ø, TRSDOS-II supplies the correct F, V or E value.

If you specify E, then DIRRD, DIRWR and LOCATE use a 24-bit, indirect, 3-byte number. The E specifies FLR records and an extended mode of specifying record numbers. The directory shows F, even though you specify the E mode.

Creation Code

One-byte field that contains a binary number "Ø," "1," or "2."

Code	Meaning
Ø	Opens the existing file, but does not reset the record length and EOF.
1	Creates a new file. Returns an error if a file of the same name already exists on the specified drive. Sets the record length and EOF at OPEN.

Code	Meaning
2	Opens a file, overwriting any existing file of the same name. Otherwise, it creates a new file. Resets the record length and EOF.

User Attribute

Lets you give your own ID number to certain types of data files. You can use any number from 32-255.

TRSDOS-II does not examine this user attribute. It is solely for your convenience.

You can assign this user attribute only to files you open with a creation code of 1 or 2. Files opened with a creation code of Ø retain the file's previously assigned user attribute. Any file created with the CREATE command has a user attribute value of Ø.

PARSER **Function Code 46**
Analyze Text Buffer

Analyzes or "parses" a text buffer into fields and subfields.

PARSER is a convenient and powerful SVC. It is useful for analyzing TRSDOS-II command lines, including keyword commands, file specifications, keyword options and parameters. You can use it also as a fundamental routine for a compiler or text editor.

Entry Conditions

A = 46
(HL) = text buffer
(DE) = list address block
DE = Ø: no lists are to be used
C = maximum length of the parse

Exit Conditions

(HL) = field position
If a delimited field was found:
(HL) = first byte of the field
If no field was found:
(HL) = terminator or non-blank separator
If the parse reached the maximum length:
(HL) = last byte of the buffer
B = actual length of the field (excluding leading and trailing blanks)
A = character preceding the field just found
If B = Ø, A = X'FF'.
C = number of bytes remaining to parse after the terminator or separator (trailing blanks have been parsed).
D = separator or terminator at the end of the field
If D = X'FF', then the parse stopped without finding a non-blank separator or terminator.
E = displacement pointer for the next parse call

Add the displacement pointer (E) to the field position (HL) to get:
a) Beginning address of the next field, or
b) Address of the byte following the last byte parsed

If the parse reached the maximum length:
E + HL = address following the end of the text buffer

If the parse did not reach the maximum length, and E = 1:
E + HL = address following the separator or terminator

If the parse did not reach the maximum length:
Z flag = Parse ended with a separator
NZ = Parse ended with a terminator
C flag = There were trailing blanks between the end of the field and the next non-blank separator or terminator
NC = There were no trailing blanks

PARSER is designed to allow repetitive calls for processing a text buffer. On exit from the parse, parameters for the next call are all readily available in the appropriate registers.

Field

A field is any string of alphanumeric characters (A-Z, a-z, Ø-9) with no embedded blanks, delimited by separators and terminators.

Separators and Terminators

Separators and terminators are used to delimit (or separate) fields within a text buffer. PARSER has pre-defined sets of field-characters and separators. You either can use these or re-define them to suit your application.

For example, the line:

BAUD=3ØØ,PARITY=EVEN WORD=7

contains 6 fields: BAUD, 3ØØ, PARITY, EVEN, WORD, and 7.

A separator is any non-alphanumeric character. PARSER stops when it encounters a separator, except when the separator is a blank (X'2Ø'). PARSER ignores leading and trailing blanks. After the trailing blanks, the parse stops at the beginning of the next field or at the first non-blank separator.

You can also define terminators, which stop PARSER regardless of whether it has found a field. Unless you specifically define these terminators, the parse stops only on non-blank separators.

Separators and terminators have the same effect on a parse. The only difference is in how they affect the F (Flag) register on exit.

A field can be delimited by paired quotation marks, also:

"field" or 'field'

When you use quotation marks, all characters (not just alphanumerics) are taken as part of the field. The quotation marks are not included in the field. For example:

'DATE(Ø7/11/79)'

is interpreted as one field that contains everthing inside the quotation marks.

When you use a quotation mark to mark the start of a field, you must use the same kind to end the field. This lets you use quotation marks within a field. For example:

"X'FFØØ'"

is parsed as one field that contains everything inside the double quotes, including the single quotes.

To Re-define the Field, Separator, and Terminator Sets

If you need to change the field and separator sets, or define terminators, you can provide three change-lists via a list address block.

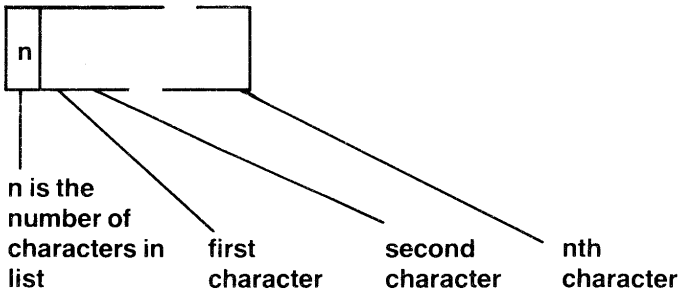
List Address Block

The list address block is six bytes long and contains three 2-byte addresses (LSB,MSB), one for each change-list:

List 1: characters to be used as terminators
List 2: additions to the set of field characters

List 3: deletions from the set of field characters (alphanumerics to be interpreted as separators)

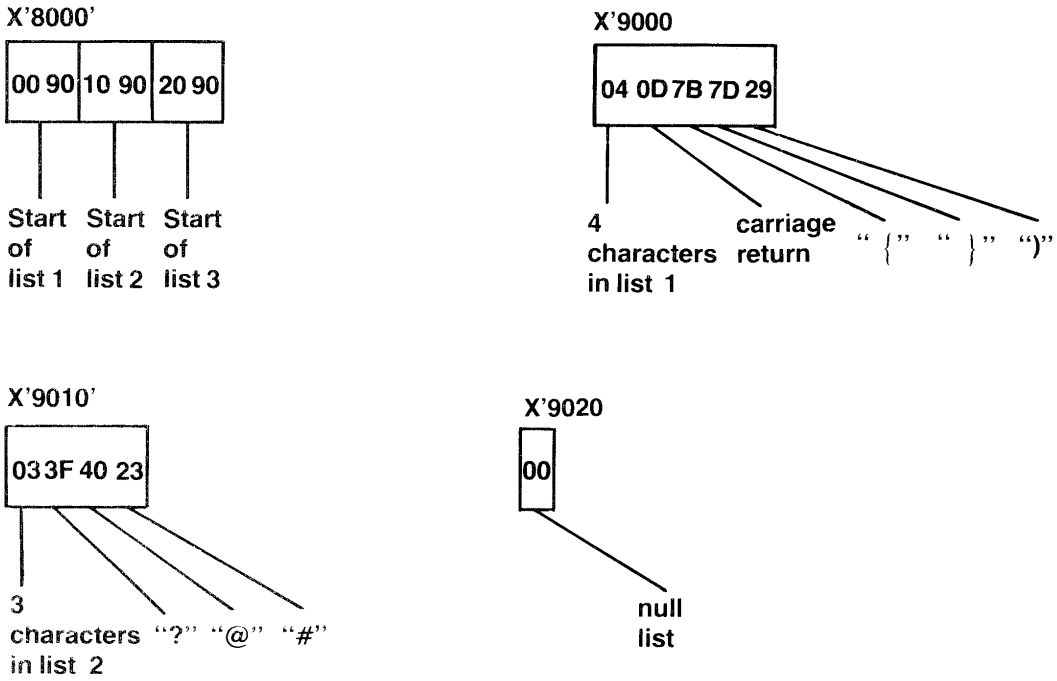
Each list has the following form:



Notes

- There are three ways to indicate a null list:
  a) Set the character-count byte (n) equal to zero.
  b) Set the pointer in the list address block to zero.
  c) Set DE equal to zero, if you aren't going to provide any lists.
- Characters are stored in lists in ASCII form.
- If a character appears in more than one list, it has the characteristics of the first list that contains it.

Here is a typical list address block with its associated lists. Assume that on entry to PARSER, DE = X'8ØØØ'.



Sample Programming

The following code shows typical repetitive uses of PARSER to break up a parameter list.

```
;-----------------PREPARE FOR PARSING LOOP-----------------------------
        LD      C,MAXLEN        ; C = Maximum length to parse
        LD      E, 0            ; For initial call to NXTFLD
        LD      HL,BUFFER       ; (HL) = string to parse
;-----------------PARSE LOOP-------------------------------------------
PARSE   CALL    NXTFLD          ; Routine to call PARSER
        CALL    HANDLR          ; Routine to handle new field
        JR      NZ,NXTRTN       ; Go to next routine if
                                ;  parsed ended on terminator
        LD      A,C             ; Else get new max length
        OR      A               ; Is it zero?
        JR      NZ,PARSE        ; If not, then continue
        LD      A,0FFH          ;
        CP      D               ; If D=0FFH then no separator
                                ;  at end of buffer.
        JR      Z,ERR           ; So go to error routine
        JR      NXTRTN          ; Else, then do next routine.
;-----------------FIELD-HANDLING ROUTINE-------------------------------
HANDLR  PUSH    AF              ; Must save status registers
                                ; and any other registers
                                ; will be changed.
                                ;
; Processing code goes here...
        POP     AF              ; Restore AF (and other
                                ; registers saved at entry)
        RET
;-----------------CALL TO PARSER---------------------------------------
NXTFLD  LD      D,0             ; Zero msb of DE
        ADD     HL,DE           ; (HL) = where to start parse
        LD      DE,LAB          ; (DE) = List address block
                                ; If DE=0 then no lists used.
        LD      A,46            ; Function Code
        RST     8
        RET
;-----------------PROGRAM CONTINUES HERE-------------------------------
NXTRTN  EQU     $
```

PRCHAR
Print Character

Function Code 18

Sends one character to the printer.

Note: Most printers do not print until their buffer is filled or they receive a carriage return. (See your printer's manual for information.)

Normally, TRSDOS-II intercepts certain codes and does not send them directly to the printer. There are several ways to override some or all of these character translations. (See the PRINIT and PRCTRL SVCs.)

While the serial printer option is selected, allowing printer output to Channel B, TRSDOS-II recognizes two characters from Channel B. These affect all serial printer output operations:

ASCII Name	Hex Code	Result
DC3, <CTRL> <S>	13	Pauses printing
DC1, <CTRL> <Q>	11	Resumes printing

Entry Conditions

A = 18
B = ASCII code for the character to send

Exit Conditions

NZ = Error
A = error code

INTERCEPTED CODES

ASCII Name	Hex Code	Result to Printer
Tab	Ø9	Sends one to eight spaces to provide the tab function.
Vertical Tab	ØB	Same as form feed below.
Form Feed	ØC	Sends enough carriage returns or line feeds to advance the paper to the next "top of form."
Carriage	ØD	Interpreted as a line feed when the current line is empty (no characters are printed since the last carriage return or line feed). This interpretation enables the correct operation of Radio Shack printers. In the auto line-feed mode, X'ØA' is sent after every X'ØD'.
Special	8D	Sends a carriage return to the printer. In the auto line-feed mode, using this code lets you send a carriage return **without** a line feed.

PRCTRL **Function Code 95**
Control Printer Operations

Gives you various printer options and lets you check the status of printer-related functions.

If you use SPOOL's capture function, TRSDOS-II sends the captured data directly to the capture file -- regardless of which control settings you select. If you use SPOOL's print function, TRSDOS-II intercepts the printed data to the selected control settings. (See the SPOOL command for details.)

By switching between the serial and parallel output modes, you can operate with two printers. One printer can use the automatic control features of TRSDOS-II, such as the auto form feed, while the other printer is controlled by the program. This is necessary because only one set of control counters is maintained.

Entry Conditions

A	= 95
C	= option value (used with option codes 3 and 4)
B	= option code

Exit Conditions

Z flag	= No error
NZ	= Error
B	= physical page length
C	= logical page length
D	= logical line length
E	= printer option E = P: parallel printer E = S: serial printer
H	= number of characters printed on the present line
L	= number of lines printed on the present page

(See the PRINIT SVC for details on physical and logical page length and logical line length.)

The option codes are:

Code	Option
Ø	Gets printer status only (see "Exit Conditions")
1	Selects the serial printer driver (you must initialize Channel B first)
2	Selects the parallel printer driver
3	Resets the current line count to the value contained in Register C
4	Resets the current character count on current line to the value contained in Register C
5	Begins the transparent mode
6	Ends the transparent mode
7	Begins the dummy mode
8	Ends the dummy mode
9	Begins the auto line-feed (after carriage return) mode
1Ø	Ends the auto line-feed (after carriage return) mode

Explanation of Options

Serial/Parallel Printer Option

While the serial printer option is selected, allowing printer output to Channel B, TRSDOS-II recognizes two input characters from Channel B. These affect all serial printer output operations:

ASCII Name	Hex Code	Result
DC3, <CTRL> <S>	13	Pauses printing
DC1, <CTRL> <Q>	11	Resumes printing

Line Count/Character Count

TRSDOS-II maintains a single line counter and a single character counter. It updates these for either serial or parallel printing. TRSDOS-II does not maintain separate counters for serial and parallel printing.

Transparent Mode

The transparent mode overrides all data translation. All data bytes go directly to the printer. TRSDOS-II does not examine the contents or update the line and character counts.

Normally, TRSDOS-II "intercepts" certain control characters and interprets them. In this way, TRSDOS-II provides printer-related features which may not be available from the printer.

For example, tabs (X'Ø9') are intercepted so that TRSDOS-II can send the appropriate number of spaces to provide the tab function. Form feeds (X'ØC' or X'ØB') are intercepted so that TRSDOS-II can advance the paper to the next top of form.

Special settings of the printer initialization values can override individual code translation. (See the PRINIT SVC for details.)

Dummy Output Mode

The dummy mode "throws away" all printer output and returns with a "good" (Z flag set) return code. During dummy mode operation, the line and character counts remain unchanged.

Auto Line Feed

Normally, the TRSDOS-II printer driver does not output line feeds after carriage returns, because most Radio Shack printers do an automatic line feed after every carriage return.

If your printer does not do automatic line feeds after carriage returns, you may enable the TRSDOS-II auto line-feed function.

While the function is enabled, TRSDOS-II outputs a line feed after each carriage return. This is true in all modes including the transparent mode.

Note: In the auto line-feed mode, you may send a carriage return with no line feed by outputting the code X'8D'. If the printer is capable of overprinting, this code returns the carriage without advancing the paper.

Precedence of Options

The priority of the available options is (in descending order):

SPOOL capture mode (See the SPOOL command)
Dummy mode
Auto line-feed (after carriage return) mode
Transparent mode
Normal mode

PRINIT **Function Code 17**
Printer Initialization

Initializes the printer driver.

It does **not** advance the printer paper and does **not** check the printer status or availability. It operates even if the printer is off-line.

Entry Conditions

A = 17
B = physical page length
C = logical page length
D = logical line length

Exit Conditions

NZ = Error

The physical page length is the number of lines that can be printed on one page. It is normally 66.

The logical page length is the number of lines you want printed on each page. It must be less than or equal to physical page length. After that number of lines is printed, TRSDOS-II sends a top-of-page request to the printer (form feed).

The logical line length is the maximum number of characters to be printed on each line. Once that number is reached, the line is printed and a new line is begun (wrap-around).

Notes

- If the logical page length = the physical page length and both are non-zero, TRSDOS-II does not do an end-of-page skip. It translates form feeds into the number of line feeds necessary to get to the top of the next page.

- If the logical page length or the physicial page length is zero, the other also must be zero. In this case, TRSDOS-II does not translate form feeds (X'ØC') and vertical tabs (X'ØB') into line feeds, but sends them directly to the printer.

If the logical line length is zero, then TRSDOS-II does not translate tab characters (X'Ø9') into spaces, but sends them directly to the printer. TRSDOS-II maintains an internal character count, with tabs counting as one character.

On exit, TRSDOS-II automatically resets the current line and character counts to zero.

TRSDOS-II assumes the printer is at the top of the form when you execute PRINIT.

TRSDOS-II resets the dummy and transparent modes to the normal mode. Auto line feed and DUAL are unaffected by PRINIT.

When you power up or reset TRSDOS-II, it assumes the following defaults:

Parameter (PRINIT register)		Value
physical page length (B)	=	66
logical page length (C)	=	6Ø
logical line length (D)	=	132

The other options set during initialization are:

Auto line-feed mode	=	off
Dummy mode	=	off
Transparent mode	=	off
Parallel printer	=	on

PRLINE **Function Code 19**
Print Line

Sends a line of characters to the printer's buffer. The line can include control characters as well as printable data.

See the PRCHAR SVC for a list of intercepted codes.

Entry Conditions

A = 19
(HL) = start of the buffer containing data and controls to send to the printer
B = line length (the number of characters to send)
C = control character (any character) to send after the last character in the buffer

Exit Conditions

NZ = Error
A = error code

RAMDIR **Function Code 53**
Get Disk Directory/Free Space into RAM)

Lets you examine a disk directory entry or the first 96 files in the directory, or it lets you determine the diskette's free space. The information is written into a RAM buffer.

Only non-system files are included in the RAM directory.

Note: We recommend restricting your use of the RAMDIR since this SVC is restricted to 96 files. If you are familiar with RAMDIR, you should change your existing programs to use RDDIR instead.

Entry Conditions

A = 53
B = drive number (binary Ø-7)
C = function switch

Contents of C	Result
Ø	Gets the entire directory into RAM in the format described below.
1-96	Gets one specified directory record (if it exists) into RAM in the format described below.
255	Gets free-space information in the format described below.

(HL) = buffer area

If C = Ø, compute the buffer size by this formula:
Number of bytes required = 34 * (# of non-system files + 1)

Because RAMDIR can read no more than 96 non-system files, the largest possible directory needs a buffer size of 3265 bytes.

If C = 1-96, the buffer must be 34 bytes long.
If C = 255 (free-space info desired), the buffer must be 4 bytes long.

Exit Conditions

Z flag = No error; (HL) = directory or free-space information
NZ = Error (Register A contains the TRSDOS-II error code)

RAM Directory Format

The directory is given in "records," one per file. Each record has the following form:

Byte Number	Contents	Comments
1	":"	ASCII colon marks beginning of each directory record in RAM.
2-16	filename/ ext:d <ENTER>	<ENTER> represents a carriage return. Trailing blanks are added as needed to fill 15 bytes.
17	"F" or "V"	Indicates fixed- or variable-length records.
18	LRL	Logical record length 1-byte binary Ø-255. Ø implies LRL =256 or VLR file.
19	# of extents	One-byte binary Ø-16, Null files=Ø.
2Ø-21	# of sectors allocated	Binary Ø-65535 in LSB-MSB sequence; null files = ØØ.
22-23	# of sectors in storage of data	Binary Ø-65535 in LSB-MSB sequence; null files = ØØ.
24	EOF byte (position of the first byte of the last record written)	Binary Ø-255; Ø implies the first byte in the last sector.
25-26	# of records written	Binary Ø-65535 in LSB-MSB sequence; null files = ØØ.
27	User attribute byte	Assigned when you created the file.

Byte Number	Contents	Comments
28	Protection level	Binary Ø-7.
29-31	Date created	Binary year, month, day.
32-34	Date last updated	Binary year, month, day.

When an entire directory is given (C = Ø on entry), a number sign (#), instead of the expected colon (:), marks the end of the directory. When C = from 1 to 96, the record always ends after the 34th byte, without a trailing number sign.

Free-Space List Format

Byte Number	Contents	Comments
1-2	# of free granules	*Binary LSB-MSB sequence
3-4	# of extents	Binary in LSB-MSB sequence

* The number of sectors equals the number of granules multiplied by 5.

RANDOM **Function Code 2Ø**
Random 1-Byte Value Return

This routine returns a random 1-byte value.

TRSDOS-II uses the time/date values in computing the random number. This method lessens the possibility of repetition.

Pass the routine a limit value. If the limit value is greater than 1, the value returned is in the range [Ø,(limit - 1)]. For example, if the limit is 255, TRSDOS-II returns a value in the range [Ø,254]. If the limit value is Ø or 1, the value returned is Ø.

Entry Conditions

A	=	2Ø
B	=	limit value

Exit Conditions

C	=	random number
NZ	=	Error

RDDIR **Function Code 32**
Reads in Directory Record

Reads in one directory record from one drive at a time. It also lets you use a wildcard mask. TRSDOS-II puts the directory record into a 128-byte ASCII string in user memory.

To read the directory of a specific file, use the filespec as the wildcard mask. (Do not use the asterisk.) RDDIR then searches for the file.

(See the DIRSET SVC for details on open file directory information.)

Entry Conditions

A = 32
BC = address of the 8-byte block in user memory defined as:

binary drive #	directory index*						**
	FF	FF	FF	FF	FF	FF	ØØ

* = FF FF FF FF FF FF sets the index to the beginning of the directory
** = end-of-list terminator

DE = wildcard mask For example:

```
MASK    DEFM    '*/BAS'
        DEFB    ØDH  ;must end with
                     <ENTER>
```

If DE = Ø, no wildcard selection takes place.

HL = address of the 128-byte area where data is to be placed

Exit Conditions

HL = address of the 128-byte area where data is placed. It is in the TRSDOS-II DIR format.
Z = No error
C,NZ = Error; EOF encountered
NC,NZ = I/O error
A = error code

Offset Table for HL Buffer

Offset	Description	# Bytes
Ø	Length byte	1
1-12	Filename	12
13	File-left-open marker	1
16-23	Date created	8
25-32	Date updated	8
39-42	Attrib	4
45	File type	1
48-51	Record length	4
53-61	Number of records	9
64-7Ø	Sectors allocated	7
73-79	Sectors used	7
8Ø-127	Reserved	48

Example

```
LOOP     LD       BC,CTLBLK
         LD       DE,MASK
         LD       HL,IMAGE

         LD       A,32
         RST      8
         JR       NZ,END          ;  END OR ERROR HAS OCCURRED

         (display "IMAGE")

         JR       LOOP

END:     JR       C,EOF           ;  INDICATES EOF
         (exit routine; otherwise, an error occurred)

CTLBLK:  DEFB     1               ;  DRIVE #
         DEFB     -1,-1,-1        ;  BEGINNING OF DIRECTORY
                                     INDEX
         DEFB     -1,-1,-1        ;
         DEFB     Ø               ;  END OF LIST TERMINATOR

MASK:    DEFM     '*/BAS'         ;  FIND /BAS FILES ONLY
         DEFB     ØDH             ;  CR

IMAGE:   DEFS     128             ;  AREA TO STORE DIR RECORD
```

READNX **Function Code 34**
Read Next Record

Reads the record following the current record (the last record accessed). If you just opened the file, READNX reads the first record.

Upon return, your record is in the record area pointed to by RECADR in the parameter list. Or, if the record length is 256 and the record type is fixed, your record is in the area pointed to by BUFADR.

Entry Conditions

A = 34
(DE) = data control block for the currently open file
HL = Reserved for use in later versions of TRSDOS-II

Exit Conditions

NZ = Error
A = error code

RENAME **Function Code 47**
Rename a File

Changes the name and/or extension of a file. You cannot change the password.

Both filespecs (old and new) must be terminated by a carriage return.

If the old filespec is password protected, you must include the password. Do not include a password in the new filespec; TRSDOS-II retains the old password.

Entry Conditions

A = 47
(HL) = old filespec
(DE) = new filespec

Exit Conditions

NZ = Error
A = error code

RETCMD **Function Code 38**
Return after Command

Sends TRSDOS-II an operator command. After the command is completed, your program regains control.

Take care that TRSDOS-II doesn't overlay your program when it loads the command file you choose. Most TRSDOS-II library commands use memory below X'27FF'. A few go up to, but do not include, X'2FFF'. Single-drive copies use all user memory.

Entry Conditions

(HL) = command string
B = length of string
A = 38

Exit Conditions

NZ = Error
A = error code

REWIND **Function Code 48**
Rewind Disk File

Rewinds a disk file. After this SVC is executed, the next read/write accesses the first record of that file.

Entry Conditions
A = 48
DE = data control block for the currently open file

Exit Conditions
NZ = Error
A = error code

RS232C Function Code 55
Initialize RS-232C Channel

Sets up or disables either Serial Channel A or B. Before you use it, be sure the correct channel is connected to the modem or other requirements.

This routine sets the standard RS-232C parameters and defines a set of supervisor calls for I/O to the channel you choose. When you initialize Channel A, TRSDOS-II defines the following SVCs: ARCV, ATX, and ACTRL. When you initialize Channel B, TRSDOS-II defines BCRV, BTX, and BCTRL.

Before re-initializing a channel, **always** turn it off. You may **not** turn off Channel A when HOST is active.

Entry Conditions

A = 55
(HL) = parameter list
B = function switch
B = Ø: turn off channel
B ≠ Ø: turn on channnel

Exit Conditions

NZ = Error
A = error code

Parameter List

This 6-byte list includes the required RS-232C parameters:

Channel	Baud Rate	Word Length	Parity	Stop Bits	End-of-List Marker

Channel -- is an ASCII "A" for Channel A or "B" for Channel B.

Baud Rate --is a binary value from "1" to" 8":

1 for 11Ø baud
2 for 15Ø baud
3 for 3ØØ baud

4 for 6ØØ baud
5 for 12ØØ baud
6 for 24ØØ baud
7 for 48ØØ baud
8 for 96ØØ baud

Word Length -- is a binary value from "5" to "8":

5 for 5-bit words
6 for 6-bit words
7 for 7-bit words
8 for 8-bit words

Parity -- is an ASCII "E," "O," or "N" (for "even," "odd," or "none").

Stop bits -- is a binary "1" or "2" for the number of stop bits.

End list marker -- is X'ØØ' for the 16-character default buffer or X'Ø1' for a larger buffer.

If you specify a larger buffer, the next five bytes specify the receive buffer, as shown below:

Byte Ø	End-of-list marker (X'Ø1')
Bytes 1-2	Buffer start address (LSB/MSB)
Bytes 3-4	Buffer end address (LSB/MSB)
Byte 5	Terminator (X'ØØ')

The buffer must be at least 49 bytes long, and the entire buffer (both front and back) must reside below X'8ØØØ'.

To determine the size of the buffer, use the following formula:

number of characters * 2 + 15 = buffer size

The number of characters to be stored must be at least 17. Remember, it must be small enough to fit below X'8ØØØ'.

SCROLL Function Code 27
Protect from Scrolling

Lets you protect part of the display from scrolling. From Ø to 22 lines at the top of the display can be protected. When scrolling occurs, only lines below the protected area are changed.

Entry Conditions

A	=	27
B	=	number of lines to protect (in range [Ø,22])

Exit Conditions

NZ	=	Error
A	=	error code

SETBRK Function Code 3
Enable and Disable <BREAK> Key

Lets you enable the <BREAK> key by defining a <BREAK> key processing program. Whenever you press the <BREAK> key, your processing program takes over.

On entry to the <BREAK> processing program, the return address of the interrupted routine is on top of the stack. You can return to it with a Z-8Ø return instruction (RET).

All Z-8Ø register contents are unchanged upon entry to the <BREAK> program.

SETBRK also lets you disable the <BREAK> key, by removing the address of the processing program. You cannot change processing programs while the <BREAK> key is enabled.

You must disable the <BREAK> function (HL=Ø) before setting HL to a new address.

The <BREAK> key processing program must reside above X'27FF'.

See "Handling Programmed Interrupts" for programming information.

Entry Conditions

A = 3
HL = address of the <BREAK> key processing program
HL = Ø: disable the <BREAK> function

Exit Conditions

NZ = Error
A = error code

If HL = Ø, then:
(HL) = address of the deleted <BREAK> key processing program

SETUSR **Function Code 2**
Set User

Sets or removes a user vector, thus giving you the ability to add SVC functions. Function codes 96-1Ø1 are available for you to define, unless the serial interface is on (see the RS232C SVC). Function codes 1Ø2-127 are always available for you to define.

Once you add an SVC function, you can call it via the RST 8 instruction, just as you can the TRSDOS-II's SVC routines.

Your routine must reside above X'27FF', and it should end with a RET instruction.

To change an already defined function, you must first remove the old vector.

You can call a non-recursive SVC (system or user-defined) from within a user-defined SVC. For example, do not call SVC 1Ø7 from within SVC 1Ø7.

Entry Conditions

A = 2
B = function code(95 < code < 128)
C = set/reset code
C = Ø: remove the vector
C ≠ Ø: add the vector

If C = Ø (remove vector):
HL = entry address of your routine

Exit Conditions

If C = Ø (remove vector):
HL = removed vector address

SORT **Function Code 56**
RAM Sort

Sorts a list of entries in RAM. All entries must have the same length, from 1 to 255 bytes. The sort is done according to a user-defined sort-key, which you may locate anywhere in the entry.

Entries with duplicate sort-keys remain in the same relative order.

The routine uses a "bubble-sort" algorithm.

Entry Conditions

A = 56
(IX) = first byte of the first entry
(DE) = first byte of the last entry
B = position key within an entry
(Zero equals the first byte of an entry.)
C = length of each entry (C > Ø)
H = sort switch
HL = Ø: use ascending sort
HL ≠ Ø: use descending sort
L = length of sort-key (L > Ø)

In general, B + L must be less than or equal to C.

Exit Conditions

NZ = Error
A = special error return code (not the standard TRSDOS-II code)

Value in A	Meaning
1	Key end exceeds the last byte of the entry
2	Key start exceeds the last byte of the entry
3	Entry length = Ø (invalid)
4	Key length = Ø (invalid)
5	Address of first entry > address of last entry

SOUND Function Code 6Ø

Sound Output

Turns on a single tone (approximately 1 khz) in computers that are capable of generating sound.

Entry Conditions

A = 6Ø
HL = sound duration (in 1/3Øth second)

Exit Conditions

None

The sound turns off automatically after n seconds, where:

n = HL/3Ø

n may vary slightly, depending on how much disk and video access is done while SOUND is on.

STCMP **Function Code 22**
String Comparison

Compares two strings to determine their collating sequence.

Entry Conditions

A	=	22
(DE)	=	string1
(HL)	=	string2
BC	=	number of characters to compare

Exit Conditions

Z flag	=	Strings are identical
NZ	=	Strings are not identical
C flag	=	string 1 (DE) precedes string2 (HL) in the collating sequence
A	=	first non-matching character (in string1)

When the strings are not equal, you can get further information from the prime registers, as follows:

HL'	=	address of the first non-matching character in string2
DE'	=	address of the first non-matching character in string1
BC'	=	number of characters remaining This includes the non-matching character.

STSCAN **Function Code 49**
String Scan

Searches through a specified text buffer for the specified string. This string can consist of any values Ø-255 (it is not strictly alphanumeric). The scan stops on the found string or on the first carriage return encountered, whichever comes first.

Entry Conditions

A	=	49
(HL)	=	text area to search
(DE)	=	compare string
B	=	length of the compare string

Exit Conditions

Z flag	=	String is found
NZ	=	Error; check Register A
		A = Ø: string is not found
		A ≠ Ø: error code
(HL)	=	start position of the matching string in the search string

TIMER **Function Code 25**
Set Timer

Lets you set a "seconds clock" to run at the same time as your program and to interrupt the program when time runs out. For example, you can specify the number of seconds for keyboard input. If input is not made within the time limit, TRSDOS-II interrupts the keyboard input routine. It interrupts, also, if you reset the timer.

After the timer counts to zero and causes an interrupt, it shuts off by itself.

See "Programming with TRSDOS-II" for information on interrupts.

Entry Conditions

A = 25
(HL) = SVC to handle interrupt
BC = number of seconds to count down

If HL and BC both = Ø, the timer turns off. If HL = Ø and BC ≠ Ø, the time count resets to the value in BC, and timing continues.

Exit Conditions

None

VDCHAR Function Code 8
Video Character

Outputs a character to the current cursor position. It is a scroll mode routine.

Control codes not listed below are ignored.

Key	Hex Code	Function
<F1>	Ø1	Turns on the blinking cursor.
<F2>	Ø2	Turns off the cursor.
<CTRL> <D> or <F3>	Ø4	Turns on the steady cursor.
<CTRL> <G>	Ø7	Sounds bell tone for approximately .25 second (in computers that generate sound).
<BACKSPACE>	Ø8	Moves the cursor back one position and blanks the character at that position.
<TAB>	Ø9	Advances the cursor to the next tab position. Tab positions are at 8-character intervals -- 8, 16, 24, 32, and so on.
<CTRL> <J>	ØA	Moves the cursor down to the next row, in the same column.
<CTRL> <K>	ØB	Moves the cursor to the beginning of the previous line. The cursor does not move into the scroll-protected area.
<ENTER>	ØD	Moves the cursor down to the beginning of the next line.
<CTRL> <N> or <F7>	ØE	Turns on dual routing.

Key	Hex Code	Function
<CTRL> <O>	ØF	Turns off dual routing.
<CTRL> <T>	14	Homes the cursor to the first non-scroll-protected position.
<CTRL> <W>	17	Erases to the end of the line. The cursor doesn't move.
<CTRL> <X>	18	Erases to the end of the screen. The cursor doesn't move.
<CTRL> <Y>	19	Sets the normal display mode (green/white on black). Remains normal until you reset it.
<CTRL> <Z>	1A	Sets the reverse display mode (black on green/white). Remains reverse until you reset it.
<ESC>	1B	Erases the screen and homes the cursor to Position Ø.
<left arrow>	1C	Moves the cursor back one space.
<right arrow>	1D	Moves the cursor forward one space.
<up arrow>	1E	Sets 8Ø characters per line and clears the display.
<down arrow>	1F	Sets 4Ø characters per line and clears the display.

Entry Conditions

A = 8
B = ASCII code for the character to be output to the display. Character codes **must** be in the range [Ø,127].

Exit Conditions

NZ = Error
A = error code

VDGRAF **Function Code 1Ø**
Video Graphics

Displays a buffer of characters, starting at the row and column you specify. It is a graphics mode routine (the cursor "wraps" the display).

Displayable Characters

This routine lets you display the 32 graphics characters (and their reverse images).

The codes are numbered from Ø through X'1F' and are pictured in the operator's manual. Codes X'2Ø' through X'7F' are displayed as standard ASCII characters.

In addition, several special control codes are available:

Hex Code	Result
F9	Sets the normal (green/white on black) mode. The cursor does not move. The mode returns to its previous state after the VDGRAF call is completed.
FA	Sets the reverse (black on green/white) mode. The cursor does not move. The mode returns to its previous state after the VDGRAF call is completed.
FB	Homes the cursor (Row Ø, Column Ø).
FC	Moves the cursor back one space. Column = Column - 1. When the column = Ø, the cursor wraps to Column 79 on the preceding row.
FD	Moves the cursor forward one space. Column = Column + 1. When the column = 79, cursor the wraps to Column Ø on the next row.
FE	Moves the cursor up one row. Row = Row - 1. When the row = Ø, the cursor wraps to Row 23.
FF	Moves the cursor down one row. Row = Row + 1. When the row = 23, the cursor wraps to Row Ø.

At exit, the cursor always is set automatically to the graphics position immediately after the last character

displayed. If the buffer length = Ø, the cursor is set to the position specified in Register Pair BC.

Entry Conditions

A = 1Ø
B = row on the screen to start displaying the buffer
B < 24. For B > 23, use B modulo 24* as the row position.
C = column on the screen to start displaying the buffer
In the 8Ø characters-per-line mode, C < 8Ø. For C > 79, use C modulo 8Ø as the column position.
In the 4Ø characters-per-line mode, C < 4Ø. For C > 39, use C modulo 4Ø as the column position.
D = buffer length (in range [Ø,255])
(HL) = beginning of the text buffer
The buffer should contain codes below X'8Ø' or the special control codes above X'F8'. Any value outside these ranges causes an error.

Exit Conditions

NZ = Error (invalid character sent)
A = error code

*MODULO - A cyclical counting system, in which number1 modulo number2 is the integral remainder after division of number1 by number2. For example: 35 modulo 24 = 11.

VDINIT **Function Code 7**

Video Initialization

Blanks the screen and resets the cursor to the upper left corner (Position Ø in the scroll mode illustration at the beginning of this section). Call VDINIT once before starting any I/O to the display.

Entry Conditions

A = 7
B = character size switch
  B = Ø: 4Ø characters per line
  B ≠ Ø: 8Ø characters per line
C = normal/reverse switch
  C = Ø: reverse mode
  C ≠ Ø: normal mode

Exit Conditions

NZ = Error
A = error code

VDLINE **Function Code 9**
Video Line

Writes a buffer of data to the display, starting at the current cursor position. The buffer should contain ASCII codes in the range [Ø,127].

Control codes that are greater than than X'2Ø' are ignored. (For more information, see VDCHAR.)

VDLINE is a scroll mode routine.

Entry Conditions

A = 9
(HL) = beginning of the text buffer
B = number of characters to be sent
C = end-of-line character
This character is sent to the display after the buffer text.

Exit Conditions

NZ = Error
A = error code

In case of an error:

B = number of characters **not** displayed, including the one causing the error
C = character causing the error

Upon return, the cursor is set automatically to the position following the last character displayed.

VDREAD **Function Code 11**
Video Read

Reads characters from the display into the buffer you specify. It is a graphics mode routine; when it reads past the last column, it wraps back to Column Ø on the next row. When it reads past Column 79 on Row 23, it wraps back to Column Ø, Row Ø.

TRSDOS-II reads reverse (black on green/white) mode characters in ASCII codes, just as it does their normal counterparts. Reverse mode is indicated when the most significant bit (bit 7) is set.

VDREAD can also be used to locate the cursor (see below).

Entry Conditions

A = 11
B = row on screen where the read starts, B < 24
If B > 23, use B modulo 24 as the row position.
C = column on screen where the read starts
In the 8Ø characters-per-line mode, C < 8Ø. For C > 79, use C modulo 8Ø as the column position.
In the 4Ø characters-per-line mode, C < 4Ø. For C > 39, use C modulo 4Ø as the column position.
D = buffer length (in range [Ø,255])
D = Ø: return the cursor position
D ≠ Ø: read the display (B = row, C = column)
(HL) = address of the text buffer

Exit Conditions

BC = current cursor position (B = row, C = column)
The cursor position at exit is the same as at entry; VDREAD doesn't change it.
NZ = Error
A = error code

VIDKEY **Function Code 12**
Combines VDLINE and KBLINE

Sends a prompting message to the display and then waits for a line from the keyboard. VIDKEY is a scroll mode routine which combines the functions of VDLINE and KBLINE.

Starting at the current cursor position, it displays the text buffer you specify. The buffer must contain codes less than X'8Ø'. (See the VDLINE SVC for a list of received control codes and other details.)

Once the text message is displayed, TRSDOS-II positions the cursor immediately after the last character displayed. (To move it to another position, you must place a control code at the end of the text buffer.)

VIDKEY then uses KBLINE to get a line from the keyboard.

(See the KBLINE SVC for a list of received control codes and other details.)

Entry Conditions

A = 12
B = number of characters to be displayed (in the range [Ø,255])
C = length of the keyboard input field (in the range [Ø,255])
(HL) = beginning of the text buffer containing the display message
(DE) = beginning of the text buffer where the keyboard input is to be stored

Exit Conditions

NZ = Error
A = error code

If Z is set (no error):

B = number of characters input from keyboard, including carriage return, if any
C = keyboard line terminator
C = Ø: input buffer was filled
C ≠ Ø: C contains the control character that ended the line (carriage return)

If Z is not set (error):
B = number of characters not displayed, including the one causing the error
C = character causing the error

VIDRAM **Function Code 94**

Transfer Video Display to RAM or Vice-Versa

Copies a screenful of graphics and/or text from a RAM buffer to the video display, or vice-versa. The programmer must be aware of the current display mode -- 8Ø characters per line or 4Ø characters per line.

Every possible data byte is treated as a displayable character. There are no cursor position or other control codes. The following table summarizes the character/code relationships for this routine:

Hex Code	Display Character
ØØ-1F	Normal Graphics
2Ø-7F	Normal ASCII Text
8Ø-9F	Reversed Graphics
AØ-FF	Reversed ASCII Text

Entry Conditions

A = 94

B = function code
B = Ø: copy from RAM to video
B ≠ Ø: copy from video to RAM

(HL) = RAM buffer The start of the buffer must be above X'28ØØ'; the end must be below X'FØØØ'.
If the video is in the 8Ø characters-per-line mode, the buffer must be 8Ø * 24 = 192Ø bytes long.
If the video is in the 4Ø characters-per-line mode, the buffer must be 4Ø * 24 = 96Ø bytes long.

Exit Conditions

None
(VIDRAM does not change the cursor position.)

WILD
Wildcard Parser

Function Code 51

Compares a filespec with a wildcard spec, depending upon the contents of B, the function code entry condition.

A mask is a filespec with asterisk(s) inserted in one or more positions in the name or extension. An asterisk means "one or more characters -- don't care what they are." For example:

- */BAS masks all files except those that have the extension /BAS.
- *LST masks all files except those that have a filename ending in LST and no extension.

The filespec is a standard TRSDOS-II filespec, terminated by a carriage return.

For details on wildcard specs, see "Wildcard" in the "Introduction."

Contents of B	Result
Ø	Sets the wildcard mask
1	Compares the filespec with the mask
2	Combines a separate name and extension into a filespec

Note: Calling any other TRSDOS-II SVC may clear the mask. Therefore, you should use this procedure:

1. Get the filespec(s) to be checked.
2. Set the mask by calling WILD with B = Ø.
3. Compare the filespec with the mask by calling WILD with B = 1.
4. Repeat Step 3 until all filespecs have been checked.
5. Each time you call another TRSDOS-II SVC, you may have to reset the mask.

Entry Conditions

A = 51
B = function switch
B = Ø: set the wildcard mask
B = 1: compare the filespec to the mask
B = 2: combine into filespec

If B = Ø:
(HL) = wildcard mask

If B = 1:
(HL) = filespec

If B = 2:
(HL) = 11-byte buffer
Bytes Ø-7 = filename; bytes 8-1Ø = extension
You must justify both fields on the left, leaving trailing spaces as needed to fill the field. The format is:

N	A	M	E	␣	␣	␣	␣	E	X	T

The symbol ␣ represents a blank space.

(DE) = 13-byte destination buffer
This buffer is to hold the compressed filespec contained in (HL).

Exit Conditions

If B = Ø:
NZ = Invalid mask specification

If B = 1:
NZ = File does not match or you have set no mask

If B = 2:
(DE) = 13-byte buffer containing filespec
The filespec is created from the 11-byte source text.

WRITNX Function Code 43
Write Next Record

Writes the record following the last record accessed. If WRITNX is the first access after the file is opened, TRSDOS-II writes the first record.

Before calling WRITNX, put your record in the record area pointed to by RECADR in the parameter list. Or, if the record length is 256 and the record type is fixed, your record is in the area pointed to by BUFADR.

Entry Conditions

A = 43
(DE) = data control block for the currently open file
HL = Reserved for use in later versions of TRSDOS-II

Exit Conditions

NZ = Error
A = error code

APPENDICES

APPENDIX A
TRSDOS-II
Character Codes

Code		Character		
			Video Display	
Dec.	Hex.	Keyboard	Scroll mode	Graphics mode
00	00	HOLD		
01	01	F1	Turns on blinking cursor	
02	02	F2	Turns off cursor	
03	03	BREAK * CTRL C		
04	04	F3 CTRL D	Turns on steady cursor	
05	05	CTRL E		
06	06	CTRL F		
07	07	CTRL G		
08	08	BACKSPACE CTRL H	Backspaces cursor and erases character	
09	09	TAB CTRL I	Advance cursor to next 8-character boundary	
10	0A	CTRL J	Line feed	
11	0B	CTRL K	Cursor to previous line.	
12	0C	F4 CTRL L		
13	0D	ENTER CTRL M	Carriage return	
14	0E	F7 CTRL N	Dual routing on.	
15	0F	CTRL O	Dual routing off.	
16	10	F6 CTRL P		
17	11	CTRL Q		
18	12	CTRL R		
19	13	F8 CTRL S		
20	14	CTRL T	Homes cursor in scroll area.	
21	15	F5 CTRL U		
22	16	CTRL V		
23	17	CTRL W	Erase to end of line	
24	18	CTRL X	Erase to end of screen	
25	19	CTRL Y	Sets white-on-black mode	
26	1A	CTRL Z	Sets black-on-white mode	
27	1B	ESC	Clears screen, homes cursor	
28	1C	←	Moves cursor back	
29	1D	→	Moves cursor forward	
30	1E	↑	Sets 80-character mode and clears display	
31	1F	↓	Sets 40-character mode and clears display	

* BREAK is always intercepted. It will **never** return a X'03'.

Code		Character		
		Key-board	Video Display	
Dec.	Hex.		Scroll mode	Graphics mode
32	20	Space Bar	∅	∅
33	21	!	!	!
34	22	"	"	"
35	23	#	#	#
36	24	$	$	$
37	25	%	%	%
38	26	&	&	&
39	27	'	'	'
40	28	(	(	(
41	29	)	)	)
42	2A	*	*	*
43	2B	+	+	+
44	2C	,	,	,
45	2D	-	-	-
46	2E	.	.	.
47	2F	/	/	/
48	30	0	0	0
49	31	1	1	1
50	32	2	2	2
51	33	3	3	3
52	34	4	4	4
53	35	5	5	5
54	36	6	6	6
55	37	7	7	7
56	38	8	8	8
57	39	9	9	9
58	3A	:	:	:
59	3B	;	;	;
60	3C	<	<	<
61	3D	=	=	=
62	3E	>	>	>
63	3F	?	?	?
64	40	@	@	@
65	41	A	A	A
66	42	B	B	B
67	43	C	C	C
68	44	D	D	D

Code		Character		
		Key-	Video Display	
Dec.	Hex.	board	Scroll mode	Graphics mode
69	45	E	E	E
70	46	F	F	F
71	47	G	G	G
72	48	H	H	H
73	49	I	I	I
74	4A	J	J	J
75	4B	K	K	K
76	4C	L	L	L
77	4D	M	M	M
78	4E	N	N	N
79	4F	O	O	O
80	50	P	P	P
81	51	Q	Q	Q
82	52	R	R	R
83	53	S	S	S
84	54	T	T	T
85	55	U	U	U
86	56	V	V	V
87	57	W	W	W
88	58	X	X	X
89	59	Y	Y	Y
90	5A	Z	Z	Z
91	5B	⊏	⊏	⊏
92	5C	CTRL 9	\	\
93	5D	⊐	⊐	⊐
94	5E	^	^	^
95	5F	—	—	—
96	60			`
97	61	A	a	a
98	62	B	b	b
99	63	C	c	c
100	64	D	d	d
101	65	E	e	e
102	66	F	f	f
103	67	G	g	g
104	68	H	h	h
105	69	I	i	i

Code		Character		
			Video Display	
Dec.	Hex.	Key-board	Scroll mode	Graphics mode
106	6A	J	j	j
107	6B	K	k	k
108	6C	L	l	l
109	6D	M	m	m
110	6E	N	m	n
111	6F	O	o	o
112	70	P	p	p
113	71	Q	q	q
114	72	R	r	r
115	73	S	s	s
116	74	T	t	t
117	75	U	u	u
118	76	V	v	v
119	77	W	w	w
120	78	X	x	x
121	79	Y	y	y
122	7A	Z	z	z
123	7B	{	{	{
124	7C	CTRL 0	¦	¦
125	7D	}	}	}
126	7E	CTRL 6	~	~
127	7F			±
128	80	*		
:	:			
:	:			
248	F8			
249	F9			Sets white-on-black mode
250	FA			Sets black-on-white mode
251	FB			Homes cursor
252	FC			Moves cursor back
253	FD			Moves cursor forward
254	FE			Moves cursor up
255	FF			Moves cursor down

* Codes 128-248 cannot be input from the keyboard or output to the display. When reading the display, a **value** greater than 127 indicates a reverse character corresponding to **value** mod 128.

APPENDIX B / Error Messages

There are three kinds of error messages you might get while using your computer:

- Boot errors, such as BOOT ERROR DC. See the Boot Errors Table for more information.

- Operating system errors, such as ERROR 24 or FILE NOT FOUND. To get a brief description of a numbered error, type ERROR followed by the error number displayed. For example, type:

  ERROR 31 <ENTER>

  and your screen shows:

  PROGRAM NOT FOUND

  For more information see the System Errors Table.

- Application program errors -- see your application program manual.

When an error message is displayed:

- Try the operation several times.

- Look up boot errors and operating system errors in the following tables and take the recommended actions. See your application program manual for explanations of application program errors.

- Trying using other diskettes.

- Reset the computer and try the operation again.

- Check all the power connections.

- Check all interconnections.

- Remove all diskettes from drives, turn off the computer, wait 15 seconds, and turn it on again.

- If you try all these remedies and continue to get an error message, contact a Radio Shack Service Center.

Note: If more than one thing is wrong, the computer might wait until you correct the first error before displaying the second error message.

RSSC = Radio Shack Service Center.

Numerical System Errors Table

Register A usually has a return code after any function call. The Z flag is set when no error occurs. Exceptions are certain computational routines, which use Registers A and F to pass back data and status information.

Code	Message	Explanation/Action
Ø	No Error Found.	No error occurred.
1	Bad Function Code On SVC Call Or No Function Exists.	Check the function code number used on the SVC call.
2	Character Not Available.	No record or character was available when you you called the SVC.
3	Parameter Error On Call.	Parameter is incorrect or a required parameter is missing.
4	CRC Error During Disk I/O.	Try the operation again, using a different diskette. If the problem occurs often, contact RSSC.
5	Disk Sector Not Found.	Try a different diskette.
6	Attempt To Open A File Which Has Not Been Closed.	Close the file before re-opening.
7	Illegal Disk Change.	TRSDOS-II detected an illegal disk swap.
8	Disk Drive Not Ready.	Drive door is open or the diskette is not in the drive. On Thinline drives, check the DRIVE command settings.

Code	Message	Explanation/Action
9	Invalid Data Provided By Caller.	Data stream to be processed has illegal characters.
1Ø	Maximum Of 16 Files May Be Open At Once.	Too many files are open at once.
11	File Already In Directory.	Filename already exists as a directory entry. Kill the existing file, choose another filename, or specify a drive number.
12	No Drive Available For An Open.	No on-line drive a. is write enabled b. has enough space to create a new file, or c. has a system directory.
13	Write Attempt To A Read Only File.	File was opened for read only, not for read/write.
14	Write Fault On Disk I/O.	Error occurred during a write operation. Try a different diskette. If the problem continues, contact RSSC.
15	Disk Is Write Protected.	Write enable the disk.
16	DCB Is Modified And Is Unusable.	DCB (used in machine-language programming) has been modified since the last disk access (while the file was open).
17	Directory Read Error.	Error occurred during an attempt to read the directory. Use a different diskette.

Code	Message	Explanation/Action
18	Directory Write Error.	Error occurred during an attempt to write to the directory. Use a different diskette.
19	Improper File Name (Filespec).	Filespec you gave does not meet TRSDOS-II standard file specifications.
2Ø	** Unknown Error Code **	
21	** Unknown Error Code **	
22	** Unknown Error Code **	
23	** Unknown Error Code **	
24	File Not Found.	Filename you gave was not found on the available disks or the file is the incorrect type for the desired operation.
25	File Access Denied Due To Password Protection.	You gave an incorrect password. See the ATTRIB command.
26	Directory Space Full.	Number of filenames has reached the amount set when you formatted the disk.
27	Disk Space Full.	No space is available on the disk.
28	Attempt To Read Past EOF.	Specified record number is past the EOF.
29	Read Attempt Outside Of File Limits.	Use valid record numbers.

Code	Message	Explanation/Action
3Ø	No More Extents Available (16 Maximum).	Use the COPY command to copy the files and reduce fragmentation. See also SAVE/RESTORE and MOVE.
31	Program Not Found.	Specified program is not found on the available disks.
32	Unknown Drive Number (Filespec).	Specified drive number is not valid.
33	Disk Space Allocation Cannot Be Made Due To Fragmentation Of Space.	Use the COPY command to copy the files and reduce fragmentation.
34	Attempt To Use A NON Program File As A Program.	File specified for execution is not a program file or the load address given is illegal. Make sure you have a system diskette in Drive Ø.
35	Memory Fault During Program Load.	Program is loaded incorrectly, possibly because of faulty memory or a "bad" load address.
36	Parameter For Open Is Incorrect.	Check the OPEN statements or the DCB for errors.
37	Open Attempt For A File Already Open.	File specified for open is already open.
38	I/O Attempt To An Unopen File.	Open the file before access.

Code	Message	Explanation/Action
39	Illegal I/O Attempt.	a. I command not given after a diskette swap. b. Can be caused by an I/O attempt to a differently formatted disk. Format the disk under the current version of TRSDOS-II or use FCOPY.
4Ø	Seek Error.	a. Data cannot be read from the disk -- faulty disk. b. When re-initializing a hard disk, you must also reformat the secondary drives.
41	Data Lost During Disk I/O (Hardware Fault).	Contact RSSC.
42	Printer Not Ready.	Check the connections, power, ribbon, on-line status, and so on.
43	Printer Out Of Paper.	Check the printer's paper supply.
44	Printer Fault (May Be Turned Off).	Check the connections, power, ribbon, on-line status, and so on.
45	Printer Not Available.	Check the connections, power, ribbon, on-line status, and so on.
46	Not Applicable To VLR Type Files.	Operation performed is not valid for VLR files.
47	Required Command Parameter Not Found.	Required parameter or argument is missing from the command.

Code	Message	Explanation/Action
48	Incorrect Command Parameter.	Option or argument given in the command is incorrect.
49	Hardware Fault During Disk I/O.	Contact RSSC.
5Ø	Invalid Space Descriptor.	The space descriptor that tells TRSDOS-II which extent to read next is invalid. Try a different diskette.
51-255	** Unknown error code **	

Alphabetical System Errors Table

Message	Explanation/Action	Code
Attempt To Open A File Which Has Not Been Closed.	Close the file before re-opening.	6
Attempt To Read Past EOF.	Specified record number is past the EOF.	28
Attempt To Use A NON Program File As A Program.	File specified for execution is not a program file or the load address given is illegal. Make sure you have a system diskette in Drive Ø.	34
Bad Function Code On SVC Call Or No Function Exists.	Check the function code number used on the SVC call.	1
Invalid Space Descriptor.	The space descriptor that tells TRSDOS-II which extent to read next is invalid. Try a different diskette.	5Ø
BOOT ERROR	See the BOOT ERROR TABLE.	
Character Not Available.	No record or character was available when you called the SVC.	2
CRC Error During Disk I/O.	Try the operation again, using a different diskette. If the problem occurs often, contact RSSC.	4
DCB Is Modified And Is Unusable.	DCB (used in machine-language programming) has been modified since the last disk file access (while the file was open).	16

Message	Explanation/Action	Code
Data Lost During Disk I/O (Hardware Fault).	Contact RSSC.	41
Directory Read Error.	Error occurred during an attempt to read the directory. Try a different diskette.	17
Directory Space Full.	Number of filenames has reached the amount set when you formatted the diskette.	26
Directory Write Error.	Error occurred during an attempt to write to the directory. Use a different diskette.	18
Disk Drive Not Ready.	Drive door is open or the diskette is not in the drive. On Thinline drives, check the Drive command settings.	8
Disk Is Write Protected.	Write enable the disk.	15
Disk Sector Not Found.	Try a different diskette.	5
Disk Space Allocation Cannot Be Made Due To Fragmentation Of Space.	Use the COPY command to copy the files and reduce fragmentation.	33
Disk Space Full.	No space is available on the disk.	27
File Access Denied Due To Password Protection.	You gave an incorrect password. See ATTRIB command.	25
File Already In Directory.	Filename already exists as a directory entry. Kill the existing file, choose another filename, or specify a drive number.	11

Message	Explanation/Action	Code
File Not Found.	Filename you gave was not found on the available disks or the file is the incorrect type for the desired operation.	24
Hardware Fault During Disk I/O.	Contact RSSC.	49
I/O Attempt To An Unopen File.	Open the file before access.	38
Illegal I/O Attempt.	a. I command not given after a diskette swap. b. Can be caused by an I/O attempt to a differently formatted disk. Format the disk under the current version of TRSDOS-II or use FCOPY.	39
Illegal Disk Change.	TRSDOS-II detected an illegal disk swap.	7
Improper File Name (Filespec).	Filespec you gave does not meet TRSDOS-II standard file specifications.	19
Incorrect Command Parameter.	Option or argument given in the command is incorrect.	48
Invalid Data Provided By Caller.	Data stream to be processed has illegal characters.	9
Maximum Of 16 Files May Be Open At Once.	Too many files are open at once.	1Ø
Memory Fault During Program Load.	Program is loaded incorrectly, possibly because of faulty memory or a "bad" load address.	35
No Drive Available For An Open.	No on-line drive a. is write enabled b. has enough space to create a new file, or c. has a system directory.	12

Message	Explanation/Action	Code
No Error Found.	No error occurred.	Ø
No More Extents Available (16 Maximum).	Use the COPY command to copy the files and reduce fragmentation. See also SAVE/ RESTORE and MOVE.	3Ø
Not Applicable To VLR Type Files.	Operation performed is not valid for VLR files.	46
Open Attempt For A File Already Open.	File specified for open is already open.	37
Parameter Error On Call.	Parameter is incorrect or a required parameter is missing.	3
Parameter For Open Is Incorrect.	Check the OPEN statements or the DCB for errors.	36
Printer Fault (May Be Turned Off).	Check the connections, power, ribbon, on-line status, and so on.	44
Printer Not Available.	Check the connections, power, ribbon, on-line status, and so on.	45
Printer Not Ready.	Check the connections, power, ribbon, on-line status, and so on.	42
Printer Out Of Paper.	Check the printer's paper supply.	43
Program Not Found.	Specified program is not found on available disks.	31
Read Attempt Outside Of File Limits.	Use valid record numbers.	29
Required Command Parameter Not Found.	Required parameter or argument is missing from the command.	47

Message	Explanation/Action	Code
Seek Error.	a. Data cannot be read from the disk -- faulty disk. b. When re-initializing a hard disk, you must also reformat the secondary drives.	4Ø
Unknown Drive Number (Filespec).	Specified drive number is not valid.	32
** Unknown Error Codes **		51-127
Write attempt to a read only file.	File was opened for read only, not for read/write.	13
Write fault on disk I/O.	Error occurred during a write operation. Try a different diskette. If the problem continues, contact RSSC.	14

Boot Errors Table

Error	Message	Explanation/Action
BOOT ERROR CK	Checksum error -- possibly a defective ROM.	Contact RSSC.
BOOT ERROR CT	Defective CTC chip.	Contact RSSC.
BOOT ERROR DC	Floppy disk controller error. a. Defective diskette. b. Floppy disk expansion unit. not on. c. Defective FDC Chip or Drive.	 a. Try a different diskette. b. Turn on the floppy disk expansion unit. c. Contact RSSC.
BOOT ERROR DM	DMA chip failure.	Contact RSSC.
BOOT ERROR DØ	Drive not ready. a. Improperly inserted diskette. b. Defective diskette. c. Defective drive.	 a. Insert the diskette again and press <RESET>. b. Try a different diskette. c. Contact RSSC.
BOOT ERROR HA	Controller error. Aborted command: Problem during boot-up of hard disk.	Re-initialize the hard disk or contact RSSC.
BOOT ERROR HC	CRC error. Invalid data in data field.	Re-initialize the hard disk or contact RSSC.

Error	Message	Explanation/Action
BOOT ERROR HD	Controller error. Busy not reset.	a. Re-initialize the hard disk. b. Power down, wait 1Ø seconds, and power up. If the error occurs again, contact RSSC.
BOOT ERROR HI	CRC error. Invalid data in ID field.	Re-initialize the hard disk.
BOOT ERROR HM	Data address mark not found.	Re-initialize the hard disk.
BOOT ERROR HN	ID not found. No Boot Track.	Re-initialize the hard disk.
BOOT ERROR HØ	Track Ø error on hard disk. a. Didn't find Track Ø before time-out. b. Secondary hard disk drives not turned on.	 a. Press <RESET>. b. Turn on your secondary hard disk drives.
BOOT ERROR HT	Time-out while waiting for Ready. a. Hard disk drive not powered up. b. Hard disk drive isn't turned on and ready within 1Ø seconds after the computer. c. Hard disk drive is disconnected.	 a. Follow correct procedure: Turn on the hard disk first. b. Press <RESET>. c. Connect the hard disk drive or operate under floppy disk control.

Error	Message	Explanation/Action
BOOT ERROR LD	Lost data during read -- FDC (floppy disk controller) or drive fault.	Try another TRSDOS-II diskette or contact RSSC.
BOOT ERROR MF	Memory failure in address range X'1ØØØ'-X'7FFF'.	Contact RSSC.
BOOT ERROR MH	Memory failure in address range X'8ØØØ'-X'FFFF'.	Contact RSSC.
BOOT ERROR ML	Memory failure in address range X'ØØØØ'-X'ØFFF'.	Contact RSSC.
BOOT ERROR PI	Defective PIO Chip.	Turn on the expansion bay if it is off. If the error occurs again, contact RSSC.
BOOT ERROR RS	The diskette in Drive Ø is not Radio Shack operating system format.	Insert a TRSDOS-II formatted diskette into Drive Ø and press <RESET>.
BOOT ERROR SC	CRC Error. Invalid data on diskette or defective diskette.	Try a different diskette.
BOOT ERROR TK	Record not found bootstrap track. Improperly formatted or defective diskette.	Re-format your diskette or try a different diskette.

Error	Message	Explanation/Action
BOOT ERROR Z8	Defective CPU.	Contact RSSC.
NOT A SYSTEM DISK	Diskette in Drive Ø isn't a TRSDOS-II operating system diskette.	Insert a TRSDOS-II operating system diskette into Drive Ø.

APPENDIX C / Alphabetical SVC Quick Reference List

Name	Description	No.
ACTRL	Performs control functions on Channel A	1ØØ
ARCV	Returns a character from Channel A	96
ATX	Outputs a character to Channel A	97
BCTRL	Performs control functions on Channel B	1Ø1
BINDEC	Converts binary to ASCII-coded decimal, and vice versa	21
BINHEX	Converts binary to ASCII-coded hexadecimal, and vice versa	24
BRCV	Returns a character from Channel B	98
BTX	Outputs a character to Channel B	99
CLOSE	Terminates access to an open file	42
CLRXIT	Clears RAM and returns to TRSDOS-II	57
CURSOR	Turns the blinking cursor on or off	26
DATE	Sets or returns the date and time	45
DELAY	Provides a delay routine	6
DIRRD	Reads the specified record (direct access)	35
DIRSET	Gets directory information on an open file	59
DIRWR	Writes the specified record (direct access)	44
DISKID	Reads the diskette ID(s)	15

Name	Description	No.
DOSCMD	Sends TRSDOS-II a command and then returns to TRSDOS-II Ready	37
ERRMSG	Returns an error message to buffer	52
ERROR	Displays "ERROR number"	39
FILPTR	Gets the pointers of an open file	58
HLDKEY	Suspends and restarts terminal output whenever you press <HOLD>	29
INITIO	Initializes all input/output drivers	Ø
JP2DOS	Returns to TRSDOS-II Ready	36
KBCHAR	Gets a character from the keyboard	4
KBINIT	Clears stored keystrokes	1
KBLINE	Gets a line from the keyboard	5
KBPUT	Puts a character in the key-ahead buffer	3Ø
KILL	Deletes the file from the directory	41
LOCATE	Returns the current record number	33
LOOKUP	Searches through a table	28
MPYDIV	Performs multiplication or division with one 2-byte value and one 1-byte value	23
OPEN	Sets up access to a new or existing file	4Ø
PARSER	Finds the alphanumeric parameter field in a text string	46
PRCHAR	Sends a character to the printer	18
PRCTRL	Controls printer operations	95

Name	Description	No.
PRINIT	Initializes the line printer driver	17
PRLINE	Sends a line to the printer	19
RAMDIR	Gets directory information into RAM	53
RANDOM	Provides a random number in the range [Ø,254]	2Ø
RDDIR	Reads a directory record	32
READNX	Gets the next record (sequential access)	34
RENAME	Renames a file	47
RETCMD	Sends TRSDOS-II a command and a return to caller	38
REWIND	Rewinds a disk file	48
RS232C	Sets or turns off Channel A or B for serial input/output	55
SCROLL	Sets the number of lines at the top of the display that are not scrolled	27
SETBRK	Sets up the <BREAK> key processing program	3
SETUSR	Sets up a user-defined SVC	2
SOUND	Turns on a tone	6Ø
SORT	Sorts a list in RAM	56
SOUND	Turns on a single tone	6Ø
STCMP	Compares two text strings	22
STSCAN	Looks for a specified string inside a text buffer	49

Name	Description	No.
TIMER	Sets a timer to interrupt a program	25
VDCHAR	Sends a character to the video (scroll mode)	8
VDGRAF	Sends a character to the video (graphics mode)	1Ø
VDINIT	Initializes the display	7
VDLINE	Sends a line to the video (scroll mode)	9
VDREAD	Reads characters from the video (scroll mode)	11
VIDKEY	Displays a message and gets a line from the keyboard	12
VIDRAM	Copies a screenful of graphics and/or text from a RAM buffer to the display, or vice versa	94
WILD	Compares a file specification with a wildcard specification	51
WRITNX	Writes the next record (sequential access)	43

APPENDIX D / Numerical SVC Quick Reference List

No.	Name	Description
Ø	INITIO	Initializes all input/output drivers
1	KBINIT	Clears stored keystrokes
2	SETUSR	Sets up a user-defined SVC
3	SETBRK	Sets up the <BREAK> key processing program
4	KBCHAR	Gets a character from the keyboard
5	KBLINE	Gets a line from the keyboard
6	DELAY	Provides a delay routine
7	VDINIT	Initializes the display
8	VDCHAR	Sends a character to the video (scroll mode)
9	VDLINE	Sends a line to the video (scroll mode)
1Ø	VDGRAF	Sends a character to the video (graphics mode)
11	VDREAD	Reads characters from the video (graphics mode)
12	VIDKEY	Displays a message and gets a line from the keyboard
15	DISKID	Reads the diskette ID(s)
17	PRINIT	Initializes the line printer driver
18	PRCHAR	Sends a character to the printer

No.	Name	Description
19	PRLINE	Sends a line to the printer
2Ø	RANDOM	Provides a random number in the range [Ø,254]
21	BINDEC	Converts binary to ASCII-coded decimal, and vice versa
22	STCMP	Compares two text strings
23	MPYDIV	Performs multiplication or division with one 2-byte value and one 1-byte value
24	BINHEX	Converts binary to ASCII-coded hexadecimal, and vice versa
25	TIMER	Sets a timer to interrupt program
26	CURSOR	Turns the blinking cursor on or off
27	SCROLL	Sets the number of lines at the top of the display that are not scrolled
28	LOOKUP	Searches through a table
29	HLDKEY	Suspends and restarts terminal output whenever you press <HOLD>
3Ø	KBPUT	Puts a character into the key-ahead buffer
32	RDDIR	Reads a directory record
33	LOCATE	Returns the current record number
34	READNX	Reads the next record (sequential access)
35	DIRRD	Reads the specified record (direct access)
36	JP2DOS	Returns to TRSDOS-II Ready

No.	Name	Description
37	DOSCMD	Sends TRSDOS-II a command and then returns to TRSDOS-II Ready
38	RETCMD	Sends TRSDOS-II a command and a return to caller
39	ERROR	Displays "ERROR number"
4Ø	OPEN	Sets up access to a new or existing file
41	KILL	Deletes the file from the directory
42	CLOSE	Terminates access to an open file
43	WRITNX	Writes the next record (sequential access)
44	DIRWR	Writes the specified record (direct access)
45	DATE	Sets or returns the date and time
46	PARSER	Finds the alphanumeric parameter field in a text string
47	RENAME	Renames a file
48	REWIND	Rewinds a disk file
49	STSCAN	Looks for a specified string inside a text buffer
51	WILD	Compares a file specification with a wildcard specification
52	ERRMSG	Returns an error message to the buffer
53	RAMDIR	Gets directory information into RAM
55	RS232C	Sets or turns off Channel A or B for serial input/output

No.	Name	Description
56	SORT	Sorts a list in RAM
57	CLRXIT	Clears RAM and returns to TRSDOS-II
58	FILPTR	Gets the pointers of an open file
59	DIRSET	Gets directory information on an open file
6Ø	SOUND	Turns on a single tone
94	VIDRAM	Copies a screenful of graphics and/or text from a RAM buffer to the display, and vice versa
95	PRCTRL	Controls printer operations
96	ARCV	Returns a character from Channel A
97	ATX	Outputs a character to Channel A
98	BRCV	Returns a character from Channel B
99	BTX	Outputs a character to Channel B
1ØØ	ACTRL	Performs control functions on Channel A
1Ø1	BCTRL	Performs control functions on Channel B

APPENDIX E / Summary of the Differences between TRSDOS and TRSDOS-II

Table 1 / General Items

Item	How TRSDOS-II Differs
Method of Allocation	Allocation is done by sectors, instead of by granules.
Number of Files	Variable up to 122Ø files, instead of fixed at 96. Defaults to 18Ø for floppy diskettes and 336 for hard disks. See the FORMAT utility.
Technique for Locking Out Flaws	Flawed areas are locked out by sectors, instead of by tracks.
Reverse Video Mode	When in the reverse video mode (black characters on green/white background) and in the scroll mode, the video scrolls a green/white line instead of a black line.
Wildcards	You can use the symbol ! (exclamation point) as a super wildcard symbol. It is the same as */* (used for files that have extensions) and * (used for files that do not have extensions). For example: KILL !:3 allows all the user files in Drive 3 to be killed.
SETCOM with FORMS	You need not execute a SETCOM command before "FORMS S" when using a serial printer. However, you still must execute SETCOM before sending data to the printer.

Table 1 (continued)

Item	How TRSDOS-II Differs
Power-Up Diagnostics	New Diagnostic Errors: BOOT ERROR CT -- indicates a defective CTC chip. Contact your Radio Shack Service Center. BOOT ERROR MF -- indicates a memory failure in address range X'1ØØØ'-X'7FFF'. Contact your Radio Shack Service Center. BOOT ERROR ML -- indicates a memory failure in address range X'ØØØØ' - X'ØFFF'. Contact your Radio Shack Service Center.
Error Codes	New Error: ERROR 5Ø -- Invalid Space Descriptor. The space descriptor that tells TRSDOS-II which extent to read next is invalid. Try a different diskette. Errors 2Ø, 21, 22, and 23 are not used. BASIC also returns an FL Error (too many files). This error was previously undefined.
Memory Requirements	X'ØØØØ'-X'27FF' Resident Area X'28ØØ'-X'2FFF' Utility Command Overlay Area X'3ØØØ'-X'EFFF' User Area X'FØØØ'-X'FFFF' TRSDOS-II Demand Resident Area

Table 2 / Supervisor Calls (SVCs)

Name	How TRSDOS-II Differs	Code
DIRRD	Enhanced -- reads the specified record	35
DIRWR	Enhanced -- writes the specified record	44
DISKID	Enhanced -- reads disk ID	15
KBPUT	New -- puts characters into a key-ahead keyboard buffer	3Ø
LOCATE	Enhanced -- returns a record number	33
OPEN	Enhanced -- opens/creates a file	4Ø
RDDIR	New -- reads the next directory record and builds an ASCII string	32
REWIND	New -- rewinds a disk file to the beginning of the file	48
RS232C	Enhanced -- initializes Serial Communication Channels A and B	55
SOUND	New -- turns on a tone in computers that are capable of generating sound	6Ø

The new SVCs are created specifically for use under TRSDOS-II.

Note: We recommend restricting your use of the RAMDIR and FILPTR SVCs since each is restricted to 96 files. If you are not familiar with these supervisor calls, you should change your existing programs to use RDDIR instead.

Table 3 / Commands and Utilities

Name	How TRSDOS-II Differs
ANALYZE	No longer available -- cannot be used under TRSDOS-II
BACKUP	Enhanced -- duplicates floppy diskettes
BUILD	Enhanced -- creates an automatic command input file
CLICK	New -- Turns the key entry sound on or off for those computers that support sound.
CLOCK	No longer available -- cannot be used under TRSDOS-II
DIR	Enhanced -- displays a diskette's directory
DRIVE	New -- lets you gain the best use of the floppy disk drive by changing the seek rate, disk swap detect, and wait for drive ready status
FC	New -- edits and repeats the last command line entered
FCOPY	New -- transfers files on floppy diskette TRSDOS 1.2, 1.2a, 2.Ø, 2.Øa, or 2.Øb to a floppy diskette formatted under TRSDOS-II (and vice versa except to TRSDOS 1.2 and 1.2a)
FILES	New -- displays an alphabetical list of filenames in a disk's directory.
FLOPPY	New -- tells TRSDOS-II to ignore the drive numbers in all file specifications
FORMAT	Enhanced -- formats a disk for data storage

Table 3 (continued)

Name	How TRSDOS-II Differs
FREE	Enhanced -- displays the number of free contiguous sectors and the total amount of free space that are on a disk
HELP	Important note only -- gives the proper syntax for TRSDOS-II commands
I	Enhanced -- lets you swap diskettes
LIB	Enhanced -- displays the library commands
RESTORE	New -- lets you retrieve the information stored in compressed form by SAVE
SAVE	New -- lets you "compress" and store information onto diskettes for archive purposes
VERIFY	Enhanced -- verifies readable data
XFERSYS	No longer available -- cannot be used under TRSDOS-II

APPENDIX F / More About SAVE and RESTORE

Backing Up Your Hard Disk

What would you do if you suddenly lost the data stored on your hard disk system? Imagine the time it would take to re-enter all the data.

If, however, you have safe, floppy-diskette copies of the data, you would not have to re-enter all the data. You could simply restore the data to the hard disk, update the information, as needed, and continue.

Two programs on your hard disk assist you in creating such duplicates.

SAVE -- stores a specified group of files on a set of floppy diskettes. These diskettes are in a special, compact format that is not directly readable from TRSDOS-II.

In this format, files consume almost half the space they normally do on floppy diskettes. SAVE also lets you store files (from the hard disk) that are normally too large to fit on a floppy diskette.

RESTORE -- lets you retrieve the saved files to the hard disk.

This is the only way to recover saved diskettes. Trying to access a saved diskette by using other TRSDOS-II commands makes the system appear "locked up" for a short time while TRSDOS-II tries to read the saved diskette.

Saving Multiple Diskettes

Because the hard disk drive is a larger storage system than the floppy diskette, you sometimes need to save information onto more than one diskette. In such a case, SAVE prompts you to insert a new diskette.

There are two terms related to SAVE with which you need to be familiar:

- **Dataset** -- A set of one or more diskettes created by SAVE. Each dataset has a unique identifier, such as 84 4E 56. RESTORE uses the identifier to ensure that you do not insert a volume from a different dataset.

- **Volume** -- A diskette that is a member of a dataset. TRSDOS-II numbers each dataset's volumes in order from Ø.

When you are saving files that require more than one volume, TRSDOS-II prompts with:

```
Insert NEXT Blank Diskette on Drive n --
Press ANY Key to Continue.
```

After you do, TRSDOS-II prompts with:

```
The Diskette Presently on Drive n
will be referred to as "VOLUME 1"
```

TRSDOS-II saves the files and then prompts:

```
Insert "VOLUME Ø" on Drive n --
Press ANY Key to Continue
```

TRSDOS-II writes its housekeeping information -- including the number of volumes in the dataset -- to Volume Ø. RESTORE uses the information to retrieve the files.

Examples

There is a variety of ways to use SAVE. The simplest is:

```
SAVE 1 TO 2 {ALL} <ENTER>
```

This copies all the files from Drive 1 into a compact form on the diskette in Drive 2.

Wildcarding

Wildcards also offer an easy way to save several files or an entire disk. For example:

```
SAVE */CBL:4 TO Ø <ENTER>
```

saves all Drive 4 files with the extension /CBL and puts them on the diskette in Drive Ø.

Using the IND Option

The indirect option lets you save groups of files by creating an indirect file, a file consisting of one or more filespecs (similar to a DO file). You can use the BUILD command to create this list of filespecs.

At TRSDOS-II READY, type:

```
BUILD PROGRAMS:Ø <ENTER>
```

This creates an indirect file called PROGRAMS.

After TRSDOS-II prompts you with:

```
Enter Command Line (1-8Ø)
.....................................
```

Enter your list of file specifications including drive numbers, for example:

```
ORDERS:5 <ENTER>
REPORTS/*:6 <ENTER>
```

To exit the BUILD and return to TRSDOS-II READY, press <BREAK>.

You are now ready to save the files (specified by the indirect file) to the specially formatted floppy diskette. Type:

```
SAVE PROGRAMS:Ø TO 1 {IND} <ENTER>
```

Both ORDERS and REPORTS are now found in the file named PROGRAMS on the diskette in Drive Ø and saved to the diskette in Drive 1.

Note: The IND option lets you save more than one file from each hard disk; it also lets you save from more than one hard disk. As a result, you might save multiple files that have the same name. Because the save and restore directory does not specify drive numbers for files, you could lose duplicate filenames.

For example, if you created an indirect file that has these files:

```
*/FOR:4
*/CBL:4
*/FOR:5
```

Drives 4 and 5 may have duplicate filenames with the /FOR extension. Before you use indirect, examine all the files to be saved. Rename any duplicate filenames before saving or create different datasets.

Using the DC and DM Options

Another way to save files is to do so with respect to their creation or modification (update) dates. For example, suppose your directory showed these creation and update dates for your files:

Filename	Created	Updated
MENU/PRG	6/1/81	9/2/81
PRGONE/PRG	6/1/81	8/16/81
PRGTWO/PRG	6/1/81	7/3Ø/81
PRGTHR/PRG	6/1/81	6/16/81
PAYROLL/DAT	9/15/81	1Ø/15/81
CHECKS/DAT	9/15/81	1Ø/15/81
TEST/PRG	1Ø/29/81	1Ø/29/81

If you want to save only those files created on June 1, 1981, use the following command:

```
SAVE */*:5 TO Ø {DC=Ø6Ø181} <ENTER>
```

The first four files are saved to the floppy diskette in Drive Ø.

In the same sense, the first four files were updated on or before September 2, 1981 (9/2/81). Type:

```
SAVE */PRG:5 TO Ø (DM<Ø9Ø281) <ENTER>
```

and all files updated on or before the specified date are saved.

How Often Should You Save Your Files?

How often you should save your files depends upon you and how much data you enter. If you enter large amounts of data, you should make frequent backups, once or twice a day.

We suggest you keep two major sets of backup files:

- **Monthly Save Set** -- At the first of each month, make a save set of the complete hard disk, including your programs.
- **Daily Save Set** -- At the end of each day, or at most every third day, make a save set of all files that have been created or updated since the monthly save set was created.

If your hard disk fails, your monthly save set supplies most of your lost data and programs. After you restore this information and that from the daily save set, the amount you need to re-enter is minimal.

Creating a Monthly Save Set

Creating a monthly save set takes time, but it is worth it. Have a supply of blank diskettes ready. Do not format them; SAVE organizes the diskettes into its own special format.

Insert a blank diskette into Drive Ø. At TRSDOS-II Ready, type:

```
SAVE :4 :Ø (SYS,ALL,ABS)  <ENTER>
```

to save all programs and data files, including system files stored on Drive 4, the primary drive. The files are stored on diskettes in Drive Ø. As one diskette becomes full, TRSDOS-II prompts you for the next diskette.

When all of the files are saved, TRSDOS-II prompts you to insert Volume Ø (of your monthly save set) into Drive Ø. TRSDOS-II now updates that diskette with housekeeping information.

When the save is finished, you are returned to TRSDOS-II Ready. Label, number, and date the diskettes.

Rotating Monthly Save Sets

The set just created is the current monthly save set. At the beginning of the next month, create a new monthly save set using different diskettes. This set becomes the current set; the other becomes the previous set.

Rotate these two sets of diskettes when making monthly save sets, always using the previous set (older) to make the current set.

Creating A Daily Save Set

When you restore lost information, your daily save sets determine how much information you must re-enter. The more often you save, the less you re-enter.

Because the daily save sets are so important, you should keep two sets of them: the current daily set and the previous daily set. This way, if something happens to the current set, you have the previous set to fall back on.

The simplest way to create a daily save set is to save those files created or updated since you created the monthly save set. To do this, type:

```
SAVE !:4 :Ø {DM>mmddyy,ABS,SYS} <ENTER>
```

The !:4 specifies all files, with or without extensions. The option DM>mmddyy saves all files updated on or after the specified date. However, if you keep a monthly save set as suggested, this also saves files created after a specified date. (The update date and the creation date in the directory are the same when the file is created.)

For example, if you create a monthly set on January 1, 1983 (Ø1/Ø1/83), then use the following command to make a daily set:

```
SAVE !:4 :Ø {DM>Ø1Ø183,ABS,SYS}
```

This saves all files created or updated on or after January 1, 1983 up to the current date. Because the absolute option is specified, you are not prompted as each file is saved.

Rotating Daily Save Sets

The set just created is the current daily save set. At the end of the next working day, create a new daily save set using different diskettes. This set becomes the current set; the other becomes the previous set.

Rotate these two sets of diskettes when making daily save sets, always using the previous set (older) to make the current set.

Restoring Your Files

If your hard disk system fails, and you must recover all files, follow these steps:

1. If Drive 4 is not bootable, or the operating system is lost, you must re-initialize your system, following the instructions given in your Hard Disk Owner's Manual. If you are sure Drive 4's operating system is not damaged, proceed to step 2.

2. Insert Volume Ø of your monthly save set, into Drive Ø and type:

   ```
   RESTORE :Ø :4 {ABS,SYS} <ENTER>
   ```

3. Insert Volume Ø of your daily save set into Drive Ø and type:

   ```
   RESTORE :Ø :4 {ABS,SYS} <ENTER>
   ```

4. Re-enter any information added since the last current daily save set was created.

To restore one file, type the following command:

RESTORE filespec:Ø :4 {ABS} <ENTER>

where filespec is the name of the file you want restored.

To restore a group of files, type:

RESTORE :Ø :4 {PROMPT} <ENTER>

Restore prompts before restoring the files. Answer by pressing <Y> or <N> (for "yes" or "no").

Always restore files from the last save set available.

When a Boot Error Occurs on Hard Disk

If your hard disk system returns a boot error, flip the RESET switch on the front of your computer. Then, try to start up your system again. If your system continues to return a boot error, you probably have lost the boot track, Track Ø.

Even when this happens, there is a way to save the contents of your primary hard disk. But, to do so, you must have at least two floppy disk drives on your system.

To save the contents of your hard disk system:

1. Transfer control to the floppy disk system (press <BREAK> <REPEAT> during "white-out").

2. Insert a diskette containing the current floppy version of TRSDOS-II in Drive Ø and start up the system so that you see TRSDOS-II Ready.

3. To be sure there is a chance to save the contents of your hard disk, try to get a directory of your primary hard disk drive. Type:

   DIR 4 <ENTER>

If you can get a directory, then you probably can save the contents of your hard disk.

4. To save the contents of your primary hard disk drive, insert a blank diskette in Drive 1 and type:

   SAVE 4 TO 1 {SYS,ALL,ABS} <ENTER>

5. After the contents of your hard disk are saved, re-initialize your primary hard disk drive. (See your Hard Disk Owner's Manual for details.)

6. When the initialization is finished, (in about 15 to 20 minutes), you can restore the files that you saved. Type:

   RESTORE 1 TO 4 {SYS} <ENTER>

INDEX

Note: An asterisk is used, when necessary, to designate the page(s) on which you will find the entry for the particular command, utility, or SVC.

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG ROAD MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

8749422- 02/83-SP

Printed in U.S.A.