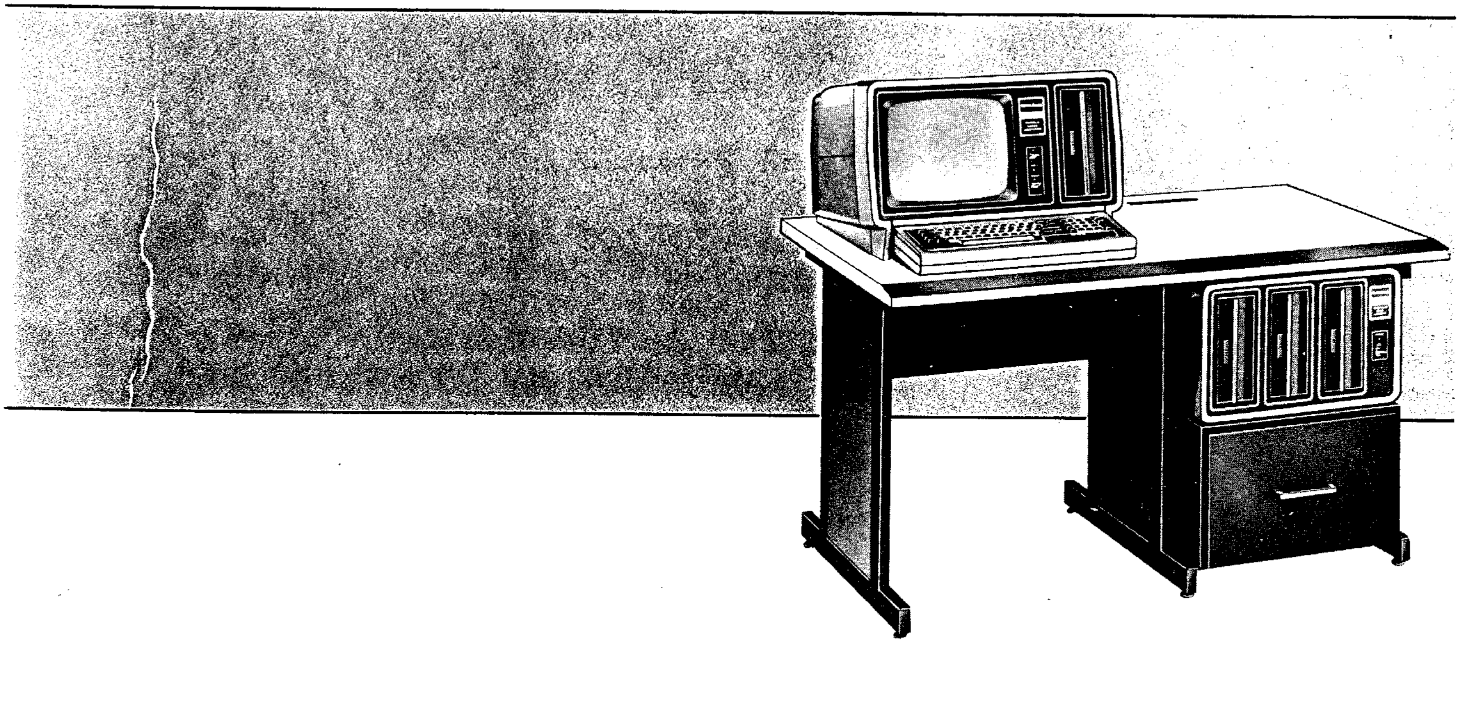
Radio Shack®

TRS-80 ™ Model II Disk Operating System Reference Manual

A Description of the Operating System: General Information, Operator Commands, Technical Information



CUSTOM MANUFACTURED IN THE U.S.A. BY RADIO SHACK T C A DIVISION OF TANDY CORPORATION

Section 0

New Release Update TRSDOS 2.0

This section pertains to the latest release of TRSDOS. It describes important operations you should perform before beginning any specific application. It also summarizes the changes and improvements contained in this version of TRSDOS.

Before attempting to follow any operations given in this section, you should read Section 1. For details on programs or utilities mentioned, see the specific entry for that subject (use the Index).

Important

Do not try to use 1.1 or 1.2 diskettes when the system is running under 2.0; do not try to use 2.0 diskettes when the system is running under 1.1/1.2. Either case could destroy the information on the diskette. 1.1/1.2 diskettes must be converted to 2.0, as explained in this section.

New Customers:

Ignore the references to TRSDOS 1.1/1.2; these are for customers who are upgrading to 2.0.

Important Disk Operations

In this section, we will outline some things you should do before you begin any applications with TRSDOS:

1. Initialize Some New Diskettes (FORMAT)
2. Duplicate the TRSDOS Diskette (BACKUP)
3. Make a Working Master (PURGE)
4. Upgrade Diskettes from Previous Versions (XFERSYS)

You will need to use several TRSDOS commands and utility programs. In this section we will list them briefly. For details, see the full descriptions elsewhere in this manual (use the Index).

1. Initialize Some New Diskettes (FORMAT)

Before any diskette can be used, it must be initialized or ''formatted'' — the data regions defined and labeled, and a table of contents or ''directory'' created.

The FORMAT utility program performs this function. We suggest you format several diskettes now. You cannot perform the other operations listed in this section until you have done this.

Start TRSDOS as explained in Chapter 1, then type in a command like this:

```
FORMAT:d
```

where *d* is one of the drives in your system.

The Computer will prompt you to mount the diskette for formatting.

If you are formatting in drive 0, you must **remove the TRSDOS diskette and insert the diskette to be formatted.**

The resultant formatted diskette will have the name TRSDOS and password PASSWORD. For diskettes that will contain TRSDOS, we suggest you use these default values. For your data diskettes, use any diskette name (ID = *name*) and any password (PW = *password*).

See FORMAT for further details.

2. Duplicate the TRSDOS 2.0 Diskette (BACKUP)

Use your factory-release TRSDOS diskette for one purpose only — as an "original master". It should not be used for applications, only for creating a "working master". You can then make changes to the working master to suit various applications.

As an added precaution, keep the factory-release diskette write-protected.

Note: In the BACKUP dialog, the term "source diskette" refers to the diskette to be duplicated; "destination diskette" refers to the diskette to contain the copy of the original.

Under TRSDOS READY, type in the appropriate BACKUP command. We suggest the following:

```
BACKUP Ø TO destination
```

where *destination* is the drive containing the formatted diskette which will become a TRSDOS diskette.

If *destination* = 0, TRSDOS will prompt you when to swap source and destination diskettes.

When the backup process is complete, remove the original diskette. Keep it in a safe place and don't use it unless you need to recreate the working master.

3. Make a Working Master (PURGE)

The factory-release TRSDOS diskette contains a variety of non-essential files. These non-essential files serve various purposes, including:

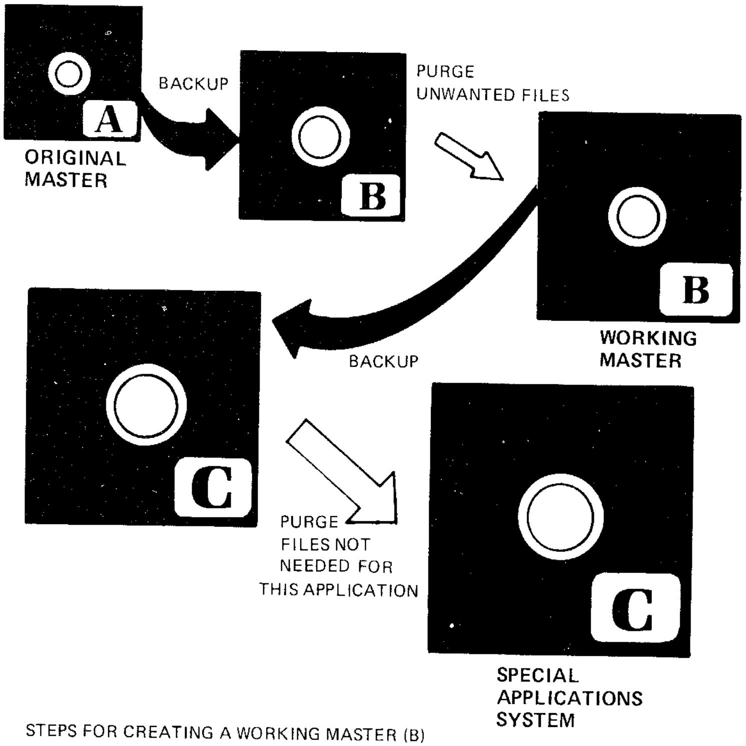
- Demonstration of programming techniques
- Modification of system input/output software
- Special features and functions

Now that your original TRSDOS diskette is safely set aside, you should create one or more customized system diskettes to suit your needs. For a discussion of "full system", "minimum system", and "data" diskettes, see **Disk File Requirements** later in this section.

Determine what files you will need for day-to-day use. You may want to have several different system diskettes — each with a different collection of system files.

If there are some files you don't want on *any* of your working diskettes, you may delete them from the working master. If there are files you don't want in a particular application, delete them from that diskette only.

To delete files, use the PURGE command.



STEPS FOR CREATING A WORKING MASTER (B)
AND A SPECIAL APPLICATION DISKETTE (C)

Here are the files included on the factory release TRSDOS 2.0 Diskette:

SYSTEM/SYS

This file is required on all TRSDOS diskettes. Do not purge it if you wish to use the diskette in drive 0.

SYSRES/SYS

This file is required on all full system diskettes, but not on minimum systems. (To find definitions, use the Index.) If you want to use the diskette to start TRSDOS, do not purge this file.

BACKUP, PATCH, XFERSYS, FORMAT, TERMINAL, MEMTEST, BASIC

These important system files should be kept on your working master. The purpose of each file is listed elsewhere in this manual (see entries in the Index). After you understand the purpose of each file, you may elect to delete any or all of them for specialized purposes.

SYSTEM64 and SYSTEM32

Only one of these two is needed in your system. If you have a 64K RAM system, keep SYSTEM64 and delete SYSTEM32; if you have a 32K RAM system, keep SYSTEM32 and delete SYSTEM64.

We will refer to the remaining file as SYSTEM*nn*. *nn* = 32 or 64, depending on which one you kept.

SYSTEM*nn* contains the high-memory routines for: HOST, SPOOL, SETCOM and all serial I/O, DO and DEBUG. You should keep this file on your working master. If you have an application which will not require any of the above-listed capabilities, you may delete it.

HERZ50, LPII, PRTBKSP

These are DO-files which allow you to make specific changes or ''patches'' to system files. See HERZ50, LPII and PRTBKSP for descriptions. If you don't need to make any of these changes, you may delete these files from your working master. Or if you do need to make them, delete them after they are used.

DOCOM*nn*, BASCOM*nn*, COMSUB*nn*, EXDATM*nn*, DATM*nn* (*nn* = 32 or 64)

These are demonstration programs and routines. See the descriptions elsewhere in this manual (use the Index).

For each file, there is a 32K RAM and a 64K RAM version. You should keep the version that matches your system's RAM size and delete the other.

These files are for demonstration and incidental purposes only. You may delete any or all of them from your working master, depending upon your needs. (Programmers should study how they work, to gain insights into techniques for using TRSDOS.)

4. Upgrade Diskettes from Previous Versions (XFERSYS)

This section applies only to customers who have been using a previous version of TRSDOS. New customers may skip it.

All diskettes — ''system'' and ''data'' — must be converted before they can be used under TRSDOS 2.0. The XFERSYS utility program does this for you. Attempts to use un-converted 1.2 diskettes under 2.0 may destroy the information on the diskettes.

See XFERSYS for the proper procedures to upgrading a diskette. Follow them carefully!

Disk File Requirements

There are three general configurations of diskettes that may be used in the TRSDOS system:

- **Full system** — The factory release copy of TRSDOS for use in drive 0.
- **Minimum system** — May be used in drive 0 *after* initialization.
- **Data** — May only be used in drives 1, 2 and 3.

"Full System"

This is a diskette which can be used to start ("bootstrap") the operating system and perform all the library commands, utilities, and supervisor calls. The factory release copy of TRSDOS is a full system.

The full system contains many files which will not be needed for day-to-day applications — but which may come in handy every now and then. For this reason, you should always keep your factory release version of TRSDOS intact, and use a backup copy as a working master.

Why and How of Full System Diskettes

If you have a Disk Expansion Unit, you will probably find it most convenient to have a Full System Diskette in drive 0 at all times. This will give you access to all of the utilities and system functions without a disk swap being necessary. For most applications, there is still plenty of room left on a full system diskette for program storage and for small to medium sized data files. Larger data files may be stored on data diskettes in the Disk Expansion Unit.

It is faster, by the way, to put different files or programs being accessed at the same time on different drives since, in most cases, the drive seek time will be reduced. On each drive, the read/write head will usually be closer to the desired location of the file currently being accessed. Using CREATE before building the data file will help insure that the file will be stored in a large block of contiguous storage on diskette. This will also help speed up disk accesses.

The working master that you create will be a full system diskette. The original master is also a full system diskette and both should be kept that way.

"Minimum System"

This is a diskette which cannot be used to start TRSDOS, but may be used after the system is fully initialized. This diskette contains the minimum amount of system files required for TRSDOS operation **after** initialization.

You may want to create a minimum system diskette for use in special applications where space in drive 0 is at a premium. You may use PURGE or XFERSYS to create a minimum system.

Why and How of Minimum System Diskettes

If you don't have a Disk Expansion Unit, but still need extra disk space for storage of large data or program files, you may need to create minimum system diskettes for use. This kind of diskette has the minimum operating system programs on it for processing, but may not be used for resetting or starting up. These diskettes may only be used in drive 0 after you have reset or started using a full system diskette. The minimum system diskette will contain the following system programs:

SYSTEM/SYS
SYSTEM*nn* (*nn* = 32 for 32K machines, 64 for 64K machines)

Note: If you do not plan to use DO, HOST, SPOOL, DEBUG, or serial I/O during the execution of your application, you will not need SYSTEM 64 or SYSTEM 32 on your minimum system diskette.

The technique for using a minimum system diskette is as follows:

1. Initialize using your working master (which must be a full system diskette).
2. Remove this diskette from drive 0 and replace with the minimum system diskette that you have created.
3. Now you may start running your application which should be on the minimum system diskette.

If your application requires more than one diskette for the storage of files or programs, then each diskette used (with your one-drive system) should be a minimum system. Your application should be written such that you only have files open on the diskette that is currently in drive 0, and that when you need to swap diskettes to get data (or programs) from another minimum system diskette, that you close the open files, then do the swap, then open the file(s) that you need from the other diskette.

You may also use minimum system diskettes in a multi-drive setup, but keep in mind that you must start or reset with a full system diskette in all cases.

To Create a Minimum System Diskette

There are two ways to create a minimum system disk. The first is to start with a copy of either your working master or any other full system disk and, from this, purge off all of the modules except SYSTEM/SYS and SYSTEM64 (or SYSTEM32). The other way to create a minimum diskette is to use XFERSYS with the MIN option. This will purge off all of the unnecessary modules and will also purge off SYSTEM64 (and/or SYSTEM32). If you plan to use DO, HOST, serial I/O, DEBUG, or SPOOL in your application, then you should use the "manual" method using PURGE.

"Data"

This is a diskette which does not have the minimum system file requirement, and therefore may only be used in the external drives (1, 2 and 3). Such diskettes have a maximum of space available for user files.

Why and How of Data Diskettes

If you have a Disk Expansion Unit, and you plan to run applications or programs that require large data files, you will want to have available as much disk space as possible for the storage of the file(s) or program(s). In this case, you will not need any of the operating system programs on such a diskette. While this kind of diskette may not be used at any time in drive 0, it will have the maximum space available for data or program storage. An empty data diskette is created by FORMAT. You may copy or write data or program files onto this kind of diskette in large, contiguous blocks for fast accessing later.

Changes to TRSDOS

New Library Commands

These are described fully in the new pages for Section 2.

Library Command	Function
ANALYZE	Gives diskette allocation information organized by track
DUAL	Duplicates output to video and line printer
ECHO	Begins echo of keyboard input to display
HELP	Helps with TRSDOS command syntax
HOST	Allows keyboard input from and video output to a remote terminal via the serial interface
MOVE	Copies multiple files and re-organizes a diskette
PRINT	Prints any file in text format
RECEIVE	Inputs Intel hex format data into RAM from the serial interface
RESET	Resets TRSDOS (like pressing the RESET switch)
SCREEN	Copies Video Display screen contents to the printer
SPOOL	Saves printer output in a disk file for later printing; also allows printing of spool file while other operations are in progress
STATUS	Displays current top of user memory and on/off status of various TRSDOS functions
T	Advances printer to top of form

New Utilities and Special Programs

These are described fully in the new pages for Section 3.

Name	Function
LPII	Modifies the TRSDOS printer driver for use with the Radio Shack Line Printer II (Catalog Number 26-1154)
MEMTEST	Checks out random access memory
PRTBKSP	Modifies the TRSDOS printer driver for use with printers capable of backspacing

Changes to 1.2 Commands and Utilities

The following have been changed:

AUTO
BACKUP
CLEAR
COPY
DEBUG
DIR
FORMAT
FORMS
FREE
I
KILL
SETCOM
TERMINAL
VERIFY
XFERSYS

For details, see the replacement pages for Section 2.

Other Changes

Keyaheads

TRSDOS allows keyaheads of up to 80 characters. This means you can type in the next command while previous ones are being executed.

Note: A keyahead will not be displayed until TRSDOS or the application program is ready to interpret it.

For example, type in these three lines, without waiting for TRSDOS READY after the first command:

```
DIR {SYS} ENTER
TIME ENTER
FREE ENTER
```

Press **BREAK** to interrupt the operation currently in progress; if a keyahead line is available, TRSDOS will begin interpreting it. Press **HOLD** to pause the operation currently in progress; press **HOLD** again to continue.

Wild Card

Certain library commands allow you to specify a collection of files by using a "wild-card" field in place of the file name and/or extension. An asterisk "*" in a file specification represents a wild-card field and means, "any sequence of one or more characters here".

The wild-card option is available on the following commands:

KILL
MOVE
DIR (wild-card field for entire file name only)

A new SVC has been added to allow Z-80 programs to use this wild-card capability.

To find a full description of **wild-card** use, use the Index.

Alternate Directory

For increased reliability of the diskette filing system, TRSDOS now sets up an alternate directory. If the main directory should become unreadable, the alternate directory will be used to allow continued access to the diskette. Details are given in the descriptions of FORMAT and BACKUP.

ABSolute Option

Many of the TRSDOS commands and utilities ask the operator to confirm whether an operation is to be completed. The ABS option has been added to eliminate these questions and thus simplify the use of the command.

The ABS option is available on the following commands:

BACKUP
COPY
FORMAT
MOVE

Disk Swap Protection

The system now has the ability to detect when diskettes have been swapped improperly, that is, while a file is open. TRSDOS will display an error message when it detects this condition, and will cancel the read or write operation. See VERIFY.

New or Changed Supervisor Calls (SVCs)

These are described in the new pages for Section 4.

Name	Function	Code
VDCHAR	Output a character to video	8
PRINIT	Initialize printer-driver parameters	17
PRCHAR	Output a character to printer	18
LOOKUP	(New) Lookup in a table	28
HLDKEY	(New) Process the HOLD key	29
OPEN	Open/Create a Program or Data File	40
RENAME	(New) Rename a file	47
WILD	(New) Process a wild-card specification	51
RAMDIR	(New) Read the directory into a RAM buffer	53
SORT	(New) Sort items in RAM list	56
FILPTR	(New) Get information about a currently open file	58
CLRXIT	(New) Execute CLEAR command and jump to TRSDOS READY	57
VIDRAM	(New) Transfer video to RAM or vice-versa	94
PRCTRL	(New) Control printer operations	95
A/BRCV	Receive character via Channel A/B	96/98
A/BTX	Transmit character via Channel A/B	97/99
A/BCTRL	(New) Control Channel A/B	100/101

Changes to Basic

New Keywords

These are described fully in the new pages for Chapter 3 of the BASIC Reference Manual.

Keyword	Operation
ERRS$ Function	Returns the last TRSDOS error number and message
NAME Statement	Renames a diskette file.

Other Changes

PRINT Zones

The PRINT zone width is 16 columns (was 14). For example:

```
PRINT "A", "B", "C", "D"
```

prints the letters in video columns 1, 17, 33 and 49.

New Error Codes

TRSDOS error codes 47-48 now indicate FC errors in BASIC.

TRSDOS error code 49 now indicates an I/O error in BASIC. (Use ERRS$ to get the TRSDOS error number and message.)

Line Printer

The maximum LPRINT or LLIST line is 255 characters (was 132).

Passwords

You can assign a password to a program when you SAVE it. This password will be required for loading or running the program.

Line Feeds

You can include line feeds in the program text, SAVE the program in ASCII format, and LOAD the program later (used to cause a Direct Statement error). Each line feed in the file requires two bytes of diskette storage.

Take care not to create a line that will require more than 255 bytes for storage. This maximum includes the line number digits and the space following the line number. If the line contains embedded line feeds, allow two bytes for each line feed.

OPEN Statement

To specify a record length of 256, you can use either 256 or 0. For example,

```
OPEN "D", 1, "FILE", 256
```

and

```
OPEN "D", 1, "FILE", 0
```

are equivalent.

NEW

In addition to erasing the BASIC program in memory, this statement now clears the screen.

Section 1

General Information

Introduction

Model II TRSDOS ("Triss-Doss") is a powerful and easy-to-use Disk Operating System, providing a full set of library commands and utility programs. In addition, many of the most useful System routines can be called directly by user programs.

Library commands are typed in from the TRSDOS command level to accomplish a variety of operations, including:

- Initialization–setting Printer parameters, date and time, etc.
- File-handling–copying, renaming, deleting, protecting, etc.
- File access–loading into memory, listing to Printer or Display, etc.
- Error identification

See the **Commands** section for details.

Utility programs provide essential services like:

- Formatting blank diskettes.
- Making backup copies of entire diskettes.

See the **Utilities** section for details.

System routines are executed via function codes instead of calls to absolute memory addresses. Routines available fall into seven categories:

- System control
- Keyboard input
- Video Display input/output
- Line Printer output
- File access
- Computational functions
- Serial communications

See the **Technical Information** section for details.

Notation

For clarity and brevity, we use some special notation and type styles in this book.

CAPITALS and punctuation

Indicate material which must be entered exactly as it appears. (The only punctuation symbols not entered are ellipses, explained below.) For example, in the line:

```
DIR  SYS
```

every letter and character should be typed exactly as indicated.

lowercase italics

Represent words, letters, characters or values you supply from a set of acceptable values for a particular command. For example, the line:

LIST *filespec*

indicates that you can supply any valid file specification (defined later) after LIST.

. . . (ellipsis)

Indicates that preceding items can be repeated. For example:

ATTRIB *filespec*{*option* , . . . }

indicates that several options may be repeated inside the braces.

ƀ

This special symbol is used occasionally to indicate a blank-space character (ASCII code 32 decimal, 20 hexadecimal).

```
RENAMEƀFILE/AƀTOƀFILE/B
```

x'*nnnn*'

Indicates that *nnnn* is a hexadecimal number. All other numbers in the text of this book are in decimal form, unless otherwise noted. For example:

```
X'7000'
```

indicates the hexadecimal value 7000 (decimal 28672).

Memory Requirements

TRSDOS occupies 6.1 tracks on the System diskette (39,040 bytes). However, only a small portion is actually in 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 0 through 10239 (X'0'–X'27FF') are reserved for the Operating System. Certain commands, called "high overlays", also use memory addresses up to X'2FFF' (details provided in the Commands section). User programs must be located above X'27FF'; and you may want to locate them above X'2FFF' to allow use of the high overlays without loss of your program.

DECIMAL ADDRESS		HEX ADDRESS
0	SYSTEM AREA	X'0000'
10240	USER AREA (SHARED WITH TRSDOS "HIGH OVERLAY COMMANDS")	X'2800'
12288	USER AREA UNTOUCHED** BY TRSDOS	X'3000'
TOP*	MAY BE RESERVED BY TRSDOS FOR SPECIAL PROGRAMMING	TOP*
32767 or 65535	Last Memory Address	X'7FFF'or X'FFFF'

MEMORY REQUIREMENTS OF TRSDOS

Note: The term "user program" applies to any program which is not a part of TRSDOS. Therefore BASIC is a user program. For memory requirements of BASIC, see the BASIC Reference Manual.

*TOP is a memory protect address set by TRSDOS. If TRSDOS is not protecting high memory, then TOP is the same as "Last Memory Address".

Single-drive COPY from one diskette to another, BACKUP and FORMAT use **all user memory.

Loading TRSDOS

See the **Operation Manual** for instructions on connection, power-up and inserting the System diskette.

Note: A System diskette must be in Drive 0 (the built-in unit) whenever the Computer is in use. Whenever the Computer is turned on or reset, it will automatically load TRSDOS from Drive 0.

After the System starts up, it will prompt you to enter the date. Type in the date in MM/DD/YYYY form and press ENTER . For example:

```
    07/04/1979 ENTER
```

for July 4, 1979.

Next the System will prompt you to enter the time. To **skip this question,** press ENTER . The time will start at 00:00:00.

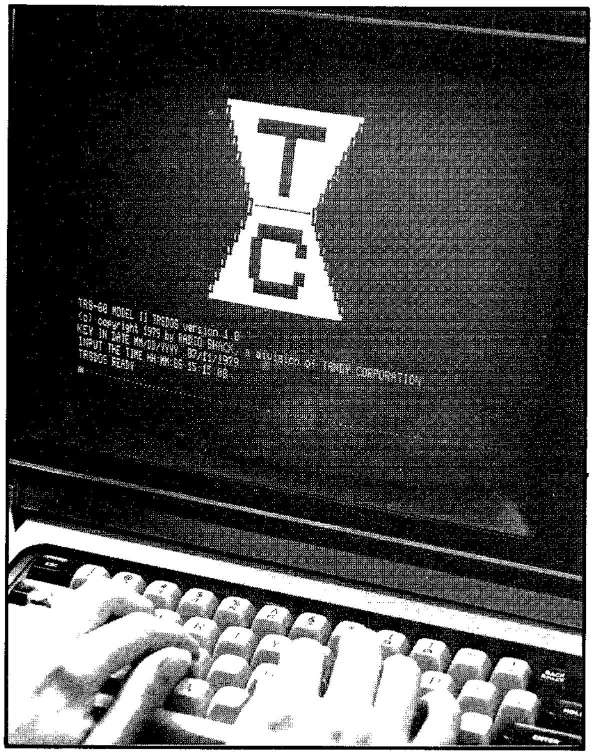
To set the time, type in the time in HH.MM.SS 24-hour form. Periods are used instead of colons since they're easier to type in. The seconds are optional. For example:

```
    14.30 ENTER
```

for 2:30 pm.

The System will record the time and date internally and return with the message:

```
    TRSDOS READY
    ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
```



```
TRS-80 MODEL II TRSDOS version 1.0
(c) Copyright 1979 by RADIO SHACK, a division of TANDY CORPORATION
KEY IN DATE MM/DD/YYYY 07/11/1979
INPUT THE TIME HH:MM:SS 15:15:00
TRSDOS READY
```

Using the Keyboard

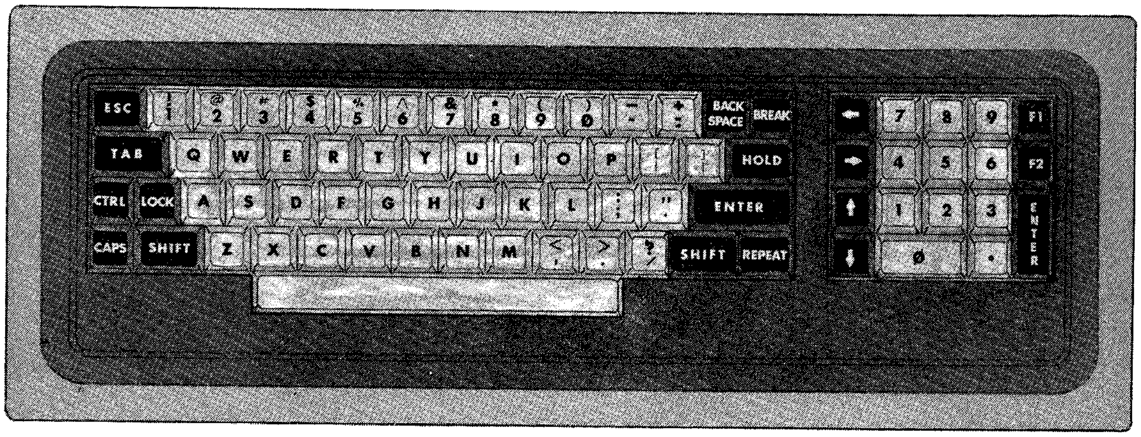
TRSDOS distinguishes between upper and lower case letters. Therefore

```
dir
```

is not the same as

```
DIR
```

Since TRSDOS commands are always capitalized, you'll probably find it convenient to operate the Keyboard in the Caps mode (press CAPS so the red light comes on). That way, all the alphabet-keys are interpreted as capital letters, regardless of whether the SHIFT key is being pressed.



Certain control keys are useful in the Command Mode:

BREAK Interrupts line entry and starts with a new line.

← **Backspaces** the cursor without erasing any characters. Use this to position the cursor for correcting a portion of a line.

→ **Forward-spaces** the cursor without erasing any characters. Use to position the cursor for correcting a portion of a line.

BACK SPACE **Backspaces** the cursor, erasing the last character you typed. Use this to correct entry errors.

ENTER Signifies end of line. When you press this key, TRSDOS will take your command. Only those characters appearing to the left of the cursor will be used.

HOLD Pauses execution of a command. Press once more to continue. Not functional in all commands.

TAB Advances the cursor to the next 8-column position. Tab positions are at columns 0, 8, 16, 24, etc.

SPACEBAR Enters a space (blank) character and moves the cursor one character forward.

If you type any other control key (non-alphanumeric, non-punctuation), a ± symbol will be displayed for that key, but the control code will be sent to the Computer. Such control keys will either be ignored or will cause a parameter error to occur. See the Keyboard Code Map in the **Appendix** for control codes.

Repeat key. For convenience when you want to repeat a single key, hold down REPEAT while pressing the desired key. For example, to backspace halfway back to the beginning of the line, hold down REPEAT and BACK SPACE .

Entering a Command

Whenever the TRSDOS READY prompt is displayed, you can type in a command, up to 80 characters. If the command line is less than 80 characters (as is usually true), you must press ENTER to signify end-of-line. TRSDOS will then "take" the command.

For example, type:

```
CLS    ENTER
```

and TRSDOS will clear the Display.

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

First it looks to see if what you've typed is the name of a TRSDOS command. If it is, TRSDOS executes it immediately.

If what you typed is not a TRSDOS command, then TRSDOS will check to see if it's the name of a program file on one of the drives.

When searching for a file, TRSDOS follows the sequence drive 0, drive 1, etc. – unless you include an explicit drive specification with the file name (described later on).

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

Keyaheads

TRSDOS also allows keyaheads of up to 80 characters. This means you can type in the next command while previous ones are being executed.

Note: A keyahead will not be displayed until TRSDOS or the application program is ready to interpret it.

Command Syntax

Command syntax is the general form of a command, like the grammar of an English sentence. The syntax tells you how to put keywords (like DIR, LIST, and CREATE) together with the necessary parameters for each keyword. In this book, we present general syntax inside gray boxes, so they're easy to recognize.

There are three general command formats:

1. No-file commands
2. One-file commands
3. Two-file commands

Syntax Forms

No-file commands

command {*options*} *comment*

options is a list of one or more parameters that may be needed by the command. Some commands have no options. The braces { } around *options* can usually be omitted when no comment is added at the end of the command line.

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

One-file commands

command filespec {*options*} *comment*

filespec is a standard TRSDOS file specification as described later in this section.

options –See description above.

comment–See description above.

Two-file commands

command filespec-1 delimiter filespec-2 {*options*} *comment*

filespec-1 and *-2* are TRSDOS file specifications as described later in this section.

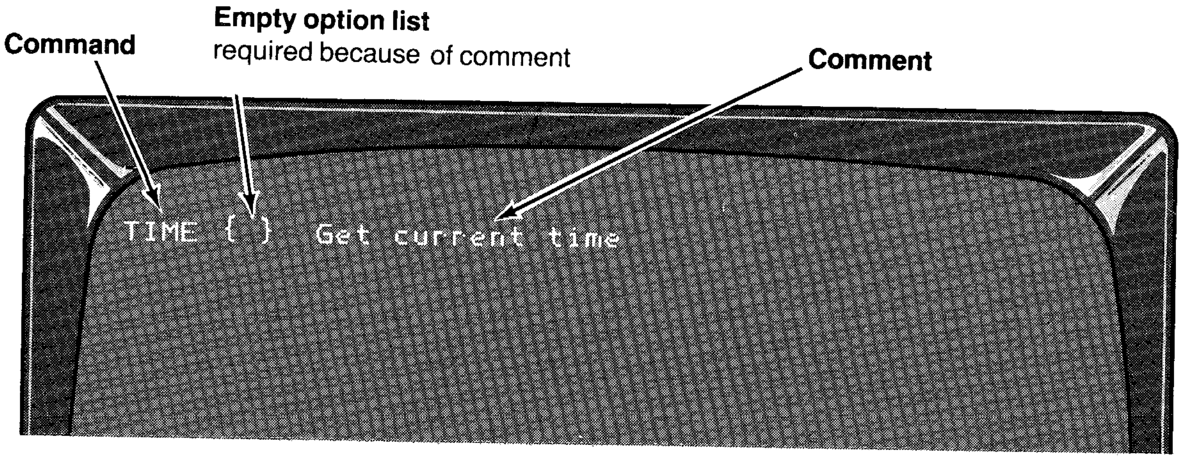
delimiter is one of the following:

blank space or spaces (indicated as b̸)
a comma, surrounded by optional spaces
b̸TOb̸ surrounded by optional spaces.

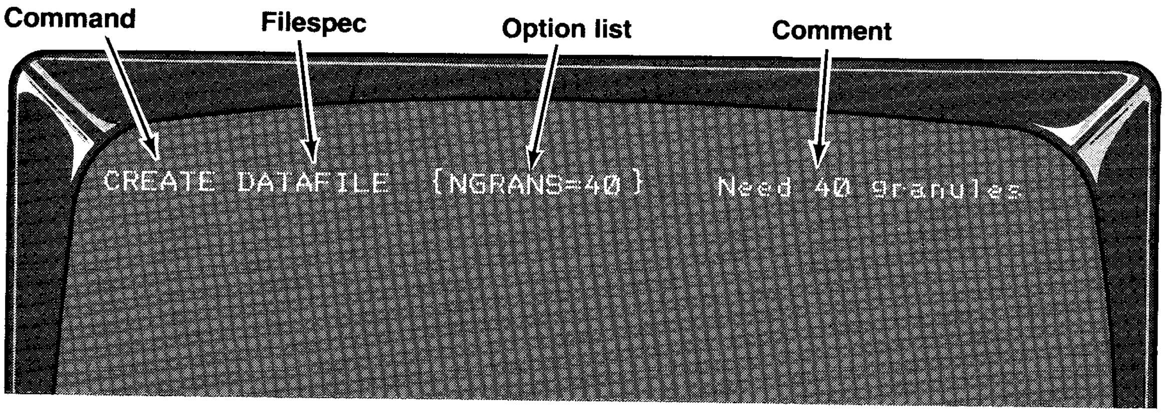
options–See description above.

comment–See description above.

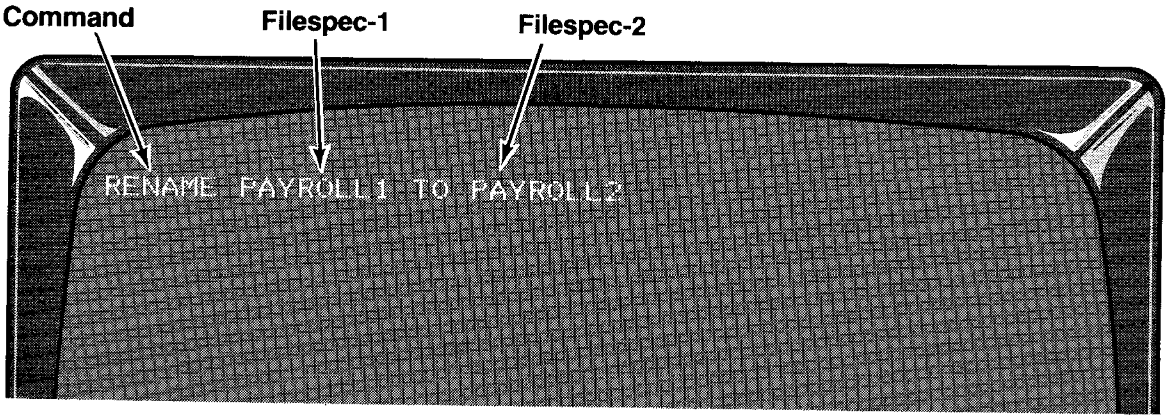
Examples of Syntax Forms



```
TIME { }  Get current time
```



```
CREATE DATAFILE  {NGRANS=4Ø}   Need 4Ø granules
```



```
RENAME PAYROLL1 TO PAYROLL2
```

File Specification

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

A file specification has the general form

filename/ext.password:d(diskette name)

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

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

.password is an optional password; *password* is a sequence of up to eight

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

(*diskette name*) is an optional field of up to 8 letters or numbers. If this field is included, it must be preceded by a drive specification.

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

For example:

```
FileA/TXT.Manager:3(ACCOUNTS)
```

references the file named FileA/TXT(ACCOUNTS) with the password Manager, on Drive 3, diskette name ACCOUNTS.

File Names

A file name consists of a name and an optional name-extension. For the name, you can choose any letter, followed by up to seven additional numbers or letters. To use a name extension, start with a diagonal slash / and add up to three numbers or letters.

For example:

```
MODEL2/TXT        INVNTORY          DATA11/BAS
NAMES/123         August/15         WAREHOUS
TEST              TEST1             TEST/1
```

are all valid and **distinct** file names.

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

/BAS	BASIC program
/TXT	ASCII text
/OBJ	Object code
/REL	Relocatable machine-language program
/DVR	Input/output driver
/SRC	Source code

Wild Card

Certain library commands allow you to specify a collection of files by using a "wild-card" field in place of the file name and/or extension. An asterisk "*" in a file specification represents a wild-card field, and means, "any sequence of one or more characters here".

Unless restrictions are stated for a command, wild-card fields may take any of the following forms:

Wild-Card Fields

*
**string*
*string**
string
*string*string*

Where *string* is a sequence of characters. The overall length of the wild-card field depends on whether it appears in the file name or file extension.

For example:

```
MOVE  */BAS:0  TO  :1
```

copies all files with the extension "/BAS" from drive 0 to drive 1.

```
DIR  */COM:0
```

gives the drive zero directory, listing all user files with the extension "/COM". (DIR allows a wild-card field only in place of the *entire* file name, and not in place of the file extension.)

```
KILL  *LST/*:0
```

kills all files with "LST" at the end of the file name, regardless of the other characters in the name and extension. TRSDOS prompts you before deleting any file.

The wild-card option is available on the following commands:

KILL
MOVE
DIR (wild-card field for entire file name only)

Drive Specification

If you give TRSDOS a file command like:

```
KILL TEST/1
```

the System will search for the file TEST/1, starting at Drive 0 and going to the other drives in sequence 1,2,3 until it finds the file.

Anytime TRSDOS has to Open a file (e.g., to List it for you) it will follow the drive lookup sequence 0,1,2,3. When TRSDOS has to create a new file, it will skip over any write-protected or full diskettes.

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

For example:

```
KILL TEST/1:3
```

tells the System to look for the file TEST/1 on drive 3 only.

Passwords

You can protect a file from unauthorized access by assigning passwords to the file. That way, a person cannot access a file simply by referring to the file name; he must also use the appropriate password for that file.

TRSDOS allows you to assign two passwords to a file:

- An "Update word", which grants the user total access to the information (execute, read, write, rename or delete).
- An "Access word", which grants the user limited access to the information (see ATTRIB).

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

A password consists of a period . followed by 1 to 8 letters or numbers. If you do not assign a password to a file, the System uses a default password of 8 blanks. In this case the file is said to be unprotected; one can gain total access simply by referring to the file name.

For example, suppose you have a file named SECRETS/BAS. and the file has MYNAME as an update and access word. Then this command:

```
    KILL SECRETS/BAS
```

will **not** cause the file to be Killed. You must include the password MYNAME in the file specification.

Suppose a file is named DOMAIN/BAS and has blank passwords. Then the command:

```
    KILL DOMAIN/BAS.GUESS
```

will not be obeyed, since GUESS is the wrong password.

Diskette Names

When you reference a file like TESTER/BAS:3, TRSDOS will use whatever diskette is in drive 3. However, if you add a diskette name to the file specification, TRSDOS will first check to see that the correct diskette is in the drive. (You assign diskette names during the Format or Backup process.)

Note: Only the COPY command looks at the diskette name and checks that the correct diskette is inserted. The other commands ignore the diskette name.

A diskette name consists of from 1 to 8 letters and numbers inside parentheses ( ). When you include the diskette name in a file specification, you must also include the drive number *:d*. Otherwise the diskette name will be ignored.

For example:

```
  COPY REPORT/TXT:Ø TO REPORT/TXT:3(TXTFILES)
```

tells TRSDOS to copy the file REPORT/TXT on drive 0 to another file named REPORT/TXT on a diskette named TXTFILES, using drive 3.

Section 2

Library Commands

Introduction

You can enter a library command whenever the TRSDOS READY prompt is displayed. (Programs can also call library commands. See **Technical Information.**)

The following commands use memory addresses from X'2800' to X'2FFF':

ANALYZE
APPEND
BUILD
COPY
CREATE
DUMP
ERROR
HELP
KILL
LIST
MOVE
PRINT
PURGE
RECEIVE
SETCOM (also SVC 55)
VERIFY

The following library commands use memory from X'2800' to TOP:

Single-Drive COPY

MOVE

RECEIVE (data goes into user area)

Other library commands use memory *below* X'2800'.

Exit Conditions of Library Commands

Any command executed from the TRSDOS READY level will return to TRSDOS.

Most library commands executed from another program can return to the program, if the program uses this capability (see SVC 38, RETCMD).

However, the following library commands will *always* exit to TRSDOS READY:

SPOOL ON
DEBUG ON
DO (exits to begin the first command in the DO-file)
HOST ON
MOVE
Single-Drive COPY

To continue program execution after any of these commands, you may use a DO-file containing library commands and ending with a command to load and execute your program.

General rules for entering commands

Don't type any leading blanks in front of the command. For example:

```
TRSDOS READY
     DIR
```

is an error. Omit the spaces before DIR.

There must be at least one space between the command and any option list or comment. For example:

```
DIR{SYS}
```

is an error. Insert a space between R and { .

There can be any number of spaces between options.

```
DIR       {SYS   ,   PRT}
```

has the same effect as:

```
DIR {SYS,PRT}
```

When no ambiguity would result, the braces around the option list can be omitted.

```
CREATE FileA NRECS=100,LRL=64
```

is acceptable, but

```
CREATE FileA NRECS=100,LRL=64    Set up file area
```

is not, since the comment "Set up file area" will be taken as an invalid parameter.

Details

When the syntax calls for a delimiter (ƀTOƀ, comma or space), other non-alphanumeric non-brace characters will also serve, unless the special punctuation is part of an option keyword, e.g., the = sign in several commands.

```
LIST TEXTFILE {PRT:SLOW}
```

is equivalent to:

```
LIST TEXTFILE {PRT, SLOW}
```

AGAIN
Repeat Last Command

```
AGAIN
```

This command tells TRSDOS to re-execute the most recently entered command.

AGAIN cannot be used after certain library commands, utilities and user programs.

Example

```
TIME
AGAIN
```

TRSDOS will re-execute the TIME command.

Sample Use

AGAIN is useful after TRSDOS has returned an Input/Output error message instead of obeying a command. For example, suppose you type:

```
KILL OLDFILE:1
```

and the diskette in drive 1 is write protected. Then you'll get an ERROR 15 message. Put a write-enable tab on the diskette and type:

```
AGAIN
```

Now TRSDOS will re-execute the command.

Suppose you are making multiple backup copies of a file from drive 0 to drive 1. Enter the COPY command once; for second and third copies, use AGAIN. For example:

```
COPY DAYSWORK:0 TO DAYSWORK:1
```

copies the file to a drive 1 diskette. Now put another diskette into drive 1 and type:

```
AGAIN
```

to repeat the copy using the new diskette.

ANALYZE
Analyze Diskette Allocation by Track

This command may take two forms:

A) ANALYZE *file* {PRT}
B) ANALYZE *drive* {PRT, T = *tracknumber*}

file is a TRSDOS file specification. If form A of the command is used, the tracks allocated to *file* will be listed in ascending order.

PRT tells TRSDOS to send the output to the printer instead of the display. This option may be used with either form of the command.

drive is a drive specification. If this form of the command is used, each track (or the track specified by TRACK = *tracknumber*) will be listed with the files that are currently using space on that track.

T = *tracknumber* tells TRSDOS to list only those files using space on the specified track. This option may only be used with form B of the command.

This command allows you to determine how the diskette tracks are allocated to system and user files. It may be useful before performing a BACKUP or XFERSYS operation.

Example

```
ANALYZE  DATAFILE
```

shows which tracks are used by the file named DATAFILE.

```
ANALYZE  1
```

shows which files are using each track on the drive 1 diskette.

APPEND
Append files

APPEND *file-1* TO *file-2*

file-1 and *file-2* are file specifications. The files must have the same type (Fixed or Variable), the same Record Length, must both be programs or both be data files (P or D in the Directory listing).

ƀTOƀ is a delimiter. A comma or a space can also be used.

APPEND copies the contents of *file-1* onto the end of *file-2*. *file-1* is unaffected, while *file-2* is extended to include *file-1*. The file types (V or F) and record lengths (for fixed length record files) must match. See DIR for more information on file types and record lengths.

Examples

```
APPEND Wordfile/2 TO Wordfile/1
```

A copy of Wordfile/2 is appended to Wordfile/1.

```
APPEND REGION1/DAT,TOTAL/DAT.guess
```

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

Sample Uses

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

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

You can combine the two files with the command:

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

PAYROLL/A will now look like this:

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

PAYROLL/B will be unaffected.

ATTRIB
Change a File's Passwords

ATTRIB *file* {ACC=*password-1,* UPD=*password-2*, PROT=*level*}

file is a file specification.

ACC=*password-1* sets the access word equal to *password-1.* If omitted, access word is unchanged.

UPD=*password-2* sets the update word equal to *password-2*. If omitted, update word is unchanged.

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

Level	**Degree of access granted by access word**
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 access word total access)

ATTRIB lets you change the passwords to an existing file. Passwords are initially assigned when the file is created. At that time, the update and access words are set to the same value (either the password you specified or a blank password). See Chapter 1 for details on access and update passwords.

Examples

```
ATTRIB  DATAFILE  ACC=JULY14, UPD=MOUSE, PROT=READ
```

Sets the access password to JULY14 and the update password to MOUSE. Use of the access word will allow only reading and executing the file.

```
ATTRIB PAYROLL/BAS.SECRET ACC=,
```

Sets the access word to blanks. The protection level assigned to the word is left unchanged.

```
ATTRIB OLD/DAT.Apples UPD=,
```

Sets the update word to blanks.

```
ATTRIB PAYROLL/BAS.PW PROT=EXEC
```

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

```
ATTRIB DATAFILE/1.PRN  PROT=,
```

Sets the access level to Kill.

Sample Uses

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

```
ATTRIB PAYROLL ACC=PAYDAY, UPD=Avocado, PROT=READ
```

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

Suppose you want to temporarily stop access to the file. Then use the command:

```
ATTRIB PAYROLL.Avocado PROT=NONE
```

Now the use of the password PAYDAY grants no access to the file. To restore the previous degree of access, use the command:

```
ATTRIB PAYROLL.Avocado PROT=READ
```

Note: The EXEC protection level should be used to protect machine-language programs (''P'' attribute in the DIR listing). Do not use this protection level for BASIC program files, or the BASIC interpreter will not be able to load and run them. BASIC programs must have a protection level of PEAD or higher.

AUTO
Automatic Command after System Start-Up

AUTO *command-line*
command-line is a TRSDOS command or the name of an executable program file.

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

When you enter an AUTO command, TRSDOS writes *command-line* into the start-up procedure for the diskette in Drive 0. TRSDOS does not check for valid commands; if the command line contains an error, it will be detected the next time the System is started up.

Examples

```
AUTO DIR {SYS}
```

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

```
AUTO BASIC
```

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

```
AUTO FORMS {W=80}  For 8-1/2" wide paper
```

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

```
AUTO PAYROLL/CMD
```

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

```
AUTO DO STARTER
```

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

To Override an AUTOmatic Command

Press (HOLD) any time during the DATE/TIME start-up initialization dialog (before the second (ENTER)).

To erase an automatic command

Type:

```
AUTO
```

This tells TRSDOS to delete any automatic key-ins and reset the start-up procedure to go directly to the TRSDOS READY mode.

Important Note

Be sure a program is fully debugged before making it an automatic command. Programs which are executed via the AUTO function should normally provide a means of exiting to the TRSDOS READY mode. (Unless the (BREAK) key is blocked by the user program, pressing (BREAK) will get you back to TRSDOS.)

Sample Use

Suppose you want the TRSDOS to run a certain BASIC program, MENU, each time it is started up. That way, an operator can turn on the Computer and get going without having to enter any TRSDOS commands.

Then use the command:

```
AUTO BASIC MENU
```

to prepare the System to run the BASIC program each time it starts up. (See BASIC Reference Manual for details on loading BASIC.)

BUILD
Create an Automatic Command Input File

BUILD *file*

file is a file specification which cannot include an extension.

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

BUILD files are primarily intended for passing command lines to TRSDOS just as if they'd been typed in at the TRSDOS READY level.

BUILDing New Files

When the file you specify does not exist, BUILD creates the file and immediately prompts you to begin inserting lines. Each time you complete a line, press ENTER . BUILD will give you another chance to re-do the line or keep it. Press ESC to erase and re-do the line; ENTER to store it and start the next line.

While typing in a line, you can use ← and → to position the cursor for corrections. BACKSPACE also works as usual. Be sure the cursor is at the end of the desired line before you press ENTER .

To end the BUILD file, simply press ENTER at the beginning of the line, i.e., when the message:

```
ENTER COMMAND LINE (1-80)
```

Is displayed.

Note: Pressing BREAK will also end the file. Only those lines that have been flagged like this:

```
*** LINE STORED IN FILE ***
```

will be saved.

Editing Existing BUILD-Files

When you specify an existing file in the BUILD command, TRSDOS assumes you want to edit that file. Before starting the edit, it copies the file into a new file with the same name but with the extension/OLD. That way, you will have a backup copy of the file as it was before being edited.

Note: Editing an existing BUILD-file requires that you have write-access to the file. That is, if the access password has a protection level which does not allow writing, then you must supply the update password.

Example

Suppose the file STARTER already exists, and you type the command:

```
BUILD STARTER
```

TRSDOS will first copy STARTER into a new file STARTER/OLD (if STARTER/OLD already exists, previous contents are lost). Then it will let you begin editing the file. As you edit the file, the updated lines will be written into STARTER.

BUILD will display the existing contents of the file, one line at a time. Beneath the line is an option list:

```
Keep, Delete, Replace, Insert or Quit?
ENTER (K/D/R/I/Q) ..?
```

Type the first letter of the desired option and press ENTER .

Keep Option: Copies the line as-is into the new file, and displays the next line for editing.

Delete Option: Deletes the line by not copying it into the new file, and displays the next line for editing.

Insert Option: Allows you to insert lines **ahead** of the line being displayed. Using this option is like entering lines into a new file as described above.

After you press ENTER , TRSDOS will give you a chance to erase and restart the insert line, or to store the insert line. Press ESC to erase. ENTER to store it. You can then insert another line.

To stop inserting, press ENTER at the beginning of the line. TRSDOS will then display the next line and the option list.

Replace Option: Deletes the displayed line and lets you insert replacement lines. Entering replacement lines is like entering lines with the insert option. Press ENTER at the beginning of a line to stop inserting. TRSDOS will display the next line and the option list.

Quit Option: Ends the editing session. All remaining lines will be copied into the new file as-is. Before closing the file, TRSDOS will ask you if you want to add new lines to the end. (If you simply want to add to a file but make no other changes, type Q at the beginning of the edit session.)

At end of file

Whenever TRSDOS reaches the end of the file, it will ask if you want to add new lines at the end. Type Y ENTER to add, N ENTER to end the editing session.

Adding lines at the end of a file is just like using the insert line option described above. Type ENTER at the beginning of a line to stop adding and close the file.

To recover a BUILD file's previous contents

There are a couple of cases in which you may need to do this. Let's assume you are editing a file named STARTER.

1. After ending the edit session, you realize that you have made an error, and you want to recover the previous version of the file.

2. You accidentally press BREAK and end the edit session; only those lines that have been flagged like this:

```
  *** LINE STORED IN FILE ***
```

will be saved in the new file named STARTER.

The previous file contents are now stored in STARTER/OLD. If you want to re-edit this file, you must Copy it or Rename it to a file name without an extension. For example, you might use this command:

```
  COPY STARTER/OLD TO STARTER  {ABS}
```

(See COPY for an explanation of the ABS parameter.) Now you can edit the previous file's contents. Type:

```
  BUILD STARTER
```

to start editing.

CLEAR
Clear User Memory

```
CLEAR
```

This command zeros user memory and returns to TRSDOS. It does not reinitialize the I/O drivers, unprotect high memory, or perform any other functions.

Example

```
CLEAR
```

CLOCK
Turn on Clock-Display

CLOCK {*switch*}
switch is one of the options, ON or OFF. If switch is not given, ON is assumed.

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

TRSDOS starts up with the clock off.

Note: The real-time clock is always running, regardless of whether the clock-display is on or off.

Examples

```
CLOCK
```

Turns on the clock-display.

```
CLOCK OFF
```

Turns off the clock-display.

CLS
Clear the Screen

```
CLS
```

This command clears the Display. Use it to erase information that you don't want others to see, for example, file specifications which include passwords.

Example

```
CLS
```

Sample Use

```
CREATE PERSONNL/BAS.secure    NGRANS=300
CLS
```

COPY
Copy a File

COPY *file-1* TO *file-2*{ABS}
file-1 and *-2* are file specifications.
␢TO␢ is a delimiter. A comma or space can also be used.
ABS is an optional parameter telling TRSDOS to copy *file-1* even if *file-2* already exists. The previous contents of *file-2* will be lost. If ABS is omitted, copy will prompt you before it overwrites an existing file.

COPY *file* TO *drive* {ABS}

drive is the drive to which the file will be copied, using the same file specification as *file*.

This command copies *file-1* into the new file defined by *file-2*. If a disk name is included in either file specification, TRSDOS will ensure that the appropriate diskette is inserted before making the copy. This allows you to copy a file from one diskette to another, using **a single drive** if necessary. For single-drive copies, specify the same drive number for each file.

When you do not add the ABS (''absolutely'') parameter, TRSDOS **will not overwrite** an existing file that matches the specification *file-2*. Instead it will prompt: ''Existing file. Copy over it? (Y/N/Q)''. Use the ABS option to overwrite (destroy) an existing file without the prompt.

Normally, COPY uses memory below X'3000''; however, when copying from one diskette to another in a **single** drive, it will use memory up to the start of unprotected memory (see **Memory Requirements**).

The diskette name, if used, must always be preceded by a drive specification; otherwise it will be ignored.

Examples

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

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

```
   COPY NAMEFILE/TXT:Ø(DEPTC) TO NAMEFILE/TXT:Ø(DEPTA)
```

This command specifies a one-drive copy from a diskette named DEPTC to another diskette named DEPTA. TRSDOS will provide the necessary prompting to accomplish the copy.

```
   COPY FILE/A TO FILE/B:1(DOUBLE)
```

This command copies FILE/A to FILE/B. TRSDOS will search all drives for FILE/A, and will require you to have or insert a diskette named DOUBLE in drive 1.

```
   COPY NEWFILE TO OLDFILE {ABS}
```

Performs the copy even if OLDFILE already exists, in which case its previous contents are lost.

```
  COPY TESTPROG TO 3
```

Copies the specified file from the first drive that contains it onto drive 3, using the file name TESTPROG.

Sample Use

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

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

CREATE
Create a Preallocated File

CREATE file {NGRANS=*n1*, NRECS=*n2*, LRL=*n3*, TYPE=*letter*}

file is a file specification.

NGRANS=*n1* indicates how many granules to allocate. If NGRANS is omitted, the number of granules allocated is determined by NRECS and LRL.

NRECS=*n2* indicates how many records to allow for. If NRECS is omitted, NGRANS determines the size of the file. When NRECS is given, LRL must also be given.

LRL=*n3* indicates the record length (Fixed-length records only). *n3* must be in the range [1,256]. If LRL is omitted, LRL=256 is used. When LRL is given, NRECS must also be given.

TYPE=*letter* specifies the record type: *letter* equals F (Fixed-length records) or V (Variable-length records). If TYPE is omitted, TYPE=F is used.

NOTE: {NGRANS} and {NRECS,LRL} are mutually exclusive.

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

With preallocated files, unused space at the end of file is **not** deallocated (recovered) when the file is Closed. With dynamically allocated files, on the other hand, unused space at the end of the file IS recovered when the file is Closed.

Note: With pre-allocated files, TRSDOS will allocate extra space when you exceed the pre-allocated amount during a write operation.

You may want to use CREATE to prepare a file which will contain a known amount of data. This will usually speed up file write operations, since TRSDOS won't have to do periodic allocations during the write operations. File reading will also be faster, since pre-allocated files are less dispersed on the diskette—requiring less motion of the read/write mechanism to locate the records.

Examples

```
CREATE DATAFILE/BAS  NRECS=300, LRL=256
```

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

```
CREATE TEXT/1  NGRANS=100, TYPE=V
```

Creates a file named TEXT/1, and allocates 100 granules. The file will contain variable-length records.

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

Creates a file named NAMES/TXT protected by the password IRIS. The file will be large enough to contain 500 records, each 30 bytes long.

Determing the size of the file

You can allocate space according to number of granules or number of records. (A granule contains 1280 bytes; a record contains from 1 to 256 bytes, depending on LRL.)

The granule is the unit of allocation in TRSDOS: if you ask for 30 granules, that's exactly how much space the file will get (38,400 bytes).

If, on the other hand, you specify the number of records, TRSDOS will give you the **number of granules** which are required to **contain** that many records. For example, if you specify 100 records and a record length of 40, you're asking for a total of 100* 40=4000 bytes. Since TRSDOS allocates spaces in units of granules (1280 bytes), you'll actually get 4 granules–containing 5120 bytes.

Record Length (Fixed-Length Files Only)

A record is the quantity of data TRSDOS processes for you during disk operations. The record length can be any value from 1 to 256.

File Type

TRSDOS.allows two types of files: Fixed-Length Record (FLR) files and Variable-Length Record (VLR) files. With FLR files, the record length (from 1 to 256) is set when the file is created, and it cannot be changed. With VLR files, the length of each record is independent of all other records in the file. For example, record 1 might have a length of 70; record 2, 33; record 3, 225; etc. Variable length records consist of a length byte followed by the data, and can contain up to 256 bytes **including** the length byte.

For further explanation of file structure, allocation and types, see **Technical Information**.

To Create a file to be used by BASIC

1. Decide how many records the file will contain. (This is just an estimate. If the file exceeds this number, it will automatically be extended.)
2. If it is a Direct access file, determine the optimum record length (from 1 to 256). If it is a sequential access file, the record length must equal 1.
3. Use a CREATE command like this:

   CREATE file {NRECS=*number*, LRL=*length*}

Sample Use

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

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

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

You could create an appropriate file with the command:

```
  CREATE PERSONNL/TXT NRECS=25Ø, LRL=128
```

Once creaed, his preallocated file will allow faster writing than would a dynamically allocated file, since TRSDOS won't have to stop writing periodically to allocate more space (until you exceed the pre-allocated amount).

DATE
Reset or Get Today's Date

DATE *mm/dd/yyyy*
mm is a two-digit month specification.
dd is a two-digit day-of-month specification.
yyyy is a four-digit year specification.
If *mm/dd/yyyy* is given, TRSDOS resets the date. If *mm/dd/yyyy* is omitted, TRSDOS displays the current date and time.

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

The operator sets the date initially when TRSDOS is started up. After that, TRSDOS updates the time and date automatically, using its built-in clock and calendar. You can enter any four-digit year after 1599.

When you request the date, TRSDOS displays it in the format:

```
THU JUL 19 1979 200 -- 14.15.31
```

for Thursday, July 19, 1979, the 200th day of the year, 2:15:31pm.

Note: If the time passes 23.59.59, TRSDOS does not start over at 00.00.00. Instead, it continues with 24.00.00. However, the next time you use the TIME or DATE command, the time will be converted to its correct 24-hour value, and the date will be updated. If you let the clock run past 59.59.59, it will recycle to 00.00.00, and the date will not be updated to include the 60-hour period.

Examples

```
DATE
```

Displays the current date and time.

```
DATE 07/18/1979
```

Resets the date to July 18, 1979, and displays the new information.

DEBUG
Start Debugger

DEBUG {*switch*}

switch is one of the following parameters:

- ON turns on the debugger.
- OFF turns off the debugger.

If *switch* is omitted and debugger is off, TRSDOS tells you so.

If *switch* is omitted and debugger is on, TRSDOS enters the debug monitor.

This command sets up the debug monitor, which allows you to enter, test, and debug machine-language programs. It also includes an Upload function to allow transmission of data from another device to the Model II, via the built-in serial interface (Channel B).

DEBUG loads into the high memory area sometimes reserved by TRSDOS for special programming (see **TRSDOS Memory Map**). While DEBUG is on, TRSDOS will automatically protect this area from being overlaid by BASIC or other user programs. **To use DEBUG from BASIC, you must turn DEBUG on before you start BASIC.**

While DEBUG is on, every time you attempt to load and execute a user program, you will enter the debug monitor. In this mode, you can enter any of a special set of single-key commands for studying how your program is working.

DEBUG can only be used on programs in the user area X'2800' to TOP).

Examples

```
DEBUG
```

If DEBUG is off, this command tells you so. If it is on, this command enters the debug monitor.

```
DEBUG OFF
```

Turns off DEBUG and un-protects high memory.

```
DEBUG ON
```

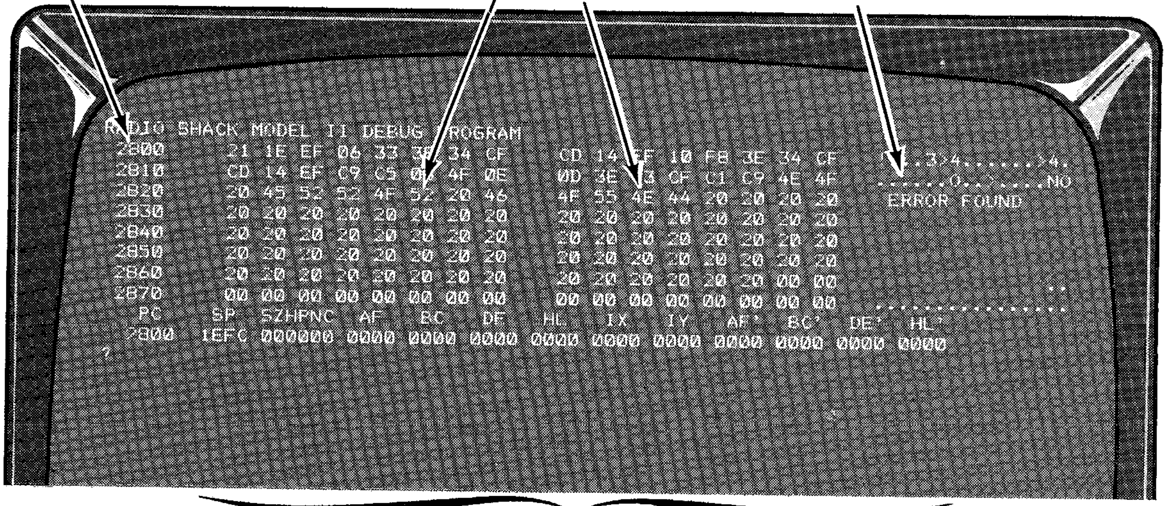
Turns on DEBUG: i.e., loads the debugger into high memory, protects high memory, and sets up a "scroll window"–a block of lines that will be scrolled. The scroll window will consist of the bottom 11 lines of the display. The top 13 lines will be used to contain the debug monitor display.

To enter the debug monitor

Type:

```
DEBUG ON
DEBUG
```

While DEBUG is on, you can also enter the debug monitor simply by typing the file specification of a user program. TRSDOS will load the program and transfer control to the debugger. The transfer address for the program will be in the PC register display.



Start address of one 16-byte "row" of RAM.

RAM display—shows hex contents of each byte.

ASCII display—period "." indicates a non-displayable character.

```
RADIO SHACK MODEL II DEBUG PROGRAM
 2800      21 1E EF 06 33 3E 34 CF     CD 14 EF 10 FB 3E 34 CF     !..3>4......>4.
 2810      CD 14 EF C9 C5 00 4F 0E     0D 3E 13 CF C1 C9 4E 4F     .......O.>....NO
 2820      20 45 52 52 4F 52 20 46     4F 55 4E 44 20 20 20 20      ERROR FOUND
 2830      20 20 20 20 20 20 20 20     20 20 20 20 20 20 20 20
 2840      20 20 20 20 20 20 20 20     20 20 20 20 20 20 20 20
 2850      20 20 20 20 20 20 20 20     20 20 20 20 20 20 20 20
 2860      20 20 20 20 20 20 20 20     20 20 20 20 20 20 00 00                   ..
 2870      00 00 00 00 00 00 00 00     00 00 00 00 00 00 00 00     ................
   PC     SP   SZHPNC  AF   BC   DE   HL   IX   IY   AF'  BC'  DE'  HL'
  2800  1EFC  000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
?
```

Z-80A register contents. SZHPNC are the flag bits in register F.

The ? is the command prompt, meaning that you can enter one of the single-key commands. Press H (for "help") to display a "menu" or list of debugger commands. To enter one of the commands, press the letter which is capitalized in the command menu. For example, to enter the memory command ("raM"), press M .

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

ESC Returns to the ? prompt and cancels the command you're in.

BACK SPACE Backspaces the cursor and erases previous character.

← Cursor back without erasing.

→ Cursor forward without erasing.

F1 In certain subcommands, homes the cursor.

TAB In certain subcommands, tabs the cursor.

Command Description

B (Breakpoint)

Press B to set a breakpoint in your program. When execution reaches a breakpoint, control returns to the debug monitor, with the program counter pointing to the breakpoint address. To continue from that point, press C. The original instruction will be executed—**but the breakpoint will not be removed.** It will still be there the next time that address is reached.

Note: Place breakpoints at the beginning byte of an opcode—**never** in the middle of an instruction.

Press B to enter the Breakpoint command. TRSDOS prompts you to enter the breakpoint number. Up to eight breakpoints are allowed, so type in a number from 1 to 8. Next TRSDOS prompts you to enter the new address for that breakpoint. If the breakpoint has previously been set, TRSDOS displays the old breakpoint address, and the original instruction that goes in that address.

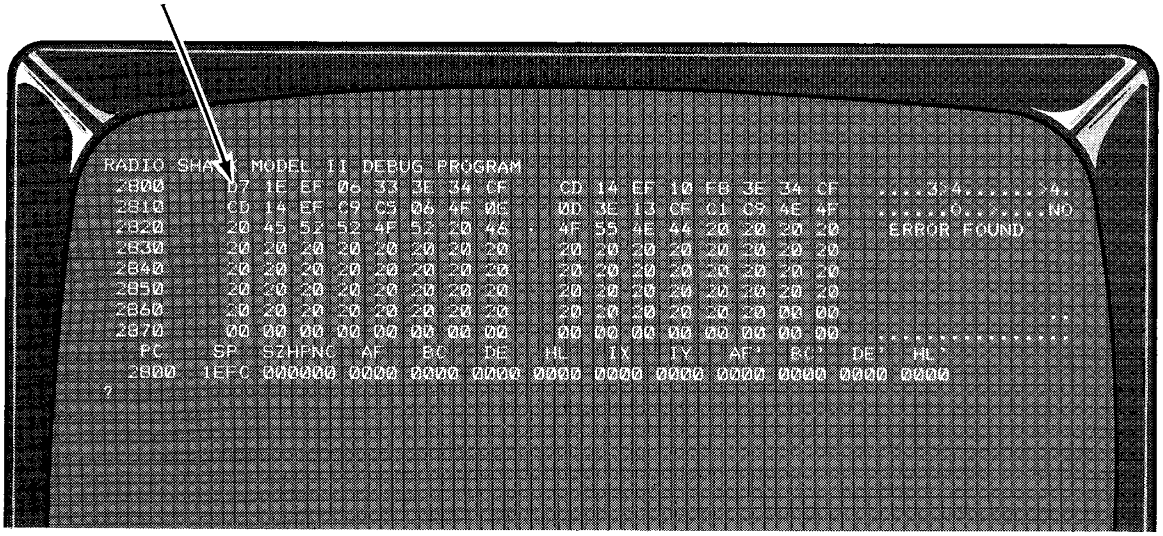
Note: While a breakpoint is in place, X'D7' is displayed in the memory display for the breakpoint address.

For example:

```
 ? B #=1 A=28ØØ
```

Puts a breakpoint (#1) at address X'2800'. The memory display for X'2800' will show a X'D7'.

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



```
RADIO SHACK MODEL II DEBUG PROGRAM
  2800      D7 1E EF 06 33 3E 34 CF   CD 14 EF 10 F8 3E 34 CF   ....3>4......>4.
  2810      CD 14 EF C9 C5 06 4F 0E   0D 3E 13 CF C1 C9 4E 4F   ......O..>....NO
  2820      20 45 52 52 4F 52 20 46   4F 55 4E 44 20 20 20 20    ERROR FOUND
  2830      20 20 20 20 20 20 20 20   20 20 20 20 20 20 20 20
  2840      20 20 20 20 20 20 20 20   20 20 20 20 20 20 20 20
  2850      20 20 20 20 20 20 20 20   20 20 20 20 20 20 20 20
  2860      20 20 20 20 20 20 20 20   20 20 20 20 20 20 00 00                 ..
  2870      00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ................
    PC     SP  SZHPNC  AF   BC   DE   HL   IX   IY   AF'  BC'  DE'  HL'
  2800  1EFC 000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
?
```

To delete a single breakpoint without affecting any others, press ENTER instead of providing a new address for the breakpoint. To delete all breakpoints, press E for "Empty breakpoint table".

C (Continue)

Press C to enter this command. It resumes execution of your program at the address pointed to by PC. Use it after the debugger has stopped at a breakpoint. The original instruction at the breakpoint address will be executed, but the breakpoint will remain in place.

D (Decimal Format)

Press D to enter this command. It displays all addresses in decimal form. However, the contents of all registers and memory addresses are still displayed in hexadecimal. In the decimal display format, you must enter all addresses as five-digit decimal numbers.

E (Empty Breakpoint Table)

Press E to empty the breakpoint table. All breakpointed instructions will be restored.

F (Find Hex String)

Press F to start this command. It will search in memory for a string up to 20 bytes long. You must enter the search string in hexadecimal format. Press ENTER when you have typed in the entire string. The debug monitor will display the first occurrence of the string. If it is not in the search area, the current memory display is unchanged.

For example:

```
    F S=28ØØ E=4ØØØ D=C3ØØ7Ø
```

Searches memory from X'2800' through X'4000' for the three-byte hexadecimal string X'C30070'. ("S = " is the prompt for start of search; "E = ", for end; "D = ", for data in hexadecimal form.)

X (Hex Format)

Press X to restore the Display to hexadecimal format. In this mode, all addresses must be entered as four-digit hexadecimal numbers.

J (Jump)

This is a two-step process:

1. Type in the J-command and the jump address. This will load the address into the PC register.
2. Then use the C command to continue execution at that point.

   For example:

   ```
   ?J A=2800
   ```

   Now press C to start execution at X'2800'.

L (Load or Copy memory to memory)

Type L to enter this command. It moves a block of memory. The debugger will prompt 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 moved.

The move is incremental: the first byte is moved to the first destination address, then the second to the second destination address, etc.

Examples:

```
    ? L S=28ØØ E=28FØ T=3ØØØ
```

Copies addresses from X'2800' to X'28F0' into memory from X'3000' to X'30F0'.

You can use this command to fill memory with a specific value, by putting the desired value in address *nnnn,* and using a command like this:

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

This will copy the value in *nnnn* into every location from *nnnn* + 1 to *xxxx* + 1. For example, if X'2800' contains a X'20', then the command:

```
? L S=28ØØ E=3ØØØ T=28Ø1
```

fills memory from 2801 to 3000 with X'20'.

O (Debug Off)

Type O to exit the debug monitor and turn off DEBUG. All breakpoints set by the B command will be removed from your program, and DEBUG will return to TRSDOS. If you simply want to exit DEBUG, type: S. DEBUG will remain "on".

P (Print Display)

Type P to send a copy of the Display to the Printer. Printer must have been initialized during TRSDOS startup or by the FORMS command.

M (Examine and Change Memory)

Type M to enter this command. The debugger will prompt you to type in the starting address of memory to be examined. As soon as you type in the complete address, the memory display will show the 128-byte area starting with that address. While the A= . . . . prompt is present, you can scroll through memory 16 bytes at a time by pressing ENTER .

To modify any memory in the display area, press F1 while the A= . . . . is displayed. The cursor will move up into the memory display area.

With the cursor in the memory display area, the cursor control keys are:

← → ↑ ↓	Cursor motion: back, forward, up, down
F1	Homes Cursor
TAB	Tabs Cursor
ENTER	Moves cursor to start of next row.
/	Moves cursor in and out of ASCII area

When the cursor is in the hexadecimal area, enter hexadecimal values. The debugger will update 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 enter this command. 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 will move over to the first byte of the register pair. While in the register modify mode, use the cursor control keys, ← and → 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 READY mode. The debugger is still on; when you load and execute a program, you will enter the debugger again.

DIR
List the Diskette Directory

DIR :*d* {SYS, PRT}

:*d* is a drive specification. (The colon : before *d* is optional.) If :*d* is omitted, drive 0 is used.

SYS tells TRSDOS to list system and user files. If SYS is omitted, only user files are listed.

PRT tells TRSDOS to list the directory to the Printer. If PRT is omitted, TRSDOS lists the directory on the Console Display.

In addition to the above syntax, DIR now allows the use of a special wild-card file specification field in place of the drive specification in which you specify an extension, but use a wild-card field in place of the file name. In the DIR command, the wild-card specification may only be used in the following manner:

DIR */*ext*:d

*/*ext* tells TRSDOS to list all files with the extension *ext*.

:*d* specifies which drive is to be used. If omitted, drive zero will be used.

The other DIR parameters may also be used with this syntax.

If the SYS option is used, only those system files with a matching extension will be included in the listing.

Examples

```
DIR
```

Displays the directory of user files in drive 0.

```
DIR 1 PRT
```

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

```
DIR { SYS, PRT }
```

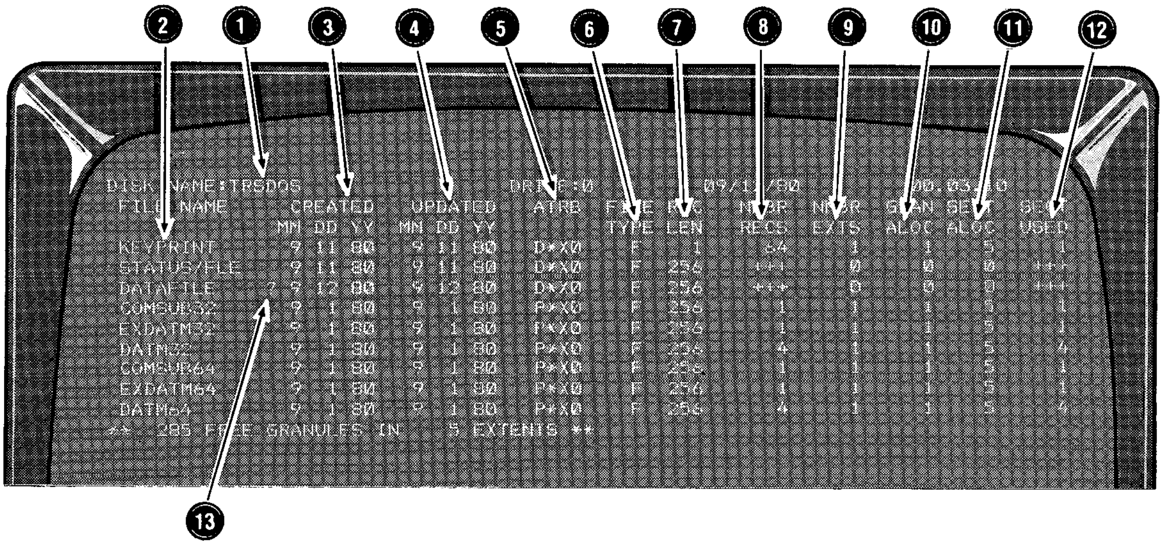
Lists to the Printer the directory of System and user files.

The braces are required to prevent TRSDOS from taking SYS as an invalid drive specification.

```
DIR */BAS:1
```

Lists all user files on drive one with the extension /BAS.

Sample Directory Listing



```
DISK NAME:TRSDOS                            DRIVE:Ø        Ø9/11/8Ø        ØØ.Ø3.1Ø
 FILE NAME      CREATED    UPDATED    ATRB   FILE RCD    NMBR  NMBR  GRAN  SECT  SECT
               MM DD YY   MM DD YY           TYPE LEN    RECS  EXTS  ALOC  ALOC  USED
 KEYPRINT        9 11 8Ø   9 11 8Ø   D*XØ    F     1      64     1     1     5     1
 STATUS/FLE      9 11 8Ø   9 11 8Ø   D*XØ    F   256     +++     Ø     Ø     Ø   +++
 DATAFILE     ?  9 12 8Ø   9 12 8Ø   D*XØ    F   256     +++     Ø     Ø     Ø   +++
 COMSUB32        9  1 8Ø   9  1 8Ø   P*XØ    F   256       1     1     1     5     1
 EXDATM32        9  1 8Ø   9  1 8Ø   P*XØ    F   256       1     1     1     5     1
 DATM32          9  1 8Ø   9  1 8Ø   P*XØ    F   256       4     1     1     5     4
 COMSUB64        9  1 8Ø   9  1 8Ø   P*XØ    F   256       1     1     1     5     1
 EXDATM64        9  1 8Ø   9  1 8Ø   P*XØ    F   256       1     1     1     5     1
 DATM64          9  1 8Ø   9  1 8Ø   P*XØ    F   256       4     1     1     5     4
** 285 FREE GRANULES IN    5 EXTENTS **
```

What the column headings mean

❶ **Disk Name** — The name assigned to the diskette when it was formatted.

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

❸ **Creation Date** — When the file was created.

❹ **Update** — When file was last modified.

❺ **Attributes** — A four-character field.

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 (no passwords).
A	The file has an access word but no update word.
U	The file has an update word but no access word.
B	The file has both update and access words.

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

0,1	Kill file and everything listed below.
2	Rename file and everything listed below.
3	Not used
4	Write and everything listed below.
5	Read and everything listed below.
6	Execute only.
7	None.

6 **File Type** — Indicates the record type for the file.

F Fixed-length records.
V Variable-length records.

7 **Record Length** — Assigned when the file was created (applies to fixed-length record files only).

8 **Number of Records** — How many logical records have been written. Plus signs ("+") signify none have been written or file has variable length records and number written cannot be calculated. If number exceeds 65535, it starts over at zero. That is, it is a modulo 65536 number. True number of records can be inferred from Sectors Used column.

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

10 **Granules Allocated** — How many granules are allocated to the file.

11 **Sectors Allocated** — How many sectors (256 byte blocks) have been allocated to the file.

12 **Sectors Used** — Shows how many sectors have data written into them. Plus sign ("+") means no data in file.

13 **A question-mark** — means the file is open or was not properly closed.

DO
Begin Auto Command Input from Disk File

DO *file*
file specifies a file created with the BUILD command

This command reads and executes the lines stored in a special-format file created with the BUILD command. The System executes the commands just as if they had been typed in from the Keyboard, except that they are not echoed to the Video Display (except for PAUSE).

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

When DO reaches the end of the automatic command input file, it relinquishes control to the last command it causes to be executed.

The DEBUG command cannot be included in an automatic command input file.

Running User Programs from a DO-file

In addition to executing TRSDOS library commands, you can load and execute user programs from a DO-file. You will probably want to make your program name be the last line in the DO-file (see **Note**). Before the DO-processor starts the last line in the DO-file, it shuts off certain special functions so that your program may execute normally (see **Note** below). For example, if you wanted to perform some library commands and then run a BASIC program called MENU, you would make this the last line in your program:

```
BASIC MENU
```

You can also "chain" do files, by putting another DO command at the end of a DO-file. The DO command *must* be the last line in the DO-file.

Note: You can run user programs from the middle of a DO-file. The program will run normally, with one important exception: pressing BREAK while your program is executing will interrupt your program, terminate DO-file processing, and return you to TRSDOS. Furthermore, your user program cannot set up a BREAK -processing program (see SETBRK).

Examples

```
    DO STARTER
```

TRSDOS will begin automatic command input from STARTER.

```
    AUTO DO STARTER
```

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

Sample Use

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

```
    FORMS W=80
    CLOCK ON
    VERIFY OFF
```

Then use BUILD to create such a file. If you called it BEGIN, then use the command:

```
    AUTO DO BEGIN
```

to perform the commands each time TRSDOS starts up.

DUAL
Duplicate Output to Video and Printer

DUAL {*switch*}

switch is one of the following:

ON	Turns on dual routing
OFF	Turns off dual routing

This command causes all video output to be copied to the printer, and all printer output to be copied to the display.

Notes:

1. Printer and display output may be different, due to intrinsic differences in the output devices and in the output software.
2. Having dual routing on will slow down the video output process.
3. If your printer is currently off-line or not powered-up the Video Display will appear to be outputting about one character every three seconds (30 seconds if you have run the LPII patch file). You should turn off DUAL or ready your printer to remedy this.

Sample Use

For a hard copy of all the system/operator dialog, turn on DUAL:

```
DUAL ON
```

DUMP
Store a Program Into a Disk File

DUMP *file*{START=*address-1*, END=*address-2*, TRA=*address-3* RELO=*address-4, RORT=letter*}

file is a file specification.

START=*address-1* specifies the start address of the memory block.

END=*address-2* specifies the end address of the memory block.

TRA=*address-3* specifies the transfer address, where execution starts when the program is loaded. If omitted, *address-4* is used.

RELO=*address-4* specifies the start address for loading the program back into memory. If omitted, *address-1* is used.

RORT=*letter* specifies whether the program is directly executable from TRSDOS. RORT stands for "RETURN OR TRANSFER". If RORT=R, then TRSDOS can load but not execute the file. If RORT=T, then TRSDOS can load and execute the file from the TRSDOS READY mode. If RORT is omitted, RORT=T is used.

Note: Addresses must be hexadecimal form, without the X' ' notation.

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

You can enter machine language programs directly into memory, via the DEBUG command.

Examples

```
DUMP LISTER/CMD  START=7000, END=7100, TRA=7004
```

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

```
DUMP PROG2/CMD START=6000, END=6F00, TRA=3010, RELO=3000
```

Creates a program file named PROG2/CMD containing the program in addresses X'6000' to X'6F00'. When loaded, PROG2/CMD will reside from X'3000' to X'3F00'. Execution will start at X'3010'. The program is executable from TRSDOS READY.

```
DUMP ROUTINE/1 START=6F00, END=6FFF, RORT=R
```

Creates a program file which cannot be executed from the TRSDOS READY level. Typically, this would be a routine to be called by another program.

ECHO
Echo Keyboard Input to the Display

```
ECHO
```

This command allows you to type information on the Display without having TRSDOS interpret it as a command. Press BREAK to stop ECHO and return to TRSDOS READY.

Sample Use

You can key data directly to the printer by turning on dual routing and then starting ECHO:

```
DUAL ON
ECHO
```

Now whatever you type will be output to the printer as well as to the display. (Most printers will not print the line until a carriage return is received or the input buffer is filled.)

ERROR
Display Error Message

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

This command displays a descriptive error message. When TRSDOS gives you a reverse (black-on-white) message like:

```
* * ERROR 47 * *
```

You type back

```
ERROR 47
```

to see the full error message.

Example

```
ERROR 3
```

Gives you the message

```
PARAMETER ERROR ON CALL
```

For a complete list of error codes, messages and explanations, see pages 4/11-4/12 of this manual.

FORMS
Set Printer Parameters

There are two syntaxes for this command:

A. FORMS {*initialization-options*}
B. FORMS {*control-switch*}

initialization-options may be any combination of the following:

{ } FORMS without any options sets all options to their default values (listed below).

P = *page length* *page length* is a decimal number from 0 to 255 telling TRSDOS how many lines to a page. If omitted, 66 is used.

L = *lines* *lines* is a decimal number from 0 to 255 telling TRSDOS the maximum number of lines to print on a page before doing an automatic top of form. If omitted, 60 is used. *lines* must always be less than or equal to *page length*. If either equals 0, both must equal 0. In this case, no automatic form feeds are done, and ASCII form feeds and vertical tabs are sent directly to the printer with no translation.

W = *width* *width* is a decimal number from 0 to 255 telling TRSDOS the maximum number of characters to print on a line before doing an automatic carriage return. If omitted, 132 is used. If W=0, no automatic carriage returns are done and ASCII TABs are sent directly to the printer with no translation.

C = *control code* *control code* is a one-byte hexadecimal code which will be output to the printer upon completion of the FORMS command.

control-switch may be any one of the following:

T tells TRSDOS to issue a form feed X'0C' character.

X tells TRSDOS to send all data directly to the printer without any translation ("transparent" mode).

D tells TRSDOS to ignore all printer output. TRSDOS will not check printer status ("dummy" mode).

N tells TRSDOS to return to "normal" (non-transparent, non-dummy) mode.

A tells TRSDOS to output a line feed after each carriage return ("auto line feed" mode) even if transparent mode is in effect. Line count is updated by carriage returns but not by line feed characters.

Q cancels auto line feed mode.

S tells TRSDOS to switch to serial Channel B printer driver (must do SETCOM before any printing can be done).

R tells TRSDOS to return to parallel printer driver.

This command lets you set up the TRSDOS Printer software to suit the Printer you have attached. If the default parameters P = 66, L = 60, W = 132, and C = 0 are appropriate, you do not need to use this command.

Summary of Special Option Combinations

Options	Result
P > L ≠ O	Auto top of form; translate X'OC' and X'OB' as carriage return/line feed to advance paper to top of next page.
P = L ≠ O	No auto top of form, but translate X'OC' and X'OB' as carriage returns/line feeds to advance paper to top of next page.
P = L = O	No auto top of form; send X'OC' and X'OB' directly to printer.
W ≠ O	Automatic carriage return after W characters in a single line; translate X'09' as 1-8 spaces to perform tab function.
W = O	No automatic carriage return; send X'09' directly to printer without translation.
P = L = W = O	Same as transparent mode except that line and width counters are still updated.

For a complete discussion of TRSDOS printer software, see PRCHAR, PRINIT, and PRCTRL in the **Technical Information** Section.

Examples

```
FORMS
```

Resets all parameters to their default values.

```
FORMS L=56
```

Resets the maximum number of printed lines per page to 56, leaving 10 lines blank on each page.

```
FORMS C=14
```

Sends the initialization code X'14' to the Printer after resetting the default parameters.

```
FORMS T
```

Advances Printer to top of form. Useful when you have done some printing and want to start next printing at top of form.

```
FORMS S
```

Sets up the serial printer driver.

Setting the Parameters

Page Size. Multiply your form length in inches by the number of printed lines per inch to get the appropriate value. Most Printers print 6 lines per inch. Therefore standard 11-inch forms have a page size of 66 lines. That's why the default is PAGE=66.

Lines per page. This number determines the number of blank lines at the bottom of each page. If you set *lines* equal to *page size*, then TRSDOS will print every line on the page. If you set *lines* equal to *page size* minus 6, then TRSDOS will leave 6 blank lines on each page. Lines per page cannot exceed *page size.*

Width. This number sets the maximum number of characters per line. If a print line exceeds this width, TRSDOS will automatically break the line at the maximum length and continue it at the beginning of the next Print line.

Control Codes. Some Printers require an initialization code (for example, to set up for double-size characters). The code you specify is sent to the Printer during execution of the FORMS command.

Using a Serial Printer

The serial printer driver uses channel B on the back panel of the display console. Connect your prịnter to this channel. Radio Shack's RS-232-2 Cable, Catalog Number 26-4403, will work with many serial printers. If channel A is not connected, place a serial terminator plug on that channel.

Before initializing the serial printer driver with FORMS, you must connect the printer and execute the SETCOM command with parameters appropriate for your printer. For example, if your printer uses 300 baud, 7-bit words, no parity and 1 stop bit, you would use a command like this:

```
SETCOM B=(300, 7, N, 1)
```

Then you would execute the FORMS command.

Technical Information

The serial printer driver uses the following pins of channel B (refer to the Model II Operation Manual for a pin diagram):

Signal Name	Pin #
GROUND	1, 7
DATA SET READY	6
CLEAR TO SEND	5
CARRIER DETECT	8
TRANSMIT DATA	/2
REQUEST TO SEND	4
DATA TERMINAL READY	20

Note: If your serial printer does not support the CLEAR TO SEND signal, connect pins 5 and 20 on channel B. If it does not support the DATA SET READY signal, connect pins 6 and 20 on channel B.

For a complete discussion of TRSDOS printer software, see PRCHAR, PRINIT, and PRCTRL in the **Technical Information** Section.

During serial printer operation, TRSDOS will recognize two characters from the printer:

ASCII Name	Hex Code	Function
DC3, "CTRL-S"	13	Pause printing
DC1, "CTRL-Q"	11	Resume printing

FREE
Display Disk Allocation Map

FREE *:d* PRT

:d is a drive specification. (The colon : before *d* is optional.) If *:d* is omitted, drive 0 is used.

PRT tells TRSDOS to send the map to the Printer. If PRT is omitted, TRSDOS sends the map to the Console Display.

This command gives you a map of granule allocation on a diskette. (A granule, 1280 bytes, is the unit of space allocation.) This information is useful when you want to optimize file access time.

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

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

Example

```
FREE
```

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

```
FREE {PRT}
```

Lists the free space map for drive 0 to the Printer. The braces are required in this example, since no drive specification is included. Otherwise TRSDOS would take PRT as an invalid drive specification.

```
FREE 2 PRT
```

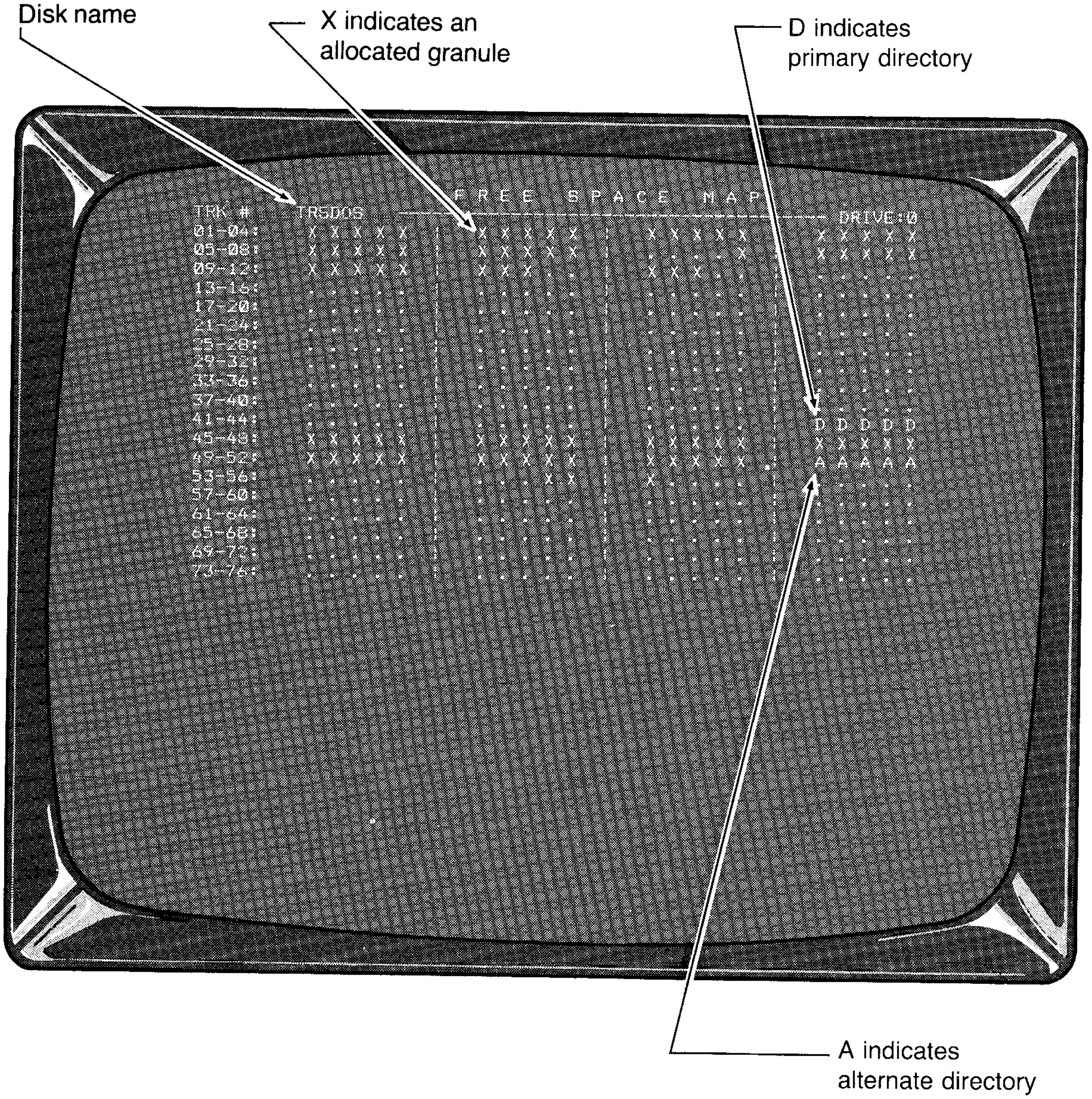
Lists the Drive 2 map to the Printer.

A Typical FREE Display

Four special symbols are used in the FREE map:

.	Unused Granule
D	Directory Information
X	Allocated Granule
F	Granule Contains a Flawed Sector (Unusable)
A	Alternate directory.

Here's a typical display:



Disk name
X indicates an allocated granule
D indicates primary directory
A indicates alternate directory

```
                         F R E E   S P A C E   M A P
TRK #     TRSDOS   --------------------------------------------- DRIVE:Ø
Ø1-Ø4:    X X X X X  |  X X X X X  |  X X X X X  |  X X X X X
Ø5-Ø8:    X X X X X  |  X X X X X  |  . . . . X  |  X X X X X
Ø9-12:    X X X X X  |  X X X . .  |  X X X . .  |  . . . . .
13-16:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
17-2Ø:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
21-24:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
25-28:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
29-32:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
33-36:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
37-4Ø:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
41-44:    . . . . .  |  . . . . .  |  . . . . .  |  D D D D D
45-48:    X X X X X  |  X X X X X  |  X X X X X  |  X X X X X
49-52:    X X X X X  |  X X X X X  |  X X X X X  |  A A A A A
53-56:    . . . . .  |  . . . X X  |  X . . . .  |  . . . . .
57-6Ø:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
61-64:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
65-68:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
69-72:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
73-76:    . . . . .  |  . . . . .  |  . . . . .  |  . . . . .
```

HELP
Help with TRSDOS Commands

```
HELP subject
```

subject specifies what TRSDOS command or general subject you need help with. If *subject* is omitted or is not in the help-list, TRSDOS will display a list of commands and general subjects for which help is available.

Sample Use

```
HELP MOVE
```

gives the syntax for the MOVE command.

```
HELP SYNTAX
```

explains the format of the HELP messages.

HOST
Operate as a Host to a Remote Terminal

HOST {*switch*}

switch is one of the following:

ON Turns host function on.

OFF Turns host function off.

If *switch* is omitted, the on/off status is displayed.

This command allows the Model II to accept keyboard input from the RS-232-C interface, and to transmit all display output to the same interface. Serial Channel A is always used. While HOST is on, the Model II keyboard and the remote terminal can both provide keyboard input; Model II output is duplicated to the Video Display and to Channel A. Remote characters have a higher priority than local characters.

When you start HOST, you have the option of enabling the remote BREAK key. If you enable remote BREAK, the terminal will be able to interrupt operations just like the console keyboard, by sending an ASCII X'03'. If you disable remote BREAK, that code will be ignored when it comes in via the serial interface. The HOST will process a remote X'00' (null char) as if it were the local HOLD key.

Notes

1. Before turning on the host function, initialize Channel A with the SETCOM command.
2. At the remote computer, use the Model II TERMINAL program, or its equivalent.
3. Channel A may not be turned off while HOST is active.

Sample Use

You have another computer connected to the Model II via a telephone modem. The remote computer is running the Terminal program, and you want the Model II to accept commands from the remote computer. Type in this command on the Model II:

```
HOST ON
```

This tells the Model II to accept ''keyboard'' input from the remote terminal, and to echo all display output to the remote terminal.

To stop HOST operation, type:

```
HOST OFF
```

I
Swap Diskettes

```
I
```

You should execute this command immediately *after* swapping diskettes. It tells TRSDOS to read the diskette ID's on all drives in the system.

Note: Do not swap diskettes while a file is open.

See VERIFY for related information.

Example

```
I
```

tells the System you have changed one of the diskettes.

KILL
Delete a File

KILL *file*

file is a file specification.

Kill also allows for wild card fields in the file name and/or extension:

KILL *wildcard/wildcard*

If no drive is specified, TRSDOS will search for all matching files on drive 0 only. If a drive is specified, TRSDOS will search only that drive.

This command deletes a file from the directory and frees the space allocated to that file. If no drive is specified, TRSDOS will search for the file, starting with drive 0. Before deleting the file, TRSDOS will display the file name and the drive that contains the file. Type Y ENTER to Kill the file, N ENTER to not kill the file, or Q ENTER to cancel the command.

Examples

```
KILL TESTPROG/BAS
```

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

```
KILL JOBFILE/IDY.foggy
```

Deletes the named file from the first drive that contains it. The file is protected with the password foggy.

```
KILL FORM/123:3
```

Deletes FORM/123 from drive 3.

```
KILL MY*:1
```

Tells TRSDOS to kill all drive 1 files which have a file name beginning with MY and have no extension. TRSDOS will prompt you before killing each file.

```
KILL */BAS
```

Tells TRSDOS to kill all files on drive 0 which have the extension /BAS.

Sample Uses

When updating a file, it is a good practice to input from the old file and output updated information to a new file. That way, if the update is wrong, you still have the old file as a backup. When you have verified that the update file is correct, you can Kill the old file.

KILL is also useful in conjunction with pre-allocated files. Suppose you have finished writing to a pre-allocated file, and one or more granules are unused in the pre-allocated file. Then you can copy the pre-allocated file to a dynamically allocated file, and afterwards Kill the pre-allocated file. This is the only way to reduce the size of a pre-allocated file.

LIB
Display Library Commands

```
LIB
```

This command lists to the Display all the Library Commands.

Example

```
LIB
```

LIST
List Contents of a File

LIST *file* {PRT, SLOW, R=*record-number*}
file is a file specificaton.
PRT tells TRSDOS to list to the Printer. If PRT is omitted, the Console Display is used.
SLOW tells TRSDOS to pause briefly after each record. If omitted, the listing is continuous.
R=*record number* tells TRSDOS the starting record for the listing. *record number* must be in the range [1,65535]. If omitted, record 1 is used.

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

To stop the listing, press HOLD . Press HOLD again to continue. Press ESC or BREAK to terminate the listing.

Examples

```
LIST DATA/BAS
```

Lists the contents of DATA/BAS.

```
LIST TEXTFILE/1 SLOW
```

Lists the contents of TEXTFILE/1, pausing after each record.

```
LIST TEXTFILE/1 R=100, A
```

The listing starts with the 100th record in TEXTFILE/1. Only ASCII characters are displayed.

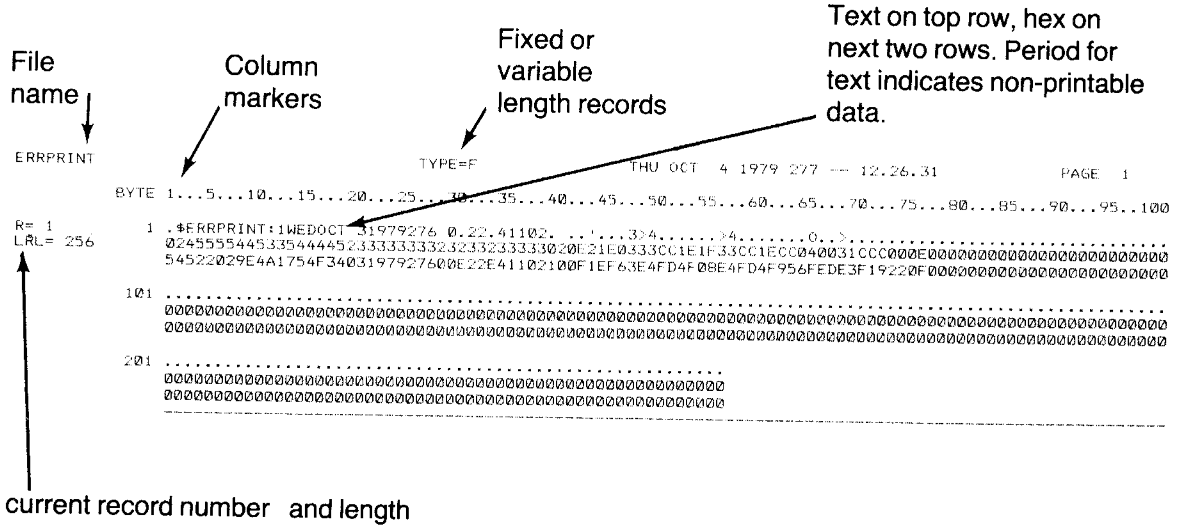
```
LIST PROGRAM/CMD PRT
```

Lists the file PROGRAM/CMD to the Printer.

Listing Format

LIST numbers each record as it is listed, and prints a heading showing the relative position of each byte in the record. Here's a sample listing after the command:

```
LIST ERRPRINT PRT
```



```
ERRPRINT                             TYPE=F             THU OCT  4 1979 277 -- 12.26.31            PAGE  1
           BYTE 1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95..100
R= 1          1 .$ERRPRINT:1WEDOCT 31979276 0.22.41102. ..'...3>4......>4.......O..>................................
LRL= 256        0245555445335444452333333332323323333020E21E0333CC1E1F33CC1ECC040031CCC000E0000000000000000000000000
                54522029E4A1754F340319792760 0E22E41102100F1EF63E4FD4F08E4FD4F956FEDE3F19220F000000000000000000000000

            101 ....................................................................................................
                0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

            201 ........................................................
                00000000000000000000000000000000000000000000000000000000
                00000000000000000000000000000000000000000000000000000000
                ----------------------------------------------------------------------------------------------------
```

File name

Column markers

Fixed or variable length records

Text on top row, hex on next two rows. Period for text indicates non-printable data.

current record number and length

LOAD
Load a Program File

LOAD *file*

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

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

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

Example

```
LOAD PAYROLL/pt1
```

Note: You may not load a program file if it would overlay TRSDOS. The code must be loaded above X'27FF'.

Sample Use

Often several program modules must be loaded into memory for use by a master program. For example, suppose PAYROLL/pt1 and PAYROLL/pt2 are modules, and MENU is the master program. Then you could use the commands:

```
LOAD PAYROLL/pt1
LOAD PAYROLL/pt2
```

to get modules into memory, and then type:

```
MENU
```

to load and execute MENU.

If PAYROLL/pt1 and PAYROLL/pt2 were Dumped with RORT=R, then you can load by typing the file name without the LOAD command, i.e.,

```
PAYROLL/pt1
PAYROLL/pt2
```

After each is loaded, TRSDOS READY returns.

MOVE
Copy Multiple User Files, Reorganize a Diskette

This command takes two forms, A) and B) below.

A) MOVE *sourcefile* TO *drive* {ABS, PROMPT}

sourcefile is required. It is a file specification in which the name and extension may be wild-cards. A password *may not* be used. The wild-card fields allow you to specify a collection of files with part of the filename/extension in common.

drive is required. It is the destination of the copy operation. It can specify any of drives 0 through 3 (single-drive MOVEs are allowed).

ABS is optional. It tells TRSDOS to perform each copy operation even if it will cause an existing file to be overwritten on the destination drive. If ABS is omitted, TRSDOS will prompt you before overwriting a file.

PROMPT is optional. It tells TRSDOS to display each file before it is copied, and give you a set of options for that file. The options are Y/N/S/Q (Yes—Copy; No—Don't Copy; Stop prompts and proceed with all copies; Quit this command—no more copies.) If PROMPT is omitted, TRSDOS will copy all files that match the wild-card specification.

B) MOVE *source-drive* TO *destin-drive* {ALL, ABS, PROMPT}

source-drive is required. It specifies which drive will contain the files to be copied.

destin-drive is required. It specifies the destination of the copy operation.

ALL tells TRSDOS to move all user files. This parameter is required for form B of the MOVE command.

ABS and PROMPT are optional. They are explained above.

This command is similar to COPY, except that it allows you to copy a collection of files with a single command line. Another difference is that the destination file is always given the same name as the source file.

Note

Only user files without passwords may be moved.

Sample Use

Suppose you want to copy the following files from drive 0 to another diskette on drive 1:

```
SORT/SRC
SORT/LST
SORT/REL
SORT/OBJ
```

Then use this command:

```
MOVE SORT/* TO 1 ABS
```

TRSDOS will automatically copy all four files, and any others which match the wild-card specification.

MOVE is also useful when you want to reduce the segmentation of files on a diskette. The files will retain their relative positions in the directory listing. BACKUP cannot serve this function, since it duplicates the file segmentation of the source diskette.

To perform such an operation from drive 0 to 1, put a formatted but empty diskette in drive 1, and use this command:

```
MOVE 0 TO 1 ALL
```

You may perform this operation in a single-drive system, too. Specify drive 0 as the destination drive. There will be at least one swap per file to be moved. TRSDOS will prompt you as required to swap source and destination diskettes.

PAUSE
Pause Execution for Operator Action

PAUSE *prompting message*

prompting message is an optional message to be displayed during the pause.

This command is intended for use inside a DO file. It makes TRSDOS print a message and then wait for the operator to press ENTER .

Example

```
PAUSE                    Insert Diskette #21
```

Prints PAUSE followed by the message and prompts the operator to press ENTER to continue.

```
PAUSE
```

Prints PAUSE and prompts the operator to press ENTER to continue. See BUILD and DO for sample uses.

PRINT
Print a Text File

PRINT *textfile* {A, V}

textfile specifies the file to be printed. For normal use of the command, the file should contain text characters only. It can have fixed-length or variable-length records.

A tells TRSDOS to treat the first byte in each record as a forms control character:

ASCII Contents of First Byte	Control Function
"1"	Do a form feed before printing (top of form)
"ƀ"	Carriage return before printing (single-space)
"Ø"	Two carriage returns before printing (double-space)
" + "	Carriage return without line-feed advance. If your printer can do a carriage return without a line-feed, this control code will cause the following characters to be overprinted on the current line.

Only use this option when *textfile* contains these forms control characters (e.g., RSCOBOL list-files).

V tells TRSDOS to output to the video as well as to the printer. If V is omitted, only the printer is used.

This command gives you printout of a text file. It does not show the record numbers and hexadecimal codes (LIST does that).

Notes:

1. PRINT always tries to output to the printer. If you do not have a printer on-line, this will normally give you an error. To avoid this, first execute the command:

   ```
   FORMS D
   ```

   which "dummies" all printer output. You can then PRINT to the video by using the V option.

2. PRINT does a top of form before it starts printing.

3. Unprintable characters (undefined control codes and codes > 127) are printed as periods.

Sample Use

Suppose you have a BASIC program named PROGRAM/TXT. It was saved in ASCII-format. You can print it without starting BASIC. Type:

```
PRINT PROGRAM/TXT V
```

for video and printer output.

PROT
Use Diskette's Master Password

PROT :*d* {OLD=*password, options*}

:*d* is a drive specification. The colon is optional

OLD=*password* specifies the diskette's current master password. (Your TRSDOS diskette is supplied with the password PASSWORD.)

options include any of the following:

NEW=*password* Gives TRSDOS the new master password (up to eight alphanumeric characters).

LOCK Tells TRSDOS to protect all user files with the latest master password. Update and access words will both be set to this password.

UNLOCK Tells TRSDOS to remove passwords from all user files.

If LOCK and UNLOCK are omitted, user file protection is left unchanged. If one is used, the other must be omitted.

PROT changes file protection on a large scale. If you know the diskette's master password, you can change it. You can also protect or un-protect all user files.

A diskette's master password is initially assigned during the format or backup process. The TRSDOS diskette is supplied with the master password PASSWORD.

Example

```
PROT 1   OLD=PASSWORD,  NEW=H2O
```

Tells TRSDOS to change the master password of the drive 1 diskette from PASSWORD to H2O.

```
PROT Ø   OLD=H2O, UNLOCK
```

Tells TRSDOS to remove passwords from every user file on the drive 0 diskette (must have the password H2O).

```
PROT Ø   OLD=H2O, NEW=ELEPHANT, LOCK
```

Tells TRSDOS to change the master password from H2O to ELEPHANT and assign the new one to every user file.

PURGE
Delete Files

PURGE :d {*file-class*}

:d is drive specification. The colon : is optional. If *:d* is omitted, drive 0 is used.

file-class is one or more of the following:

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

If *file-class* is omitted, DATA is used.

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

Before eliminating any system files, read **Disk File Requirements** in Section 0.

When the command is entered, TRSDOS will ask for the diskette's password. Type in up to 8 characters, and press ENTER. The System will then display user file names one at a time, prompting you to Kill or leave each file or Quit the operation.

Example

```
PURGE 1
```

TRSDOS will let you purge data files from drive 1. **This would include BASIC programs.**

```
PURGE
```

TRSDOS will let you purge data files from drive 0.

RECEIVE
Receive Object Code via RS-232-C

RECEIVE {CH = *channel, offset*}

channel is one of the serial channels, A or B. CH = *channel* is required.

offset is optional. It tells TRSDOS to offset each load address specified in the incoming data (see "Required Data Format", below). It must use one of the following forms:

offset	**Result**
ADD = *hhhh*	The data load address is incremented by the hexadecimal value *hhhh.*
SUB = *hhhh*	The data load address is decremented by the hexadecimal value *hhhh.*

If *offset* is omitted, the data loads at the address specified in the incoming data.

This command lets you receive object code into RAM from another device (a Model I, II, or III, or other computer). The data must be sent in Intel Hex Format as described later on.

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

Notes

1. The data will be 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 last value.)

2. 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 = *hhhh* option to specify the original load address.

Sample Uses

1. You have initialized Channel A properly, and have established a connection with the sending device. You know that the data load address is above X'2FFF' and below the end of user memory.

   Now type:

   ```
   RECEIVE CH = A
   ```

   If the connection is good, Model II will display:

   ```
   Ready to receive          # =
   ```

   The sending device can now begin the transmission. The number of the current record will be displayed after # = . When the transmission is complete, Model II will display:

   ```
   START ADDRESS = aaaa     LAST ADDRESS = bbbb     TRA ADDRESS = cccc
   TRSDOS READY
   ```

   aaaa, bbbb, cccc are hexadecimal addresses.

   The code will be in the memory area specified in the data itself. You can now use the DUMP command to create a program file on diskette.

2. You have initialized channel B properly, and you have established a connection with the sending device. But the load address is X'2800'. The end address is X'37FF'. You need to add X'0800' to this address so that the receive program won't be overlaid. Type:

   ```
   RECEIVE   CH = B   ADD = 0800
   ```

   This command causes the data to be loaded starting at X'2800' + X'0800' = X'3000'.

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

   ```
   DUMP   PROGRAM1   START = 3000   END = 3FFF   RELO = 2800
   ```

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

Required Data Format

The transmitting program must send the data in "Intel® Hex Format", described below.

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

1) high nibble (most significant four bits), sent as first byte of pair.
2) low nibble (least significant four bits), sent as 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, data is always in the range [X'30', X'3A'] or [X'41', X'46']. Values outside this range will terminate reception and produce an error message.

RECORD FORMAT

Records must be sent as follows:

CHARACTER NUMBER	CONTENTS	COMMENTS
1	“:”	Sync-character to indicate beginning of record.
2	High nibble of record length (N)	This 2-byte sequence gives the number of byte PAIRS in this record. Zero means 256 byte pairs follow.
3	Low nibble of record length (N)	
4	High nibble of msb of load addr.	This 4-byte sequence gives address where the data is to start loading. Address specified must be in the user area [X'2800', TOP].
5	Low nibble of msb of load addr.	
6	High nibble of lsb of load addr.	
7	Low nibble of lsb of load addr.	
8	High nibble of lsb of EOF (end of file) code	This byte-pair gives the EOF code. Any non-zero value means end of file (no more records follow). A value of zero means more records follow.
9	Low nibble of EOF code	
10	First byte of first data pair	First byte is ASCII code for first hex digit; second byte is ASCII code for second hex digit.
11	Second byte of 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	

CHARACTER NUMBER	CONTENTS	COMMENTS
10 + (N*2)	First byte of data checksum (high nibble)	This pair represents 2's complement of the data (all byte pairs after the ":" up to but not including the checksum). Note that each byte pair is converted 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	"0"	30
3	"2"	32
4	"2"	32
5	"8"	38
6	"0"	30
7	"0"	30
8	"0"	30
9	"0"	30
10	"3"	33
11	"7"	37
12	"7"	37
13	"0"	30
14	"A"	41
15	"3"	33

This record will contain 2 byte-pairs of data:
"3" "7" representing the value X'37'
"7" "0" representing the value X'70'
and will start loading at X'2800'. The one-byte sum of the original bytes (represented in pairs by characters 2 through 13) is X'5D'. The 2's complement of X'5D' is X'A3' — which is represented in bytes 14 and 15.

RENAME
Rename a File

RENAME *file-1* TO *file-2*

file-1 and *file-2* are file specifications. If *file-2* includes a drive specification or password, it will be ignored. The file will retain its former password, if any.

b̸TOb̸ is a delimiter. A comma or space may also be used.

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

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

Examples

```
   RENAME Miss/BAS TO Ms/BAS
```

TRSDOS will search for Miss/BAS starting with drive 0, and will rename it to Ms/BAS.

```
   RENAME REPORT/AUG:3 TO REPORT/SEP
```

Renames REPORT/AUG on drive 3 to REPORT/SEP.

```
   RENAME MASTER.12345678 TO MASTER/A
```

Searches for MASTER and renames it to MASTER/A. The password 12345678 must grant at least RENAME access (see Passwords in chapter 1). The renamed file has the same password.

RESET
Reset/Restart TRSDOS

```
RESET
```

Executing the command is equivalent to pressing the RESET switch. The RESET command closes all open files.

Sample Use

```
RESET
```

SCREEN
Copy Screen to Printer

```
SCREEN
```

This command reads the contents of the display and outputs it to the printer. Graphics characters will be represented as periods, and reverse alphanumeric characters will be represented as normal characters.

Sample Use

You want to save a copy of a particular command line which is still on the screen. Type:

```
SCREEN
```

SETCOM
Set Up RS-232C Communications

SETCOM { A=(*baud rate, word length, parity, stop bits*),
B=(*baud rate, word length, parity, stop bits*) }

A=(*options*) tells TRSDOS to initialize channel A.
To turn channel A off, use A=OFF instead of A = (*options*)
If A = (*options*) is omitted, status of channel A is unchanged.

B=(*options*) tells TRSDOS to initialize channel B.
To turn channel B off, use B=OFF.
If B = (*options*) is omitted, status of channel B is unchanged.

The options tell TRSDOS what RS-232C parameters to use. The following parameters are available:

baud rate	110, 150, 300, 600, 1200, 2400, 4800, 9600 If not specified, 300 is used.
word length	5, 6, 7, 8 If not specified, 7 is used.
parity	E for even, O for odd, N for none If not specified, even is used.
stop bits	1, 2 If not specified, 1 is used.

Every option but the last must be followed by a comma. The options are positional, e.g., the third item in an option list must always specify parity. To use a default value, omit the option. If you want to list subsequent options, you must include a comma for each default.

SETCOM without any options tells TRSDOS to display the status of both serial channels.

This command initializes RS-232C communications via channels A and B on the back panel. Before executing it, you should connect the communications device (modem, etc.) to the Model II.

To **change** the settings on a currently active channel, you must first turn the channel **off**. If the channel is already off when you try to turn it off, you'll get an error message.

See the Model II Operation Manual for a description of RS-232C signals used in channels A and B. For hard-wired connection from one Model II to another, see the wiring diagram in Technical Information, RS232C supervisor call.

SETCOM uses the Special Programming Area above TOP (see Memory Requirements). To use the serial I/O channels from BASIC you must execute SETCOM **before** starting BASIC.

Once you initialize a channel, you can begin sending and receiving data, using six system routines that are set up during initialization:

ARCV	Channel A receive, function code 96
ATX	Channel A transmit, function code 97
BRCV	Channel B receive, function code 98
BTX	Channel B transmit, function code 99
ACTRL	Channel A control, function code 100
BCTRL	Channel B control, function code 101

These system routines are only available when the respective channel has been initialized, See **Technical Information** for details.

Examples

```
SETCOM A=( )
```

Sets up channel A for serial communications, using all the default parameters. System function calls 96 and 97 are available for serial I/O. The status of channel B is unchanged.

```
SETCOM B=(4800, 8, , 2), A=OFF
```

Sets up channel B:

baud rate	4800
word length	8 bits
parity	Even (default)
stop bits	2

and turns off channel A.

```
SETCOM A=(2400, 8, O), B=( , , , 2)
```

Sets up channels A and B:

	Channel A	**Channel B**
baud rate	2400	300 (default)
word length	8	7 (default)
parity	Odd	Even (default)
stop bits	1 (default)	2

```
SETCOM
```

displays the status of both channels.

```
SETCOM  A = OFF, A = ()
```

resets channel A to default parameters.

SPOOL
Capture Printer Output or Print a Spool File

SPOOL {*switch*}

switch controls the spool function. If *switch* is omitted, the SPOOL status is displayed.

switch may be any one of the following:

ON activates the spooler. This switch must be used before any of the following switches can be used.

OFF turns off the spooler, and closes the capture- and print-files.

N, F = *file* closes the current capture-file and opens a new one specified by *file*. If *file* is omitted, subsequent printer output will be dummied (ignored) until a capture file is named with N, F = *file*.

P, F = *file*, K, C = *copies*, L = *line* begins spool-printing.

file is the file to be printed.

K Tells TRSDOS to keep the print-file after printing it. If K is omitted, the file will be deleted after it is printed. TRSDOS will not delete a print-file if the file is closed by a SPOOL S command or if a disk error occurs in the print-file.

C = *copies* Specifies how many copies you want. If omitted, one copy is made. C may be any number from 1 to 255.

L = *line* Specifies the line number where printing starts. A line is defined sequence of characters terminated by a carriage return. If omitted, printing starts at line one. L may be any number from 1 to 65535.

H Halts spool-printing but saves the current position for later resumption ("R" switch).

R, L = *line* Resumes spool-printing after a halt (H switch), or displays the current line number if the spooler has not been halted. If L = *line* is used, printing resumes at the specified line. If L = *line* is omitted, printing resumes from where it was halted.

S Stop printing. Closes but does not kill the print-file; leaves the capture-file open.

The purpose of a spooler is to increase the efficiency of the system (i.e., reduce the CPU idle time), and to allow you to use the system while a print operation is in progress. The TRSDOS spooler can perform two functions:

1. It saves or ''captures'' the data that would normally go to the printer. This captured data may be thrown away, or it may be saved in a capture-file for later use.

2. It prints data from a disk file *while* other operations are in progress. That is, you can be using the Model II system — everything except for the printer — *while* the file is being printed.

The two functions may be used one at a time or simultaneously. In the latter case, the spool-file is printed and real-time printer output is captured for later use.

Sample Uses

1. **(Capture-File)** You are going to run a program that outputs to the printer. Instead of waiting while the printing is done, you would like to capture it in a disk file, and print it all out later. We'll call the capture-file SPOOL1. Type:

```
SPOOL  ON
SPOOL  N,F = SPOOL1
```

Now all printer output will be saved in SPOOL1.

To stop capturing the printer output in SPOOL1, type:

```
SPOOL  OFF
```

Now SPOOL1 is a text file which may be LISTed or PRINTed normally.

2. **(Print-file)** You need to print a file created by the spooler, but you want to use the system at the same time. Suppose the file is named SPOOL 1, from the previous example. The spooler is off. Type:

```
SPOOL  ON
SPOOL  P,F = SPOOL1
```

   TRSDOS will begin printing the file as a ''background task'', meaning that printing is performed only when the system is not busy with some higher-priority operation like interpreting and executing your keyboard commands. Since we did not include the K or C = *copies* option, TRSDOS will delete SPOOL1 after it is printed, and will print only one copy.

   Spooler does not turn itself off after completing a print-file. To turn it off, type:

```
SPOOL  OFF
```

3. **(Simultaneous Capture-File and Print-File)** You want to save real-time printer output at the same time as the spooler is printing a file. In some applications, this will be common.

   For this purpose, you need one capture-file and one print-file. We'll use the names SPOOL1 and SPOOL2.

First turn the spooler on, and begin capturing printer output in SPOOL1:

```
SPOOL  ON
SPOOL  N,F = SPOOL1
```

Now use the Computer normally, until you are ready to begin printing out SPOOL1. When you are ready, type:

```
SPOOL  N,F = SPOOL2
```

This closes SPOOL1, and makes SPOOL2 the new capture-file. To begin printing SPOOL1, type:

```
SPOOL  P,F = SPOOL1
```

Now SPOOL1 will be printed, and any real-time printing will be saved in SPOOL2.

Suppose you want to halt the print-file operation. Then type:

```
SPOOL  H
```

This does not affect the capture-file operation. To resume printing, type:

```
SPOOL  R
```

Using the Spooler

1. Spooler is invoked by the console command: SPOOL ON. This will cause a high-memory module to be loaded and linked with the operating system. Whenever spooler is active (resident in high-RAM), the user must insure that this high-RAM space is not overlayed with any programs or subroutines.

2. Before getting into the details of spooler, a brief explanation of how it works is in order. Spooler itself can be thought of as a high level "supervisor" whose role is to monitor the use of the "physical" printer. As such, spooler can be said to "control" or "own" the printer. Spooler will accept your commands to print out disk files of text to the real printer and will, upon your command, save all data which was intended to be printed onto a disk file.

   As long as spooler is active (or until you issue the command line: SPOOL OFF), all data that would normally go to the printer will be intercepted by spooler. This is called the "capture" function of the spooler. "Capturing" can be done to a disk file of the name you specify in the command line:
   SPOOL N F = *filespec*
   "Capturing" can also be done such that all of the printed data bytes will get "thrown" away — this is called capturing to a dummy file. In this case, no file is opened and no data gets saved. There is a good use for this, as we will discuss later. Because of this capturing function, your program is now said to be outputting data to a "logical" printer, not the "physical" printer.

While spooler is active, and while it is printing out a previously captured file, it is using the extra processor cycles that would normally be wasted. Previously, the computer would just wait for certain operations to complete; now, these small time periods are used by the spooler to get the data from a disk file, then output them to the real printer. Not only are wait times used, but, just to insure that your printed report doesn't stop running altogether, some extra "time slices" are given to the spool printer. The spooler, as you can now see, is a true multi-tasked operation. The technique is more correctly known as background/foreground processing, where the spool printer is a background task, and your application program is a foreground task.

The capture function and the printing functions of spooler are separate. It should be noted that the capture function is directly controlled by the application program or utility that creates printed data. In BASIC, the "LPRINT" verb directs data to the printer. Every time a byte is output to the printer, spooler, when active, will intercept this byte and either output it to a disk file, as mentioned above, or to throw it away. The time it takes to write the data out to a disk is much faster than if it were going out to a real printer, due to the differences in speed between a disk drive and even the fastest printers that you might use. This feature alone can be used to speed up long jobs that prepare a printed report as an output. The trade-off for the increased speed is, of course, disk space. Printed reports usually take a lot of disk space to store. More on this later.

3. Some of the features that spooler print has that should be helpful are listed below:

   A. The operator has control over the printing of the report from the disk file. Such controls include pausing the report, stopping it altogether, resuming the report, restarting the report on a certain line, and starting it up from a certain line. The line positioning feature is especially useful when you only need a certain part of a printed report, or if you ran out of paper, or the paper jammed, while you were printing the report for the 1st time.
   B. Other operator controls include number of copies, and whether to keep or delete the spool disk file after printing is complete.
   C. Assuming none of the high-RAM where the spooler program resides has been used for user subroutines, etc., then the operator has the operational flexibility to run spooler printing as desired. Again, spooler was designed to be used without any application program modification to printer functions.
   D. Special code has been installed that can sense printer faults in parallel printers. This causes the printing function to just wait for the problem to get corrected before the printing will resume. This should be especially helpful when you have a paper jam or run out of paper. As soon as the printer goes back on-line and is ready, spooler print will resume. Also, you may stop the printer by taking it off-line (parallel printer only) so that you may look at the printed material, and spooler will just "wait" for the printer to go back on-line. If you have a serial printer that has the ability to send the DC3 character (X'13') and the DC1 (X'11') character, then you may use the DC3 to

pause the printer until the Model II receives the DC1 (resume) character. This is really a new feature in the printer software, not spooler, but applies to spooler also.

4. While the spooler print function is running, programs may be run concurrently with the printing. As a background task, spool printing will usually run as fast as it can using the left-over computer cycles. There are, however, some programs that cause there to be few extra computer cycles; in this case, the printing function will slow down. This is to be expected. Every effort has been made to insure that the foreground task, your application program, will not be unduly affected by the running of a spooler print file.

5. Other things which affect the speed of the spooler printing function are listed below:
   A. The number and frequency of disk input/output operations, i.e., OPENS, CLOSES, READS, and WRITES.
   B. The printer you have, especially the size of its RAM buffer, and whether it is a serial or parallel printer.
   C. How often the processing program waits for keyboard input.

   These all affect the spool print operation because, as a background task, the spooler must wait for your processing program to finish some operations before it can get control. Your program has a higher priority than the spooler and if there is any competition for the same software in the operating system then the spooler will take a "back seat" to your program. An example would be: spooler printer needs another disk record to continue its printing. Your program is currently needing a disk record itself to continue processing. Spooler, in this case, will wait for your disk operation to complete before it will request its next disk record.

6. Spooler has, in its capture function, the ability to save or to throw away the printed data. When it is first brought up, the capture function is set up to "throw away" the printed data until you specify (with the SPOOL NF = *filespec* command) what the capture-file's file name will be. As soon as you do that, all data bytes normally going to the printer will be captured into the data file you've specified. The printed bytes will continue to be captured into this file until you issue another SPOOL NF = command, which tells the capture function to close out the first file, then open up another file to start capturing into. As soon as the first file is closed, it may be printed. You may find it useful to delay printing of this file until a later time. The only requirement is that the capture file must be closed before it can be printed by the spooler print function.

   When a capture file is open, you will notice that it appears in the directory with a '?' after the file name. This is because the file is open and has not been closed. Any other file which was opened and not closed will be marked in the same way (see DIR command for details).

7. You may find it useful to setup the dummy capture mode, which is invoked by the command SPOOL N F=<CR>. In this mode, all printed bytes will not get saved on a disk file, but instead will be thrown away. The most common use for this feature is when you have already run, and printed, the desired report, but you must rerun the program so that it can re-update or re-build the disk files that it created. This will speed up the running of the program (in comparison to having to print the report again). Independent of this mode, you may still be printing out another report, captured earlier, with the spooler print function.

8. It should be noted that, while spooler printer is running, the line count and character counts that are kept by the operating system (see SVC PRCTRL) are pertinent to the printing that is taking place by the spooler. This will usually have no relationship to the data that your processing program might be outputting to the capture file. Because of this, if your application program uses this new printer control supervisor function, it will not give correct line counts, character counts, etc. to your program if spooler is running. This is because the print software only keeps track of one set of line counts, character counts, etc. and those values will be affected only by the spooler print function, not by the data which is getting intercepted by the capture function. Consider this before writing subroutines to use the new printer control supervisor function in your applications. The use of these new functions will preclude the use of spooler print while your program is running. Only when your program is outputting data directly to the physical printer (which it is not the case if spooler is running) will these line and character counts correctly reflect what the printer is doing at the time.

   Also, if your programs use the functions of setting line counts or character counts, your programs should not be run when spooler is running. The results will be unpredictable, and most assuredly, undesirable. The print control functions relate to the physical printer, not to the "logical" printer that exists when spooler is active. Your program, when spooler is active, outputs to the "logical" printer which has no line counts, character counts, etc.

9. The amount of disk space that will be required to store a printed report that has been captured is strictly a function of the size of the report itself— basically, the number of bytes of data. Certain techniques may be utilized to help speed up the capture and to insure that you can get the most printed data in a file. These are listed below:

   A. It is recommended that you always use an empty diskette (a data diskette with no other files on it is best) for the capture file, when you have a Disk Expansion Unit and have an extra drive to dedicate to the capture file.
   B. If your application or system doesn't allow the use of a separate, dedicated drive for the capture file, then use a diskette with the most space available on it that will be on-line when capturing is to take place.
   C. In either case above, using CREATE to set-aside the largest possible space for the capture file will speed up the capturing itself— additional disk space allocation will not be needed during the run.

D. Do not keep a capture file any longer than necessary so that the space on the disk may be used for another capture file. The default, when printing out a captured file, is for the captured file to be deleted upon completion of the printing.

E. Try to avoid the outputting of large number of spaces in a printed report. If at all possible, use the tab character (X'09') instead of spaces, to "position" to the next printed column or field. The capture file will only have to store the single tab character instead of the spaces. This will save on disk storage space.

F. You might be able to move some of your data files from one disk to another (in a multi-drive system) to free up the largest amount of disk space onto a single disk for the capture file. This could slow down your other disk accesses, depending on a lot of other factors, but this still might be faster just because of using the capture function instead of a real, "physical" printer.

Some applications, because of the amount of free space on the disks, or because of the size of the printed reports, will preclude the use of spooler. Sample runs should show you what disk space requirements you might have and what limitations of printed report size you might experience.

10. Spooler (either capture or printing) is compatible with:
    A. Host
    B. Serial Printer (SETCOM, FORMS S)
    C. Do File Processing
    D. Dual Routing
    E. LIB commands PRT option (DIR, FREE, ANALYZE, LIST)
    F. Communications (Some timing difficulties might be experienced at higher baud rates)
    G. Most utilities and LIB commands

11. Spooler should never be run when formatter, or backup is running. In addition, the disk that has the capture file on it should never be removed from the drive until that capture file is closed and/or spooler is de-activated. Keep in mind that when a printing function is currently in progress, the disk with the file being printed must not be removed from the drive it is in, until the file is closed (and deleted, if so desired). Depending on what else is going on, sometimes the print file will remain open for several minutes after the printing has been completed. Make sure by doing a directory, looking for the print file in the DIR before removing the diskette. The print file will not appear in the directory with a "?" next to the name, because it was only opened for reading, not for writing. Only files opened for writing will have this "?" appear in the DIR listing.

Notes:

1. The SPOOL ON command cannot be called from BASIC or any user program, because it always jumps to TRSDOS READY upon completion. (Other spooler commands *may* be called from BASIC i.e. SPOOL N F = *filespec* etc.).
2. When a capture file is first opened, the spooler writes a header record at the front of the file. This consists of a "length" byte followed by the TRSDOS time-date text. This information will not be printed by the SPOOL P command. You can create a spool print-file by following this format. If no header is to be used, the first byte of the file should be a binary zero. In this case, spool printing will begin with the second byte. Spool files must have fixed-length, 256-byte records.
3. While the spooler is on and you have not named a capture file, all printer output is ignored. To begin capturing the output, you must name a file (SPOOL N F = file).
4. While spool is ON and data is being captured and/or printed, the capture-file and/or print-file are open. DO NOT REMOVE THE CAPTURE-FILE OR PRINT-FILE DISKETTES UNTIL YOU HAVE TURNED THE SPOOLER OFF. Do not Backup or Format to these disks, either.
5. The capture-file traps all printer bytes — no translation is done. This is true regardless of what print control options may have been selected (e.g., auto line-feeds). On the other hand, the print-file is output using the current selection of print control options. For example, a X'OC' will be captured literally; but a X'OC' in a print-file may be interpreted before it is printed, depending on the currently selected print control options. See FORMS and SVC PRCTRL.
6. When the spooler completes a print-file, it will not kill the print-file immediately, but at the earliest opportunity. If you issue a DIR command immediately after completion of a spool print operation, you may notice that the file is still in the DIRectory. The file will be killed during execution of any user program or any "high-overlay" TRSDOS command such as LIST.
7. Once it has been closed, a spooler capture-file may be printed via the PRINT command or by the spooler's print facility. However, the PRINT command will not operate as a "background" operation allowing you to key in commands during printing, and the time/date text will be printed at the beginning of the file.

STATUS
Display System Status Information

```
STATUS
```

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

Sample Use

You want to locate a Z-80 program at the top of memory. Use STATUS to find this address:

```
STATUS
```

T
Advance Printer Paper to Top of Form

```
    T
```

This command works like FORMS with the T option. Use it whenever you adjust the printer paper position, or whenever you want to start a new page. If spooler is currently active and capturing, this will send Top-Of-Forms character X'OC' to the spooler capture file.

Sample Use

After PRINTing a file, you want to start a new page:

```
  T
```

TIME
Reset or Get the Time

TIME *hh.mm.ss*

hh is a two-digit hour specification.
mm is a two-digit minute specification.
ss is a two-digit second specification. If *.ss* is omitted, .00 is used.

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

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

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

The operator sets the time initially when TRSDOS is started up. After that, TRSDOS updates the time and date automatically, using its built-in clock and calendar.

When you request the time, TRSDOS displays it in this format:

```
THU JUL 19 1979 2ØØ -- 14.15.31
```

for Thursday, July 19, 1979, the 200th day of the year, 2:15:31 pm.

Note: If the time passes 23:59:59, TRSDOS does not start over at 00:00:00. Instead, it continues with 24:00:00. However, the next time you use the TIME or DATE command, the time will be converted to its correct 24-hour value, and the date will be updated. If the clock is allowed to run past 59.59.59, it will re-cycle to zero, and the date will not be updated to include the 60-hour period.

Examples

```
TIME
```

Displays the current date and time.

```
TIME 13.2Ø.ØØ
```

Resets the time to 1:20:00 pm.

```
TIME 18.24
```

Resets the time to 6:24:00

Note: Periods are used instead of the customary colons since periods are easier to type in — you don't have to press SHIFT.

VERIFY

There are two syntaxes for this command:
A. VERIFY {*switch*}
B. VERIFY {DETECT = *switch*}

switch is either ON or OFF.

In syntax form A, *switch* tells TRSDOS whether to verify all diskette writes (read after each write). TRSDOS starts with this switch ON. If omitted, the current status is given.

In syntax form B, *switch* controls the diskette ID sensing feature. If DETECT = ON, then TRSDOS will automatically check the diskette ID before any diskette access. TRSDOS starts with the DETECT switch ON. If DETECT = OFF, then TRSDOS will not automatically check the diskette ID before each diskette read. In this case, you must use the "I" command immediately after swapping diskettes.

This command controls the verify function. When it is on, TRSDOS will read after each write operation, to verify that the data is readable. If the data is not readable, after retries, TRSDOS will return an error message, so you'll know that the operation was not successful.

Note: TRSDOS always verifies directory writes. User writes (writing data into a file) are only verified when VERIFY is ON.

TRSDOS starts up with VERIFY ON. For most applications, you should leave it ON.

While DETECT = ON: If the operator swaps diskettes while a file is open, TRSDOS will abort any attempted I/O to that file with error 7. When the correct diskette is reinserted, TRSDOS will be able to perform the desired I/O operation.

When improper swapping is a possibility, programmers should write error-7 recovery procedures which prompt the operator to insert the correct diskette.

Examples

```
VERIFY ON
```

Turns on the verity function.

```
VERIFY OFF
```

Turns off the verify function.

```
VERIFY
```

Displays the status of the verify switch.

```
VERIFY  DETECT = OFF
```

Turns off the diskette swap detection feature for reads only.

Section 3

Utility Programs

Introduction to Utility Programs

NAME	PURPOSE
BACKUP	Duplicate a Diskette
FORMAT	Organize a Diskette
MEMTEST	Checks out random access memory.
PATCH	Change contents of a disk file
TERMINAL	Communications program
XFERSYS	Transfer operating system

Several other ''incidental'' utilities are included: BASCOM, COMSUB and DOCOM (all for serial communications); EXDATM and DATM (for date calculations); HERZ50 (for non-USA users); PRTBKSP; LPII.

The following utilities use memory from X'2800' to TOP, and exit to TRSDOS READY upon completion:

BACKUP
FORMAT
MEMTEST
X FERSYS

To ''chain'' utilities and user programs, include them as commands in a DO-file. The last command in the file may be one to load and execute your program.

BACKUP
Duplicate a Diskette

BACKUP *source* TO *destination* {*options*}

source and *destination* are drive specifications. If omitted, TRSDOS will prompt you to specify the source and destination drives.

{} if no options are given, BACKUP use defaults for all the options. The defaults are described below.

options may be any combination of the following:

PW = *source-password* tells TRSDOS the master password of the source diskette. TRSDOS will not duplicate a diskette unless you give the correct password. If this option is omitted, TRSDOS will assume the password is PASSWORD.

NEW = *dest-password* tells TRSDOS the password to assign to the destination diskette. The master password allows access to all user files via the PROT command, as well as full BACKUP privileges. If omitted, TRSDOS will use the password of the source diskette.

ID = diskette-name tells TRSDOS the diskette name to assign to the destination diskette. If omitted, TRSDOS will use the diskette name of the source diskette.

SYS tells TRSDOS to copy system files only. If omitted, all files will be copied (subject to PROMPT selections).

ABS tells TRSDOS not to prompt the operator for diskette information, but to use the information found on the source diskette. If this option is used, TRSDOS will overwrite data on the destination diskette *without* first warning the operator. If this option is omitted, the operator may be prompted several times.

PROMPT tells TRSDOS to prompt the user before each non-system file is copied, allowing a selective backup. If PROMPT is omitted, TRSDOS will copy all files without prompting.

NOAUTO tells TRSDOS not to copy the AUTO command input (if any). If omitted, TRSDOS will copy any AUTO command input. (See AUTO.)

This utility allows you to duplicate some or all of the files on a system or data diskette. The destination diskette must already be formatted.

Any two drives may be used for source and destination diskettes, or a single drive may be used. In this latter case, TRSDOS will prompt you to swap source and destination diskettes several times during the backup process.

Prompting Messages

Here are the prompting messages that may be displayed (depending on the options given in the BACKUP command):

```
Source drive number? (0-3) ..
```

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

```
Destination drive number? (0-3) ..
```

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

```
Source diskette ready? (Y/Q) ..
```

When the source diskette is in the proper drive, type Y (ENTER). To cancel the backup operation, type Q (ENTER).

```
DESTINATION Disk Ready? (Y/Q) ..
```

When the destination diskette is in the proper drive, type Y (ENTER). To cancel the backup operation, type Q (ENTER).

```
Diskette contains DATA— use it? (Y/Q) ..
```

This warning will be given only if the ABS option is omitted and the destination diskette contains file data. If you wish to overwrite existing information, type Y (ENTER). To cancel BACKUP, type Q (ENTER).

```
Change Diskette Information?
```

This prompt will only be given if you do not use the ABS option. Type Y (ENTER) if you wish to change the password or diskette name on the destination diskette. Type N (ENTER) if you wish to copy this information from the source diskette.

If you selected the Change option, TRSDOS will prompt you as follows:

```
Enter New Password. . . . . . . . .
```

Type in the password to be assigned to the destination diskette and press (ENTER).

```
Enter New Disk Name. . . . . . . . .
```

Type in the diskette name to be assigned to the destination diskette and press (ENTER).

If you selected the PROMPT option, the following prompt will be displayed before each user file is copied:

file ? (Y/N/S) ..

file indicates the current file. To copy it, type Y (ENTER). To skip it, type N (ENTER). To copy it and stop the prompts, type S (ENTER).

Primary and Alternate Directories

The primary directories of the source and destination diskettes must be on the same track.

The alternate directories may be on different tracks; in fact, the alternate directory need not be present on either diskette. If the destination diskette does have an alternate directory, it will be used as such, even if the source diskette has no alternate directory. If the destination has no alternate directory, none will be created, even if the source diskette has an alternate directory.

During the BACKUP process, the source diskette's primary directory will be used unless it is found to be flawed. In this case, the alternate, if available, will be used.

If the destination diskette contains an alternate directory on a track which is used for data on the source diskette, the backup will be cancelled.

Other Messages and Error Conditions

This is not a comprehensive listing; self-explanatory messages are not included.

```
Source diskette ERROR
Only GOOD files can be copied
Continue anyway? (Y/Q)
```

TRSDOS cannot copy the current file, due to a flaw in the diskette or some other error. Type Y (ENTER) to continue with the next file; Q (ENTER) to cancel the backup.

If any of the following errors occurs, you should:

1. Recreate the steps which caused the problem and duplicate the error.
2. Call Radio Shack Customer Service for a system error report.

```
DUAL ALLOCATION
```

Two files contain overlapping space allocations.

```
ALLOCATED ON FLAWED TRACK
```

A file has space allocated on a flawed track.

```
ALLOCATION ERROR
```

There is a conflict between space allocations and the free space map.

Using BACKUP To Recover Data (Alternate Directory Required).

Of all tracks on a diskette, the directory is used the most, therefore it tends to wear out first. When a primary becomes unreadable, TRSDOS cannot perform normal disk I/O using that diskette. If an alternate directory is available, TRSDOS will use it automatically. This will slow down disk I/O.

If you begin having diskette errors with a particular diskette, use BACKUP with a destination diskette which is known to be good (no flaws). During the backup process, TRSDOS will use the alternate directory, if one is available and is needed, to recreate a good directory on the destination diskette. (If the destination has both primary and alternate directories, both will be contain the same good information.)

FORMAT
Erase and Initialize a Diskette

FORMAT *drive* {*options*}

drive Specifies which drive is to be used for the format operation. *drive* is the drive number. 0, 1, 2, or 3. If omitted, TRSDOS will prompt you for the drive number.

{} If no options are given, TRSDOS will use defaults, as given below.

ABS Tells TRSDOS *not* to warn the operator if the destination diskette contains data. If ABS is omitted, TRSDOS will always warn the operator before overwriting a version 2.0 diskette which contains file data.

ID = *diskette-name* Tells TRSDOS the name to assign to the diskette. If omitted, TRSDOS will be used.

PW = *password* Tells TRSDOS the master password to assign to the diskette. If omitted, PASSWORD will be used. The master password allows access to all user files (via the PROT command), and also allows full BACKUP privileges.

DIR = *tracknumber* Tells TRSDOS where to place the primary directory. If this option is omitted, track 44 will be used. The primary directory may be placed on any track from 1-76.

Warning: SYSTEM/SYS *requires* 7 tracks following directory if diskette is not a data diskette.

ALT = *tracknumber* Tells TRSDOS where to place the alternate directory. If *tracknumber* = 00, no alternate directory will be created. If the ALT option is omitted, the primary directory track number plus 8 will be used. If the specified or computed *tracknumber*>76, track 1 will be used.

verification level	is one of the following:
FULL	Reads each sector and compares the value against what was written during initialization.
NONE	No verification is done.

If no verification level is specified, FULL is used.

This program initializes a diskette by defining the tracks and sectors, and writing system information onto the diskette. (For more information on the TRSDOS diskette format, see **Technical Information**.)

The diskette may be blank (new or bulk-erased) or it may already be formatted. When you re-format a diskette, all previous information is lost.

Verification

FORMAT also does a specified amount of verification, or checking for areas on the diskette which cannot store data due to flaws in the recording surface. If it finds a flawed area, TRSDOS ''locks out'' the affected track and will never try to use that track.

Whenever you format a new diskette, use the FULL verify option. This takes longer, but gives greater protection against lost data due to a flawed track.

When you are re-formatting a diskette which has been functioning without diskette errors, the NONE option may be sufficient. NONE is faster than FULL, since no verification is done. But remember, for greater reliability, FULL verification is recommended.

Main and Alternate Directories

TRSDOS maintains a primary directory on each diskette. Unless you use the ALT = 00 option, TRSDOS will also maintain an alternate directory.

For general applications, you should always have an alternate directory. If a diskette's primary directory should become unreadable, the alternate directory will automatically be used instead — so that you don't lose access to the data on that diskette. See BACKUP, Using BACKUP to Recover Lost Data.

Selecting the Directory Tracks

TRSDOS allows you to specify where the directories will be placed. The default locations are:

Primary = 44   Alternate = 52

For general purposes, this will be the most efficient arrangement.

Notes on Directory Location

1. If you are going to put TRSDOS on the diskette, be sure there are seven contiguous good tracks immediately after the primary directory.

2. The greater the spread between the primary and alternate directories, the longer it will take TRSDOS to update them. A spread of 8 tracks is recommended. This isolates the directories without separating them unduly.

3. If the diskette is to contain the TRSDOS system (full or minimum), there must be seven good contiguous tracks following the primary directory. For this reason, the primary directory track number must be less than or equal to 69.

4. FORMAT will not place a directory (primary or alternate) on a flawed track. If you specify a track (DIR = or ALT =) which is flawed, TRSDOS will use the default track (DIR = 44, ALT = 52). If the default track is bad, TRSDOS will place the directory on the first good track after this one. TRSDOS will abort the operation if it doesn't encounter a good track within five tracks of the default track.

5. In some applications you may want to use different directory positions. For example, if a data (non-system) diskette is to contain only one user file, it may be efficient to place the directory on track 1 so that tracks 2-76 are available for the file.

   For another example, suppose tracks 44-49 are flawed, preventing TRSDOS from placing the directory there. In this case, you may locate the primary directory on a good track, so the diskette will still be usable.

Examples

```
FORMAT
```

TRSDOS will prompt you for the drive to be used, but will use defaults for all the options.

```
FORMAT  1  {ID = ACCOUNTS, PW = MOUSE}
```

TRSDOS will use drive 1; the diskette will be named ACCOUNTS with the password MOUSE.

```
FORMAT  {ABS, NONE}
```

TRSDOS will prompt you for the drive to be used, will use no verification, and all other options will be defaulted.

```
FORMAT  1  {DIR = 01, ALT = 02}
```

Drive 1, primary directory on track 1, alternate on track 2. This leaves all unused space in one extent. The resultant diskette can not be used to contain TRSDOS, since there are not seven free tracks immediately following the primary directory.

```
FORMAT  2  {DIR = 40}
```

Drive 2, primary directory on track 40, alternate on track 40 + 8 = 48.

When to Format

To prepare a new diskette.

Before you can use a new diskette, you must format it. After formatting, record the disk name, date of creation and password in a safe place. This will help you estimate how long a diskette has been in use, and prevent your forgetting the master password. (For this application, always use the FULL verify option.

To erase all data from a diskette.

To "start over" with a diskette, you can format it.

To lock out flawed areas.

After long use, flaws may develop on a diskette. Reformat the diskette to lock out these tracks while leaving the good tracks available for data storage. Use the FULL verify option for this application.

MEMTEST

```
MEMTEST
```

This utility tests the random access memory in the Model II. You may select either a full memory test (X'0000' to end) or a user memory test (X'3000' to top of user memory).

To execute the test, type under TRSDOS READY:

```
MEMTEST (ENTER)
```

Note: After running the full memory test, you must reset the system. After running the user memory test, control returns to TRSDOS READY. All user memory is cleared. None of the high-memory routines may be active while MEMTEST is running. (The STATUS command will tell you if any are active.)

PATCH
Change the contents of a disk file

PATCH *programfile* A=*address,* F=*findstring,* C=*changestring*

This is the form to use when you are patching a program stored with the "P" (program) attribute (see DIR). Files created with DUMP will fall into this category.

programfile specifies the file you want to change. If it is a system file, no password is necessary. If it is a protected user file, the password must be included.

A=*aaaa*

aaaa is the starting address of the data to be changed. This is where the data resides in memory when the program is loaded. *aaaa* is a four-digit hexadecimal value without the X' ' notation.

F=*findstring* specifies the string that is currently in the patch area.

C=*changestring* specifies what data is to replace *findstring. changestring* must contain the same number of bytes as does *findstring.*

Both *findstring* and *changstring* can be in hexadecimal or ASCII form. In hexadecimal form, each byte to be changed is represented as a two-digit hexadecimal value. In ASCII form, each byte to be changed is represented by the ASCII character corresponding to that byte value. ASCII strings must be enclosed by single or double quotes.

PATCH *datafile* R=*record,* B=*startingbyte,* F=*findstring,* C=*changestring*

This is the form to use when you are patching a data file, i.e., a file stored with the "D" attribute (see DIR). BASIC programs and data files fall into this category.

datafile specifies the file you want to change. If it is a system file, no password is necessary. If it is a protected user file, the password must be included.

R=*record*

record tells which record contains the data to be changed, and is a decimal number from 1 to 65536.

B=*startingbyte*

startingbyte specifies the position of the first byte to be changed. It is a decimal number from 1 to 256.

F=*findstring* and C=*changestring* are described above

This utility lets you make minor corrections in any disk file, provided that:
1. You know the existing contents and location of the data you want to change.
2. You want to replace one string of code or data with another string of the same length.

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

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

Note: If you press BREAK during a patch operation, before any changes have been made in the file, PATCH will close the file and return you to TRSDOS. The file will be unchanged. Once PATCH has begun changing the file, pressing BREAK will have no effect.

Using PATCH on a TRSDOS System File

When Radio Shack releases a modification to TRSDOS, you will receive a printout of the exact PATCH commands that are required to perform the change.

To implement such a change, you would follow these steps:

1. Make a backup copy of the diskette to be patched.
2. Insert the TRSDOS disk to be changed into one of the drives. The diskette must be write-enabled.
3. In the TRSDOS READY mode, type in the specified PATCH command.
4. After completion of the patch, test the diskette in drive zero to see that it is operational as a TRSDOS system diskette. You will have to reset the Computer.

Using PATCH on a Program File

Remember that in this context, "program files" refers strictly to those files stored with the "P" attribute. Use the DIR command to find out the attributes of a file. BASIC programs have the "D", not the "P", attribute. (See instructions for changing data files.) Program files are created with DUMP.

Suppose you want to change seven bytes in a machine-language program file. First determine where the seven-byte sequence resides in RAM when the program is loaded. Then make sure that your replacement string is the same length as that of the original string. For example, you might write down the information as follows:
File to be changed: VDREAD
Start address: X'5280'
Sequence of code to be changed: X'CD2C25E5'
Replacement code: X'000000C9'

Then you could use the following command:

```
    PATCH   VDREAD A=5280, F=CD2C25E5, C=000000C9
```

Using PATCH on a data file (including BASIC programs)

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

For example, suppose in a file called NAMEFILE you need to change a 12-byte sequence. When you LIST the file, you find that the sequence is located in record 128, and that the sequence starts at byte 14. Write down the information like this:
File to be changed: NAMEFILE
Record number: 128
Starting byte: 14
Sequence of text to be changed: "JOHN'S DINER"
Replacement text: "JACK'S PLACE"

Then use the following command:

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

Notice that either string can include a single-quote, as long as the string is surrounded by double-quotes. If you wanted to include a double-quote inside either string, you would have to enclose that string in single-quotes.

Note: The string you are changing must be wholly contained inside the specified record. If it spans two records, you will have to perform the patch operation twice, once for each record.

Error Conditions

If a TRSDOS error occurs during the patch operation, you will receive the appropriate ** ERROR *nn* ** message, and the patch will be terminated without changing the file.

PATCH can also produce the following messages:

PATCH STRING TOO LONG–ABORT	This occurs when you are patching a data file and the patch string spans two records. You will need to perform the patch in two steps, one for each record that contains a part of the string to be changed.
FILE CONTAINS VARIABLE-LENGTH RECORDS–ABORT	You can only patch fixed length record files.
STRING NOT FOUND	The find-string string was not found at the patch location you specified. Before patching a file, you must know the exact patch location and the existing contents of that 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 is outside the RAM area where the program resides when it is loaded. Check the A=*aaaa* parameter. Also be sure that the *findstring* and *changestring* aren't longer than you intended for them to be.

TERMINAL

```
TERMINAL
```

This is a versatile program designed to allow communications between the Model II and another computer running a host program. TERMINAL is designed primarily for transmission and reception of ASCII text rather than machine-language object code.

Input/output is through serial channel A. In most applications, hookup will be through telephone lines via a modem.

TERMINAL has three modes of operation:

- **Menu** — Allows you to select or change options, even execute TRSDOS library commands
- **Interactive terminal**—Transmits your keyboard input and displays incoming data
- **Transmit from RAM**—For high-speed transfer of prepared data. Incoming data is displayed on the screen.

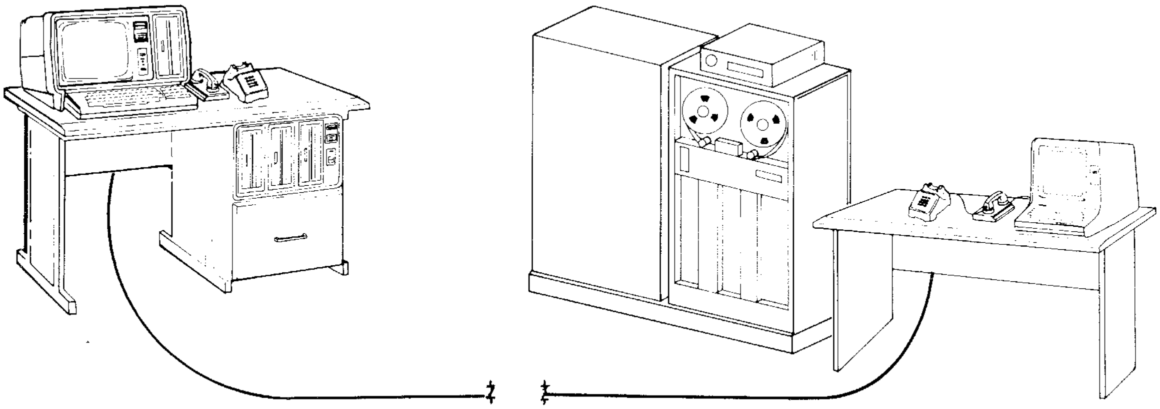


Figure 1. A typical terminal/host configuration.

Setting Up

For communications through ordinary telephone lines, you will need a modem such as the Radio Shack Telephone Interface II, Catalog Number 26-1171, and the Model II RS-232 Cable, 26-4403.

1. Set up the modem according to its instructions, and connect it to channel A on the back panel of the Model II display console. Unless channel B is connected to another device, you must install the serial terminator on that channel.
2. Find out what RS-232-C parameters are required by the host program you are going to use:
   Baud rate
   Word length
   Parity
   Number of stop bits
   You will need to initialize channel A accordingly (see "Running Terminal").
3. Set the modem to originate or answer mode — whichever is *different* from the mode used by the host program you are going to communicate with. Also set it to full or half duplex, again depending on the requirements of the host program.
4. Turn on the modem and the Model II computer system.

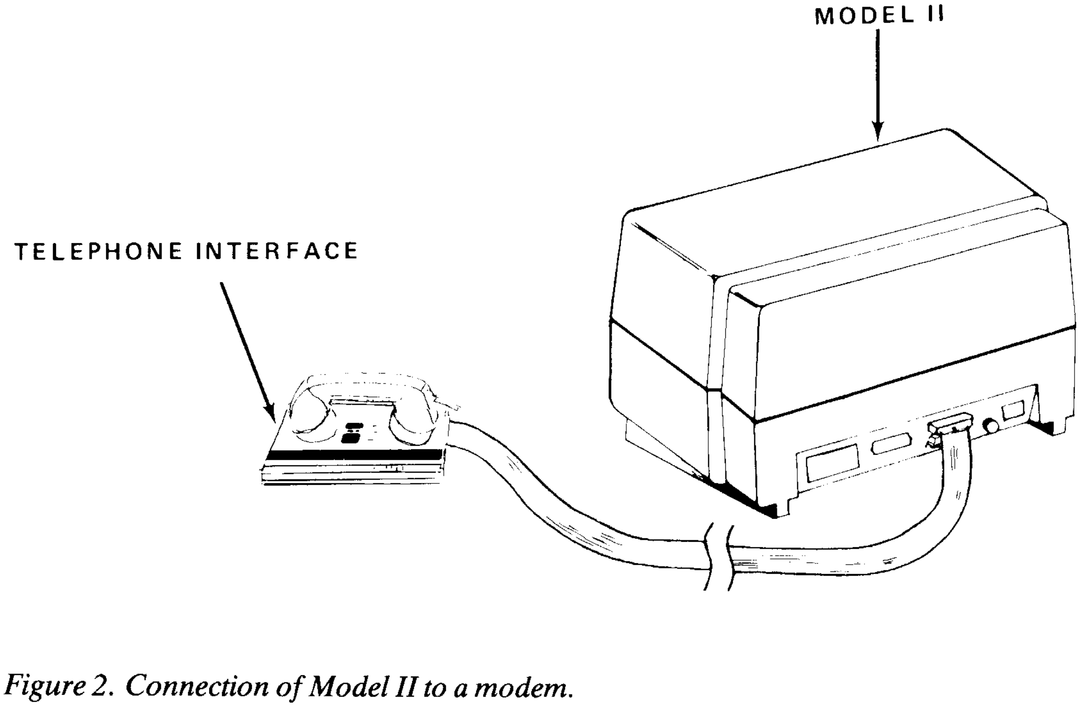


Figure 2. Connection of Model II to a modem.

Running Terminal

Since TERMINAL allows you to enter any TRSDOS library command, it is not necessary to initialize channel A or the printer before starting.

From the TRSDOS READY mode, you can start TERMINAL by typing:

```
TERMINAL
```

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

```
-- ENTER MENU SELECTION ..
```

Note: When we show computer prompts and user input in the same example, we highlight the user input with a gray background.

Now is a good time to initialize channel A according to the requirements of the host program you are going to communicate with. Type:

```
-- ENTER MENU SELECTION S
```

The program will prompt you to type in a TRSDOS command. Type in the SETCOM command just as you would in the TRSDOS READY mode. For example,

```
ENTER TRSDOS COMMAND (1-79)
SETCOM A=(300,7,N,2) ENTER
```

would enable channel A with 300 baud, seven-bit words, no parity and two stop bits. After executing the command, control will return to TERMINAL's menu mode.

If you plan to use the printer option of TERMINAL (described later on), you should also initialize the printer now, with the FORMS command. Type:

```
-- ENTER MENU SELECTION S ENTER
```

and enter the appropriate FORMS command.

To select another menu command, type in the letter specified in the menu table. For example, type:

```
-- ENTER MENU SELECTION M ENTER
```

to redisplay the entire menu.

Modes of Operation

Menu Mode

This is an off-line mode, i.e., you cannot transmit characters to the host program, and if characters are sent to you, they will be lost. This is the only mode in which you can select menu options described later on. From it, you can also enter the transmit from RAM or interactive terminal mode.

Interactive Terminal Mode

You can enter this mode from the menu with the T command; you also enter this mode automatically after completion of an auto sign-on or a transmission from the RAM buffer.

In the interactive terminal mode, characters you type are sent to the host program, and incoming characters are displayed as they are received. If the host program echoes your transmissions, they too will appear on the display; if not, you can select the echo option and TERMINAL will display your keyboard input.

Incoming characters can be saved in the RAM buffer (R option) and can be output to the printer (P option).

If transmission errors occur, TERMINAL will display a descriptive error message and wait for the error condition to be corrected. When it is, normal I/O will resume in the interactive terminal mode.

To return to the menu mode, press (BREAK)

Transmit from RAM (and Auto Sign-On)

You enter this mode via the X command. The contents of the RAM buffer are sent to the host program, and control passes to the interactive mode. Auto sign-on (O command) works just like transmit from RAM; the following comments apply to both operations.

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

During the transmission, incoming text will be displayed on the screen. If the host program echoes your transmissions, you will be able to verify that the data was sent accurately.

During the transmissions, you can adjust the delay between characters by repeatedly pressing the ↑ (faster) and ↓ (slower) keys. If echoed data appears garbled, slow down the transmissions. If not, you might want to speed it up.

If a break character or sequence is received in this mode, TERMINAL will pause until the next character is received. If a X'13' is received, TERMINAL will pause until a X'11'' is received. (By convention, X'13' is called the DC3 signal and means pause; X'11' is called the DC1 signal and means resume).

If transmission errors occur, TERMINAL will display a descriptive error message and wait for the error condition to be corrected. When it is, normal I/O will resume.

To exit from this mode at any time, press BREAK . You will be returned to the menu.

Details of the Menu

M Display Menu

After you have entered several menu commands, the menu begins scrolling off the display. Use the M command to clear the display and redisplay the menu.

S Perform System Command

Whenever you need to perform a TRSDOS library command, use this command as shown previously. After execution of a TRSDOS library command, control will return to TERMINAL's menu.

Some TRSDOS commands and programs always return to TRSDOS READY. Refer to pages 2/4 and 3/3 for lists. If you execute any of these via the S command, control will *not* be returned to TERMINAL, but to TRSDOS READY.

B Set/Change Break Character or Sequence

This command lets you select which incoming code will be interpreted as a "break". It also lets you define a key to send that same break character or to send a break sequence.

For the break character, any code from 0 to 255 may be specified. For the break sequence, any duration from 1 to 451 milliseconds may be specified. (The correct time period duration for a break sequence is determined by the host program, and depends on the baud rate.) For the user-defined break key, any key except (BREAK) or (CTRL)(C) may be used.

When TERMINAL receives the specified break character (or whenever it receives a break sequence), it will warn you so you may take appropriate action. If it is in the transmit from RAM mode, it will pause the transmission until the next character is received.

The following example shows how you would set up X'0A' as the break character, and CTRL D as the break key.

```
—Enter Menu Selection B ENTER
Break Key is Now 1B Hex
Change? (Y/N) Y ENTER
Enter New Key (1) CTRL D        Break Key is Now 04 Hex
Type of Break is Now CHR
Change? (Y/N) N ENTER
Break Char is Now 03 Hex
Enter new CHAR Value in Hex (2) 0A ENTER
Break Char is Now 0A Hex
```

The following example shows how you would establish the break sequence and define ESC as a break-sequence key.

```
—Enter Menu Selection B ENTER
Break Key is Now 1B Hex
Change? (Y/N) N ENTER
Type of Break is Now CHR
Change? (Y/N) Y ENTER
Type of Break is Now SEQ
Break SEQ is 250 mil secs
Change? (Y/N) N ENTER
```

W Set/Change Prompt Wait Character

This command affects TERMINAL operation in the transmit from RAM mode and during auto sign-on. It does not affect operation in the interactive terminal mode.

The prompt wait feature allows you to use the high-speed transmit from RAM mode — even when the host program can only accept one line at a time. Typically, the host program sends you a prompt such as a question mark or colon when it is ready for the next line. In the interactive keyboard mode, you would simply wait until this prompt is displayed; the prompt wait feature makes TERMINAL do the same thing while in the transmit from RAM mode.

When the feature is on, TERMINAL will send one line at a time, and wait for a prompt character from the host program before sending each line. (A line is defined as a string of characters terminated by a carriage return X'0D'.)

You can define the prompt wait character as any keyboard character from X'20' to X'7F'.

Leave the prompt wait feature off when the host program is simply storing characters as received, and is not sending a ready-for-next-line prompt. TERMINAL will transmit text from RAM in a continuous stream.

Suppose the host program sends a question-mark (code X'3F') when it is ready for a line of text. Then type:

```
---ENTER MENU SELECTION W     ENTER
PROMPT OPTION NOW OFF
PRESS ENTER TO LEAVE AS-IS OR ENTER NEW CHARACTER ?
PROMPT OPTION ON WITH '?' AS THE CHARACTER
---ENTER MENU SELECTION ..
```

From now on during auto sign-on or transmit from RAM, TERMINAL will wait for the question-mark prompt before it sends a line.

Note: When you start the transmit from RAM (X option) or auto sign-on (O option), the first line will be sent immediately, without waiting for a prompt. Each subsequent line will be sent after a prompt has been received.

To turn the prompt wait feature off, press HOLD when the program asks for a new character.

F Set/Change F1 & F2 Keys

This command lets you program F1 and F2 to output any code from zero to 255. This is useful when you will be using a particular code frequently.

For example, suppose the host program accepts a X'13' as a pause control, and X'11' as a resume. After receiving a pause character, it waits for a resume character before sending any more text.

Although you can send these codes from the keyboard (CTRL Q is a X'11' and CTRL S is a X'13'), it would be more convenient to program F1 and F2 to send these codes. 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
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
F2 KEY WILL SEND A 13 HEX CODE
 --- ENTER MENU SELECTION ..
```

Now when you type F1 in the interactive terminal mode, TERMINAL will transmit the resume control X'11'; for F2 , the pause control X'13'.

L Toggle Line Feed Option

This command tells TERMINAL how to handle an incoming line feed X'0A'. When the option is on, all line feeds are ignored. When it is off, they are not ignored.

The option is useful if the host program always sends a line feed after a carriage return. Since the TRSDOS display and printer drivers automatically perform a line feed after a carriage return is sent, the incoming line feed is redundant. Therefore the line feed option should normally be on.

To toggle (change the state of) the line feed option, type:

```
    --- ENTER MENU SELECTION L ENTER
```

The new state of the option (on or off) will be displayed, and the menu prompt will return.

P Toggle Printer Option

This command turns the printer option on and off. When the option is on, incoming text will be copied to the printer as it is received and displayed. Whenever you use D command while this option is on, the RAM buffer text will be copied to the printer.

Be sure you have initialized the printer with the FORMS command before attempting to use it.

To toggle the printer option, type:

```
    --- ENTER MENU SELECTION P ENTER
```

The new state of the option (on or off) will be displayed, and the menu prompt will return.

TERMINAL uses a circular buffer for efficient output to the printer. However, if characters come in too fast, they will not be printed. They *will* be displayed, though, and will be saved in RAM if the buffer is open. (Check your printer's specifications for maximum character input rate. At 300 baud, 7-bit characters may come in as fast as 30 per second.)

To minimize hookup time, don't use the printer option while on-line with the host program. Save the incoming text in RAM instead. After completion of the hookup, turn the printer option on and use the D command to get a hard copy of the data.

E Toggle Self Echo Option

Some host programs echo the text you send. That is, as the host receives each character, it sends it right back to you. In this case, what you send will be displayed on the screen. When communicating with this type of host program, set your modem to full duplex.

If the host program does not echo your text, then what you send (keyboard input or text from RAM) will not be displayed. In this case, you should use the self-echo option, which displays everything you type or transmit from RAM. With such host programs, set your modem to half duplex.

To toggle the echo option, type:

```
    --- ENTER MENU SELECTION E (ENTER)
```

The new state of the option (on or off) will be displayed, and the menu prompt will return.

R Toggle RAM Buffer Option

This command pertains to the interactive terminal mode only. It lets you save in RAM some or all the data that is received, by "opening" and "closing" the RAM buffer. That way, you can examine the data later with the D command, or save the data in a disk file with the C command.

Whenever you open the RAM buffer, you have a choice of resetting it or retaining its current contents. In the latter case, new incoming text will be loaded after the existing text in the RAM buffer.

To toggle the RAM buffer option, type:

```
    --- ENTER MENU SELECTION R (ENTER)
```

The new state of the option (on or off) will be displayed. If you have just opened the buffer, you will receive the following prompt:

```
    RAM BUFFER NOW OPEN
    RESET RAM BUFFER? (Y/N) ..
```

If you type Y (ENTER), the buffer will be reset and previous contents will be lost. For further information, see "Using the RAM Buffer."

A Build Auto Sign-On Message

This command lets you prepare an automatic sign-on to be sent to the host program with the O option. Typically, the message will contain responses to the standard sign-on questions provided when you first call up a host program.

The message can be up to 60 characters, consisting of any characters that can be entered from the keyboard. The message can include control characters. To embed a carriage return (X'0D') in the message, press (↓). It will be echoed as ±, but a carriage return will be stored.

If you enter any other control character in the message, it will also be displayed as a ±, but the true control code will be sent. (When you display a message, control codes will not be shown at all.)

For example, suppose the host terminal requires responses to the following prompts during sign-on:

```
USER ID?
USER PASSWORD?
PROGRAM NAME?
```

Instead of typing in information each time you call up the host program, you can store the responses in an auto sign-on buffer. In this case, the buffer might contain the following information:

```
FTW-779   <X'ØD'>
LUMMOX    <X'ØD'>
MENU      <X'ØD'>
```

To set up this auto sign-on message, type:

```
-- ENTER MENU SELECTION A   ENTER
THE CURRENT AUTO-SIGN ON IS

CHANGE? (Y/N) Y   ENTER
ENTER AUTO SIGN-ON MESSAGE (1-6Ø)
FTW (↓) LUMMOX (↓) MENU  ENTER
THE CURRENT AUTO SIGN-ON IS
FTW
LUMMOX
MENU
--ENTER MENU SELECTION ..
```

The blank line above indicates that the original auto sign-on was blank or contained non-display characters.

G Get Disk File into RAM Buffer

This command lets you load text stored in a disk file into the RAM buffer. Then you can send it to the host program via the transmit from RAM command. The previous contents of the RAM buffer are lost.

The disk file can contain fixed- or variable-length records, and fixed length records can have any length. However, only ASCII files should be loaded and sent. You can send BASIC programs as long as you saved them with the A (ASCII) option.

For example, suppose you have created a document stored in the file DOCUMENT/TXT. You want to send it to the host program. To get the file into the RAM buffer, type:

```
--- ENTER MENU SELECTION G  ENTER
ENTER FILESPEC (1-34)
DOCUMENT/TXT   ENTER
```

TERMINAL will load the file and return to the menu. The RAM buffer will be closed.

If the host program is ready to accept data, you can now send it with the X command. After transmission is complete, TERMINAL will go to the interactive terminal mode.

C Copy RAM Buffer to Disk

This command creates a disk file copy of the text in the RAM buffer. The new file will have a record length of one. Use this command to save data that has been received into the RAM buffer in the interactive terminal mode. To minimize hookup time, you will probably want to do this after ending the connection to the host program. Or if the RAM buffer is full, save it in a disk file, then reset it and re-open it to accept more data.

For example, suppose you have just received a report in the interactive terminal mode, and you want to save it in a disk file named REPORT. Type:

```
--- ENTER MENU OPTION C   ENTER
ENTER FILESPEC (1-34)
REPORT   ENTER
```

The new file will be created (if REPORT already exists, it will be overwritten with the new data), and the RAM buffer contents and status will be unchanged.

To stop the copy process, press BREAK . The disk file will be closed and you will be returned to the menu.

D Display RAM Buffer

This command displays the contents of the RAM buffer. To pause the display, press HOLD . To continue, press F2 HOLD . If the printer option is on when you issue this command, the text will also be output to the printer. To enter the command, type:

```
--- ENTER MENU SELECTION D ENTER
```

To stop the display function, press BREAK . You will be returned to the menu.

X Transmit RAM Buffer and Enter Term Mode

This option puts you in the transmit from RAM mode, in which the current contents of the RAM buffer are sent 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 . You will be returned to the menu.

O Enter Terminal Mode with Auto Sign-On

This command starts transmission of the current auto sign-on message. After the message is sent, TERMINAL enters the interactive terminal mode. For details, see "Transmitting from RAM."

To stop transmitting the auto sign-on, press BREAK . You will be returned to the menu.

Note: Most host programs cannot receive anything until the host program has sent the first prompting message. Because of this, you should:

1. Go to the interactive terminal mode (T option) 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 option).

T Enter Terminal Mode

This command puts you directly into the interactive terminal mode. While in this mode, press BREAK to return to the menu.

For details, see "Interactive Terminal."

V Toggle Video Filter

Some data characters cause undesirable results when output to the display. For example, if an ESC (X'1B') is output, it clears the screen and homes the cursor.

The video filter option lets you prevent this from happening by "filtering" these characters from the display. If the RAM buffer is open, they will be saved in RAM, regardless of the state of this option.

Here is a list of filtered codes when the option is on (all codes are given in hexadecimal):

01, 02, 03, 04, 05, 06, 07, 0B, 0C, 0E, 0F,
10, 11, 12, 13, 14, 15, 16, 1E, 1F

If any of these characters is received while the video filter is on, a "±" will be displayed in its place.

Q Quit

This command returns control to TRSDOS. Any data in the RAM buffer will be lost (i.e., you cannot restart TERMINAL and recover it).

Using the RAM Buffer

The RAM buffer is used to store incoming text (R option) and prepared text from a disk file (G option) so that it can be sent rapidly. The RAM buffer helps reduce costly hookup time, by letting you perform time-consuming operations — preparing data or printing it out — while you are off-line.

Size

For 32K Model II systems, the buffer can contain 11,493 bytes of text; for 64K systems, 44,261.

For 32K systems using 300 baud (i.e., 30 characters per second), it will take approximately 7 minutes to fill the buffer in the interactive terminal mode; for 64K systems, 25 minutes.

If the buffer becomes filled during a load from disk (G command) or while receiving data in the interactive terminal mode, a warning message will be displayed and the buffer will be closed. If you are loading a disk file, you will be returned to the menu and the buffer will contain the data that was loaded. If you are in the interactive terminal mode, normal I/O will continue, except that it will no longer be saved in the buffer.

Saving the RAM Buffer

When the buffer is filled in the interactive mode (or when you suspect it will be soon), do the following:

1. Transmit a pause or break control character to the host program.
2. Return to the menu by pressing BREAK .
3. Copy the RAM buffer to a disk file (C command).
4. Reset the RAM buffer (R command).
5. Return to the interactive terminal mode (T command).

Opening and Closing the RAM Buffer

Often during interactive I/O, you want to save only portions of the text. The R command lets you do this. Each time you are about ready to receive some important data, do the following:

1. Transmit a pause or break control character to the host program.
2. Return to the menu by pressing BREAK .
3. Toggle the RAM buffer status. If it is not off, toggle it again. If it is on, you have the option of resetting it or leaving it as is. To add new data onto the end of old, *do not* reset it. To delete old data, *do* reset it. For details and examples, see R command later on.
4. Return to the interactive terminal mode (T command).
5. Direct the host program to resume transmission. The data will now be saved in the RAM buffer as it is received.

Saving the Options You Have Selected

You can create a customized version of TERMINAL — one which starts up with the options you have selected. For example, the break character and key, F1 and F2 characters and auto sign-on message, could all be saved so you won't have to set them each time you start the program.

Options which may be saved in a customized program:

- 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 option
- Printer option
- Self-echo option
- Video filter option
- Auto sign-on option
- Receive into RAM option (not recommended)
- Speed of transmit from RAM and auto sign-on (as set by the ↑ and ↓ keys). Once you find out the maximum rate of transmission the host program can handle, you'll probably want that to become the default rate.

After selecting the options for your customized version, you use the DUMP command to create a new program file. Terminal resides from X'3000' to X'3FFF', and its entry point is X'3000'.

You must give this customized program a name other than TERMINAL — and leave TERMINAL in its original configuration. Suppose you want to call your customized version MINE. Then type:

```
    -- ENTER MENU SELECTION 5  ENTER
    ENTER TRSDOS COMMAND (1-79)
    DUMP MINE START=3ØØØ, END=3FFF   ENTER
```

Now when you type:

```
     TRSDOS READY
     MINE ENTER
```

your customized version of TERMINAL will start.

Sample Uses

To Send a BASIC Program

The program must be stored in an ASCII-format disk file. For example, suppose you are in BASIC and you are ready to save a program on drive 1 with the file name SORTDATA. Then type:

```
Ready
>SAVE "SORTDATA:1", A  ENTER
Ready
>SYSTEM  ENTER
TRSDOS READY
TERMINAL  ENTER
```

Once you have set up the modem and initialized channel A as explained previously, call up the host program and place the telephone handset into the modem. The modem's ready light should come on, indicating that you are receiving the carrier signal from the host program.

```
-- ENTER MENU SELECTION T  ENTER
```

Now go through the necessary sign-on with the program. When you want to send the BASIC program, press BREAK to return to the menu. (If you want to use the prompt wait option, select it now.) Then type:

```
-- ENTER MENU SELECTION G   ENTER
ENTER FILESPEC (1-34)
SORTDATA:1    ENTER
```

Terminal will load the program into RAM. Make sure the host is ready to receive the program, then type:

```
-- ENTER MENU SELECTION X  ENTER
```

and the program will be sent to the host. Press BREAK if you want to stop the transmission for any reason. You will return to the menu. Otherwise you will go into the interactive terminal mode when the program has been sent.

To Receive a BASIC Program

Suppose you are communicating with the host program and it is ready to send you an ASCII-format BASIC program. Before telling the host to go ahead, first go to the menu and type as follows:

```
    -- ENTER MENU SELECTION R ENTER
```

If the buffer is now closed, repeat the R command and TERMINAL will display the message:

```
    RAM BUFFER NOW OPEN
    RESET RAM BUFFER (Y/N) Y    ENTER
```

This opens and clears the buffer. Now go back to the interactive terminal mode (T option) and tell the host to send the program.

After the entire program has been received, press BREAK to return to the menu; then type:

```
    -- ENTER MENU SELECTION C    ENTER
    ENTER FILESPEC (1-34)
    NEWPROG   ENTER
```

This will copy the BASIC program in RAM into a disk file (we named this one NEWPROG).

To try out the program, exit TERMINAL and go into BASIC. Then type:

```
      Ready
      >LOAD "NEWPROG" ENTER
```

After the program has loaded, you should examine it to see if it needs modification to run under Model II BASIC. Pay particular attention to BASIC statements and functions involving input/output:

- Keyboard
- Video Display
- Line Printer
- Disk Files

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 will display an error message in reverse video (black on white); if possible, it will continue normal I/O.

Here are the messages that may be produced while you are in the interactive terminal mode:

P	**Parity error.** The received character will be displayed after the reverse P.
O	**Over-run.** At least one character has been received but not picked up by TERMINAL. This will happen if you are in the menu mode while the host program is sending characters.
F	**Framing error.** The received character will be displayed after the reverse F. Check your SETCOM parameters to see that they match the requirements of the host program.

The following messages can be produced in any mode except the menu:

DATA CARRIER LOST **DATA CARRIER RESTORED**	Check the telephone/modem connection. TERMINAL will pause until the carrier is restored. If TERMINAL was transmitting from RAM or sending an auto sign-on, it will start over at the beginning of the text when the data carrier is restored.
BREAK SEQUENCE RECEIVED	If the host program sends a break sequence, or sends TERMINAL's own break character, this message will be displayed; if TERMINAL is in the transmit from RAM or auto sign-on mode, it will pause until the next character is received from the host.

TERMINAL: A Quick Summary

Feature/ Option	Menu	Transmit From RAM & Auto Sign-On	Interactive Terminal
Return To Menu With BREAK	N/A	Yes	Yes
Execute TRSDOS Commands	Use "S" Command	No	No
Break Character & Key	Use "B" Option	When Break Sequence Or Break Is Received, Pauses Transmission until next Character Is Received	Transmits Character From Keyboard
Wait For Prompt	"W" Option	Yes	No
Program F1 And F2	"F" Option	No	Yes
Ignore Line Feeds After Carriage Returns	"L" Option	Yes	Yes
Output To Printer	"P" Option	Yes	Yes
Self-Echo	Use "E" Option	Yes	Yes
Receive Into RAM	"R" Option	No	Yes
Auto Sign-On	Set Up' With "A" Option; Send With "O" Command	Active Here	Enters This Mode When Done
Video Filter	"V" Option	Yes	Yes

Continued on next page

TERMINAL: A Quick Summary, continued

Feature/ Option	Menu	Transmit From RAM & Auto Sign-On	Interactive Terminal
Get Disk File Into Ram	"G" Command	No	No
Copy Ram To Disk	"C" Option	No	No
Transmit Ram Buffer	"X" Option	Active Here	Enters This Mode When Done
Recognize DC1/DC3 Resume/Pause	No	Yes	No
Display Ram Buffer	"D" Command	No	No
Adjust Speed Of Transmit	No	Yes, With ↑ And ↓	No
Redisplay Menu	"M" Command	No	No
Save Customized Program	"S" Command Dump X'3000' X'3FFF'	No	No

XFERSYS
Transfer, Modify or Upgrade System Files

XFERSYS *:d* {*option*}

:d Specifies the drive containing the destination diskette. If omitted, TRSDOS will prompt for the drive number.

{} If no option is given, FULL is used (see below).

option may be one of the following (these are summaries; details are given later in this section):

FULL tells TRSDOS to purge all conflicting system files from the destination diskette, and to copy all system files from the source to the destination diskette.

MIN tells TRSDOS to purge all system files from the destination diskette, and to copy only the minimum system from the source to the destination diskette.

DATA tells TRSDOS to purge all system files from the destination diskette.

XFERSYS is a multi-purpose utility allowing various system file operations. Here is a summary of its major purposes:

1. Conversion
   Older version (1.1 and 1.2) system and data diskettes must be converted before they can be used under the 2.0 system.

2. Creation of Minimum System or Data Diskettes
   For definitions of ''minimum system'' and ''data'' diskettes, see Section 0 of this manual.

3. Merging System Files from Different Diskettes
   This is useful for combining different system language packages onto a single diskette.

4. Relocating System Files
   XFERSYS allows you to create full or minimum system diskettes on which the directory and system files are moved to a non-standard area of the diskette.

Now we will describe each of these operations in detail, one at a time.

Conversion

The diskette format used by TRSDOS 2.0 is different from that of previous versions. All earlier-version diskettes, both system and data, must be converted before they can be used by TRSDOS 2.0.

Follow this procedure for converting your older version diskettes.

1. Make backups.
   Reset the Computer and insert the old version of TRSDOS. Use this version to make a backup copy of all the diskettes that you plan to convert to the new version. Keep each backup copy in a safe place until the conversion is complete and confirmed by use. THIS IS VERY IMPORTANT.

2. Check free space map.
   Note: This step only applies if the destination diskette is going to contain a full or minimum system. Skip this step for all data-only diskettes.

   Using the older version of TRSDOS, examine the free space map of each of the diskettes.

   TRSDOS 2.0 requires seven tracks immediately following the directory. Any diskette that has these seven unused tracks following the directory (usually tracks 45-51) may be converted as-is.

   Any diskette that does not have these seven free tracks cannot be converted until the files located on these tracks are moved or killed. You should try to copy the files onto another less-full diskette, then kill the files from the diskette you are converting. If you don't know which files are stored on these tracks, then continue with Step 3. XFERSYS will list exactly which files need to be removed.

   For the same reason, there should be no flawed tracks on these same seven tracks. If there are any flaws, you must backup the diskette onto another one which has no flaws in this area.

   Note: If the 1.1 or 1.2 diskette has the TRSDOS system on it, then tracks 45-48 will usually be allocated to TRSDOS files. You do not have to move these files, since XFERSYS will automatically delete them. However, if user files are allocated to tracks 49-51, these must be moved.

3. Reset the Computer and insert your 2.0 working master created according to the instructions in Section 0 of this manual.

   Note: The working master contains only those system files that you use from day to day. The more system files on your master, the longer the XFERSYS operation will take, and the greater the chance that XFERSYS will be aborted due to limited disk space.

4. Conversion Step.

4-A. If you want the destination diskette to have a full system, use this command:

```
XFERSYS  :d
```

where *d* is the drive which will contain the destination diskette. Multi-drive users should always use drive 1, 2 or 3, since no disk swaps will be required. Single-drive users must use drive 0; XFERSYS will prompt you to swap diskettes whenever required. Remember: the SOURCE diskette is your 2.0 working master; the DESTINATION diskette is the one you are converting.

4-B. If you want the destination diskette to have a minimum system diskette, use this command:

```
XFERSYS  :d  MIN
```

where *d* is as explained in step 4-A.

4-C. If you want the destination diskette converted to a non-system data diskette, use this command:

```
XFERSYS  :d  DATA
```

where *d* specifies the destination drive, either 1, 2 or 3.

5. XFERSYS runs in three phases (detailed below). The screen will show which phase is in progress. The complete conversion may take up to 10 minutes, depending on the number of system files on your source diskette.

   If there are not seven contiguous free tracks following the directory, XFERSYS will list the files that must be moved to free up this area.

   In this case, XFERSYS will terminate without making any changes to the destination diskette. Write down the file names listed. Reset the Computer, insert the older version of TRSDOS, and kill the files listed by XFERSYS (make copies first unless you don't need them!). Then start over at step 3.

6. If the conversion is successful, the message CONVERSION COMPLETE will be displayed. The resultant (destination) diskette is now a TRSDOS 2.0 full system, minimum system or data diskette, depending on the option selected.

Creation of Minimum System or Data Diskettes

XFERSYS can also take a 2.0 diskette and remove some or all system files to create a minimum system or a data diskette. Conversely, it can add some or all system files to a data diskette. The steps are exactly like those described previously, except:

- The *entire* operation is performed under TRSDOS 2.0.
- There is no conversion phase.
- You may use ANALYZE to determine which files (if any) need to be moved.

Merging System Files

There are times when you want to move the system files from one diskette onto another — without eliminating the system files already on the destination diskette. Suppose, for example, you have two Radio Shack language packages on separate diskettes, and you want to combine both packages onto a single diskette. XFERSYS can accomplish this via the FULL option.

Look back at the syntax block at the opening of this section. Compare the descriptions given for the three options. Notice that the FULL option purges only *conflicting* system files from the destination diskette, but *leaves other system files alone*. This feature is what allows the merging of system files.

Note: ''System'' files can be identified by looking at the ATTRB column in the directory listing. All system files will have the ''S'' attribute.

1. Backup the destination diskette.
2. Type in the command:

```
XFERSYS  :d  FULL
```

where *d* is the drive number for the destination diskette.

First XFERSYS will purge from the destination diskette all system files that are duplicated on source diskette. It will not purge the system files unique to the destination diskette (i.e., the language-package files).

Next XFERSYS will copy all system files from the source. This includes all of the standard TRSDOS files on the source plus the language files which have the ''S'' attribute in the directory listing.

Note: Do not attempt to merge and perform a conversion at the same time. Do the conversion first and then you may merge.

Relocating System Files

For general applications, the standard location of the TRSDOS directory and system files is most efficient. Each time you use BACKUP to duplicate a system diskette, the system location is preserved. (In fact, the entire destination diskette becomes a sector-for-sector duplicate of the source.)

However, in special cases, you might want to locate the directory and system files in some area other than tracks 44-51. For example, by putting this system information on tracks 1-8, you leave the rest of the diskette as a single extent. If your application involves the creation and maintenance of a single data file, this arrangement might be desirable.

Follow this procedure for moving a full or minimum system to a non-standard area:

1. Use FORMAT to initialize a diskette with the directory in the desired location. For example:

   ```
   FORMAT  :d  DIR = 1
   ```

   where *d* is the destination drive, would place the primary directory on track 1.

2. Now you use XFERSYS to transfer a full or minimum system onto the formatted diskette from Step 1.

2-A. Full System. Use a command like this:

```
XFERSYS  :d  FULL
```

where d is the destination drive, will move a full system onto the destination diskette. The system files will occupy tracks 2-8.

2-B. Minimum System. Use a command like this:

```
XFERSYS  :d  MIN
```

where *d* is the destination drive, will move a minimum system onto the destination diskette. The system files will occupy tracks 2-8.

Details of the XFERSYS Process

XFERSYS has three phases.

Phase 1, Part 1:

XFERSYS determines whether the diskette is in the current (2.0) format or in an earlier (1.1 or 1.2) format. If it is in the current format, XFERSYS skips to Phase 1, Part 2 described below.

If the diskette is in an earlier format, XFERSYS removes the old system files. Next it converts the diskette to the current format. But before this is done, XFERSYS must make sure there are seven unused tracks immediately following the directory on the destination diskette. If there aren't, XFERSYS will display the message,

```
The following must be killed or moved
```

and then list the files which are ''in the way''. Then TRSDOS will cancel the operation, leaving the diskette unchanged.

Phase 1, Part 2

If ''Full'' option and destination disk is:

1.1 or 1.2	2.0
TRSDOS kills certain system files which do not work under 2.0.	TRSDOS kills only those system files whose names are in the source diskette's directory.

If the 'FULL' option is *not* used, TRSDOS kills all system files on the destination diskette (whether it is a 1.1, 1.2 or 2.0).

Phase 2

TRSDOS now copies the system file SYSTEM/SYS. This is the file residing on seven contiguous tracks following the directory.

Phase 3

TRSDOS copies the remaining system programs (marked with an ''S'' in the directory-listing attribute column). The more system files on the diskette, the longer the XFERSYS operation will take.

Note: If the XFERSYS fails after the diskette has been partially modified, you will get the message

```
INCOMPLETE Conversion
```

The diskette will be unusable. Make a backup from your ''safe copy'' and start over.

BASCOM (BASIC Program)
COMSUB (USR Subroutine)
DOCOM (TRSDOS DO-File)
Serial I/O Demonstration

BASCOM*nn*
is the name of the BASIC program; *nn* = 32 for 32K computers; *nn* = 64 for 64K computers.

COMSUB*nn*
is the name of the USR subroutine; *nn* = 32 or 64 as above.

DOCOM*nn*
is the name of a TRSDOS DO-file that automatically initializes channel A for serial I/O, loads COMSUB*nn* and starts BASIC; *nn* = 32 or 64 as above.

The BASIC communication program BASCOM and the communications subroutine COMSUB together perform a "terminal" function, demonstrating an application for Model II's serial interface. The programs are provided to give you a feel for interfacing programs at the assembly-language level with BASIC programs. BASCOM calls the machine-language COMSUB via the USR function. Like TERMINAL, BASCOM and COMSUB are included on your system diskette and may be examined by employing the TRSDOS command LIST (or the BASIC command LIST in the case of BASCOM).

The programs allow you to use the keyboard of the Model II to send data in the form of ASCII characters to another computer or device; at the same time, characters transmitted on the other device will be received by the Model II and printed on the Display.

Serial channel A is used for sending and receiving. (You must put a terminator plug on channel B.) The programs alternately check channel A for a character received, and the keyboard for a character typed. Characters typed will be automatically echoed to the Video Display, though this can be defeated by making a two-byte modification to COMSUB; the NOPs at X'6F9F' and X'6FA0' (COMSUB32) or X'EF9F' and X'EFA0' (COMSUB64) should be modified to a LD (HL), 00.

Sample Use

Before using the two programs, make sure that channel A is connected to a modem and that channel B is fitted with the serial terminator or connected to some other serial device (e.g., a serial printer). Then, under TRSDOS READY, type:

```
DO DOCOM32
```

or

```
DO DOCOM64
```

depending on whether you have a 32K or 64K computer. DOCOM32 and DOCOM64 name DO files which

1. Execute the SETCOM command (parameters are set to default values)
2. Load the appropriate COMSUB subroutine
3. Load BASIC and reserve enough memory for COMSUB. When the BASIC prompt appears on the screen, type:

```
RUN "BASCOM32"
```

or

```
RUN "BASCOM64"
```

The program will load and begin.

Error Handling

When a transmit or receive error is detected, COMSUB will print the word ERROR followed by an 8-bit error code. The leftmost digit represents bit 7; the right-most digit, bit zero. (The program will continue to attempt serial I/O.)

If bit 3 is on, then the modem carrier was lost. Check the telephone connection and all other connections.

If bits 0, 1, and 2 are all off, the error will be a receive error. See ARVC, page 4/78 in the TRSDOS manual, for further details on the error code.

If bits 4, 5, 6, and 7 are all off, the error is a transmit error. See ATX, page 4/79 in the TRSDOS manual, for details.

Source Listing of COMSUB

The following fully commented listing is provided to aid assembly-language programmers in writing their own serial I/O routines.

Notice that the program is ORGed at X'EF80'. This is an appropriate address for 64K machines. For 32K machines, use a start address of X'6F80'.

```
;               SUBROUTINE FOR BASIC COMMUNICATIONS PROGRAM

;           THIS ROUTINE MUST BE EXECUTED AT 3ØØ BAUD OR HIGHER

        ORG     ØEF8ØH          ;ON ENTRY DE POINTS TO A 3 BYTE STRING DESCRIPTOR
        INC     DE              ;DE NOW POINTS TO LSB OF STRING ADDDRESS
        LD      A,(DE)          ;LSB OF STRING ADDRESS TO ACCUMULATOR
        LD      L,A             ;LSB OF STRING ADDRESS TO REGISTER L
        INC     DE              ;DE NOW POINTS TO MSB OF STRING ADDRESS
        LD      A,(DE)          ;MSB OF STRING ADDRESS TO ACCUMULATOR
        LD      H,A             ;MSB OF STRING ADDRESS TO REGISTER H
        LD      A,(HL)          ;1 BYTE STRING TO ACCUMULATOR
        CP      Ø               ;SEE IF CHARACTER IS ZERO
        JR      NZ,XMITER       ;IF NOT ZERO TRANSMIT CHARACTER, ELSE FALL THROUGH TO RECIEVE CHARACTER
        LD      A,96            ;SVC CALL;PORT A RECIEVE
        RST     8               ;
        JR      C,ERROR         ;QUIT ON ERROR IF MODEM CARRIER NOT PRESENT
        RET     NZ              ;RETURN IF NO CHARACTER RECIEVED
        OR      A               ;SET STATUS BITS
        JR      NZ,ERROR        ;QUIT ON ERROR IF ANY STATUS BITS ARE SET
        LD      (HL),B          ;PASS RECIEVED CHARACTER TO STRING LOCATION
        RET

XMITER  LD      B,A             ;CHARACTER TO BE TRANSMITTED TO REGISTER B
        LD      DE,ØFFFFH       ;LOOP COUNT IF TRANSMITER BUSY STATUS ENCOUNTERED
XMIT1   LD      A,97            ;SVC CALL;PORT A TRANSMIT
        RST     8               ;
        JR      C,ERROR         ;QUIT ON ERROR IF MODEM CARRIER NOT PRESENT
        NOP                     ;INSERT "LD     (HL),ØØ" HERE WHEN USING MODEM IN HALF-DUPLEX MODE
        NOP                     ;
        RET     Z               ;RETURN IF CHARACTER TRANSMITTED
        BIT     Ø,A             ;CHECK CLEAR TO SEND STATUS BIT
        JR      NZ,ERROR        ;QUIT ON ERROR IF STATUS BIT SET
        LD      A,D             ;MSB OF LOOP COUNT TO ACCUMULATOR
        OR      E               ;LSB OF LOOP COUNT
        DEC     DE              ;REDUCE LOOP COUNT
        JR      NZ,XMIT1        ;LOOP IF COUNT IS NOT ZERO, ELSE FALL THROUGH TO AN ERROR
ERROR   LD      (HL),ØØ         ;DO NOT DISPLAY CHARACTER IF ERROR ENCOUNTERED
        LD      B,8             ;LOOP COUNT (8 BIT STATUS BYTE)
        LD      HL,BITST3       ;STORAGE AREA
BITEST  BIT     7,A             ;CHECK BIT 7 OF ACCUMULATOR FOR COMMUNICATIONS STATUS
        LD      (HL),'Ø'        ;LOAD ASC-II ZERO
        JR      Z,BITST1        ;JUMP IF STATUS BIT NOT SET
        INC     (HL)            ;ASC-II ZERO => ASC-II ONE IF STATUS BIT SET
BITST1  RLCA                    ;ROTATE ACCUMULATOR LEFT
        INC     HL              ;MOVE TO NEXT STORAGE POSITION
        DJNZ    BITEST          ;LOOP TO CHECK STATUS OF 8 BITS
        LD      HL,BITST2       ;ERROR MESSAGE TO BE DISPLAYED
        LD      B,14            ;LENGTH OF MESSAGE
        LD      C,'#'           ;CHARACTER TO BE INSERTED AT THE END OF ERROR MESSAGE
        LD      A,9             ;SVC CALL;VIDEO LINE
        RST     8               ;
        RET

BITST2  DEFM    'ERROR#'        ;ERROR MESSAGE
BITST3  DEFS    8               ;STORAGE AREA FOR ERROR STATUS BITS TO BE DISPLAYED
```

DATM (Machine-Language Subroutine)
EXDATM (Console Program)
Date Calculations

DATM*nn*
is a machine-language subroutine that performs date calculations; *nn* = 32 for 32K computers; *nn* = 64 for 64K computers.

EXDATM*nn*
is a console program (can be executed from TRSDOS READY mode) that loads and uses DATM*nn*; *nn* = 32 or 64 as above.

EXDATM is a console routine which calls the date calculation program DATM and allows you to pass data to it from the keyboard.

DATM performs two functions:

1. Given a date and a timespan measured in days, the program adds the timespan to the date and calculates the resulting date. The timespan can't be greater than 65535 days; the year of the date must be at least 1600 and not larger than 9999.
2. Given two dates, the program calculates the number of days between the dates. The same parameter limitations apply.

Sample Uses

Under TRSDOS READY, type:

```
EXDATM64
```

or

```
EXDATM32
```

depending on whether you have a 64K or 32K machine. The program will ask you:

```
DATE ADDITION OR DIFFERENCE (A/D) ..
```

Date Addition

To add a date and a timespan, type:

```
A ENTER
```

The program will ask:

```
ENTER PARAMETERS ....................
```

to which a sample reply might be:

```
Ø5/14/1977 ØØ18Ø ENTER
```

The program will compute and print the date which is 180 days after May 14, 1977:

```
THU NOV 1Ø 1977 314
```

followed by the system time. The number 314 means that November 10 is the 314th day of 1977 (see the TRSDOS DATE command).

In general to find a new date from a date and a timespan, the date and timespan should be entered like this:

mm/dd/yyyyb̸jjjjj

where *mm* is a two-digit number 01-12; *dd*, a two-digit number 01-31; *yyyy*, a four-digit number 1600-9999; and *jjjjj* (Julian measure of time), a five-digit number 00001-65535. (Julian time refers to the measure of time in days rather than hours, years, months, etc.) Be sure to use leading zeroes as required to make up the specified number of digits. And don't omit the space (b̸) between *yyyy* and *jjjjj*.

Date Difference

To find the number of days between two days, type D ENTER in response to the initial prompt (A/D). In response to the ENTER PARAMETERS prompt, type in your data as in this sample:

Ø4/15/1979 12/31/1979 ENTER

The program will print the number of days between the dates you entered: 625.

In general to find a timespan between two dates, the dates should be entered like this:

mm/dd/yyyyb̸mm/dd/yyyy

where *mm* is a two-digit number 01-12, *dd*, a two-digit number 01-31; and *yyyy*, a four-digit number 1600-9999. The earlier date must come first. Since the maximum timespan that can be returned is 65535 days, the two dates must not be separated by more than 179 years. Use leading zeroes where necessary to make up the specified number of digits, and don't omit the space (b̸) between the two dates.

DATM Source Listing For Assembly-Language Programmers

Notice that the program is ORGed at X'0000'. This is simply to make it easier for you to relocate the program. DATM32 is actually ORGed at X'6C60'; DATM64, at X'EC60'.

In general, the entry and exit conditions for this subroutine are the same as those given for EXDATM.

Entry Conditions

B = Function Switch
If B = 0 then compute date addition
If B ≠ 0 then compute date difference

For date addition (B = 0)

(HL) = 16-byte text buffer located above X'27FF', as follows:
mm/dd/yyyyƀjjjjj
just like the input text for EXDATM described previously.
(DE) = 26-byte output buffer located above X'27FF', as follows:

NAME OF DAY	MONTH	DAY OF MONTH	YR.	DAY OF YR.	TIME	MONTH OF YR.	DAY OF WEEK
3	3	2	4	3	8	2	1

For date difference (B ≠ 0)

(HL) = 21-byte text buffer located above X'27FF', as follows:
mm/dd/yyyyƀmm/dd/yyyy
just like the input text for EXDATM described previously.

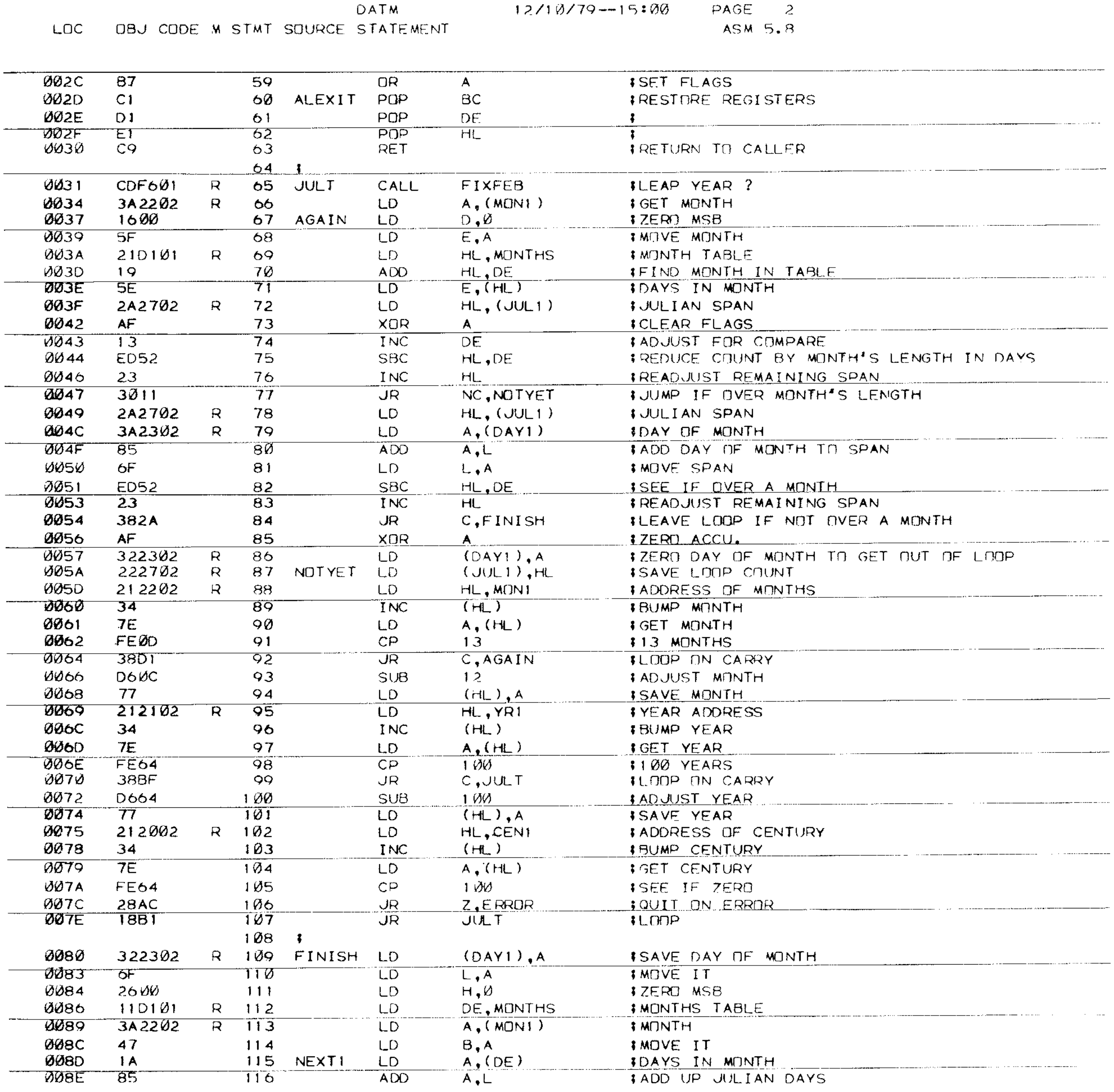
(DE) = 5-byte output buffer, located above X'27FF', as follows:
jjjjj
with leading blanks supplied as necessary.

To call DATM, execute a CALL to the start address.

Exit conditions

(DE) = output text, in the format described previously.
NZ = Error occurred
A = Error code

```
                                   DATM           12/1Ø/79--15:ØØ     PAGE    1
LOC    OBJ CODE M STMT SOURCE STATEMENT                                ASM 5.8

                       1  ;===========================================================================
                       2  ;                            JULIAN DATE MATH ROUTINE
                       3  ;===========================================================================
                       4  ;
                       5  ;                            B=FUNCTION SWITCH
                       6  ;
                       7  ;                                      IF B=Z THEN MM/DD/Y.YYY+JJJJJ=NEW DATE
                       8  ;
                       9  ;                                      IF B=NZ THEN MM/DD/YYYY-MM/DD/YYYY=JJJJJ
                      1Ø  ;
                      11  ;
                      12  ; B=Z    ENTRY : DE=>26 BYTE OUTPUT FIELD (SEE SYSTEM DATE ROUTINE)
                      13  ;                HL=>MM/DD/YYYY JJJJJ
                      14  ;
                      15  ;                     MM=MONTH
                      16  ;                     DD=DAY
                      17  ;                   YYYY=YEAR (16ØØ <= YYYY <= 9999)
                      18  ;                  JJJJJ=JULIAN TIME SPAN IN DAYS (JJJJJ <= 65535)
                      19  ;
                      2Ø  ; B=NZ   ENTRY  : DE=>5 BYTE OUTPUT FIELD (JJJJJ)
                      21  ;                 HL=>MM/DD/YYYY MM/DD/Y.YYY (EARLIER DATE, LATER DATE)
                      22  ;
                      23  ;        EXIT   : A=Z GOOD INDICATION
                      24  ;                 A=NZ A CONTAINS ERROR CODE
                      25  ;
                      26  ;===========================================================================
                      27  ;
                      28  ;                FOR 64K VERSION RELOCATE AT EC6ØH
                      29  ;
                      3Ø  ;                FOR 32K VERSION RELOCATE AT 6C6ØH
                      31  ;
                      32  ;===========================================================================
                      33  ;
 ØØØØ   E5            34  DATM     PUSH    HL                  ;SAVE CALLER'S REGISTERS
 ØØØ1   D5            35           PUSH    DE                  ;
 ØØØ2   C5            36           PUSH    BC                  ;
                      37  ;
 ØØØ3   7C            38           LD      A,H                 ;CHECK FOR BAD BUFFER ADDRESS
 ØØØ4   FE28          39           CP      28H                 ;
 ØØØ6   3822          4Ø           JR      C,ERROR             ;QUIT ON ERROR
 ØØØ8   7A            41           LD      A,D                 ;CHECK FOR BAD BUFFER ADDRESS
 ØØØ9   FE28          42           CP      28H                 ;
 ØØØB   381D          43           JR      C,ERROR             ;QUIT ON ERROR
 ØØØD   ED5381Ø1 R    44           LD      (OUTDSS),DE         ;SAVE OUTPUT BUFFER POINTER
 ØØ11   AF            45           XOR     A                   ;ZERO ACCU.
 ØØ12   B8            46           CP      B                   ;LOOK AT OPTION SWITCH
 ØØ13   C2A1Ø2   R    47           JP      NZ,TWODAT           ;ACT ON OPTION
 ØØ16   CD2EØ2   R    48  JULDAT   CALL    EDIT                ;EDIT INPUT
 ØØ19   2ØØF          49           JR      NZ,ERROR            ;QUIT ON ERROR
 ØØ1B   3E2Ø          5Ø           LD      A,' '               ;BLANK
 ØØ1D   BE            51           CP      (HL)                ;CHECK BLANK
 ØØ1E   2ØØA          52           JR      NZ,ERROR            ;QUIT ON ERROR
 ØØ2Ø   23            53           INC     HL                  ;BUMP INFORMATION POINTER
 ØØ21   3E15          54           LD      A,15H               ;CONVERT
 ØØ23   CF            55           RST     8                   ;SVC
 ØØ24   ED5327Ø2 R    56           LD      (JUL1),DE           ;SAVE JULIAN SPAN
 ØØ28   28Ø7          57           JR      Z,JULT              ;QUIT ON ERROR
 ØØ2A   3EØ3          58  ERROR    LD      A,3                 ;PARAMETER
```



```
                                 DATM           12/1Ø/79--15:ØØ      PAGE   2
  LOC    OBJ CODE M STMT SOURCE STATEMENT                             ASM 5.8

  ØØ2C   87            59           OR      A                ;SET FLAGS
  ØØ2D   C1            6Ø   ALEXIT  POP     BC               ;RESTORE REGISTERS
  ØØ2E   D1            61           POP     DE               ;
  ØØ2F   E1            62           POP     HL               ;
  ØØ3Ø   C9            63           RET                      ;RETURN TO CALLER
                       64   ;
  ØØ31   CDF6Ø1    R   65   JULT    CALL    FIXFEB           ;LEAP YEAR ?
  ØØ34   3A22Ø2    R   66           LD      A,(MON1)         ;GET MONTH
  ØØ37   16ØØ          67   AGAIN   LD      D,Ø              ;ZERO MSB
  ØØ39   5F            68           LD      E,A              ;MOVE MONTH
  ØØ3A   21D1Ø1    R   69           LD      HL,MONTHS        ;MONTH TABLE
  ØØ3D   19            7Ø           ADD     HL,DE            ;FIND MONTH IN TABLE
  ØØ3E   5E            71           LD      E,(HL)           ;DAYS IN MONTH
  ØØ3F   2A27Ø2    R   72           LD      HL,(JUL1)        ;JULIAN SPAN
  ØØ42   AF            73           XOR     A                ;CLEAR FLAGS
  ØØ43   13            74           INC     DE               ;ADJUST FOR COMPARE
  ØØ44   ED52          75           SBC     HL,DE            ;REDUCE COUNT BY MONTH'S LENGTH IN DAYS
  ØØ46   23            76           INC     HL               ;READJUST REMAINING SPAN
  ØØ47   3Ø11          77           JR      NC,NOTYET        ;JUMP IF OVER MONTH'S LENGTH
  ØØ49   2A27Ø2    R   78           LD      HL,(JUL1)        ;JULIAN SPAN
  ØØ4C   3A23Ø2    R   79           LD      A,(DAY1)         ;DAY OF MONTH
  ØØ4F   85            8Ø           ADD     A,L              ;ADD DAY OF MONTH TO SPAN
  ØØ5Ø   6F            81           LD      L,A              ;MOVE SPAN
  ØØ51   ED52          82           SBC     HL,DE            ;SEE IF OVER A MONTH
  ØØ53   23            83           INC     HL               ;READJUST REMAINING SPAN
  ØØ54   382A          84           JR      C,FINISH         ;LEAVE LOOP IF NOT OVER A MONTH
  ØØ56   AF            85           XOR     A                ;ZERO ACCU.
  ØØ57   3223Ø2    R   86   NOTYET  LD      (DAY1),A         ;ZERO DAY OF MONTH TO GET OUT OF LOOP
  ØØ5A   2227Ø2    R   87           LD      (JUL1),HL        ;SAVE LOOP COUNT
  ØØ5D   2122Ø2    R   88           LD      HL,MON1          ;ADDRESS OF MONTHS
  ØØ6Ø   34            89           INC     (HL)             ;BUMP MONTH
  ØØ61   7E            9Ø           LD      A,(HL)           ;GET MONTH
  ØØ62   FEØD          91           CP      13               ;13 MONTHS
  ØØ64   38D1          92           JR      C,AGAIN          ;LOOP ON CARRY
  ØØ66   D6ØC          93           SUB     12               ;ADJUST MONTH
  ØØ68   77            94           LD      (HL),A           ;SAVE MONTH
  ØØ69   2121Ø2    R   95           LD      HL,YR1           ;YEAR ADDRESS
  ØØ6C   34            96           INC     (HL)             ;BUMP YEAR
  ØØ6D   7E            97           LD      A,(HL)           ;GET YEAR
  ØØ6E   FE64          98           CP      1ØØ              ;1ØØ YEARS
  ØØ7Ø   38BF          99           JR      C,JULT           ;LOOP ON CARRY
  ØØ72   D664         1ØØ           SUB     1ØØ              ;ADJUST YEAR
  ØØ74   77           1Ø1           LD      (HL),A           ;SAVE YEAR
  ØØ75   212ØØ2    R  1Ø2           LD      HL,CEN1          ;ADDRESS OF CENTURY
  ØØ78   34           1Ø3           INC     (HL)             ;BUMP CENTURY
  ØØ79   7E           1Ø4           LD      A,(HL)           ;GET CENTURY
  ØØ7A   FE64         1Ø5           CP      1ØØ              ;SEE IF ZERO
  ØØ7C   28AC         1Ø6           JR      Z,ERROR          ;QUIT ON ERROR
  ØØ7E   18B1         1Ø7           JR      JULT             ;LOOP
                      1Ø8   ;
  ØØ8Ø   3223Ø2    R  1Ø9   FINISH  LD      (DAY1),A         ;SAVE DAY OF MONTH
  ØØ83   6F           11Ø           LD      L,A              ;MOVE IT
  ØØ84   26ØØ         111           LD      H,Ø              ;ZERO MSB
  ØØ86   11D1Ø1    R  112           LD      DE,MONTHS        ;MONTHS TABLE
  ØØ89   3A22Ø2    R  113           LD      A,(MON1)         ;MONTH
  ØØ8C   47           114           LD      B,A              ;MOVE IT
  ØØ8D   1A           115   NEXT1   LD      A,(DE)           ;DAYS IN MONTH
  ØØ8E   85           116           ADD     A,L              ;ADD UP JULIAN DAYS
```

```
                                            DATM             12/1Ø/79--15:ØØ     PAGE    3
     LOC     OBJ CODE  M STMT SOURCE STATEMENT                                    ASM 5.8

     ØØ8F    3ØØ1          117           JR      NC,NEXT2         ;JUMP IF NO CARRY
     ØØ91    24            118           INC     H                ;BUMP MSB
     ØØ92    6F            119  NEXT2    LD      L,A              ;MOVE LSB
     ØØ93    13            12Ø           INC     DE               ;BUMP TABLE ADDRESS
     ØØ94    1ØF7          121           DJNZ    NEXT1            ;LOOP
     ØØ96    2224Ø2     R  122           LD      (JULIAN),HL      ;SAVE JULIAN DAYS
                           123  ;
     ØØ99    218AØ1     R  124  LDBUFF   LD      HL,TABLX         ;LENGTH TABLE
     ØØ9C    2293Ø1     R  125           LD      (TABDRS),HL      ;SAVE TABLE ADDRESS
     ØØ9F    212ØØ2     R  126           LD      HL,CEN1          ;ADDRESS OF CENTURY
     ØØA2    6E            127           LD      L,(HL)           ;CENTURY
     ØØA3    ØE64          128           LD      C,1ØØ            ;*1ØØ
     ØØA5    Ø6ØØ          129           LD      B,Ø              ;MULTIPLY PARAMETER
     ØØA7    6Ø            13Ø           LD      H,B              ;ZERO H
     ØØA8    3E17          131           LD      A,17H            ;MULTIPLY
     ØØAA    CF            132           RST     8                ;SVC
     ØØAB    3A21Ø2     R  133           LD      A,(YR1)          ;GET YEAR
     ØØAE    16ØØ          134           LD      D,Ø              ;ZERO MSB
     ØØBØ    5F            135           LD      E,A              ;MOVE YEAR
     ØØB1    19            136           ADD     HL,DE            ;GET TOTAL YEAR
     ØØB2    114ØØ6        137           LD      DE,16ØØ          ;BASE YEAR 16ØØ
     ØØB5    B7            138           OR      A                ;ZERO CARRY
     ØØB6    ED52          139           SBC     HL,DE            ;YEARS SINCE BASE
     ØØB8    E5            14Ø           PUSH    HL               ;SAVE YEARS SINCE BASE
     ØØB9    Ø6Ø1          141           LD      B,1              ;DIVIDE PARAM
     ØØBB    ØEØ4          142           LD      C,4              ;/4
     ØØBD    3E17          143           LD      A,17H            ;DIVIDE
     ØØBF    CF            144           RST     8                ;SVC
     ØØCØ    23            145           INC     HL               ;ADJUST FOR DAY ZERO SAT->MON
     ØØC1    23            146           INC     HL               ;SAME
     ØØC2    23            147           INC     HL               ;SAME
     ØØC3    23            148           INC     HL               ;SAME
     ØØC4    23            149           INC     HL               ;SAME
     ØØC5    E5            15Ø           PUSH    HL               ;SAVE MOD 4 YEARS
     ØØC6    3A2ØØ2     R  151           LD      A,(CEN1)         ;GET CENTURY
     ØØC9    D61Ø          152           SUB     16               ;BASE CENTURY
     ØØCB    F5            153           PUSH    AF               ;SAVE DIFFERENCE
     ØØCC    6F            154           LD      L,A              ;MOVE DIFFERENCE
     ØØCD    26ØØ          155           LD      H,Ø              ;ZERO MSB FOR DIVIDE
     ØØCF    ØEØ4          156           LD      C,4              ;/4
     ØØD1    3E17          157           LD      A,17H            ;DIVIDE
     ØØD3    CF            158           RST     8                ;SVC
     ØØD4    F1            159           POP     AF               ;DIFFERENCE
     ØØD5    95            16Ø           SUB     L                ;SUB # OF QUADRENNIALS
     ØØD6    6F            161           LD      L,A              ;MOVE ADJUSTMENT FOR CENTURY
     ØØD7    E5            162           PUSH    HL               ;SAVE ADJUSTMENT
     ØØD8    CDDEØ1     R  163           CALL    LPORNT           ;DETERMINE PROPER ADJUSTMENT (LEAP OR NOT)
     ØØDB    E1            164           POP     HL               ;ADJUSTMENT
     ØØDC    2ØØ1          165           JR      NZ,UPDAT3        ;NOT QUADRENNIAL
     ØØDE    2C            166           INC     L                ;INC ADJUSTMENT
     ØØDF    EB            167  UPDAT3   EX      DE,HL            ;ADJUSTMENT TO DE
     ØØEØ    E1            168           POP     HL               ;MOD 4 YEARS
     ØØE1    AF            169           XOR     A                ;ZERO FLAGS
     ØØE2    ED52          17Ø           SBC     HL,DE            ;ADJUST MOD 4 YEARS
     ØØE4    EB            171           EX      DE,HL            ;MOD 4 YEARS TO DE
     ØØE5    E1            172           POP     HL               ;TOTAL YEARS
     ØØE6    19            173           ADD     HL,DE            ;TOTAL DAYS OFF FROM BASE TO YEAR IN ?
     ØØE7    ED5B24Ø2   R  174           LD      DE,(JULIAN)      ;GET JULIAN DAY
```

```
                                       DATM            12/1Ø/79--15:ØØ     PAGE    4
  LOC   OBJ CODE M STMT SOURCE STATEMENT                                    ASM 5.8

 ØØEB   19            175             ADD    HL,DE          ;ADD JULIAN DAYS
 ØØEC   ØE07          176             LD     C,7            ;/7
 ØØEE   Ø6Ø1          177             LD     B,1            ;DIVIDE PARAM
 ØØFØ   3E17          178             LD     A,17H          ;DIVIDE
 ØØF2   CF            179             RST    8              ;SVC
 ØØF3   2126Ø2      R 18Ø             LD     HL,WK1         ;DAY OF WEEK
 ØØF6   71            181             LD     (HL),C         ;SAVE ABOVE
 ØØF7   Ø6ØØ          182             LD     B,Ø            ;DATE PARAMETER
 ØØF9   2A81Ø1      R 183             LD     HL,(OUTDSS)    ;OUTPUT BUFFER ADDRESS
 ØØFC   3E2D          184             LD     A,2DH          ;DATE
 ØØFE   CF            185             RST    8              ;SVC
 ØØFF   C22AØØ      R 186             JP     NZ,ERROR       ;QUIT ON ERROR
 Ø1Ø2   2195Ø1      R 187             LD     HL,NAMDAY      ;GET DAY OF WEEK TABLE
 Ø1Ø5   79            188             LD     A,C            ;MOVE DAY OF WEEK
 Ø1Ø6   CBØ7          189             RLC    A              ;DOUBLE IT
 Ø1Ø8   81            19Ø             ADD    A,C            ;TRIPLE IT
 Ø1Ø9   85            191             ADD    A,L            ;ADD TABLE ADDRESS
 Ø1ØA   6F            192             LD     L,A            ;FIND TABLE ADDRESS
 Ø1ØB   3ØØ1          193             JR     NC,DAYOFW      ;JUMP IF MSB NC
 Ø1ØD   24            194             INC    H              ;BUMP MSB ON C
 Ø1ØE   Ø1Ø3ØØ        195    DAYOFW   LD     BC,3           ;3 LOOPS
 Ø111   CD8ØØ1      R 196             CALL   LOADSS         ;LD BUFFER
 Ø114   3A22Ø2      R 197             LD     A,(MON1)       ;GET MONTH
 Ø117   21AAØ1      R 198             LD     HL,NAMONT      ;GET TABLE
 Ø11A   47            199             LD     B,A            ;MOVE MONTH
 Ø11B   CBØ7          2ØØ             RLC    A              ;DOUBLE IT
 Ø11D   8Ø            2Ø1             ADD    A,B            ;TRIPLE IT
 Ø11E   85            2Ø2             ADD    A,L            ;ADD TABLE ADDRESS
 Ø11F   6F            2Ø3             LD     L,A            ;FIND TABLE ADDRESS
 Ø12Ø   3ØØ1          2Ø4             JR     NC,MONOFY      ;JUMP IF MSB NC
 Ø122   24            2Ø5             INC    H              ;INC MSB ON C
 Ø123   Ø1Ø3ØØ        2Ø6    MONOFY   LD     BC,3           ;3 LOOPS
 Ø126   CD8ØØ1      R 2Ø7             CALL   LOADSS         ;LOAD BUFFER
 Ø129   3A23Ø2      R 2Ø8             LD     A,(DAY1)       ;GET DAY
 Ø12C   CD5DØ1      R 2Ø9             CALL   CONSUB         ;CONVERT
 Ø12F   3A2ØØ2      R 21Ø             LD     A,(CEN1)       ;GET CENTURY
 Ø132   CD5AØ1      R 211             CALL   CONCAR         ;CONVERT
 Ø135   3A21Ø2      R 212             LD     A,(YR1)        ;GET YEAR
 Ø138   CD5AØ1      R 213             CALL   CONCAR         ;CONVERT
 Ø13B   ED5B24Ø2    R 214             LD     DE,(JULIAN)    ;JULIAN DAY
 Ø13F   B7            215             OR     A              ;CLEAR CARRY
 Ø14Ø   CD61Ø1      R 216             CALL   CONSU1         ;CONVERT AND MOVE
 Ø143   EB            217             EX     DE,HL          ;OUTPUT BUFFER ADDRESS TO HL
 Ø144   Ø1Ø8ØØ        218             LD     BC,8           ;COUNT FOR MOVE
 Ø147   CD8ØØ1      R 219             CALL   LOADSS         ;BUMP OVER TIME ALREADY IN OUTPUT BUFFER
 Ø14A   3A22Ø2      R 22Ø             LD     A,(MON1)       ;GET MONTH
 Ø14D   CD5DØ1      R 221             CALL   CONSUB         ;CONVERT
 Ø15Ø   3A26Ø2      R 222             LD     A,(WK1)        ;DAY OF WEEK
 Ø153   CD5DØ1      R 223             CALL   CONSUB         ;CONVERT
 Ø156   AF            224             XOR    A              ;CLEAR ACCUM
 Ø157   C32DØØ      R 225             JP     ALEXIT         ;JUMP TO EXIT
                      226    ;
 Ø15A   37            227    CONCAR   SCF                   ;SET CARRY FLAG (DON'T CALL EDITM)
 Ø15B   18Ø1          228             JR     NOCARY         ;JUMP OVER CLEARING CARRY FLAG
 Ø15D   B7            229    CONSUB   OR     A              ;CLEAR CARRY FLAG (CALL EDITM)
 Ø15E   16ØØ          23Ø    NOCARY   LD     D,Ø            ;ZERO MSB FOR CONVERT
 Ø16Ø   5F            231             LD     E,A            ;VALUE TO CONVERT
 Ø161   211BØ2      R 232    CONSU1   LD     HL,BUFF5       ;BUFFER CONVERT ADDRESS
```

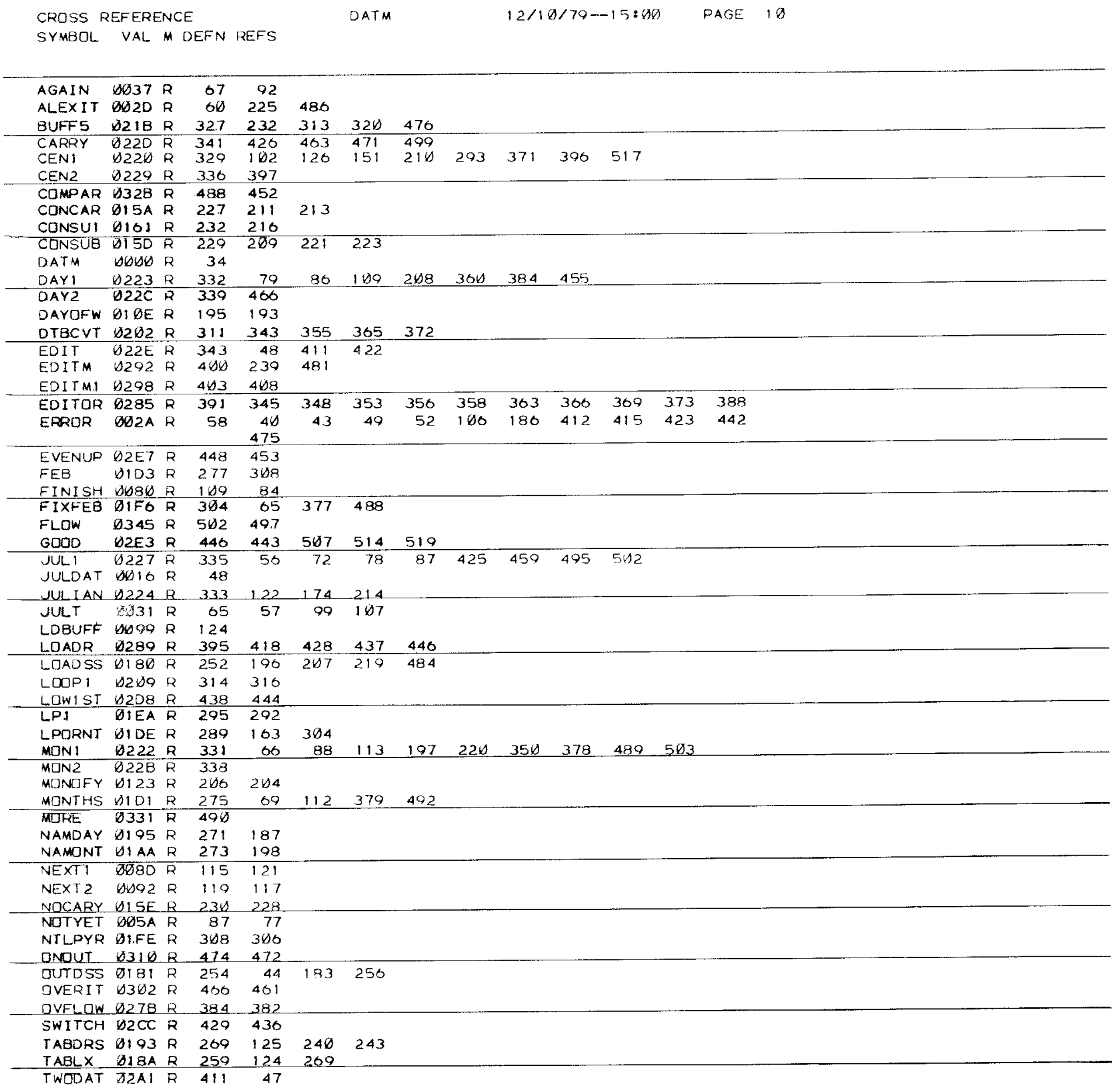
```
                                      DATM           12/10/79--15:00    PAGE   5
 LOC    OBJ CODE M STMT SOURCE STATEMENT                                  ASM 5.8

  Ø164   Ø6ØØ        233           LD      B,Ø              ;CONVERSION PARAMETER
  Ø166   E5          234           PUSH    HL               ;SAVE ADDRESS
  Ø167   F5          235           PUSH    AF               ;SAVE FLAGS
  Ø168   3E15        236           LD      A,15H            ;CONVERT
  Ø16A   CF          237           RST     8                ;SVC
  Ø16B   F1          238           POP     AF               ;GET FLAGS
  Ø16C   D492Ø2    R 239           CALL    NC,EDITM         ;EDIT CONVERSION ON NO CARRY
  Ø16F   2A93Ø1    R 24Ø           LD      HL,(TABDRS)      ;TABLE COUNT
  Ø172   4E          241           LD      C,(HL)           ;LENGTH TO MOVE
  Ø173   23          242           INC     HL               ;BUMP
  Ø174   2293Ø1    R 243           LD      (TABDRS),HL      ;SAVE ADDRESS
  Ø177   AF          244           XOR     A                ;ZERO ACCUM.
  Ø178   47          245           LD      B,A              ;ZERO B
  Ø179   57          246           LD      D,A              ;ZERO D
  Ø17A   3EØ5        247           LD      A,5              ;LENGTH OF CONVERSION BUFFER
  Ø17C   91          248           SUB     C                ;DISPLACEMENT IN CONVERSION BUFFER
  Ø17D   5F          249           LD      E,A              ;GET VALUE
  Ø17E   E1          25Ø           POP     HL               ;FRONT OF CONVERSION BUFFER
  Ø17F   19          251           ADD     HL,DE            ;FIND FIRST DIGIT
                     252  LOADSS   EQU     $
  Ø18Ø   11ØØØØ      253           LD      DE,Ø             ;GET BUFFER ADDRESS
                     254  OUTDSS   EQU     $-2
  Ø183   EDBØ        255           LDIR                     ;MOVE
  Ø185   ED5381Ø1  R 256           LD      (OUTDSS),DE      ;SAVE ADDRESS
  Ø189   C9          257           RET
                     258  ;
  Ø18A   Ø2          259  TABLX    DEFB    2
  Ø18B   Ø2          26Ø           DEFB    2
  Ø18C   Ø2          261           DEFB    2
  Ø18D   Ø3          262           DEFB    3
  Ø18E   Ø2          263           DEFB    2
  Ø18F   Ø2          264           DEFB    2
  Ø19Ø   Ø2          265           DEFB    2
  Ø191   Ø2          266           DEFB    2
  Ø192   Ø1          267           DEFB    1
                     268  ;
  Ø193   8AØ1      R 269  TABDRS   DEFW    TABLX
                     27Ø  ;
  Ø195   4D4F4E54    271  NAMDAY   DEFM    'MONTUEWEDTHUFRISATSUN'
                     272  ;
  Ø1AA   2Ø2Ø2Ø4A    273  NAMONT   DEFM    '   JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC'
                     274  ;
  Ø1D1   ØØ          275  MONTHS   DEFB    ØØ               ;NO MONTH
  Ø1D2   1F          276           DEFB    31               ;JAN
  Ø1D3   1C          277  FEB      DEFB    28               ;FEB
  Ø1D4   1F          278           DEFB    31               ;MAR
  Ø1D5   1E          279           DEFB    3Ø               ;APR
  Ø1D6   1F          28Ø           DEFB    31               ;MAY
  Ø1D7   1E          281           DEFB    3Ø               ;JUN
  Ø1D8   1F          282           DEFB    31               ;JUL
  Ø1D9   1F          283           DEFB    31               ;AUG
  Ø1DA   1E          284           DEFB    3Ø               ;SEP
  Ø1DB   1F          285           DEFB    31               ;OCT
  Ø1DC   1E          286           DEFB    3Ø               ;NOV
  Ø1DD   1F          287           DEFB    31               ;DEC
                     288  ;
  Ø1DE   3A21Ø2    R 289  LPORNT   LD      A,(YR1)          ;YEAR
  Ø1E1   6F          29Ø           LD      L,A              ;MOVE IT
```

```
                              DATM                 12/1Ø/79—15:ØØ      PAGE    6
 LOC    OBJ CODE  M STMT  SOURCE STATEMENT                             ASM 5.8

 Ø1E2   FEØØ         291            CP       Ø                ;SEE IF QUAD OR CEN
 Ø1E4   2ØØ4         292            JR       NZ,LP1           ;JUMP IF NOT
 Ø1E6   212ØØ2     R 293            LD       HL,CEN1          ;CENTURY
 Ø1E9   6E           294            LD       L,(HL)           ;MOVE IT
 Ø1EA   26ØØ         295  LP1       LD       H,Ø              ;ZERO MSB
 Ø1EC   Ø6Ø1         296            LD       B,1              ;DIVIDE PARAMETER
 Ø1EE   ØEØ4         297            LD       C,4              ;/4
 Ø1FØ   3E17         298            LD       A,17H            ;DIVIDE
 Ø1F2   CF           299            RST      8                ;SVC
 Ø1F3   AF           3ØØ            XOR      A                ;ZERO ACCU.
 Ø1F4   B9           3Ø1            CP       C                ;REMAINDER
 Ø1F5   C9           3Ø2            RET
                     3Ø3  ;
 Ø1F6   CDDEØ1     R 3Ø4  FIXFEB    CALL     LPDRNT           ;LEAP YEAR ?
 Ø1F9   3E1C         3Ø5            LD       A,28             ;TWENTY EIGHT DAYS OF FEB.
 Ø1FB   2ØØ1         3Ø6            JR       NZ,NTLPYR        ;JUMP IF NOT LEAP YEAR
 Ø1FD   3C           3Ø7            INC      A                ;29 DAYS OF FEB.
 Ø1FE   32D3Ø1     R 3Ø8  NTLPYR    LD       (FEB),A          ;MOD MONTHS TABLE
 Ø2Ø1   C9           3Ø9            RET
                     31Ø  ;
 Ø2Ø2   Ø6Ø3         311  DTBCVT    LD       B,3              ;COUNT FOR LEADING ZEROS
 Ø2Ø4   3E3Ø         312            LD       A,'Ø'            ;LEADING ZEROS
 Ø2Ø6   111BØ2     R 313            LD       DE,BUFF5         ;CONVERT BUFFER
 Ø2Ø9   12           314  LOOP1     LD       (DE),A           ;LOAD LEADING ZEROS
 Ø2ØA   13           315            INC      DE               ;BUMP
 Ø2ØB   1ØFC         316            DJNZ     LOOP1            ;LOOP
 Ø2ØD   ØEØ2         317            LD       C,2              ;COUNT FOR MOVE
 Ø2ØF   EDBØ         318            LDIR                      ;MOVE
 Ø211   E5           319            PUSH     HL               ;SAVE POINTER TO INFORMATION
 Ø212   211BØ2     R 32Ø            LD       HL,BUFF5         ;FRONT OF BUFFER
 Ø215   Ø4           321            INC      B                ;CONVERSION PARAMETER
 Ø216   3E15         322            LD       A,15H            ;CONVERT
 Ø218   CF           323            RST      8                ;SVC
 Ø219   E1           324            POP      HL               ;POINTER TO INFORMATION
 Ø21A   C9           325            RET
                     326  ;
 Ø21B                327  BUFF5     DEFS     Ø5               ;CONVERSION BUFFER
                     328  ;
 Ø22Ø   ØØ           329  CEN1      DEFB     ØØ               ;CENTURY STORE
 Ø221   ØØ           33Ø  YR1       DEFB     ØØ               ;YEAR STORE
 Ø222   ØØ           331  MON1      DEFB     ØØ               ;MONTH STORE
 Ø223   ØØ           332  DAY1      DEFB     ØØ               ;DAY STORE
 Ø224   ØØØØ         333  JULIAN    DEFW     ØØ               ;JULIAN DAYS IN YEAR
 Ø226   ØØ           334  WK1       DEFB     ØØ               ;DAY OF WEEK STORE
 Ø227   ØØØØ         335  JUL1      DEFW     ØØ               ;JULIAN DAYS STORE
 Ø229   ØØ           336  CEN2      DEFB     ØØ               ;CENTURY STORE
 Ø22A   ØØ           337  YR2       DEFB     ØØ               ;YEAR STORE
 Ø22B   ØØ           338  MON2      DEFB     ØØ               ;MONTH STORE
 Ø22C   ØØ           339  DAY2      DEFB     ØØ               ;DAY STORE
                     34Ø  ;
 Ø22D   ØØ           341  CARRY     DEFB     ØØ               ;CARRY ACCUMULATOR
                     342  ;
 Ø22E   CDØ2Ø2     R 343  EDIT      CALL     DTBCVT           ;CONVERT STRING INFORMATION
 Ø231   BB           344            CP       E                ;CP MONTH TO ZERO
 Ø232   2851         345            JR       Z,EDITOR         ;QUIT ON ERROR
 Ø234   3EØC         346            LD       A,12             ;12 MONTHS
 Ø236   BB           347            CP       E                ;MONTHS
 Ø237   384C         348            JR       C,EDITOR         ;QUIT ON ERROR
```

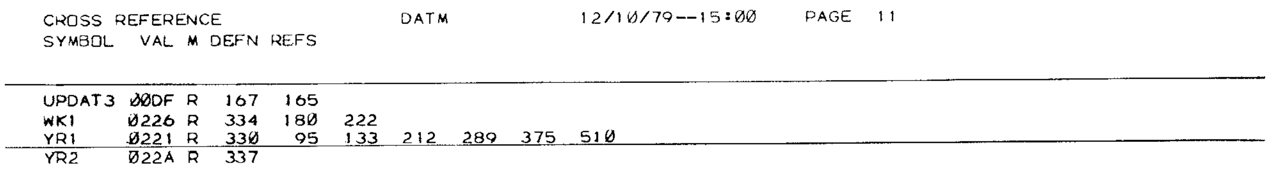
```
                                  DATM            12/10/79--15:00     PAGE   7
  LOC    OBJ CODE  M STMT SOURCE STATEMENT                            ASM 5.8

  Ø239   7B             349           LD      A,E           ;MONTHS
  Ø23A   3222Ø2     R   35Ø           LD      (MON1),A      ;SAVE MONTHS
  Ø23D   3E2F           351           LD      A,'/'         ;DATE DELIMITER
  Ø23F   BE             352           CP      (HL)          ;COMPARE TO STRING
  Ø24Ø   2Ø43           353           JR      NZ,EDITOR     ;QUIT ON ERROR
  Ø242   23             354           INC     HL            ;BUMP POINTER
  Ø243   CDØ2Ø2     R   355           CALL    DTBCVT        ;CONVERT STRING
  Ø246   2Ø3D           356           JR      NZ,EDITOR     ;QUIT ON ERROR
  Ø248   BB             357           CP      E             ;CP DAY TO ZERO
  Ø249   283A           358           JR      Z,EDITOR      ;QUIT ON ERROR
  Ø24B   7B             359           LD      A,E           ;DAY
  Ø24C   3223Ø2     R   36Ø           LD      (DAY1),A      ;SAVE DAY
  Ø24F   3E2F           361           LD      A,'/'         ;DATE DELIMITER
  Ø251   BE             362           CP      (HL)          ;CP TO STRING
  Ø252   2Ø31           363           JR      NZ,EDITOR     ;QUIT ON ERROR
  Ø254   23             364           INC     HL            ;BUMP POINTER
  Ø255   CDØ2Ø2     R   365           CALL    DTBCVT        ;CONVERT STRING
  Ø258   2Ø2B           366           JR      NZ,EDITOR     ;QUIT ON ERROR
  Ø25A   3EØF           367           LD      A,15          ;15TH CENTURY
  Ø25C   BB             368           CP      E             ;CENTURY
  Ø25D   3Ø26           369           JR      NC,EDITOR     ;QUIT ON ERROR
  Ø25F   7B             37Ø           LD      A,E           ;CENTURY
  Ø26Ø   322ØØ2     R   371           LD      (CEN1),A      ;SAVE CENTURY
  Ø263   CDØ2Ø2     R   372           CALL    DTBCVT        ;CONVERT STRING
  Ø266   2Ø1D           373           JR      NZ,EDITOR     ;QUIT ON ERROR
  Ø268   7B             374           LD      A,E           ;YEAR
  Ø269   3221Ø2     R   375           LD      (YR1),A       ;STORE YEAR
  Ø26C   E5             376           PUSH    HL            ;SAVE POINTER TO INFORMATION
  Ø26D   CDF6Ø1     R   377           CALL    FIXFEB        ;FIX FEB.
  Ø27Ø   3A22Ø2     R   378           LD      A,(MON1)      ;MONTH
  Ø273   21D1Ø1     R   379           LD      HL,MONTHS     ;MONTHS TABLE
  Ø276   85             38Ø           ADD     A,L           ;ADJUST TABLE ADDRESS
  Ø277   6F             381           LD      L,A           ;MOVE LSB OF MONTH ADDRESS
  Ø278   3ØØ1           382           JR      NC,OVFLOW     ;JUMP ON NO OVERFLOW
  Ø27A   24             383           INC     H             ;MSB FOR OVERFLOW
  Ø27B   3A23Ø2     R   384    OVFLOW LD      A,(DAY1)      ;DAY
  Ø27E   3D             385           DEC     A             ;ADJUST FOR COMPARE
  Ø27F   BE             386           CP      (HL)          ;COMPARE DAY
  Ø28Ø   E1             387           POP     HL            ;RESTORE POINTER
  Ø281   3ØØ2           388           JR      NC,EDITOR     ;QUIT ON ERROR
  Ø283   AF             389           XOR     A             ;CLEAR FLAGS
  Ø284   C9             39Ø           RET
  Ø285   3EØ3           391    EDITOR LD      A,3           ;ERROR ON CALL
  Ø287   B7             392           OR      A             ;SET FLAG
  Ø288   C9             393           RET
                        394    ;
  Ø289   Ø6Ø4           395    LOADR  LD      B,4           ;LOOP COUNT
  Ø28B   212ØØ2     R   396           LD      HL,CEN1       ;FRONT OF FIRST DATE INFO.
  Ø28E   1129Ø2     R   397           LD      DE,CEN2       ;FRONT OF SECOND DATE INFO.
  Ø291   C9             398           RET                   ;RETURN TO SENDER
                        399    ;
  Ø292   ØEØ5           4ØØ    EDITM  LD      C,5           ;# SIGN. DIGITS COUNT
  Ø294   Ø6Ø4           4Ø1           LD      B,4           ;MAX SPACES TO INSERT
  Ø296   3E3Ø           4Ø2           LD      A,'Ø'         ;LEADING ZERO VALUE
  Ø298   BE             4Ø3    EDITM1 CP      (HL)          ;ASC-II ZERO ?
  Ø299   CØ             4Ø4           RET     NZ            ;LEAVE IF NOT
  Ø29A   ØD             4Ø5           DEC     C             ;DEC SIGN. DIGIT COUNT
  Ø29B   362Ø           4Ø6           LD      (HL),' '      ;PUT IN LEADING SPACE
```

```
                                 DATM             12/1Ø/79--15:ØØ     PAGE    8
 LOC    OBJ CODE  M STMT SOURCE STATEMENT                               ASM 5.8

 Ø29D   23          4Ø7               INC     HL              ;NEXT BYTE
 Ø29E   1ØF8        4Ø8               DJNZ    EDITM1          ;LOOP
 Ø2AØ   C9          4Ø9               RET                     ;RETURN
                    41Ø   ;
 Ø2A1   CD2EØ2    R 411   TWODAT      CALL    EDIT            ;EDIT INPUT INFORMATION
 Ø2A4   C22AØØ    R 412               JP      NZ,ERROR        ;QUIT ON ERROR
 Ø2A7   3E2Ø        413               LD      A,' '           ;BLANK
 Ø2A9   BE          414               CP      (HL)            ;COMPARE WITH DELIMITER IN STRING
 Ø2AA   C22AØØ    R 415               JP      NZ,ERROR        ;QUIT ON ERROR
 Ø2AD   23          416               INC     HL              ;BUMP INFORMATION POINTER
 Ø2AE   E5          417               PUSH    HL              ;SAVE POINTER
 Ø2AF   CD89Ø2    R 418               CALL    LOADR           ;LOAD REGISTERS WITH FRONT OF DATE INFO.
 Ø2B2   Ø1Ø4ØØ      419               LD      BC,4            ;LOOP COUNT
 Ø2B5   EDBØ        42Ø               LDIR                    ;MOVE
 Ø2B7   E1          421               POP     HL              ;POINTER TO INFORMATION
 Ø2B8   CD2EØ2    R 422               CALL    EDIT            ;EDIT INFORMATION
 Ø2BB   C22AØØ    R 423               JP      NZ,ERROR        ;QUIT ON ERROR
 Ø2BE   21ØØØØ      424               LD      HL,ØØ           ;ZERO HL
 Ø2C1   2227Ø2    R 425               LD      (JUL1),HL       ;ZERO JULIAN COUNT
 Ø2C4   212DØ2    R 426               LD      HL,CARRY        ;CARRY ACCUMULATOR
 Ø2C7   36ØØ        427               LD      (HL),Ø          ;ZERO IT
 Ø2C9   CD89Ø2    R 428               CALL    LOADR           ;LOAD REGISTERS WITH FRONT OF DATE INFO.
 Ø2CC   1A          429   SWITCH      LD      A,(DE)          ;GET FIRST DATE
 Ø2CD   4E          43Ø               LD      C,(HL)          ;MOVE 1ST DATE
 Ø2CE   77          431               LD      (HL),A          ;
 Ø2CF   79          432               LD      A,C             ;
 Ø2DØ   12          433               LD      (DE),A          ;
 Ø2D1   23          434               INC     HL              ;BUMP POINTER
 Ø2D2   13          435               INC     DE              ;BUMP POINTER
 Ø2D3   1ØF7        436               DJNZ    SWITCH          ;LOOP
 Ø2D5   CD89Ø2    R 437               CALL    LOADR           ;LOAD REGISTERS WITH FRONT OF DATE INFO.
 Ø2D8   1A          438   LOW1ST      LD      A,(DE)          ;2ND DATE
 Ø2D9   BE          439               CP      (HL)            ;1ST DATE
 Ø2DA   23          44Ø               INC     HL              ;BUMP
 Ø2DB   13          441               INC     DE              ;BUMP
 Ø2DC   DA2AØØ    R 442               JP      C,ERROR         ;QUIT ON ERROR
 Ø2DF   2ØØ2        443               JR      NZ,GOOD         ;QUIT LOOKING IF DDD2 IS SMALLER DATE
 Ø2E1   1ØF5        444               DJNZ    LOW1ST          ;LOOP
                    445   ;
 Ø2E3   CD89Ø2    R 446   GOOD        CALL    LOADR           ;LOAD REGISTERS WITH FRONT OF DATE INFO.
 Ø2E6   Ø5          447               DEC     B               ;REDUCE COUNT BY ONE
 Ø2E7   1A          448   EVENUP      LD      A,(DE)          ;GET 2ND DATE
 Ø2E8   BE          449               CP      (HL)            ;CP TO NEW DATE (FIRST DATE)
 Ø2E9   23          45Ø               INC     HL              ;BUMP
 Ø2EA   13          451               INC     DE              ;BUMP
 Ø2EB   2Ø3E        452               JR      NZ,COMPAR       ;ADD A MONTH AND UPDATE
 Ø2ED   1ØF8        453               DJNZ    EVENUP          ;LOOP UNTIL UPDATED
                    454   ;
 Ø2EF   3A23Ø2    R 455               LD      A,(DAY1)        ;EARLIER DAY
 Ø2F2   5F          456               LD      E,A             ;MOVE IT
 Ø2F3   AF          457               XOR     A               ;ZERO ACCU.
 Ø2F4   57          458               LD      D,A             ;MOVE IT
 Ø2F5   2A27Ø2    R 459               LD      HL,(JUL1)       ;TOTAL JULIAN DAYS
 Ø2F8   ED52        46Ø               SBC     HL,DE           ;SUBTRACT DAYS
 Ø2FA   3ØØ6        461               JR      NC,OVERIT       ;SKIP IF NO CARRY
 Ø2FC   E5          462               PUSH    HL              ;SAVE TOTAL
 Ø2FD   212DØ2    R 463               LD      HL,CARRY        ;CARRY ACCUMULATOR
 Ø3ØØ   35          464               DEC     (HL)            ;ADJUST ACCORDING TO CARRY
```

```
                                   DATM              12/1Ø/79--15:ØØ       PAGE   9
 LOC    OBJ CODE  M STMT SOURCE STATEMENT                                  ASM 5.8

Ø3Ø1    E1            465         POP    HL              ;TOTAL
Ø3Ø2    3A2CØ2    R   466  OVERIT LD     A,(DAY2)        ;THE OTHER DAY
Ø3Ø5    5F            467         LD     E,A             ;MOVE IT
Ø3Ø6    19            468         ADD    HL,DE           ;ADD DAYS IN MONTH TO JULIAN TOTAL
Ø3Ø7    EB            469         EX     DE,HL           ;TOTAL TO DE
Ø3Ø8    3EØØ          47Ø         LD     A,Ø             ;ZERO ACCU. (KEEP FLAGS)
Ø3ØA    212DØ2    R   471         LD     HL,CARRY        ;CARRY ACCUMULATOR
Ø3ØD    3ØØ1          472         JR     NC,ONOUT        ;GO ON IF NO CARRY
Ø3ØF    34            473         INC    (HL)            ;BUMP CARRY ACCUMULATOR
Ø31Ø    BE            474  ONOUT  CP     (HL)            ;SEE IF ANY CARRYS ARE LEFT
Ø311    C22AØØ    R   475         JP     NZ,ERROR        ;QUIT ON ERROR
Ø314    211BØ2    R   476         LD     HL,BUFF5        ;CONVERSION BUFFER
Ø317    Ø6ØØ          477         LD     B,Ø             ;CONVERSION PARAMETER
Ø319    E5            478         PUSH   HL              ;SAVE FRONT OF BUFFER
Ø31A    3E15          479         LD     A,15H           ;CONVERT
Ø31C    CF            48Ø         RST    8               ;SVC
Ø31D    CD92Ø2    R   481         CALL   EDITM           ;EDIT CONVERSION
Ø32Ø    E1            482         POP    HL              ;FRONT OF BUFFER
Ø321    Ø1Ø5ØØ        483         LD     BC,5            ;COUNT
Ø324    CD8ØØ1    R   484         CALL   LOADSS          ;MOVE TO OUTPUT BUFFER
Ø327    AF            485         XOR    A               ;GOOD INDICATION
Ø328    C32DØØ    R   486         JP     ALEXIT          ;END
                      487  ;
Ø32B    CDF6Ø1    R   488  COMPAR CALL   FIXFEB          ;FIX FEB.
Ø32E    3A22Ø2    R   489         LD     A,(MON1)        ;MONTH
Ø331    16ØØ          49Ø  MORE   LD     D,Ø             ;ZERO MSB
Ø333    5F            491         LD     E,A             ;MOVE MONTH
Ø334    21D1Ø1    R   492         LD     HL,MONTHS       ;MONTHS TABLE
Ø337    19            493         ADD    HL,DE           ;FIND MONTH
Ø338    5E            494         LD     E,(HL)          ;GET # OF DAYS IN MONTH
Ø339    2A27Ø2    R   495         LD     HL,(JUL1)       ;TOTAL FOR JULIAN DAYS
Ø33C    19            496         ADD    HL,DE           ;ADD UP TOTAL
Ø33D    3ØØ6          497         JR     NC,FLOW         ;JUMP IF NO OVERFLOW
Ø33F    E5            498         PUSH   HL              ;SAVE TOTAL
Ø34Ø    212DØ2    R   499         LD     HL,CARRY        ;CARRY ACCUMULATOR
Ø343    34            5ØØ         INC    (HL)            ;BUMP CARRY
Ø344    E1            5Ø1         POP    HL              ;TOTAL
Ø345    2227Ø2    R   5Ø2  FLOW   LD     (JUL1),HL       ;SAVE TOTAL
Ø348    2122Ø2    R   5Ø3         LD     HL,MON1         ;MONTH
Ø34B    34            5Ø4         INC    (HL)            ;BUMP MONTH
Ø34C    7E            5Ø5         LD     A,(HL)          ;MONTH TO ACCU.
Ø34D    FEØD          5Ø6         CP     13              ;13 MONTHS
Ø34F    3892          5Ø7         JR     C,GOOD          ;LOOP ON CARRY
Ø351    D6ØC          5Ø8         SUB    12              ;READJUST
Ø353    77            5Ø9         LD     (HL),A          ;STORE IT
Ø354    2121Ø2    R   51Ø         LD     HL,YR1          ;YEAR
Ø357    34            511         INC    (HL)            ;BUMP YEAR
Ø358    7E            512         LD     A,(HL)          ;YEAR TO ACCU.
Ø359    FE64          513         CP     1ØØ             ;1ØØ YEARS
Ø35B    3886          514         JR     C,GOOD          ;LOOP ON CARRY
Ø35D    D664          515         SUB    1ØØ             ;READJUST
Ø35F    77            516         LD     (HL),A          ;STORE IT
Ø36Ø    212ØØ2    R   517         LD     HL,CEN1         ;CENTURY
Ø363    34            518         INC    (HL)            ;BUMP CENTURY
Ø364    C3E3Ø2    R   519         JP     GOOD            ;LOOP
```



```
CROSS REFERENCE                DATM             12/1Ø/79--15:ØØ      PAGE  1Ø
SYMBOL   VAL  M DEFN REFS

AGAIN    ØØ37 R   67    92
ALEXIT   ØØ2D R   6Ø   225   486
BUFF5    Ø21B R  327   232   313   32Ø   476
CARRY    Ø22D R  341   426   463   471   499
CEN1     Ø22Ø R  329   1Ø2   126   151   21Ø   293   371   396   517
CEN2     Ø229 R  336   397
COMPAR   Ø32B R  488   452
CONCAR   Ø15A R  227   211   213
CONSU1   Ø161 R  232   216
CONSUB   Ø15D R  229   2Ø9   221   223
DATM     ØØØØ R   34
DAY1     Ø223 R  332    79    86   1Ø9   2Ø8   36Ø   384   455
DAY2     Ø22C R  339   466
DAYOFW   Ø1ØE R  195   193
DTBCVT   Ø2Ø2 R  311   343   355   365   372
EDIT     Ø22E R  343    48   411   422
EDITM    Ø292 R  4ØØ   239   481
EDITM1   Ø298 R  4Ø3   4Ø8
EDITOR   Ø285 R  391   345   348   353   356   358   363   366   369   373   388
ERROR    ØØ2A R   58    4Ø    43    49    52   1Ø6   186   412   415   423   442
                       475
EVENUP   Ø2E7 R  448   453
FEB      Ø1D3 R  277   3Ø8
FINISH   ØØ8Ø R  1Ø9    84
FIXFEB   Ø1F6 R  3Ø4    65   377   488
FLOW     Ø345 R  5Ø2   497
GOOD     Ø2E3 R  446   443   5Ø7   514   519
JUL1     Ø227 R  335    56    72    78    87   425   459   495   5Ø2
JULDAT   ØØ16 R   48
JULIAN   Ø224 R  333   122   174   214
JULT     ØØ31 R   65    57    99   1Ø7
LDBUFF   ØØ99 R  124
LOADR    Ø289 R  395   418   428   437   446
LOADSS   Ø18Ø R  252   196   2Ø7   219   484
LOOP1    Ø2Ø9 R  314   316
LOW1ST   Ø2D8 R  438   444
LP1      Ø1EA R  295   292
LPORNT   Ø1DE R  289   163   3Ø4
MON1     Ø222 R  331    66    88   113   197   22Ø   35Ø   378   489   5Ø3
MON2     Ø22B R  338
MONOFY   Ø123 R  2Ø6   2Ø4
MONTHS   Ø1D1 R  275    69   112   379   492
MORE     Ø331 R  49Ø
NAMDAY   Ø195 R  271   187
NAMONT   Ø1AA R  273   198
NEXT1    ØØ8D R  115   121
NEXT2    ØØ92 R  119   117
NOCARY   Ø15E R  23Ø   228
NOTYET   ØØ5A R   87    77
NTLPYR   Ø1FE R  3Ø8   3Ø6
ONOUT    Ø31Ø R  474   472
OUTDSS   Ø181 R  254    44   183   256
OVERIT   Ø3Ø2 R  466   461
OVFLOW   Ø27B R  384   382
SWITCH   Ø2CC R  429   436
TABDRS   Ø193 R  269   125   24Ø   243
TABLX    Ø18A R  259   124   269
TWODAT   Ø2A1 R  411    47
```



```
CROSS REFERENCE                DATM             12/1Ø/79--15:ØØ      PAGE  11
SYMBOL   VAL  M DEFN REFS

UPDAT3   ØØDF R  167   165
WK1      Ø226 R  334   18Ø   222
YR1      Ø221 R  33Ø    95   133   212   289   375   51Ø
YR2      Ø22A R  337
```

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

```
DO  HERZ50
```

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

This utility is provided for customers in areas where the AC power is 50 rather than 60 Hz. It should not be used by any other customers.

HERZ50 is a DO-file that makes a few changes in the video display software of TRSDOS. Only the drive zero diskette is changed. Be sure it is write-enabled before you start the DO-file. Once the HERZ50 changes are done, they will remain in effect for that diskette.

To perform the change, type under TRSDOS READY:

```
DO  HERZ50
```

The program provides several opportunities for you to abort the process by pressing BREAK . Pressing any other key during a pause will cause the program to continue.

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

LPII
Modify Printer Driver for use with Line Printer II

```
DO  LPII
```

This DO-file changes the TRSDOS printer driver to make it properly operate the Radio Shack Line Printer II. It makes a permanent change in the TRSDOS diskette. The change will take effect when the Computer is reset or started using the modified TRSDOS.

Sample Use

```
DO  LPII
```

PRTBKSP
Modify Printer Driver for Backspacing

```
DO  PRTBKSP
```

Some printers are capable of backspacing for the purpose of overprinting and other special applications. For example, the Radio Shack Daisy Wheel printers have this capability.

The TRSDOS printer driver ordinarily ignores the backspace code (ASCII X'08'). If you are using a printer which can handle this code properly, you may want to modify TRSDOS so that it will send the backspace code to the printer.

To make the modification, execute the DO-file named PRTBKSP. This will make a permanent change in the TRSDOS diskette. The change will not take effect until the Computer is started or reset using the modified TRSDOS.

After the modification has been made, TRSDOS will send the backspace character to the printer, and will decrement the character-in-line count.

Sample Use

```
DO   PRTBKSP
```

Section 4

Technical Information

This chapter explains TRSDOS on a technical level. You do not need it to use the Operator Commands, nor do you need it to run BASIC applications programs on the Computer. You do need it to write assembly programs which use System routines. You may also find the information incidentally useful in programming with BASIC.

Diskette Organization

Model II uses single-sided, double-density diskettes. Each diskette contains 77 tracks, numbered 0-76.

Each track contains 26 sectors, numbered 1-26. Each sector contains 256 bytes, except for track 0 sectors, which contain 128 bytes. The total capacity of a diskette is:

$$(76 * 26 * 256) + (1 * 26 * 128) = 509{,}184 \text{ bytes}$$

Disk Space Available to User

Sector 26 of each track is reserved for System use, giving the user 25 sectors per track. On System diskettes, 65 tracks are available for the user; on non-System diskettes, 75 tracks are available.

Details: Track 0 is reserved by the System. Another track (usually track 44) is reserved by the System for the diskette directory. On Operating System diskettes, ten additional tracks are used for System files.

Unit of Allocation

The only unit of disk space allocation is the "granule". A TRSDOS granule is defined as 5 sectors. Therefore the smallest non-empty file consists of 5 sectors; i.e., one granule.

NON-SYSTEM DISKETTE	TRACKS	GRANULES	SECTORS	BYTES
1	75	375	1875	480,000
–	1	5	25	6400
–	–	1	5	1280
–	–	–	1	256

Table 4.1 Space Available to User

Disk Files

Methods of File Allocation

Model II provides two ways to allocate disk space for files: Dynamic Allocation and Pre-Allocation.

Dynamic Allocation

With Dynamic Allocation, the System allocates granules only at the time of write. For example, when a file is first Opened for output, no space is allocated. The first space allocation is done at the first write. Additional space is added as required by subsequent writes.

With dynamically allocated files, unused granules are de-allocated (recovered) when the file is Closed.

Pre-Allocation

With Pre-Allocation, the file is allocated a specified number of granules when it is Created. Pre-Allocated files can only be created by the operator command CREATE.

TRSDOS will dynamically extend (enlarge) a Pre-Allocated file as needed for subsequent write operation. However, TRSDOS will not de-allocate unused granules when a pre-allocated file is Closed. The way to reduce the size of a Pre-Allocated file is to Copy it to a dynamically allocated file and Kill the Pre-Allocated one.

Record Length

The Model II transfers data to and from diskettes one sector at a time; i.e., in 256-byte blocks. These are the System's "physical" records.

User records or "logical" records are the buffers of data you wish to transfer to or from a file. These can be from 1 to 256 bytes long.

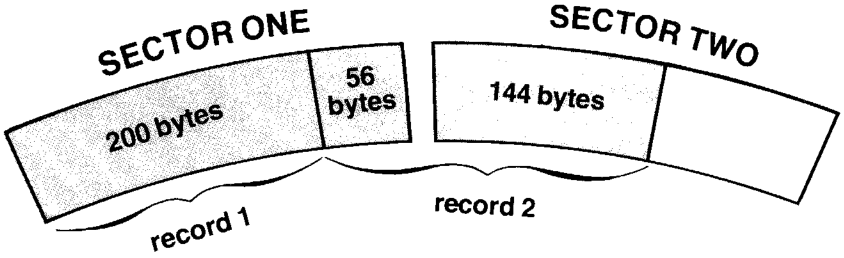
The Operating System will automatically "block" your logical records into physical records which will be transferred to disk, and "de-block" the physical records into logical records which are used by your program. Therefore your **only** concern during file-access is with logical records. You never need to worry about physical records, sectors, tracks, etc. This is to your benefit, since physical record lengths and features may change in later TRSDOS 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 will automatically be spanned across sectors.

For example, if the record length is 200, Sectors 1 and 2 will look like this:



Fixed-Length and Variable Length Records

Model II files can have either fixed-length or variable-length records. Files with fixed-length records will be referred to as FLRs; files with variable length records, VLRs.

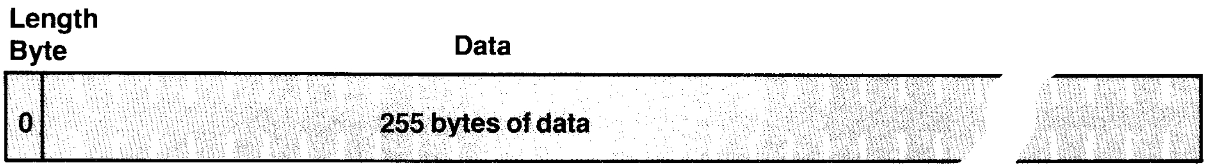
Record length in an FLR file is set when the file is Opened for the first time. This length can be any value from 1 to 256 bytes. Once set, the record length in an FLR cannot be changed, unless the file is being over-written with new data.

Record length in a VLR file is specified in a one-byte length-field at the beginning of each record. The record-lengths in a VLR file can vary. For example, the first record in a file might have a length of 32; the second, 17; the third, 250; etc.

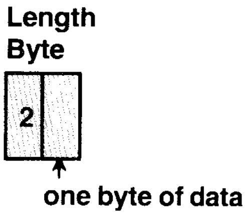
The record-length byte indicates the entire length of the record, **including** the length-byte. This can be any value from 0 to 255. A value of 1 can be used, but it has no meaning.

Examples:

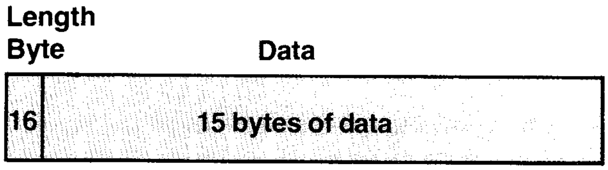
A length-byte value of zero indicates that the record contains 255 bytes of data:



A length-byte value of 2 indicates that the record contains 1 byte of data:



A length-byte value of 16 indicates that the record contains 15 bytes of data:



Record Processing Capabilities

Model II TRSDOS allows both Direct and Sequential file access. Direct access—sometimes called "random access", but "direct" is more descriptive — allows you to process any record you specify.

Note: A file can contain up to 65535 records. Records are numbered from 0 (beginning of file) to 65534. A record number of 65535 indicates the end of file (EOF). These limits will be changed in a later release of TRSDOS.

Sequential access allows you to process records in sequence: Record N, N+1, N+2, ... With sequential access, you do not specify a record number; instead, the Operating System accesses the next record following the last record processed.

For files with fixed length records (FLRs) you can position the current record pointer to the beginning of the file, end of file, or to any record in the file. In short, you can use Direct and/or Sequential Access with FLRs at any time during processing.

For files with variable length records (VLRs) you can only position to the beginning of the file or to the end of file. You **cannot** position to any other record in the file, since the position of interior VLRs cannot be calculated. If short, you can only use Sequential access with VLRs.

The Direct access routines are Direct-Read and Direct-Write; the Sequential access routines are Read-Next and Write-Next. Direct access routines always access the record you specify. Sequential access routines always access the record **following** the last record processed. (When the file is first opened, sequential processing starts with record 0.)

Examples

Assume you have a Fixed Length Record file currently Open. Here are some typical sequences you can accomplish via the file processing routines.

1. Read and/or write records in the file — in any order
This is done using Direct-Write and Direct-Read routines. You could read record 5, write at end of file, read record 3, write record 3, etc.

2. Sequential Read (or Write) beginning anywhere in the file.
First you would do a Direct-Read to the record where you want to start reading or writing. After that, you would do sequential reads or writes until done.

3. **Sequential Write starting at end of file.**
First do a Direct-Write to the end of the file. Then do sequential writes until done.

4. **Determine the number of records in a file.**
First do a Direct-Read to end of file, then use the LOCATE routine to get the current record number, which now equals *number of records +1.*

Examples with Variable Length Records

Here are examples of ways to read or write to VLR files:

1. **Start reading or writing sequentially at first record**
Open the file and start reading or writing sequentially until done.

2. **Sequential Write starting at end of file**
(same process as Example 3 above).

Note: Whenever you write to a VLR, the end of the file is **automatically** reset to the last record you write. This means you cannot update a VLR file directly; you must read in the file and output the updated information to a new VLR file.

How to Use the Supervisor Calls

Supervisor Calls (SVCs) are Operating System routines available to any user program. The routines alter certain System functions and conditions; provide file access; perform I/O to the Keyboard, Video Display, and Printer; and perform various computations.

All the SVCs leave memory above X'2FFF' untouched. Only those Z-80 registers used to pass parameters from the SVC are altered. All others are unaffected. However, all the prime registers are used by the System; they are not restored.

Each SVC is assigned a Function Code. These codes run from 0 through 127. Only the first 96 are defined by the System; codes 96-127 are available for user definition.

To specify a given Supervisor Call, your program refers to the SVC's Function Code.

Calling Procedure

All SVCs are accomplished via the RST 8 instruction.

1. Load the Function Code for the desired SVC into the A register. Also load any other registers which are needed by the SVC, as detailed under "Supervisor Calls."
2. Execute a RST 8 instruction.
3. Upon return from the SVC, the Z flag will be set if the function was successful. If the Z flag is not set, there was an error. The A register contains the appropriate error code (except after certain computational SVCs, which use the A register to return other information).

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,10                    ; B=CHARACTER COUNT
        LD     C,0DH                   ; 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
```

Error Codes and Messages

Register A usually contains a return code after any function call. The Z flag is set when no error occurred. Exceptions are certain computational routines, which use the A and F registers to pass back data and status information.

CODE	MESSAGE
0	No Error Found
1	Bad Function Code On SVC Call Or No Function Exists
2	Character Not Available
3	Parameter Error On Call
4	CRC Error During Disk I/O Operation
5	Disk Sector Not Found
6	Attempt To Open A File Which Has Not Been Closed
7	Illegal Disk Change
8	Disk Drive Not Ready
9	Invalid Data Provided By Caller
10	Maximum Of 16 Files May Be Open At Once
11	File Already In Directory
12	No Drive Available For An Open
13	Write Attempt To A Read Only File
14	Write Fault On Disk I/O
15	Disk Is Write Protected
16	DCB Is Modified And Is Unusable
17	Directory Read Error
18	Directory Write Error
19	Improper File Name (filespec)
20	FAD Read Error
21	FAD Write Error
22	FID Read Error
23	FID Write Error
24	File Not Found
25	File Access Denied Due To Password Protection
26	Directory Space Full
27	Disk Space Full
28	Attempt To Read Past EOF
29	Read Attempt Outside Of File Limits
30	No More Extents Available (16 Maximum)
31	Program Not Found
32	Unknown Drive Number (filespec)
33	Disk Space Allocation Cannot Be Made Due To Fragmentation Of Space
34	Attempt To Use A Non Program File As A Program
35	Memory Fault During Program Load
36	Parameter For Open Is Incorrect

(Continued on next page)

CODE	MESSAGE
37	Open Attempt For A File Already Open
38	I/O Attempt To An Unopen File
39	Illegal I/O Attempt
40	SEEK Error
41	Data Lost During Disk I/O (Hardware Fault)
42	Printer Not Ready
43	Printer Out Of Paper
44	Printer Fault (May Be Turned Off)
45	Printer Not Available
46	Not Applicable To VLR Type Files
47	Required Command Parameter Not Found
48	Incorrect Command Parameter
49	Hardware Fault During Disk I/O
50-255	** Unknown Error Code **

Supervisor Calls

In this section we will use the following notation:

NOTATION	MEANING
RP = data	The register-pair RP contains the data.
n1 < R< n2	The register R contains a value greater than n1 and less than n2
(RP) = data	The register-pair RP contains the address of ("points to") the data.
NZ = Error	If Z flag is not set, an error occurred.

When a range is not given, any representable number can be used. For example, a register can contain any value from 0-255.

The contents of this section are:

System Control

Supervisor calls described in this section:

FUNCTION CODE	NAME	PURPOSE
0	INITIO	Initializes all I/O drivers
2	SETUSR	Sets up a user-defined SVC
3	SETBRK	Sets up BREAK key processing program
15	DISKID	Reads a diskette ID
25	TIMER	Set timer to interrupt program
29	HLDKEY	Process HOLD
36	JP2DOS	Returns to TRSDOS (TRSDOS READY)
37	DOSCMD	Sends TRSDOS a command and then returns to TRSDOS READY
38	RETCMD	Sends TRSDOS a command and return to caller
39	ERROR	Displays "ERROR *number*"
52	ERRMSG	Returns Error Message to Buffer
57	CLRXIT	Clear RAM and return to TRSDOS

Table 4.3. System Control Supervisor Calls

INITIO
Initialize I/O (function code 0)

This routine initializes all input/output drivers. It calls all of the other initialization routines. There are no parameters.

Note: This routine has been done already by the System. Users never need to call it, except in extreme error conditions.

Error Conditions

A = 0

Exit Conditions

NZ = Error
A = Error Code

SETUSR
Set User (function code 2)

This routine sets or removes a user vector. This gives you the ability to add SVC functions. Function codes 96-101 are available for user definition, unless the serial interface is on (see RS232C). Function codes 102-127 are always available for user definition.

Once added, such a function can then be called via the RST 8 instruction, just like the System's SVC routines.

Your routine must reside above X'27FF', and should end with a RET instruction.

To change a previously defined function, you must first remove the old vector.

Entry Conditions

HL = Entry address of your routine (when C is not equal to 0)
B = Function code to be used, 95< code< 128
C = Set/Reset code. If C=0, remove the vector. Otherwise, add the vector.
A = 2

Exit Conditions

HL = Removed vector address (when C=0 on entry)

SETBRK
Set BREAK (function code 3)

This routine lets you enable the BREAK key by defining a BREAK-key processing program. Whenever the BREAK key is pressed, your processing program takes over. On entry to the BREAK processing program, the return address of the interrupted routine is on the top of the stack and can be returned to with a RETurn instruction. All Z-80 register contents are unchanged upon entry to the BREAK program.

The routine also lets you disable the BREAK key, by removing the address of the processing program. While BREAK key is enabled, you cannot change processing programs; you must disable it first.

The BREAK key processing program must reside above X'27FF'.

See "Handling Programmed Interrupts" for programming information.

Entry Conditions

HL = Address of BREAK key processing program. When BREAK is pressed, control transfers to this address.
If HL = 0, then address of previous processing program is removed.

A = 3

Exit Conditions

NZ = Error

A = Error Code

(HL) = Address of deleted BREAK key processing program, if HL = 0 on entry

DISKID (function code 15)

This routine reads the diskette ID from any or all of drives 0 through 3. (The diskette ID is assigned by the FORMAT and BACKUP utilities.) This routine is useful when the program needs to ensure that the Operator has inserted the proper diskette.

Entry Conditions

B = Drive Select Code. If B = 0, read from drive 0, etc.
B must be one of the following: 0, 1, 2, 3, or 255. If B = 255, then routine reads from all four drives.

(HL) = Buffer to hold the diskette ID(s).
If B = 0, 1, 2 or 3, then buffer must be 8-bytes long. If B = 255, then buffer must be 32 bytes long. Drive 0 ID will be placed in first 8 bytes, then drive 1, etc.

A = 15

Exit Conditions

The Diskette ID(s) are placed in the buffers pointed to by register-pair HL. If a drive is not ready, blanks are placed into the buffer.

NZ = Error.

A = Error Code.

TIMER
(function code 25)

This routine lets you start a timer to interrupt a program when time runs out. Unlike the DELAY routine, TIMER runs concurrently with your program. One application would be to give an operator a specified number of seconds for keyboard input, and to interrupt the keyboard input routine if no input was made within the time limit.

When setting the timer, you tell it how many seconds to count down. TRSDOS will then continue executing your program, until the timer counts down to zero or you reset the timer.

This is a "one-shot" timer. When it counts to zero and causes an interrupt, it automatically shuts off.

See Programming with TRSDOS for information on interrupts.

Entry Conditions

(HL) =	Routine to handle interrupt when timer counts to zero
BC =	Number of seconds to count down
A =	25

If HL and BC both equal zero, then timer is turned off.
If HL = 0 and BC is not equal to zero, then time count is reset to the value in BC, and timing continues.

HLDKEY
Process the HOLD Key (function code 29)

This routine enables the HOLD key pause-function.

Entry Conditions

A = 29

B = Function Code

Contents of B	Result
B = 0	Turn off HOLD processor. Pressing HOLD will generate keyboard data X'00'.
B = 1	Turn on HOLD processor. Pressing HOLD will not generate keyboard data; it will be intercepted by TRSDOS.
B > 1	Check for HOLD key. If it has been pressed one or more times, pause until it is pressed again. No matter how many times HOLD is pressed before this function is called, it will be interpreted as a single keystroke, thus starting the pause.

Exit Conditions

NZ HOLD processor is off.

User Notes

To use this function, first turn on the HOLD processor by calling the SVC with B = 1. Then, when you want to check whether HOLD has been pressed, call it again with B > 1. If HOLD has been pressed, the routine will not return until HOLD is pressed a second time.

Returning to the TRSDOS READY command level turns off the HOLD processor. However, you can execute a command without turning off the HOLD processor. See SVC's JP2DOS and DOSCMD. Some commands like DIR, LIST, etc., will reset the HOLD processor.

JP2DOS
Jump to DOS (function 36)

This program simply returns control to the command level (TRSDOS READY). All Open files are Closed automatically.

Entry Conditions

A = 36

Note: An alternate way to perform this function is to execute an RST Ø instruction. There are no entry conditions for this method.

DOSCMD
DOS Command (function code 37)

This routine sends TRSDOS a command. After the command is executed, control returns to TRSDOS (TRSDOS READY). All Open files are closed automatically.

Entry Conditions

(HL) =	TRSDOS command string
B =	Length of command string
A =	37

RETCMD
Return after Command (function code 38)

This routine sends TRSDOS an operator command. After completion of the command, control returns to your program. All Open files are Closed automatically.

Note: Take care that TRSDOS doesn't overlay your program while loading the command file you specified. Most TRSDOS library commands use memory below X'27FF'; a few go up to but not including X'2FFF'. Single-drive, single-disk copies use all user memory. See **Library Commands** for details.

Entry Conditions

(HL) =	TRSDOS command string
B =	Length of command string.
A =	38

Exit Conditions

NZ =	Error
A =	Error Code

ERROR
(function code 39)

This routine displays the message ERROR followed by the specified error code. The message appears at the current cursor position.

Entry Conditions

B =	Error Code
A =	39

Exit Conditions

NZ =	Error
A =	Error Code

ERRMSG
Error Message (function code 52)

This routine returns an 80-byte descriptive error message to the specified buffer area. (See list of Error Codes and Messages.)

Entry Conditions

B =	Error Code corresponding to message
(HL) =	80-byte buffer area in user area (above X'27FF')
A =	52

Exit Conditions

NZ =	Error
A =	Error Code

CLRXIT
Clear User Memory and Jump to TRSDOS (function code 57)

This routine zeroes user memory and transfers control to TRSDOS READY.

Entry Conditions

A = 57

Exit Conditions

None

Keyboard

Supervisor calls described in this section*:

FUNCTION CODE	NAME	PURPOSE
1	KBINIT	Clears stored keystrokes.
4	KBCHAR	Gets a character from keyboard.
5	KBLINE	Gets a line from keyboard.
12	VIDKEY	Display message and get line from KB.

Table 4.4. Keyboard Supervisor Calls

*VIDKEY is described later on under "Video Display."

KBINIT
Keyboard Initialize (function code 1)

This routine initializes the keyboard input driver. This call should be made before you start keyboard input. It clears all previous keystrokes.

Entry Conditions

A =	1

Exit Conditions

NZ =	Error
A =	Error Code

KBCHAR
Keyboard Character (function code 4)

This routine gets one character from the keyboard. The routine returns immediately either with or without a character in register B.

The BREAK key is masked from the user — it will never be returned, since it is intercepted by the System. If a SETBRK routine is enabled, control passes to the processing program (see SETBRK) whenever BREAK is pressed. Otherwise, control pass to TRSDOS READY.

Entry Conditions

A =	4

Exit Conditions

B =	Character found, if any. Only codes within the range [0,127] can be returned. If no character is returned, B is unchanged.
NZ =	No character present
A =	Error Code

KBLINE
Keyboard Line (function code 5)

This routine inputs a line from the Keyboard into a buffer, and echoes the line to the Display, starting at the current cursor position. As each character is received and displayed, the cursor advances to the next position (Scroll Mode – see "Video Display" section on the following pages).

On entry to this routine, the input buffer is filled with periods, and these periods appear on the Display, indicating the length of the input field for the operator's convenience.

The line ends when a carriage return is typed or when the input buffer is filled. A carriage return and erase-to-end-of-screen are always sent to the Display upon termination of line input; a carriage-return is stored only if the Operator actually pressed ENTER .

Entry Conditions

(HL) =	Start of input buffer.
B =	Maximum number of characters to receive, 0 < B
A =	5

Exit Conditions

B =	Actual number of characters input, including carriage return
C =	0 if input buffer was filled without carriage return. If line ended with a carriage return, then C = X'0D'.

Control codes **not** listed below are placed in the buffer and represented on the display with ± symbols,

KEY	HEX CODE	FUNCTION
BACKSPACE	08	Backspaces the cursor and erases a character.
ENTER	0D	Terminates line. Clears trailing periods on display but not in buffer.
CTRL W	17	Fills remainder of input buffer with blanks, blanks remainder of Display line.
CTRL X	18	Fills remainder of input buffer with blanks, blanks to end of Display.
ESC	1B	Reinitializes input function by filling input buffer with periods and restoring cursor to original position.
←	1C	Backspaces the cursor to allow editing of line. Does not erase characters.
→	1D	Advances the cursor to allow editing of line. Does not erase characters.

Table 4.5. Received Control Codes, code < 32

Video Display

Supervisor Calls described in this section:

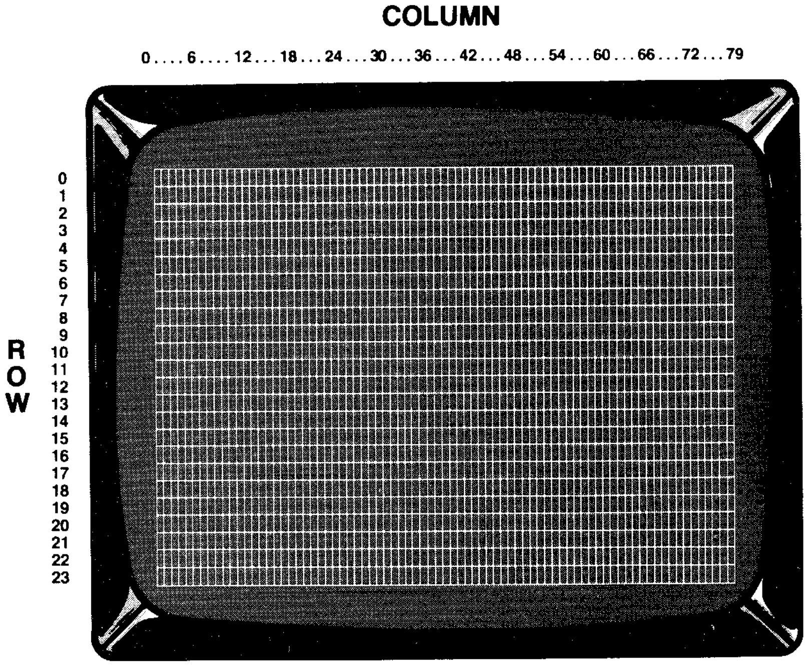
FUNCTION CODE	NAME	PURPOSE
7	VDINIT	Initializes Display
8	VDCHAR	Sends a character, Scroll Mode
9	VDLINE	Sends a line, Scroll Mode
10	VDGRAF	Sends characters, Graphics Mode
11	VDREAD	Reads characters, Graphics Mode
12	VIDKEY	Displays message, and gets line from KB
26	CURSOR	Turns cursor on or off
27	SCROLL	Sets number of lines at top of display which are not scrolled
94	VIDRAM	Video/RAM Transfer

Table 4.6. Video Display Supervisor Calls

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 80 by 24 matrix, as illustrated below:



DISPLAY POSITIONS, GRAPHICS MODE

Note: The Display has two character sizes: 80 characters per line and 40 characters per line. The illustration above shows the 80 character per line mode.

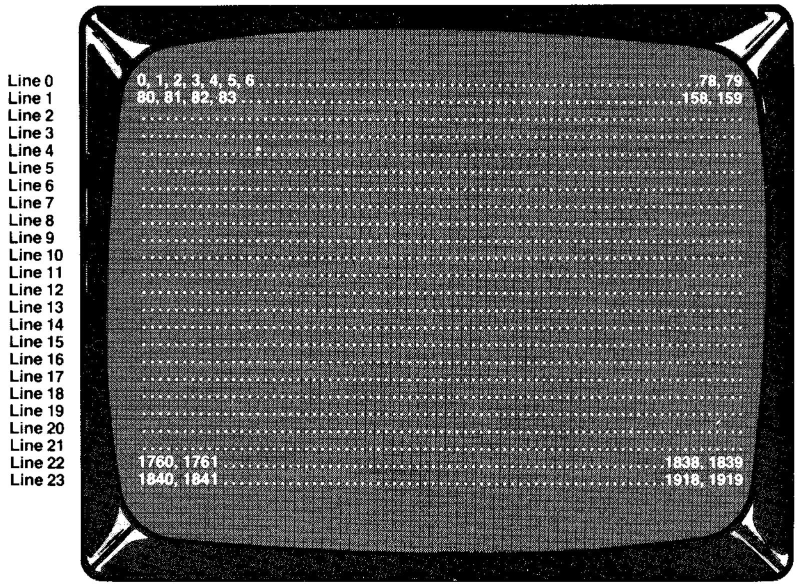
Each time an acceptable display character is received, it is displayed at the current cursor position (which is set on entry to the Graphics Mode routines). Before displaying the next character, the cursor position is advanced, as follows:

- 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 0 on the next row.

Cursor motion works the same way in all directions. For example, if the cursor is at Row 23, Column 40, and the X'FF' (graphics-down) code is received, the cursor wraps around to Row 0 in the same column.

Scroll Mode

In the Scroll Mode, the Display can be thought of as a sequence of 1920 display positions, as illustrated below:



DISPLAY POSITIONS, SCROLL MODE

Note: The Display has two character sizes: 80 characters per line and 40 characters per line. The illustration above shows the 80 character per line mode.

In the scroll mode, each time an acceptable display character is received, it is displayed at the current cursor position, and 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 when the bottom line is filled, the entire Display is "scrolled":

1. Line 0 is deleted
2. Lines 1-23 are moved up on one line
3. Line 23 is blanked
4. The cursor is set to the beginning of line 23.

Note: Lines 0 to 22 on the Display can be protected from scrolling via the SCROLL function call.

VDINIT
Video Initialization (function code 7)

Call this initialization routine once before starting any I/O to the Display. It blanks the screen and resets the cursor to the top left corner (position 0 in the Scroll Mode illustration).

Entry Conditions

B = Character size switch. If B=0 then size is set to 40 characters/line. Otherwise, size is set to 80 characters/line.

C = Normal/Reverse switch. If C = 0 then sets Reverse mode, black on white background. Otherwise sets Normal mode, white on black background.

A = 7

Exit Conditions

NZ = Error

A = Error Code

VDCHAR
Video Character (function code 8)

This routine outputs a character to the current cursor position. It is a Scroll Mode routine, as described above.

Control Codes not listed below are ignored.

KEY	HEX CODE	FUNCTION
F1	01	Blinking cursor on.
F2	02	Cursor off.
CTRL D	04	Turns on steady cursor.
BACKSPACE	08	Moves cursor back one position and blanks the character at that position.
TAB	09	Advances cursor to next tab position. Tab positions are at 8-character boundaries, 8, 16, 24, 32, . . .
CTRL J	0A	Line feed—cursor moves down to next row, same column position.
CTRL K	0B	Positions cursor to beginning of previous line. Cursor will not move into the scroll-protected area
ENTER	0D	Moves cursor down to beginning of next line.
CTRL N	0E	Turns dual routing on.
CTRL O	0F	Turns dual routing off.
CTRL T	14	Homes cursor to the first non scroll-protected position.
CTRL W	17	Erases to end of line, cursor doesn't move.
CTRL X	18	Erases to end of screen, cursor doesn't move.
CTRL Y	19	Sets Normal Display mode (white on black). Remains Normal until reset by programmer.
CTRL Z	1A	Sets Reverse Display mode (black on white). Remains Reverse until reset by programmer.
ESC	1B	Erases screen and homes cursor (position 0).
←	1C	Moves cursor back one position.
→	1D	Moves cursor forward one position.
↑	1E	Sets 80 character/line and clears Display.
↓	1F	Sets 40 character/line and clears Display.

Table 4.7. Received Control Codes, Code< X'20'

Entry Conditions

B = ASCII code for character to be output to the Display; character codes **must** be in the range [0,127]

A = 8

Exit Conditions

NZ = Error

A = Error Code

VDLINE
Video Line (function code 9)

This routine writes a buffer of data to the Display, starting at the current cursor position. It is a Scroll Mode routine.

The buffer should contain ASCII codes in the range [0,127].

Received control codes, code X'20', are handled as with VDCHAR.

Entry Conditions

(HL) = Beginning of the buffer containing characters to be set to the Display

B = Number of characters to be sent

C = End of line character. This character will be sent to the Display after the buffer text

A = 9

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 always set to the position following the last character displayed.

VDGRAF
Video Graphics (function code 10)

This function displays a buffer of characters, starting at a specified row and column. 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 0 through X'1F', and are pictured in the **Operator's Manual.** Codes X'20' through X'7F' are displayed as standard ASCII characters.

In addition, several special control codes are available:

HEX CODE	FUNCTION
F9	Sets Normal (white on black) mode. Cursor does not advance. Mode reverts to previous state after completion of VDGRAF call.
FA	Sets Reverse (black on white) mode. Cursor does not advance. Mode reverts to previous state after completion of VDGRAF call.
FB	Homes cursor (Row 0, Column 0).
FC	Moves cursor back one space. Col. = Col-1. When column equals 0, cursor wraps to Col. 79 on the preceding row.
FD	Moves cursor forward one space. Col. = Col. + 1. When column equals 79, cursor wraps to Col. 0 on the next row down.
FE	Moves cursor up one row. Row = Row-1. "Wraps" to Row 23 when Row = 0.
FF	Moves cursor down one row. Row = Row + 1. "Wraps" up to Row 0 when Row = 23.

Table 4.8. Special Graphics Control Codes

At exit, the cursor is always set to the Graphics position immediately after the last character displayed. If the Buffer length was zero, the cursor is set to position specified in BC registers.

Entry Conditions

B = Row on screen to start displaying the buffer, B<24. If B>23, then B modulo 24* is used as row position.

C = Column on screen to start displaying the buffer. In 80 character/line mode, C < 80. For C > 79, C modulo 80 is used as column position.

In 40 character/line mode, C < 40. For C > 39, C modulo 40 is used as column position.

D = Length of buffer, in range [0,255]

(HL) = Beginning of text buffer. The buffer should contain codes below X'80' or the special control codes above X'F8'. Any value outside these ranges will cause an error.

A = 10

Exit Conditions

NZ => Error (invalid character sent)

A = Error Code

**Modulo*—A cyclical counting system. For modulus *n*, *x* modulo *n* is the integer remainder after division of *x* by *n*. For example, 85 modulo 80=5.

VDREAD Video Read (funtion code 11)

This routine reads characters from the Video Display into a specified buffer. It is a Graphics Mode routine; when it reads past the last column, it wraps back to column 1 on the next row. When it reads past column 79 on row 23, it wraps back to row 0, column 0.

Reverse (black on white) mode characters are read in as ASCII codes just like their Normal counterparts; Reverse mode is indicated when the most significant bit (bit 7) is set.

This routine can also be used just to locate the cursor (see below).

Entry Conditions

B = Row on screen where read starts, B < 24
If B > 23, then B mod 24 is used as row position.

C = Column on screen where read starts
In 80 character/line mode, C < 80
For C >79, C mod 80 is used as column position.
In 40 character/line mode, C < 40
For C > 39, C mod 40 is used as column position.

D = Length of buffer, in range [0,255]. If D = 0, then B and C are ignored. Current cursor position will be returned as row, column in BC register pair.

(HL) = Beginning of text buffer

A = 11

Exit Conditions

BC = Current cursor position, B = row, C = column. CURSOR position at exit is the same as at entry – VDREAD does not change it.

NZ = Error

A = Error Code

VIDKEY
(function code 12)

This routine sends a prompting message to the Display and then waits for a line from the Keyboard. It is a Scroll Mode routine, combining the functions of VDLINE and KBLINE.

The routine writes the specified text buffer to the Display, starting at the current cursor position. The text buffer must contain codes<X’80’. Refer to VDLINE for a list of Received Control Codes and other details.

After the Video write, the cursor will be positioned immediately after the last character displayed. (To move it to another position, control codes can be placed at the end of the text buffer.)

Next, the routine gets a line from the Keyboard.

Note: Before starting the line input, all previously stored keystrokes are cleared.

Refer to KBLINE for a list of Received Control Codes and other details.

Entry Conditions

(HL) = Beginning of text buffer containing display message

B = Number of characters to be displayed, B in the range [0,255]

C = Length of Keyboard Input field, C in the range [0,255]

(DE) = Beginning of text buffer where Keyboard input will be stored

A = 12

Exit Conditions

NZ = Error (illegal value in display buffer)

A = Error Code

If Z is set (no error), then registers B and C contain:

B = Number of characters input from Keyboard, including carriage return, if any

C = Keyboard Line termination. If C = 0, then input buffer was filled. Otherwise C = control character that terminated the line (carriage return).

If Z is not set (error), the registers B and C contain:

B = Number of characters not displayed, including the one causing the error

C = Character causing the error

CURSOR (function code 26)

This routine turns the blinking cursor on or off. The System still keeps track of the current cursor position, whether it is on or off.

Entry Conditions

B = Function Switch. If B = 0 then cursor will be turned off. If B< >0 then cursor will be turned on.

A = 26

SCROLL (function code 27)

This routine lets you protect a portion of the Display from scrolling. From 0 to 22 lines at the **top** of the Display can be protected; when scrolling occurs, only lines below the protected area will be changed.

Entry Conditions

B = Number of lines to be protected, in range [0,22].

A = 27

VIDRAM
Transfer Video Display to RAM or Vice-Versa (function code 94)

This command allows a screenful of graphics and/or text to be copied from a RAM buffer to the video display, or vice-versa. The programmer must be aware of the current display mode—80 characters/line or 40 characters/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
00-1F	Normal Graphics
20-7F	Normal ASCII Text
80-9F	Reversed Graphics
A0-FF	Reversed ASCII Text

Entry Conditions

A = 94

B = Function Code
If B = 0, copy from RAM to video
If B ≠ 0, copy from video to RAM

(HL) = RAM Buffer. Start of buffer must be above X'2800'; end of buffer must be below X'F000'.
If video is in 80 cpl mode, buffer must be 80 * 24 = 1920 bytes long.
If video is in 40 cpl mode, buffer must be 40 * 24 = 960 bytes long.

Exit Conditions

The cursor position is unchanged by this SVC.

Line Printer

Supervisor Calls described in this section:

FUNCTION CODE	NAME	PURPOSE
17	PRINIT	Initializes the Line Printer Driver.
18	PRCHAR	Sends a character to the Printer.
19	PRLINE	Sends a line to the Printer.
95	PRCTRL	Controls printer operations.

Table 4.9. Line Printer Supervisor Calls

PRINIT
Printer Initialization (function code 17)

This routine initializes the printer driver only. It does not advance the printer paper, and it does not check the printer status. It will operate whether or not the printer is on-line.

Entry Conditions

A = 17
B = Page length
C = Printed lines per page
D = Maximum number of characters per line

Register C must always be less than or equal to B.

If B = C ≠ O, TRSDOS will not do any automatic top of form. However, it will translate a form feed code X'OC' or X'OB' into the correct number of carriage returns or line feeds to get the paper to the top of the next form.

If B = O, C must also equal O. Similarly, if C = O then B must also equal O. In such a case, form feeds X'OC' and vertical tabs X'OB' are not translated but are sent directly to the printer.

If D = O, TRSDOS will not translate tabs X'09' as one to eight spaces. It will send the character directly to the printer. It will continue to update the character count, with tab X'09' counting as a single character.

Exit Conditions

The current character count and current line count are set to zero. TRSDOS assumes that the paper is already at the top of form when this routine is called.

''Dummy'' and ''Transparent'' printer modes are both returned to ''Normal''. (See SVC PRCTRL.) The serial/parallel and auto linefeed options are unchanged by PRINIT.

User Notes

1. TRSDOS maintains current line-on-page and current character-in-line counts regardless of which initialization settings are used. See SVC PRCTRL for further details on these counts.
2. The printer need not be ready when PRINIT is called.
3. Whenever TRSDOS is started (reset or power-on), this initialization is done automatically, with the following parameters:

Parameter (PRINIT register)	**Value**
Page Length (B =)	66
Printed lines/page (C =)	60
Max. characters/line (D =)	132

The other options set during initialization are:
Parallel printer
Non-auto line feed
Non-dummy
Non-transparent

PRCHAR
Print Character (function code 18)

This routine sends one character to the Printer.

Note: Most printers do not print until their buffer is filled or a carriage return is received.

Entry Conditions

B = ASCII code for character to send

A = 18

Exit Conditions

NZ = Error
A = Error Code

Notes

While the serial printer option is selected, allowing printer output to Channel B, two INPUT characters are recognized from Channel B. These will affect all serial printer output operations:

ASCII Name	Hex Code	Result
DC3, "CTRL-S"	13	Pause printing
DC1, "CTRL-Q"	11	Resume printing

Certain codes are normally intercepted by TRSDOS and are not sent directly to the printer. There are several ways to override some or all of these character translations. See SVC PRINIT and SVC PRCTRL.

Table of Intercepted Codes

ASCII Name	Hex Code	Result to Printer
Tab	09	From one to eight spaces are sent to provide a tab function.
Vertical Tab	0B	Same as form feed below.
Form Feed	0C	TRSDOS sends enough carriage returns or line feeds to the printer to advance the paper to the next top of form.
Carriage Return	0D	When the current line is empty (no characters printed since the last carriage return or line feed), this is translated as a line feed to allow correct operation of Radio Shack printers. In the auto line feed mode, X'0A' is sent after every X'0D'.
Special	8D	TRSDOS sends a carriage return to the printer. In the auto line feed mode, using this code allows you to send a carriage return *without* a line feed.

PRLINE
Print Line (function code 19)

This routine sends a line to the Printer. The line can include control characters as well as printable data. See PRCHAR for a list of intercepted codes.

See PRCHAR for addition notes on serial output and on special character handling.

Entry Conditions

(HL) = Start of text buffer containing data and controls to send to Printer

B = Length of buffer (number of characters to send)

C = Control Character (any character) to send after last character in buffer

A = 19

Exit Conditions

NZ = Error

A = Error Code

PRCTRL
Control Printer Operations (function code 95)

This routine lets you select various printer options and check the status of printer-related functions.

Note: If you are using the spooler's capture function, the captured data is sent directly to the capture-file—regardless of what control settings are selected. If you are using the spooler's print function, the printed data will be interpreted according to the currently selected control settings. See SPOOL for details.

Entry Conditions

A = 95
C = Used with certain options (see option list below)
B = Option code:

Contents of B (Decimal)	Option (details are given later)
0	Get printer status only (see Exit Conditions)
1	Select serial printer driver (you must initialize channel B first)
2	Select parallel printer driver
3	Reset current line count; register C contains new count
4	Reset current character count on current line; register C contains the new count
5	Begin transparent mode
6	End transparent mode
7	Begin dummy mode
8	End dummy mode
9	Begin auto line-feed after carriage return
10	End auto line-feed after carriage return

Exit Conditions

NZ = Error Occurred.

Z = SVC completed without error.

B = Page Length

C = Maximum number of lines to be printed on each page

D = Maximum number of characters to be printed on each line

E = ASCII ''P'' or ''S'' (Indicates which printer option, Parallel or Serial Printer, is selected.)

H = Number of characters printed on current line since last carriage return

L = Number of lines printed since last top of form

Explanation of Options

Serial/Parallel Printer Option

While the serial printer option is selected, allowing printer output to Channel B, two *input* characters are recognized from Channel B. These will affect all serial printer output operations:

ASCII Name	Hex Code	Result
DC3, "CTRL-S"	13	Pause printing
DC1, "CTRL-Q"	11	Resume printing

Line count/Character Count

TRSDOS maintains a single line counter and a single character counter. These are updated for either serial or parallel printing. TRSDOS 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. There is no examination of content. The line and character counts are not updated.

Normally, TRSDOS "intercepts" certain control characters and interprets them. This allows TRSDOS to provide printer-related features which may not be available from the printer.

For example, tabs (X'09') are intercepted so that TRSDOS can send the appropriate number of spaces to provide the tab function. Form feeds (X'OC' or X'OB') are intercepted so that TRSDOS may advance the paper to the top of the next form.

Code translation may be overridden individually by special settings of the printer initialization values. See SVC PRINIT 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 count and character count remain unchanged.

Auto Line Feed

Normally, the TRSDOS printer driver does not print line feeds after carriage returns. This is because most Radio Shack printers do an automatic line-feed after every carriage return.

If your printer does not perform automatic line-feeds after carriage returns, you may enable the TRSDOS auto line-feed function.

While the function is enabled, a line-feed will be output 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 latter code will return the carriage without advancing the paper.

Precedence of Options

The following table shows which options have priority over others. Options are listed in order of descending priority.

Spool Capture-Mode (See SPOOL command)
Dummy Mode
Auto Line-Feed after Carriage Return
Transparent Mode
Normal Mode

User Notes

1. If you are using the spooler's capture function, the captured data is sent directly to the capture-file—regardless of what printer control settings are selected. If you are using the spooler's print function, the printed data will be interpreted according to the currently selected forms control settings. See SPOOL for details.

2. You can operate with two printers, by switching between the serial and parallel output modes. One printer can use the automatic control features of TRSDOS (auto form feed, etc.), while the other printer is controlled by the program. This is necessary since *only one set of control counters is maintained.*

One possible approach: Select the transparent operation before you switch to the serial printer and select normal operation before you switch to the parallel printer. That way, the line and character count will remain accurate for the parallel printer. It will be up to your program to control the serial printer.

File Access

Supervisor calls described in this section:

Function Code	Name	Function
33	LOCATE	Returns the current record number.
34	READNX	Gets next record (Sequential Access).
35	DIRRD	Reads specified record (Direct Access).
40	OPEN	Sets up access to new or existing file.
41	KILL	Deletes the file from the directory.
42	CLOSE	Terminates access to an Open file.
43	WRITNX	Writes next record (Sequential Access).
44	DIRWR	Writes specified record (Direct Access).
47	RENAME	Renames a file.
51	WILD	Compares a file specification with a wild-card specification.
53	RAMDIR	Gets directory information into RAM.
58	FILPTR	Gets pointers of an open file.

Table. 4.10. File Access Calls

LOCATE
(function code 33)

This function returns the number of the current record, i.e., the number of the last record accessed. You can use this call only with Fixed Length Record files.

Entry Conditions

(DE) =	Data Control Block for currently Open file (see OPEN)
HL =	Reserved for use in later versions of TRSDOS
A =	33

Exit Conditions

BC =	Current Record Number
NZ =	Error
A =	Error Code

READNX
Read Next Record (function code 34)

This routine reads the next record after the current record. (Current record is the last record accessed.) If the file has just been Opened, READNX will read the first record.

Entry Conditions

(DE) =	Data Control Block for currently Open file (see OPEN)
HL =	Reserved for use in later versions of TRSDOS
A =	34

Exit Conditions

NZ = Error

A = Error Code

Upon return, your record is in the Record Area pointed to by RECADR in the parameter list, or, if RL=256 and record type is Fixed, your record is in the area pointed to by BUFADR.

DIRRD
Direct Read (function code 35)

This routine reads the specified record, allowing direct access.

Note: With VLR files, you can only use it to read the first record or to read the end of file.

Entry Conditions

(DE) = Data Control Block for currently Open file (see OPEN)
BC = Desired record number
BC = 0 means position to beginning of file
BC = X'FFFF' means position to end of file

HL = Reserved for use in later versions of TRSDOS

A = 35

Exit Conditions

NZ = Error

A = Error Code

Upon return, your record will be in the Record Area pointed to by RECADR in the parameter list, or, if RL = 256 and record type is Fixed, your record is in the area pointed to by BUFADR.

OPEN
(function code 40)

This one call handles both the creation and opening of files.

A given file can only be Open under one Data Control Block at a time. Because of the versatile file processing routines, this one DCB is sufficient to handle the various I/O applications.

Entry Conditions for OPEN

(DE) =	60-byte Data Control Block (see below)
(HL) =	11-byte Parameter List (see below)
A =	40

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, as described below.

Data Control Block (60 bytes)

The Data Control Block (DCB) is used by the System for file access bookkeeping. You will also use it to pass the filespec for the file you want to Open, as follows:

Before calling OPEN, place the filespec at the beginning of the DCB, followed by a carriage return. See **File Specification** in Section 1 of this manual.

For example ($ signifies a carriage return):

CONTENTS OF FIRST BYTES OF DCB BEFORE OPEN

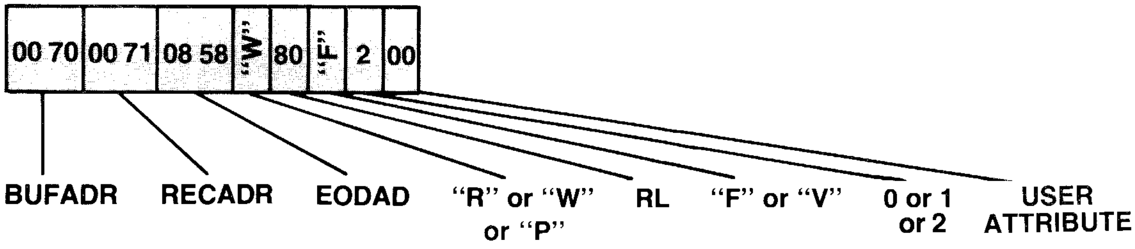
```
FILENAME/EXT.PASSWORD:d(DISKETTE)$
```

While a file is Open, the filespec is replaced with information used by the System for bookkeeping. When the file is Closed, the original filespec (except for the password) will be put back into the DCB.

Important Note: Do not ever modify any portion of the DCB while the file is Open. If you do, the results will be unpredictable.

Parameter List (11 bytes)

CONTENTS OF PARAMETER LIST (SAMPLE)



BUFADR (Buffer Address). This two-byte field must point to the beginning of the Buffer Area.

The Buffer Area is the space TRSDOS will use to process all file accesses. If spanned records are possible, you must reserve 512 bytes. If no spanning is possible, reserve only 256 bytes.

With Fixed Length Record files, spanning is only required 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 exactly, 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 will have to be spanned. In this case, you will need to reserve 512 bytes.

With Variable Length Record files, you must always reserve 512 bytes for processing. This is because spanning may be required, depending on the lengths of the individual records in the file.

RECADR (Record Address). This two-byte field must point to the beginning of the Record Area.

For disk reads, this is where TRSDOS will place the record. For disk writes, this is where you put the record to be written.

Exception: For FLR files with a record length of 256, this address is not used. Your record will be in the buffer area pointed to by BUFADR.

For Fixed Length Record files with record length not equal to 256, this buffer should be the same size as the record length. For Variable Length Files, this area should be long enough to contain the longest record in the file (including the length-byte). If you are not sure what the longest record will be, reserve 256 bytes.

EODAD (End of Data Address). This two-byte field can be used to give TRSDOS a transfer address to use in case the end of file is reached during an attempted read; control will transfer to the EODAD address if the end of file is reached during a read operation. If EODAD = 0 and end of file is reached, the SVC will simply return with the end of file error code in register A.

ACCESS Type

The seventh byte specifies the type of access that is desired. It can be any of the following:

ASCII **Character**	**Meaning**
R	Read-only
W	Read/Write a data file
P	Write a program file

The last option is new. It allows a programmer to create a file which can be loaded and executed directly from TRSDOS. The TRSDOS DUMP command also creates program files, and should be used except for unusual cases. This file will appear in the DIR with the "P" attribute.

It is the programmer's responsibility to ensure that the program bytes are written in the correct format and sequence, as defined in the section, Programming with TRSDOS.

0 or 1 or 2 (Creation Code). This one-byte field contains a binary number 0, 1 or 2.

CODE	MEANING
0	Open the file only if it already exists. Do not create a new file in directory. Record Length and end of file are **not** reset.
1	Create a new file only; do not Open an existing file. Record Length and end of file are set at Open time.
2	Open existing file; if file not found, create it (record length and end of file will be reset).

Table 4.11. File Creation Code

If a new file is being created (creation code = 1 or 2) and the file does not already exist, TRSDOS will use the first write-enabled drive with at least one free granule.

RL (Record Length). This one-byte field specifies the record length to be used. Zero indicates a record length of 256. For Variable Length record files, this field is ignored. If the file already exists, and the Creation Code is 0, the System will supply the correct RL value, regardless of what you put there.

"V" or "F" (Variable or Fixed Length). This one-byte field contains either an ASCII "V" for Variable or an ASCII "F" for Fixed. Once a file has been created, this attribute cannot be changed. If the file already exists, and the Creation Code is 0, the System will supply the correct "F" or "V" value, regardless of what you put in the parameter list.

USER ATTRIBUTE — Byte

The 11th byte in the open parameter list contains a USER ATTRIBUTE byte.

The USER ATTRIBUTE byte may be used by the Z-80 programmer to mark certain types of data files. Values 0-31 have been reserved for use by Radio Shack programs. Values 32-255 are available for user definition. TRSDOS will not examine the USER ATTRIBUTE byte; it is solely a convenience for the programmer.

Note: All files created under versions prior to 2.0 should have a USER ATTRIBUTE value of zero. All files created with the CREATE command will have a USER ATTRIBUTE value of zero.

The USER ATTRIBUTE is assigned when a file is created or opened with creation code 1 or 2. When a file is opened with creation code 0, the file's previously assigned user attribute will not be changed; it will be stored in the 11th byte of the parameter list as an output parameter.

KILL
(function code 41)

This routine deletes the specified file from the directory. A file must be Closed before it can be Killed.

Entry Conditions

(DE) =Data Control Block, contaning standard TRSDOS filespec (see illustration in description of OPEN)

A = 41

Exit Conditions

NZ = Error

A = Error Code

CLOSE
(function code 42)

This routine terminates access to the file. If there are records in the Buffer Area not yet written, they will be written at this time.

Entry Conditions

(DE) = Data Control Block for currently Open file

A = 42

Exit Conditions

NZ => Error

A = Error Code

Upon return, the filespec (except for the password) will be put back into the DCB.

WRITNX
Write Next Record (function code 43)

This routine writes the next record after the last record accessed; that is, it writes sequentially. If WRITNX is the first access after the file is Opened, the first record will be written.

Entry Conditions

(DE) = Data Control Block for currently Open file

HL = Reserved for use in later versions of TRSDOS

A = 43

Before calling WRITNX, put your record in the record area pointed to by RECADR in the Parameter List, or, if RL=256 and record type is Fixed, your record is in the area pointed to by BUFADR.

Exit Conditions

NZ = Error

A = Error Code

DIRWR
Direct Write (function code 44)

This routine writes the specified record. It writes your record into the specified record position of the file.

Note: For VLR files, you can only position to the beginning or end of file. When you write to a VLR file, the end of file is reset to the last record written.

Entry Conditions

(DE) =	Data Control Block for currently Open file
BC =	Record number you want to write BC = 0 means write first record in file BC = X'FFFF' means write record at end of file
(HL) =	Reserved for use in later versions of TRSDOS
A =	44

Before calling DIRWR, put your record into the Record Area pointed to by RECADR in the Parameter list, or, if RL=256 and record type is Fixed, put record in area pointed to by BUFADR.

Exit Conditions

NZ =	Error
A =	Error Code

RENAME
Rename a File (function code 47)

This routine changes the name and/or extension of a file. The password cannot be changed.

Entry Conditions

A = 47
(HL) = Old file specification, followed by a carriage return. If file is password-protected, password must be included.
(DE) = New file specification followed by a carriage return. A password cannot be used here (the old one will be retained).

Exit Conditions

NZ	Error occurred
A	Error Code

WILD
Wild Card Parser (function code 51)

This routine compares a file specification with a wild-card specification. For details on wild-card specifications, use the Index.

Entry Conditions

B = Function Code:

Contents of B	Result
0	Set wild-card mask.
1	Compare file specification with mask
2	Combines a separate name and extension into a file specification

When B = 0, the other register contents are:

(HL) = Wild card mask, *followed by a carriage return*. A mask is a file specification with asterisks 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 with the extension /BAS.

*LST masks all files except those which have a filename ending in LST and no extension.

When B = 1, the other register contents are:

(HL) = File specification to be checked against the mask; file specification must be terminated with a carriage return.

When B = 2, the other register contents are:

(HL) = An 11-byte buffer defined as follows: First comes an eight-byte name field. Then comes a three-byte extension field. Both must be left justified with trailing spaces as required to fill the field. For example:
NAME b̸b̸b̸b̸ EXT
The symbol "b̸" represents a blank space and is used only where needed for emphasis or illustration.

(DE) = A 13-byte destination buffer to hold the compressed file specification contained in (HL).

Exit Conditions

After entry with B = 0, the exit conditions are:

NZ = Invalid mask specification

After entry with B = 1, the exit conditions are:

NZ = File does not match mask or no mask has been set

After entry with B = 2, the exit conditions are:

(DE) = 13-byte buffer containing a filename/extension followed by a carriage return. The filename/extension is created from the 11-byte source text (HL) on entry with B = 2.

Following the example given above, (DE) would contain: NAME/EXT followed by a carriage return

User Notes

Calling any other TRSDOS SVC may clear the mask. Therefore you should use this procedure:

1. Get the file specification(s) to be checked.
2. Set mask by calling WILD with B = 0
3. Compare the file specification with the mask by calling WILD with B = 1
4. Repeat Step 3 until all file specifications have been checked.
5. Each time you call another TRSDOS SVC, you may have to reset the mask.

RAMDIR
Get Diskette Directory/Free Space into RAM (function code 53)

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

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

Entry Conditions

A = 53
B = Drive Number, binary 0, 1, 2, or 3
C = Function switch:

Contents of C	Result
0	Gets entire directory into RAM in the format described below.
1-96	Gets one specified directory record into RAM (if it exists) in the format described below.
255	Gets free-space information in the format described below.

(HL) = Buffer area:

If C = 0, compute buffer size by this formula:
Number of bytes required = 34 * (# of non-system files) + 1.

On a data diskette, there can be a maximum of 96 non-system files; therefore the largest possible directory would require a buffer size of 3265 bytes.

If C = 1-96, buffer must be 34 bytes long.
If C = 255 (free space info desired), buffer must be 4 bytes long.

Exit Conditions

NZ = Error occurred. Register A contains TRSDOS error code.
Z = No error. (HL) = directory or free-space info.

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 (CR)	(CR) represents a carriage return. Trailing blanks are added as required to fill 15 bytes.
17	"F" or "V"	Indicates Fixed- or Variable-length records.
18	LRL	Logical Record Length one byte binary 0-255. 0 implies LRL = 256 or VLR file.
19	# of extents	One byte binary 0-16, Null files = 0.
20-21	# of sectors allocated	Binary 0-65535 in LSB-MSB sequence; null files = 00.
22-23	# of sectors used in storage of data	Binary 0-65535 in LSB-MSB sequence; null files = 00.
24	EOF byte (position of first byte of last record written	Binary 0-255, 0 implies first byte in last sector.
25-26	# of records written	Binary 0-65535 in LSB-MSB sequence; null files = 00.
27	User attribute byte	Assigned when file was created.
28	Protection level	Binary 0-7.
29-31	Date created	Binary year, month, day.
32-34	Date last updated	Binary year, month, day.

When an entire directory is given (C = 0 on entry), a "#" sign instead of the expected ":" marks the end of the directory. When C = 1-96, record always ends after 34th byte without a trailing "#".

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.

FILPTR
Get Pointers of a Currently Open File (function code 58)

This routine provides information on any user file that is currently open.

Entry Conditions

A = 58
(DE) = Data Control Block (DCB) defined when file was opened.

Exit Conditions

NZ = Error occurred. Register A contains TRSDOS error code.

Z = No error. The following registers are set up:
  B = Which drive contains the file (binary 0, 1, 2, or 3)
  C = Position of file in the diskette directory (binary 1-96)

User Notes

This routine is especially intended for use in conjunction with the SVC RAMDIR. After opening a file, the programmer 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 pertinent information about that file (current file space allocated/used, protection level, etc.).

Computational

Supervisor calls described in this section:

Function Code	Name	Function
6	DELAY	Provides a delay-loop
20	RANDOM	Provides a random number, range [0,254]
21	BINDEC	Converts binary to ASCII-coded decimal, and vice versa
22	STCMP	Compares two text strings
23	MPYDIV	Performs 8 bit * 16 bit multiplication and 16 bit / 8 bit division
24	BINHEX	Converts binary to ASCII-coded hexadecimal, and vice-versa
28	LOOKUP	Searches through a table.
45	DATE	Sets or returns the time and date.
48	PARSER	Finds the alphanumeric parameter field in a text string
49	STSCAN	Looks for a specified string inside a text buffer
56	SORT	Sorts a list in RAM.

Table 4.12 Computational Supervisor Calls

DELAY (function code 6)

This routine provides a delay routine, returning control to the calling program after the specified time has elapsed.

Entry Conditions

BC = Delay Multiplier. If BC = 0, then delay time will be 426 milliseconds. If BC > 0, then delay time will be: 6.5 * (BC − 1) + 22 microseconds.

A = 6

RANDOM (function code 20)

This routine returns a random one-byte value. To extend the cycle of repetition, the instantaneous time/date are used in generating the number.

You pass the routine a limit value; the value returned is in the range [0,limit−1]. For example, if the limit is 255, then the value returned will be in the range [0,254].

Entry Conditions

B = Limit value

A = 20

Exit Conditions

C = Random number
For B > 1, number returned is in range [0,B−1]
For B = 0 or 1, number returned = 0

BINDEC
Binary/Decimal (function code 21)

This routine converts a two-byte binary number to ASCII-coded decimal, and vice versa. Decimal range is [0,65535].

Entry Conditions

B = Function Switch:
If B = 0, then convert binary to ASCII decimal
If B is not 0, then convert ASCII decimal to binary

Contents of other registers when B = 0, binary to decimal:

DE = Two-byte binary number to convert

(HL) = 5-byte area to contain ASCII-coded decimal value upon return. The field will contain decimal digits (X'30'-X'39'), leading zeroes on the left as necessary to fill the field; for example, the number 21 would be:

00021

Contents of other registers when B is not 0, decimal to binary:

(HL) = 5-byte area containing ASCII decimal value to be converted to binary

A = 21

Exit Conditions

(HL) = Decimal value

DE = Binary value

STCMP
String Comparison (function code 22)

This routine compares two strings to determine their collating sequence.

Entry Conditions

(DE) = First string

(HL) = Second string

BC = Number of characters to compare

A = 22

Exit Conditions

Status bits indicate results, as follows:

Z flag set indicates strings are identical.

NZ indicates strings not identical.

Carry flag set indicates first string (pointed to by DE) precedes second string (pointed to by HL) in collating sequence.

Other register contents:

A = First non-matching character in first string

When strings are not equal, you can get further information from the prime registers, as follows:

HL' = Address of first non-matching character in second string

DE' = Address of first non-matching character in first string

BC' = Number of characters remaining, including the non-matching character

MPYDIV
Multiply Divide (function code 23)

This routine does multiplication and division with one 2-byte value and one 1-byte value.

Entry Conditions

B = Function Switch:
If B = 0 then multiply
If B not 0 then divide
A = 23

For multiplication:
HL = Multiplicand
C = Multiplier

For division:
HL = Dividend
C = Divisor

Exit Conditions

HL = Result (product HL * C or quotient HL/C)

A = Overflow byte (multiplication only)

C = Remainder (division only)

Status bits affected by division:
Carry flag set if dividing by zero. Divide not attempted.
Z flag set only if the quotient is zero.

Status bits affected by multiplication:
Carry Flag set if overflow.
Z flag set only if result is zero.

LOOKUP
Look Up in a Table (function code 28)

This routine performs a lookup function on a table of three-byte entries. Each entry looks like this:

Byte 1	Bytes 2-3
Search Key	Data, e.g., an address in LSB, MSB sequence

At the end of the table, you must place one-byte X‘FF’. Since X‘FF’ is the table terminator, your search keys must be between X‘OO’ and X‘FE’.

Given a one-byte search argument, the routine locates the first matching entry in the table. The routine uses a sequential search algorithm.

Entry Conditions

(HL) = First byte of table
B = Search argument
A = 28

Exit Conditions

NZ = Search argument not found

If Z is set, then

HL = Data. H contains byte three, L contains byte two.

User Notes

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.

BINHEX
Binary/Hexadecimal (function code 24)

This routine converts a two-byte binary number to ASCII-coded hexadecimal, and vice versa. Hexadecimal range is [0,FFFF].

Entry Conditions

B = Function Switch:
If B = 0, then convert binary to ASCII hexadecimal
If B is not 0, then convert ASCII hexadecimal to binary

A = 24

Contents of other registers when B = 0:

DE = Two-byte binary number to convert

(HL) = 4 byte area to contain ASCII coded hexadecimal value upon return. The field will contain hexadecimal digits with leading zeroes on the left as necessary to fill the field, for example, the number X'FF' would be:
00FF

Contents of other registers when B not 0:

(HL) = 5-byte area containing ASCII hexadecimal value to be converted as described above

A = 24

Exit Conditions

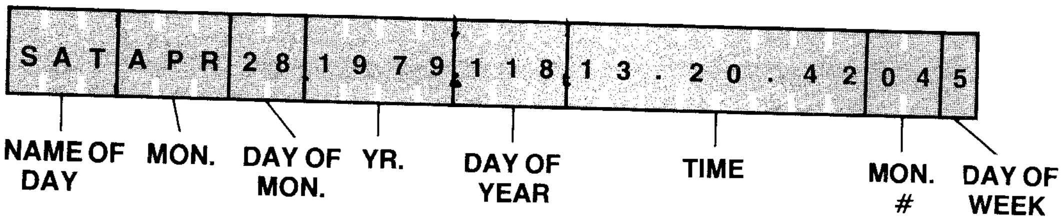
(HL) = Hexadecimal value

DE = Binary value

DATE
(function code 45)

This routine sets or returns the real-time (time and date). The data is returned as a 26-byte ASCII string containing 8 fields.

CONTENTS OF TIME/DATE STRING (SAMPLE)



Example Time/Date string:

SATAPR28197911813.20.42045

Represents the data "Saturday, April 28, 1979, 118th day of the year, 13:20:42 hours, 4th month of the year, 5th day of the week.

Notes: DAY OF WEEK Field: Monday is day 0. The date calculations are based on the Julian Calendar.

Entry Conditions

B = Function Switch

If B = 0 (Get time/date):
(HL) = 26-byte buffer where date/time will be stored

If B = 1 (Set date):
(HL) = 10-byte buffer containing date in this form:
MM/DD/YYYY

If B = 2 (Set time):
(HL) = 8+byte buffer containing time in this form:
HH.MM.SS

PARSER (function code 48)

This general-purpose routine "parses" (analyzes) a text buffer into fields and subfields. PARSER is useful for analyzing TRSDOS command lines including keyword commands, file specifications, keyword options and parameters. It can also be used as a fundamental routine for a compiler or text editor.

By necessity, the description of PARSER is rather long and detailed. In actual use, the routine is as convenient as it is powerful. For example, PARSER is designed to allow repetitive calls for processing a text buffer; on exit from the routine, parameters for the next call are all readily available in appropriate registers.

The routine has pre-defined sets of field-characters and separators; you can use these or re-define them to suit your application.

In general, a field is any string of alphanumeric characters (A-Z, a-z, 0-9) with no embedded blanks. Fields are delimited by separators and terminators, defined below. For example, the line:

```
BAUD=300, PARITY=EVEN WORD=7
```

contains 6 fields: BAUD, 300, PARITY, EVEN, WORD, and 7.

However, a field can also be delimited by paired quote marks:
"field" or 'field'
When the quote marks are used, **any** characters, not just alphanumerics, are taken as part of the field. The quote marks are not included in the field. For example, the line:

```
'DATE(07/11/79)'
```

will be interpreted as one field containing everything inside the quotes. When a quote mark is used to mark the start of a field, the same type of quote mark must be used to mark the end of the field. This allows you to include quotes in a field, for example:

```
"X'FF00'"
```

will be parsed as one field containing **everything** inside the double quotes " ", including the single quote marks ' '.

A separator is any non-alphanumeric character. PARSE will always stop when a separator is encountered, **except** when the separator is a blank (X'20'). Leading and trailing blanks are ignored. After trailing blanks, PARSE stops at the beginning of the next field, or on the first non-blank separator.

You can also define terminators, which will stop the parse regardless of whether a field has been found. Unless you specifically define these, PARSE will only stop 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.

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, explained later.

Entry Conditions

(HL) = Text buffer

(DE) = List Address Block
DE = 0 indicates no lists are to be used
C = Maximum length of parse
A = 48

Exit Conditions

(HL) = Field-position:
(HL) = First byte of field, if a delimited field was found
(HL) = Terminator or non-blank separator if no field was found
(HL) = Last byte of buffer if parse reached maximum length

B = Actual lengths of field, excluding leading and trailing blanks

A = Character preceding the field just found. If B = 0, A = X'FF'

C = Number of bytes remaining to parse after terminator or separator. Note that trailing blanks have been parsed.

D = Separator or terminator at end of field. If D = X'FF' then parse stopped without finding a non-blank separator or terminator.

E = Displacement pointer for next parse call. Add E to HL to get:
a) Beginning address of next field, or
b) Address of byte following the last byte parsed. Note that if parse reached maximum length, then E + HL = Address following end of text buffer. If parse did not reach maximum length, and E = 1, then E + HL = Address following separator or terminator.

Status bits (F register) affected when parse did **not** reach maximum length:

Z flag:
Z (set) if parse ended with a separator
NZ (not set) if parse ended with a terminator

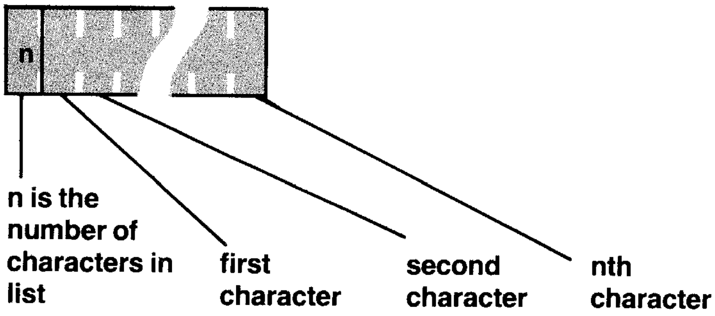
C flag:
C (set) if there were trailing blanks between end of field and next non-blank separator or terminator
NC (not set) if there were no trailing blanks

List Address Block

The List Address Block is six bytes long and contains two-byte addresses (lsb,msb) for three change-lists:

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, i.e., alphanumerics to be interpreted as separators

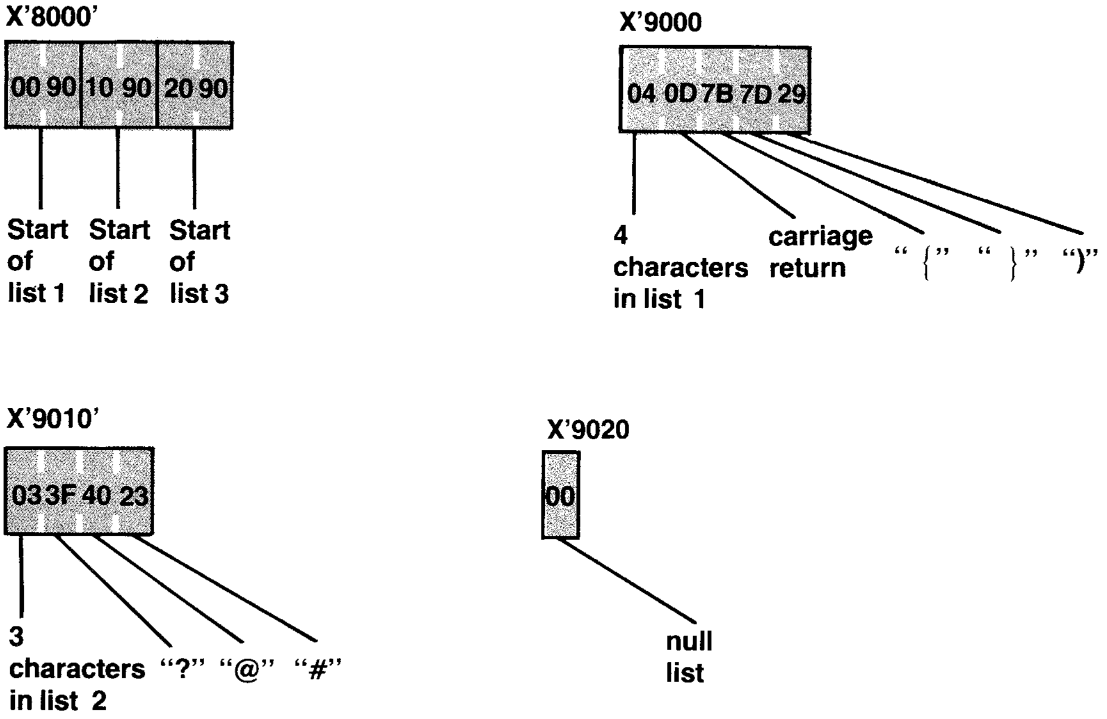
Each list has the following form:



Notes:

1. 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=0 if you aren't going to provide any lists.
2. Characters are stored in lists in ASCII form.
3. If a character appears in more than one list, it will have 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'8000'.



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, Ø            ; 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,ØFFH          ;
        CP      D               ; If D=ØFFH 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,Ø             ; Zero msb of DE
        ADD     HL,DE           ; (HL) = where to start parse
        LD      DE,LAB          ; (DE) = List address block
                                ; If DE=Ø then no lists used.
        LD      A,46            ; Function Code
        RST     8
        RET
;----------------PROGRAM CONTINUES HERE---------------------------
NXTRTN  EQU     $
```

STSCAN
String Scan (function code 49)

This is a general purpose string scan. It searches through a specified text buffer for the specified string. This string can consist of any values 0-255 (it is not strictly alphanumeric). The scan stops on the found string or on the first carriage return encountered, whichever comes first.

Entry Conditions

(HL) = Text area to be searched

(DE) = Compare string

B = Length of compare string

A = 49

Exit Conditions

NZ = String not found

Z = String found

(HL) = Start position of matching string in search string

SORT
RAM Sort (function code 56)

This routine 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 may be located anywhere in the entry.

Entries with duplicate sort-keys will remain in the same relative order.

The routine uses a ''bubble-sort'' algorithm.

Entry Conditions

A = 56
(IX) = First byte of first entry in list
(DE) = First byte of last entry in list (*not* the first byte following the list)
B = Position of key within an entry (0 = first byte of an entry)
C = Length of each entry, C>0

H = Sort switch:
If H = 0, then use ascending sort
If H ≠ 0, then use descending sort
L = Length of sort-key, L>0

In general, (B + L) must be less than or equal to C.

Exit Conditions

NZ Error occurred
A = Special error return code (not the standard TRSDOS code)

Value in A	Meaning
1	Key end exceeds last byte of entry
2	Key start exceeds last byte of entry
3	Entry length = 0 (invalid)
4	Key length = 0 (invalid)
5	Address of first entry > address of last entry

Serial Communications

Supervisor Calls described in this section:

Function Code	Name	Function
55	RS232C	Set or turn off channel A or B for serial input/output.
96	ARCV	Channel A receive
97	ATX	Channel A transmit
98	BRCV	Channel B receive
99	BTX	Channel B Transmit
100	ACTRL	Channel A control
101	BCTRL	Channel B control

These routines allow you to use the Model II's RS-232C interface, channels A and B on the back panel. See the **Model II Operation Manual** for a description of signals available.

Using the Serial Communications SVC's

The Model II contains two serial channels, A and B. If you are going to use only one channel, the other channel must be connected to the Serial Dummy Terminator described on page 4/77.

Initialization

Before performing any serial I/O, you must initialize the desired channel with either the library command SETCOM or the SVC RS232C.

Input/Output

All serial I/O is character-oriented, rather like keyboard input and video output. The major difference is that your program must check the communications status at various times to ensure that the communications link is in place and that data is not being lost for one reason or another.

After any attempted serial I/O, the communications status is returned to your program in Z-80 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 to receive operations; others to all serial I/O.

In the following tables, bit 7 refers to the most significant bit in a byte; bit 0, to the least significant.

Status byte in register A after serial output (A/BTX):

BIT	MEANING WHEN SET
0	Clear to Send (CTS) was not detected.
1	Not used.
2	Transmitter is busy.
3	Modem carrier was lost.
4	Not used
5	Not used
6	Not used
7	Not used

Status byte in register A after serial input (A/B RCV):

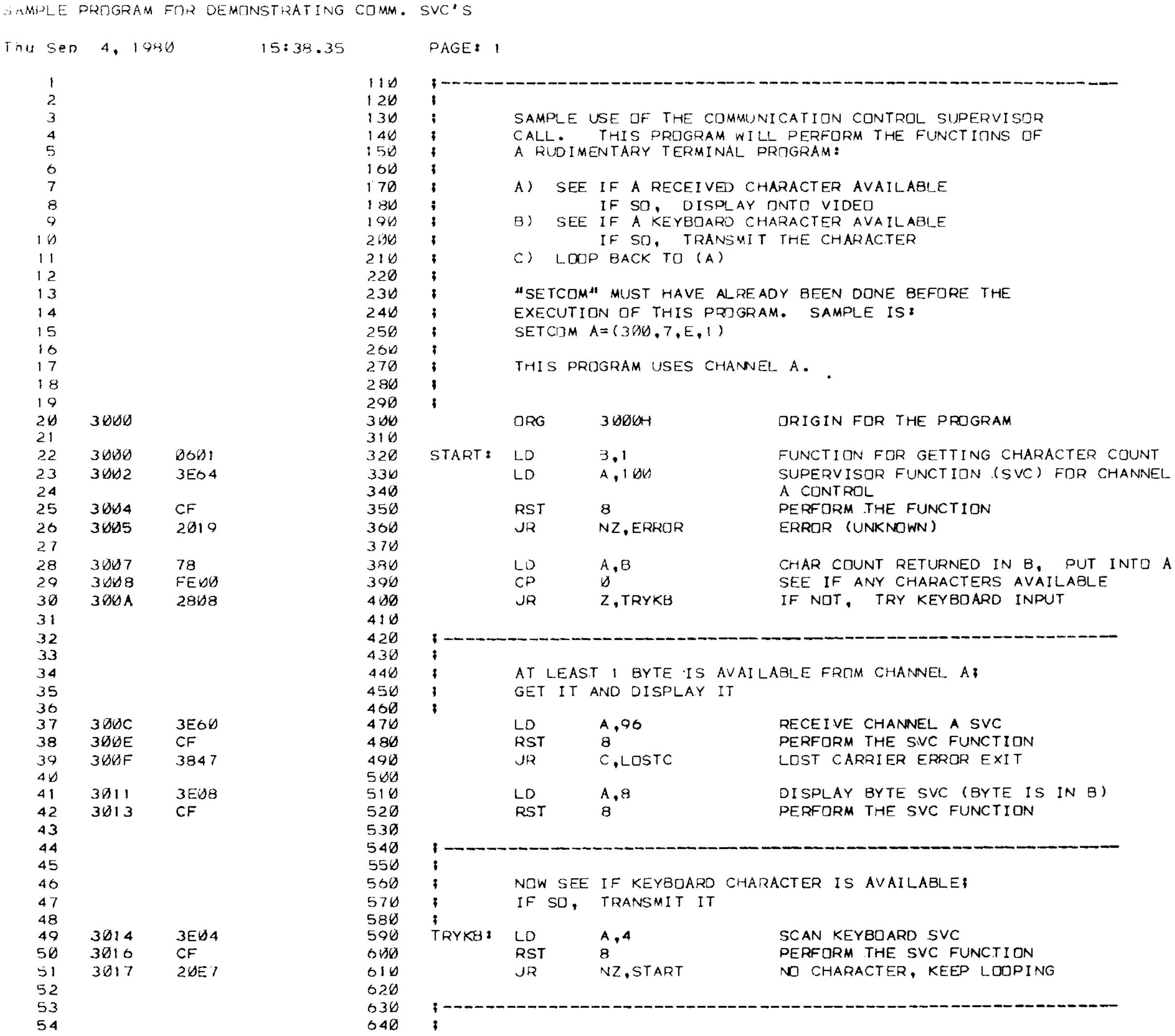
BIT	MEANING WHEN SET
0	Not used
1	Not used
2	Not used
3	Modem carrier was lost
4	Parity error occured on character found in register B.
5	Data lost—more than 16 characters received between SVC's. B. contains last character received.
6	Framing error occurred on last character received.
7	A "break sequence" (extended null character) was received. B will have a X'00' in it.

It is possible to obtain the status of a serial channel without attempting I/O, using A/BCTRL. These SVC's return the following information in register B.

BIT	MEANING WHEN SET
0	Clear to Send is not present,
1	Not used.
2	Transmitter is busy.
3	Modem carrier is not present.
4	Parity error is occurring on character currently being received.
5	Data is being lost due to overflow at hardware level (a rare occurrence)
6	Framing error is occurring on the character now being received.
7	A break sequence (extended null character) is now being received.

Sample Program

This program demonstrates the use of the serial communications SVC's in a hypothetical ''terminal'' program.

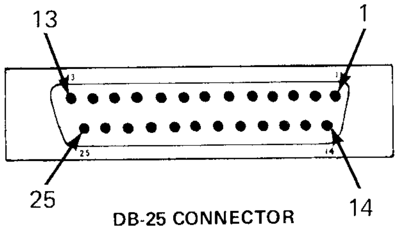


```
SAMPLE PROGRAM FOR DEMONSTRATING COMM. SVC'S

Thu Sep  4, 1980         15:38.35          PAGE: 1

                    110  ;-----------------------------------------------------------------
                    120  ;
                    130  ;        SAMPLE USE OF THE COMMUNICATION CONTROL SUPERVISOR
                    140  ;        CALL.   THIS PROGRAM WILL PERFORM THE FUNCTIONS OF
                    150  ;        A RUDIMENTARY TERMINAL PROGRAM:
                    160  ;
                    170  ;        A)  SEE IF A RECEIVED CHARACTER AVAILABLE
                    180  ;                IF SO,  DISPLAY ONTO VIDEO
                    190  ;        B)  SEE IF A KEYBOARD CHARACTER AVAILABLE
                    200  ;                IF SO,  TRANSMIT THE CHARACTER
                    210  ;        C)  LOOP BACK TO (A)
                    220  ;
                    230  ;        "SETCOM" MUST HAVE ALREADY BEEN DONE BEFORE THE
                    240  ;        EXECUTION OF THIS PROGRAM.  SAMPLE IS:
                    250  ;        SETCOM A=(300,7,E,1)
                    260  ;
                    270  ;        THIS PROGRAM USES CHANNEL A.
                    280  ;
                    290  ;
3000                300           ORG     3000H              ORIGIN FOR THE PROGRAM
                    310
3000      0601      320  START:   LD      B,1                FUNCTION FOR GETTING CHARACTER COUNT
3002      3E64      330           LD      A,100              SUPERVISOR FUNCTION (SVC) FOR CHANNEL
                    340                                      A CONTROL
3004      CF        350           RST     8                  PERFORM THE FUNCTION
3005      2019      360           JR      NZ,ERROR           ERROR (UNKNOWN)
                    370
3007      78        380           LD      A,B                CHAR COUNT RETURNED IN B,  PUT INTO A
3008      FE00      390           CP      0                  SEE IF ANY CHARACTERS AVAILABLE
300A      2808      400           JR      Z,TRYKB            IF NOT,  TRY KEYBOARD INPUT
                    410
                    420  ;-----------------------------------------------------------------
                    430  ;
                    440  ;        AT LEAST 1 BYTE IS AVAILABLE FROM CHANNEL A;
                    450  ;        GET IT AND DISPLAY IT
                    460  ;
300C      3E60      470           LD      A,96               RECEIVE CHANNEL A SVC
300E      CF        480           RST     8                  PERFORM THE SVC FUNCTION
300F      3847      490           JR      C,LOSTC            LOST CARRIER ERROR EXIT
                    500
3011      3E08      510           LD      A,8                DISPLAY BYTE SVC (BYTE IS IN B)
3013      CF        520           RST     8                  PERFORM THE SVC FUNCTION
                    530
                    540  ;-----------------------------------------------------------------
                    550  ;
                    560  ;        NOW SEE IF KEYBOARD CHARACTER IS AVAILABLE;
                    570  ;        IF SO,  TRANSMIT IT
                    580  ;
3014      3E04      590  TRYKB:   LD      A,4                SCAN KEYBOARD SVC
3016      CF        600           RST     8                  PERFORM THE SVC FUNCTION
3017      20E7      610           JR      NZ,START           NO CHARACTER, KEEP LOOPING
                    620
                    630  ;-----------------------------------------------------------------
                    640  ;
```

```
SAMPLE PROGRAM FOR DEMONSTRATING COMM. SVC'S

Thu Sep  4, 198Ø          15:38.45          PAGE: 2

  55                          65Ø   ;        WE HAVE A CHARACTER FROM THE KEYBOARD (IN B);
  56                          66Ø   ;        TRANSMIT IT
  57                          67Ø   ;
  58   3Ø19    3E61           68Ø   XMIT:    LD       A,97               TRANSMIT TO CHANNEL A SVC
  59   3Ø1B    CF             69Ø            RST      8                  PERFORM THE SVC
  6Ø   3Ø1C    2ØFB           7ØØ            JR       NZ,XMIT            MUST BE BUSY,  TRY AGAIN
  61                          71Ø
  62   3Ø1E    18EØ           72Ø            JR       START              GOOD, KEEP LOOPING
  63                          73Ø
  64                          74Ø   ;--------------------------------------------------------------
  65                          75Ø   ;
  66                          76Ø   ;        ERROR HANDLERS HERE
  67                          77Ø   ;
  68   3Ø2Ø    21293Ø         78Ø   ERROR:   LD       HL,MSG1            ERROR MESSAGE
  69   3Ø23    46             79Ø            LD       B,(HL)             GET LENGTH OF MESSAGE FROM FRONT
  7Ø   3Ø24    23             8ØØ            INC      HL                 GET HL => TEXT ITSELF
  71   3Ø25    3EØ9           81Ø            LD       A,9                DISPLAY LINE SVC
  72   3Ø27    CF             82Ø            RST      8                  PERFORM THE SVC
  73   3Ø28    C9             83Ø            RET                         RETURN TO TRSDOS
  74                          84Ø
  75   3Ø29    2E             85Ø   MSG1:    DEFT     'UNKNOWN ERROR USING CHANNEL A CONTROL FUNCTION'
  76                          86Ø
  77                          87Ø   ;--------------------------------------------------------------
  78                          88Ø   ;
  79                          89Ø   ;        LOST DATA CARRIER - DISPLAY A MESSAGE,  WAIT FOR CARRIER
  8Ø                          9ØØ   ;        TO RETURN,  AND START BACK AT TOP OF LOOP
  81                          91Ø   ;
  82   3Ø58    216D3Ø         92Ø   LOSTC:   LD       HL,MSG2            ERROR MSG
  83   3Ø5B    46             93Ø            LD       B,(HL)             GET LENGTH FROM FRONT
  84   3Ø5C    23             94Ø            INC      HL                 HL => TEXT ITSELF
  85   3Ø5D    3EØ9           95Ø            LD       A,9                DISPLAY LINE SVC
  86   3Ø5F    CF             96Ø            RST      8                  PERFORM THE SVC
  87                          97Ø
  88   3Ø6Ø    Ø6ØØ           98Ø   CLOOP:   LD       B,Ø                FUNCTION FOR GETTING STATUS
  89   3Ø62    3E64           99Ø            LD       A,1ØØ              CH A CONTROL SVC
  9Ø   3Ø64    CF            1ØØØ            RST      8                  PERFORM THE SVC
  91   3Ø65    2ØB9          1Ø1Ø            JR       NZ,ERROR           ERROR (BAD FUNCTION ?)
  92                         1Ø2Ø
  93   3Ø67    CB58          1Ø3Ø            BIT      3,B                SEE IF CARRIER BIT STILL ON
  94   3Ø69    2ØF5          1Ø4Ø            JR       NZ,CLOOP           IF ON (IE - CARRIER NOT THERE), WAIT
  95   3Ø6B    1893          1Ø5Ø            JR       START              IF CARRIER THERE, RETURN TO MAIN LOOP
  96                         1Ø6Ø
  97   3Ø6D    1E            1Ø7Ø   MSG2:    DEFT     'LOST DATA CARRIER ON CHANNEL A'
  98                         1Ø8Ø
  99                         1Ø9Ø
 1ØØ   3ØØØ                  11ØØ            END      START              TRANSFER CONTROL TO "START"
ØØØØØ  Assembly Errors
```

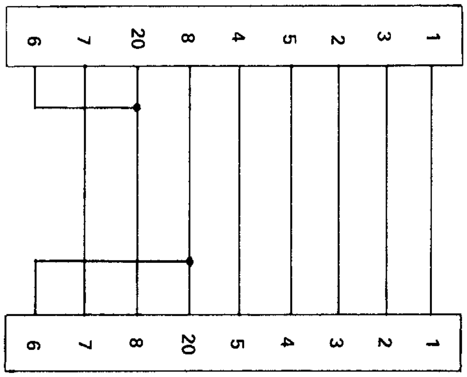
Model II / Model II Communications

For hard-wiring between two Model II's without a modem, use the wiring arrangement described below (Model II to Model II **only**)



DB-25 CONNECTOR

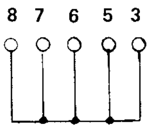
DB-25 MALE CONNECTOR
CONNECT TO RS-232 MODEL II

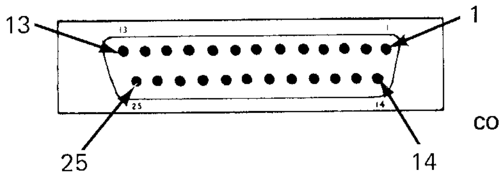


DB-25 MALE CONNECTOR
CONNECT TO CHANNEL A
MODEL II

Connection Diagram, Model II (Channel A or B) to Model II (Channel A or B). Use stranded wire, 24-gauge, to connect two DB-25 connectors as illustrated. If wire length exceeds 50 feet, twist lines 7 (GND), 2 (TD) and 3 (RD). Refer to the **Model II Operation Manual** for a description of signals available.

In addition, if you are only going to use one of the serial channels, pins 3, 5, 6, 7, and 8 on the other channel must be tied together **before** you initialize the channels with SETCOM or RS232C. Prepare a DB-25 male terminator plug for the unused channel:





DB-25 CONNECTOR

CONNECT TO CHANNEL B
MODEL II

JUMPER DIAGRAM

DUMMY TERMINATOR FOR MODEL II
(SERIAL TERMINATOR PLUG)

RS232C
Initialize RS−232C Channel
(Function Code 55)

This routine sets up or disables either channel A or B. Before using it, the appropriate channel should be connected to the modem or other equipment.

This routine sets the standard RS-232C parameters, and defines a pair of supervisor calls for I/O to the specified channel. When you initialize Channel A, SVC's 96, 97 and 100 are defined; when you initialize Channel B, SVC's 98, 99 and 101 are defined. See ARCV, ATX, BRCV, BTX, ACTRL and BCTRL.

Before re-initializing a channel, **always** turn it off. You may *not* turn off channel A when HOST is active.

Entry Conditions

(HL) = Parameter list described below
B = Function switch:
If B is not equal to zero then turn on the channel and define I/O SVC's for that channel
If B is equal to zero then turn off the channel and delete I/O SVC's for that channel. In this case only the first byte in the parameter list ("A" or "B") is used.
A = 55

Parameter List

This six-byte list includes the necessary RS−232C parameters:

CHANNEL	BAUD RATE	WORD LENGTH	PARITY	STOP BITS	END 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 110 baud
2 for 150 baud
3 for 300 baud
4 for 600 baud
5 for 1200 baud
6 for 2400 baud
7 for 4800 baud
8 for 9600 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

ARCV
BRCV
Channel A/B Receive (function codes 96/98)

These routines perform character input from channel A or B. They are analogous to keyboard character input (see KBCHAR).

TRSDOS sets up A/BRCV, A/BTX and A/BCTRL when you initialize channel A/B with RS232C. If you call any of these routines while the channel is not initialized (active), you will get an error return code of 1 (no function code exists).

Input Buffer

Each channel (A and B) has its own internal 16-character receive buffer. The buffer is established when the channel is initialized. This should reduce data overruns when receiving data at high speeds.

When each byte is received, the communications status at that time is also stored in the receive buffer. Each time you get a character via the SVC A/B RCV, the first (oldest) character in the buffer, if any, is returned in register B, and the status at the time that byte was received is returned in register A.

An overrun occurs only when the 17th byte is received into the already-full buffer. The 16th character is replaced with the 17th. When this character is retrieved, an overrun will be indicated in register A.

If there are no characters ''waiting'' in the buffer, the communications status returned in register A will reflect the *current* status of the serial interface.

Entry Conditions

A = 96 (for ARCV)
A = 98 (for BRCV)

Exit Conditions

B = Character found, if any.
NZ flag = No character found (check communications status).
C flag = Modem carrier was not present when SVC was entered.
A = Communications status:

BIT	MEANING WHEN SET
0	Not used.
1	Not used.
2	Not used.
3	Modem carrier not present.
4	Parity error occurred on last character received (in register B).
5	Data lost due to overflow (register B contains last character).
6	Framing error occurred on last character received.
7	A break sequence (extended null character) has been received.

ATX
BTX
Channel A/B Transmit (function codes 97/99)

These routines perform character output to channel A or B. They are analogous to video character output (see VDCHAR).

TRSDOS sets up A/BTX, A/BRCV and A/BCTRL when you initialize channel A/B with RS232C. If you call any of these routines while the channel is not initialized (active), you will get an error return code of 1 (no function code exists).

Data bytes will be transmitted even if no carrier is present. You must check the status flags and register A for error conditions. Carry flag set means that no carrier was present. You may ignore this flag if your application does not "care" whether the carrier is present during transmission.

Register A contains status information upon return from this SVC.

Entry Conditions

A = 97 (for ATX)
A = 99 (for BTX)
B = ASCII code for character to be sent

Exit Conditions

NZ flag = No character sent. Check register A for communications status.

C flag = Modem carrier not present when SVC was entered.

A = Communications status (bit 7 is msb; 0 is lsb):

BIT	MEANING WHEN SET
0	Clear to Send (CTS) not present.
1	Not used.
2	Transmitter busy.
3	Modem carrier not present.
4	Parity error occurred on last character received.*
5	Data lost due to hardware-level overflow.*
6	Framing error occurred on last character received.*
7	A break sequence (extended null character) has been received.*

* Bits 4 through 7 are only used when the character was not sent.

ACTRL
BCTRL
Control Channel A/B (function codes 100/101)

These routines control the RS-232-C interface, Channels A and B.

Entry Conditions

A = 100 for ACTRL, 101 for BCTRL.
B = Option switch

Contents of Register B	Result
0	Get current status of serial I/O Channel into register B
1	Get current character count in the serial receive buffer into B
2	Turn on Request To Send. On when channel is initialized.
3	Turn off RTS
4*	Start transmission of a break-sequence.
5	Stop transmission of break-sequence.
6	Reset serial receive buffer count to zero
7	Reset SIO error condition

* Before starting to transmit a break sequence, make sure the transmit buffer is empty. It is the programmer's responsibility to send the break sequence for the appropriate time period, as required by the application or host computer.

Exit Conditions

NZ = Error occurred.
B = Status, as follows:

Bit	Meaning when set
7	Break sequence is now being received
6	Framing error in a byte now being received
5	Data overflow due to a byte now being received
4	Parity error in byte now being received
3	Modem data carrier not present
2	Transmitter busy
1	Unused
0	Clear To Send not present

Programming with TRSDOS

This section tells you how to execute your own machine-language programs under TRSDOS. It includes two sections:

- Program Entry Conditions—how control is transferred to your program after it is loaded from disk.
- Handling Programmed Interrupts—how to write an interrupt service routine for BREAK key processing and TIMER interrupts.

To create and use a program:

1. Enter the program into memory, either with DEBUG, an assembler, or via the serial interface channel from another device.
2. Use the DUMP command to save the program as an executable disk file, setting load and transfer addresses.
3. To run the program, input the file name to the TRSDOS command interpreter (TRSDOS READY mode).

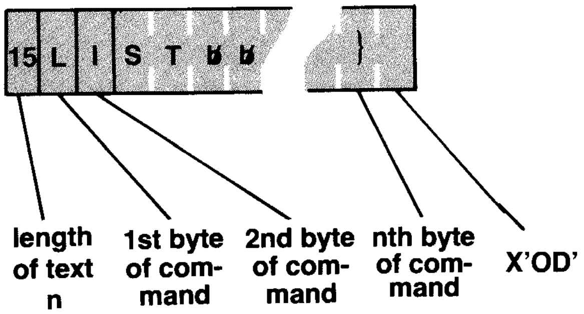
Program Entry Conditions

Upon entry to your program, TRSDOS sets up the following registers:

BC = Address of first byte following your program, i.e., the first free byte for use by your program

DE = Highest memory address not protected by TRSDOS, i.e., the end of memory which can be used by your program

HL = Address of buffer containing the last command entered to the TRSDOS command interpreter. The first byte of the buffer contains the length of the command line, not including the carriage return. The text of the command follows this length byte. For example:



Handling Programmed Interrupts

TRSDOS allows two user-programmed interrupts as described under SETBRK and TIMER. When either kind of interrupt is received (BREAK key is pressed or TIMER counts to zero), control transfers to your interrupt handling routine.

Note: System routines called by your program are also subject to interrupts. Interrupt handlers can also be interrupted.

Upon entry to your interrupt processing routine, TRSDOS sets up the registers as follows:

(SP) = The 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 finished processing, restore all registers and execute a return to continue with the interrupted program.

It is good practice to keep interrupt handling routines short; ideally, the routine simply flags the main program that an interrupt has occurred and returns. The main program can then respond to the interrupt flag when convenient.

Always end your interrupt handler with the RET instruction and with all registers intact.

TRSDOS is serially reusable but not always re-entrant. More specifically, your interrupt routine should not call TRSDOS SVC's, since under some conditions this will produce unpredictable results.

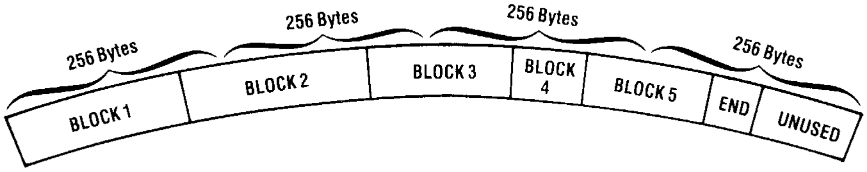
Program Files

In this section we will:

- Describe the required format and structure of program files.
- Summarize the procedure for writing a program file using TRSDOS file-access SVC's.
- Present an illustrated sample program file.

Program File Format

A program file is stored on the diskette in blocks, which might look like this:



The lengths of the blocks will vary, depending on the type of block and the amount of information in that block. The blocks must be contiguous: there can be no unused bytes after the end of one block and the beginning of the next.

There are three major types of blocks:

1. **Program data blocks.** These contain the actual program data, prefixed by four bytes of header information.

2. **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.

3. **Trailer blocks.** Each program file ends with a trailer block. It marks the end of file and tells TRSDOS 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 always four bytes long.

The first byte in a block identifies the block type, as follows:

Contents of Byte # Hex	Block Type
01	Program data
05	Comment block
02	Trailer block — jump after loading
03	Trailer block — return to caller after loading
00, 04, 06 - FF	*Reserved for system use* — loader will fail if one of these codes is used.

Details of Block Structure

Program data blocks consist of the following:

Byte #	Contents
1	Block Identifier (= 1)
2	Length — number of bytes of program data plus two for load address
3-4	Load address — where the following program data starts loading into RAM, LSB-MSB format.
5-end	Program data

Notes:

1. The block identifier always equals binary "1" for a program data block.
2. The length byte gives the number of bytes in the rest of the block — *following* the two-byte load address. This sum may range from three (two-byte addresses plus one byte of data) to 258 (two-byte address plus 256 bytes of data), but it must be translated into the range 0-255.

To do this, take the number of program bytes and increment by two. Note that values greater than 255 "wrap around" to 0, 1, and 2. Here is a table:

Number of bytes after the address (program data only)	Use this value for the length byte itself
1	3
2	4
3	5
. . .	. . .
253	255
254	0
255	1
256	2

3. The load address tells TRSDOS where the data in this block will be loaded into RAM. Bytes will be loaded serially starting at the load address. The load address must allow the entire block of program data to load in the "User" area of RAM (see "Memory Requirements of TRSDOS").

Comment Blocks

Comment blocks consist of the following:

Byte #	Contents
1	Block Identifier (= 5)
2	Length — number of bytes in the comment
3-end	Comment

Notes:

1. For comments, the block identifier always equals binary "5".

2. The length byte gives the number of bytes in the comment, i.e., the number of bytes after the length byte itself. This sum ranges from 0 to 255:

Length Byte	Length of Comment
0	0
1	1
2	2
. . .	
255	255

Trailer Blocks

Each program file ends with a single trailer block which tells TRSDOS what to do next:

A. Jump to a specified "transfer" address.
- or -
B. Return to the caller.

Trailer blocks consist of the following:

Byte #	Contents
1	Block Identifier (= 2 or 3)
2	Length (= 2)
3-4	Transfer address in LSB-MSB sequence

Notes

1. For trailer blocks, the identifier can equal binary "2" or "3".

   If ID = 2, TRSDOS will jump to the transfer address after loading the program.

   If ID = 3, TRSDOS will not jump to the transfer address, but will return to the caller after loading the program, with the following prime registers set:

   HL' = Transfer address taken from trailer block
   DE' = Address of last usable byte
   BC' = First byte following the program just loaded.

2. The transfer address is in LSB-MSB sequence. It must be in the user area of RAM (see "Memory Requirements of TRSDOS").

Procedure for Writing a Program File

Program files must have fixed length records of length 256, and must have the "P" (Program) access type. Both of these attributes are set when the file is created (Creation code 1 or 2). See SVC OPEN.

The TRSDOS 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", i.e., there are no unused bytes between blocks. It is the programmer's responsibility to:

1. Store the correct byte sequence in the logical record area

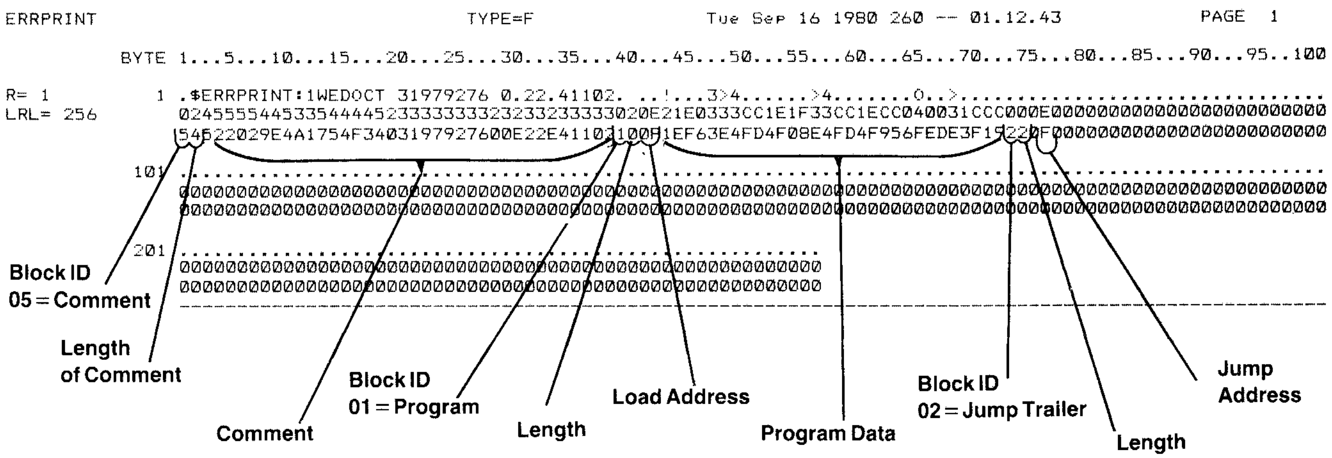
2. Call the disk write SVC(DIRWR) when each 256-byte record is filled. Blocks can and must "span" sectors. The only unused bytes in the file will be after the trailer block. See the illustration on page 4/87.

User Notes

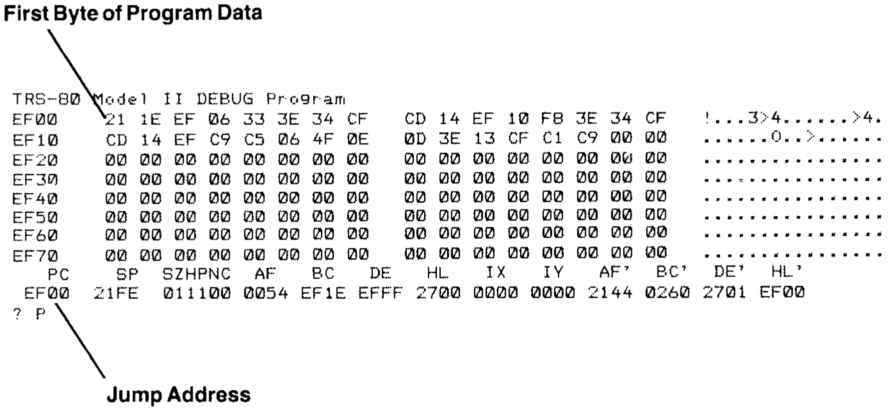
1. The TRSDOS DUMP command always writes a comment as the first block in the file. The comment consists of the file specification followed by the current TRSDOS time/date text.

2. TRSDOS will not load a user program *below* "user RAM". The programmer should ensure that blocks do not load *above* "user RAM". See "Memory Requirements of TRSDOS".

Illustrated Program File Listing



```
ERRPRINT                              TYPE=F              Tue Sep 16 1980 260 -- 01.12.43               PAGE  1

               BYTE 1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95..100

R= 1              1 .$ERRPRINT:1WEDOCT 31979276 0.22.41102. ..!...3>4......>4.......O..>................................
LRL= 256            0245555445335444452333333332323323333302 0E21E0333CC1E1F33CC1ECC040031CCC000E000000000000000000000000
                    54622029E4A1754F3403197927600E22E41102100F1EF63E4FD4F08E4FD4F956FEDE3F19220F000000000000000000000000

                101 ....................................................................................................
                    0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                    0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

                201 ........................................................
                    00000000000000000000000000000000000000000000000000000000
                    00000000000000000000000000000000000000000000000000000000
                    ----------------------------------------------------------------------------------------------------
```



```
TRS-80 Model II DEBUG Program
EF00      21 1E EF 06 33 3E 34 CF    CD 14 EF 10 F8 3E 34 CF    !...3>4......>4.
EF10      CD 14 EF C9 C5 06 4F 0E    0D 3E 13 CF C1 C9 00 00    .......O..>.....
EF20      00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
EF30      00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
EF40      00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
EF50      00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
EF60      00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
EF70      00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00    ................
    PC     SP    SZHPNC   AF    BC    DE    HL    IX    IY    AF'   BC'   DE'   HL'
  EF00   21FE   011100 0054 EF1E EFFF 2700 0000 0000 2144 0260 2701 EF00
? P
```

High-Memory Modules and Their Addresses and Use

As mentioned earlier, there are five high-memory modules that are conditionally loaded. There are DO, SPOOLER, HOST, DEBUG and communications drivers. In a 64K machine, these reside above X'F000'; in a 32K machine, these reside at X'7000' and above. The approximate memory map is as follows:

Note: Addresses may change in future releases.

Module	32K	64K
Comm drivers	X'7000'	X'F000'
DEBUG	X'7400'	X'F400'
HOST*	X'7400'	X'F400'
SPOOLER*	X'7640'	X'F640'
DO*	X'7C80'	X'FC80'

*Mutually exclusive with DEBUG

Some applications require a large amount of RAM to be able to execute (e.g., a large BASIC program or a machine-language subroutine that must link to BASIC and you need full RAM for the BASIC program itself). If this is the case, you may use the upper portion of RAM for your application if you consider the following:

A. The use of RAM above the X'7000' (32K machine) or X'F000' (64K machine) boundary will preclude the use of certain high-memory modules. If you invoke a system function that loads this high-RAM with a module, and then your application loads something on top of this, you will certainly experience problems. Refer to the memory map above for details. The system cannot prevent you from overlaying an already loaded high-memory module. You must take the appropriate care to prevent any problems if you choose to use this RAM space.

B. The operating system now, upon program-load, will either give X'EFFF' (64K machine) or X'6FFF' (32K machine) in register DE upon entry to the program as the high-RAM address. To the user of BASIC, this will give you slightly less RAM for program and variable storage than was available in earlier versions of TRSDOS. This may be over-ridden, if necessary, by specifying the command line: BASIC – M:XXXXX where XXXXX is the decimal high-RAM address to use as the upper limit. *This must never exceed decimal 63487 for a 64K machine!* The same rule applies to an applications program. The intent of this RAM restriction is that if you never use the RAM above this high-memory mark, you will be able to use all of the high-memory routines during the execution of your programs.

C. The most common high-memory module used is probably the communications drivers, as these are used by the serial printer software in the operating system. If you use a serial printer, you will not want to overlay X'F000-F3FF' (64K machine) or X'7000-73FF' (32K machine). The code in that area must be left intact at all times that the comm drivers are active. Overlaying anything here will cause problems, some of which could be very serious.

D. SPOOLER is another high-memory module that can be used by almost any application that has printed output. It has been designed to be used without any changes in the application software. It may be used at any time, and the decision to use it may rest in the operator of the Model II. To provide, in your application software, the most flexibility for the operator, you should not use the RAM area of 'F640-F800' (64K machine) or '7640-7C80' (32K machine). If you need this RAM, you must make it clear to the operator of the system that spooler may not be used. Serious problems will occur if any of the spooler code space is overlayed after the system has loaded the spooler code in that area.

E. HOST is another high-RAM routine that is designed to be ''transparent'' to the application software. Host allows for a remote terminal to execute your applications and programs as if the remote were local (the local keyboard and video). If this high-RAM space is overlayed with a user-subroutine, serious problems will occur. Operation flexibility suffers if an application is written that depends upon the use of this space for a subroutine or program.

As you can see, the use of any high-RAM space will prevent the use of some or all of the system functions that use this space. Careful planning of what will be needed, when it will be needed, and what you want to instruct the operator of the system to do (or not to do) when the operator runs your application, may provide you with the extra RAM space you need.

An example might be that your application program will not create printed output. In this case, even if you have a serial printer, you will not need the spooler RAM area or the comm drivers RAM area (if comm drivers were needed just for the serial printer support). If you make sure that the spooler and the comm drivers are not active before or during the run of the application, then you may use this RAM space. One way to be sure of this is to start execution of the application with a DO file (which resides in the top of RAM). In this DO file, you may shut off spooler and comm before the application itself is executed. That way, you can be sure that these high-RAM modules will not be resident and will not be overlayed. Remember that you must shut down both comm channels (A & B) before the high-RAM comm drivers are considered inactive by the system.

The example programs supplied on the original system disk that use high-RAM subroutines (DATMxx, EXDATMxx, COMSUBxx, BASCOMxx, etc.) and the DO files to setup these (DOCOMxx) all demonstrate the correct way to use high-RAM subroutines below the lowest of the system's high-RAM modules. Examine these for the techniques, and addresses used. For full operational flexibility, it is recommended to stay away from the high-RAM areas.

It is also possible that in future releases of the TRSDOS operating system, more features or routines will be available that will use this space in high-RAM. You should keep this in mind whenever you design your applications; the next features might be exactly what you need (or desire) and that, in order to use these features, you will have to modify your high-RAM subroutines. Consider the efforts required to make these program changes, especially if your application program(s) are not easy to modify, or it is not easy to get revised copies out to the users.

Section 5

Appendix

TRSDOS Character Codes
Keyboard Code Map
Decimal-Hexadecimal Conversion

TRSDOS Character Codes

Code		Character		
		Key-board	Video Display	
Dec.	Hex.		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	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	CTRL L		█
13	0D	ENTER CTRL M	Carriage return	┳
14	0E	CTRL N	Dual routing on .	╋
15	0F	CTRL O	Dual routing off.	┼
16	10	CTRL P		▁
17	11	CTRL Q		▪
18	12	CTRL R		▄
19	13	CTRL S		▮
20	14	CTRL T	Homes cursor in scroll area.	▌
21	15	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	b̸	b̸
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- board	Video Display	
Dec.	Hex.		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		
		Key-board	Video Display	
Dec.	Hex.		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 Ø	¦	¦
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

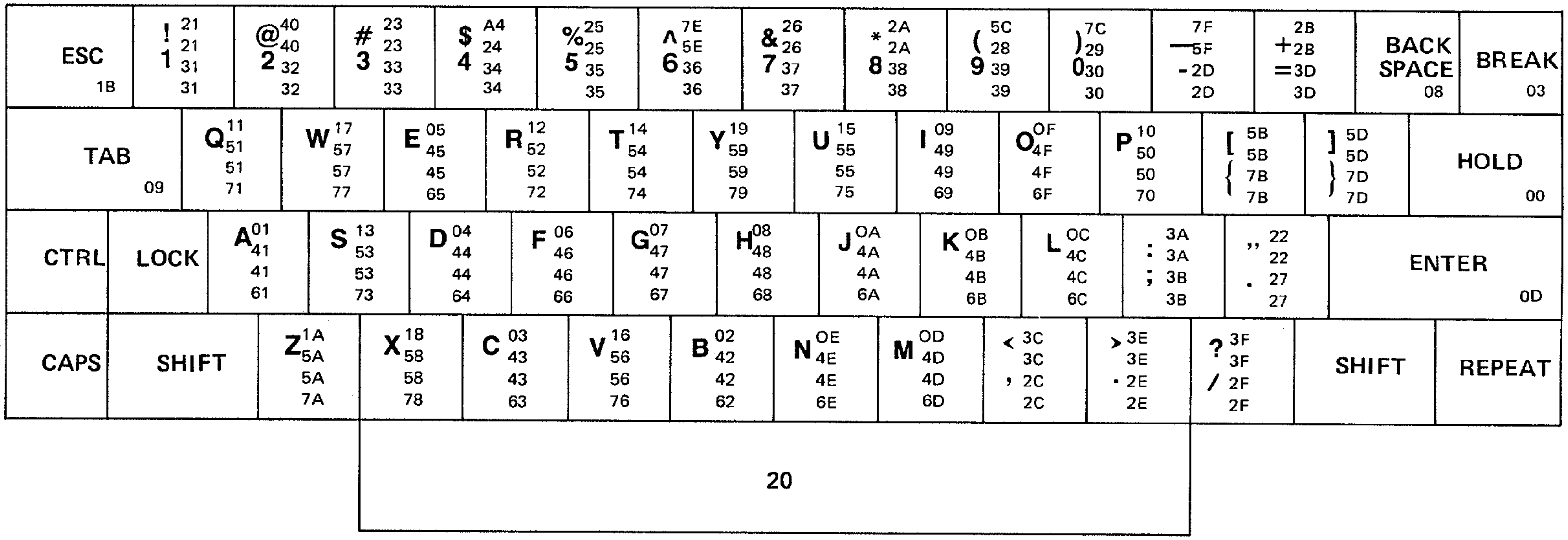
* Codes 128-248 cannot be input from the keyboard or output to the display. When reading the display, a **value** greater than 127 indicates a reverse character corresponding to **value** mod 128.

Keyboard Code Map

The keyboard map (see foldout page) presents the actual code that TRSDOS will return to the user for each key on the keyboard, in each of the four modes—unshift, shift, caps, and control.

A program executing under TRSDOS—for example, BASIC—may translate some of these codes into other values. Consult the program's documentation for details.

Note: The **BREAK** key (code X'03') is always intercepted by TRSDOS. It will never be returned as a character to the user.



ESC 1B	! 1 21 21 31 31	@ 2 40 40 32 32	# 3 23 23 33 33	$ 4 A4 24 34 34	% 5 25 25 35 35	^ 6 7E 5E 36 36	& 7 26 26 37 37	* 8 2A 2A 38 38	(9 5C 28 39 39	) 0 7C 29 30 30	— - 7F 5F 2D 2D	+ = 2B 2B 3D 3D	BACK SPACE 08	BREAK 03

TAB 09	Q 11 51 51 71	W 17 57 57 77	E 05 45 45 65	R 12 52 52 72	T 14 54 54 74	Y 19 59 59 79	U 15 55 55 75	I 09 49 49 69	O 0F 4F 4F 6F	P 10 50 50 70	[{ 5B 5B 7B 7B	] } 5D 5D 7D 7D	HOLD 00

CTRL	LOCK	A 01 41 41 61	S 13 53 53 73	D 04 44 44 64	F 06 46 46 66	G 07 47 47 67	H 08 48 48 68	J 0A 4A 4A 6A	K 0B 4B 4B 6B	L 0C 4C 4C 6C	: ; 3A 3A 3B 3B	" ' 22 22 27 27	ENTER 0D

CAPS	SHIFT	Z 1A 5A 5A 7A	X 18 58 58 78	C 03 43 43 63	V 16 56 56 76	B 02 42 42 62	N 0E 4E 4E 6E	M 0D 4D 4D 6D	< , 3C 3C 2C 2C	> . 3E 3E 2E 2E	? / 3F 3F 2F 2F	SHIFT	REPEAT

20

LEGEND:

XX	CONTROL
XX	SHIFT
XX	CAPS
XX	UNSHIFT

← 1C	7	8 38	9 39	F1 01
→ 1D	4 34	5 35	6 36	F2 02
↑ 1E	1 31	2 32	3 33	E N T E R 0D
↓ 1F	Ø 30		. 2E	

Decimal–Hexadecimal Conversion

DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX
Ø	ØØ	35	23	7Ø	46	1Ø5	69
1	Ø1	36	24	71	47	1Ø6	6A
2	Ø2	37	25	72	48	1Ø7	6B
3	Ø3	38	26	73	49	1Ø8	6C
4	Ø4	39	27	74	4A	1Ø9	6D
5	Ø5	4Ø	28	75	4B	11Ø	6E
6	Ø6	41	29	76	4C	111	6F
7	Ø7	42	2A	77	4D	112	7Ø
8	Ø8	43	2B	78	4E	113	71
9	Ø9	44	2C	79	4F	114	72
1Ø	ØA	45	2D	8Ø	5Ø	115	73
11	ØB	46	2E	81	51	116	74
12	ØC	47	2F	82	52	117	75
13	ØD	48	3Ø	83	53	118	76
14	ØE	49	31	84	54	119	77
15	ØF	5Ø	32	85	55	12Ø	78
16	1Ø	51	33	86	56	121	79
17	11	52	34	87	57	122	7A
18	12	53	35	88	58	123	7B
19	13	54	36	89	59	124	7C
2Ø	14	55	37	9Ø	5A	125	7D
21	15	56	38	91	5B	126	7E
22	16	57	39	92	5C	127	7F
23	17	58	3A	93	5D	128	8Ø
24	18	59	3B	94	5E	129	81
25	19	6Ø	3C	95	5F	13Ø	82
26	1A	61	3D	96	6Ø	131	83
27	1B	62	3E	97	61	132	84
28	1C	63	3F	98	62	133	85
29	1D	64	4Ø	99	63	134	86
3Ø	1E	65	41	1ØØ	64	135	87
31	1F	66	42	1Ø1	65	136	88
32	2Ø	67	43	1Ø2	66	137	89
33	21	68	44	1Ø3	67	138	8A
34	22	69	45	1Ø4	68	139	8B

Decimal—Hexadecimal Conversion, continued

DEC	HEX	DEC	HEX	DEC	HEX
14Ø	8C	18Ø	B4	22Ø	DC
141	8D	181	B5	221	DD
142	8E	182	B6	222	DE
143	8F	183	B7	223	DF
144	9Ø	184	B8	224	EØ
145	91	185	B9	225	E1
146	92	186	BA	226	E2
147	93	187	BB	227	E3
148	94	188	BC	228	E4
149	95	189	BD	229	E5
15Ø	96	19Ø	BE	23Ø	E6
151	97	191	BF	231	E7
152	98	192	CØ	232	E8
153	99	193	C1	233	E9
154	9A	194	C2	234	EA
155	9B	195	C3	235	EB
156	9C	196	C4	236	EC
157	9D	197	C5	237	ED
158	9E	198	C6	238	EE
159	9F	199	C7	239	EF
16Ø	AØ	2ØØ	C8	24Ø	FØ
161	A1	2Ø1	C9	241	F1
162	A2	2Ø2	CA	242	F2
163	A3	2Ø3	CB	243	F3
164	A4	2Ø4	CC	244	F4
165	A5	2Ø5	CD	245	F5
166	A6	2Ø6	CE	246	F6
167	A7	2Ø7	CF	247	F7
168	A8	2Ø8	DØ	248	F8
169	A9	2Ø9	D1	249	F9
17Ø	AA	21Ø	D2	25Ø	FA
171	AB	211	D3	251	FB
172	AC	212	D4	252	FC
173	AD	213	D5	253	FD
174	AE	214	D6	254	FE
175	AF	215	D7	255	FF
176	BØ	216	D8		
177	B1	217	D9		
178	B2	218	DA		
179	B3	219	DB		

Section 6

Index

TRSDOS INDEX