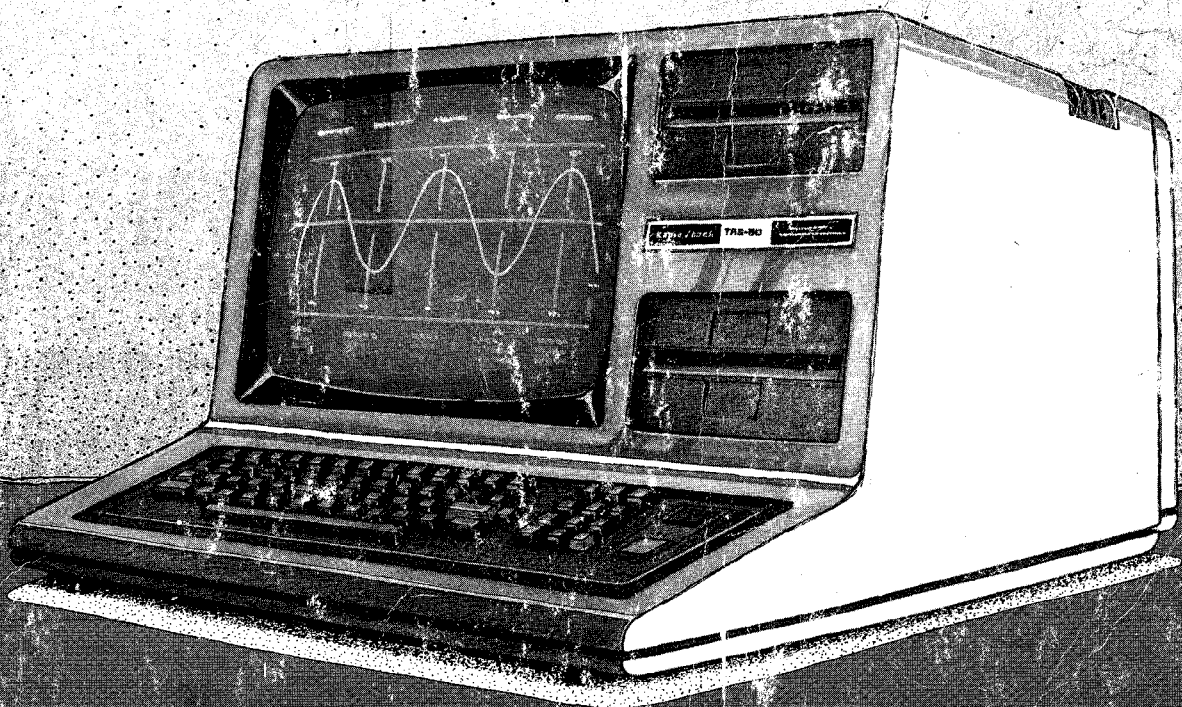


TRS-80™ Model III

Operation and
BASIC Language
Reference Manual



Radio Shack

The biggest name in little computers™

CUSTOM MANUFACTURED IN THE USA BY RADIO SHACK  A DIVISION OF TANDY CORPORATION

The FCC Wants You to Know . . .

This equipment generates and uses radio frequency energy. If not installed and used properly, that is, in strict accordance with the manufacturer's instructions, it may cause interference to radio and television reception.

It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna
- Relocate the computer with respect to the receiver
- Move the computer away from the receiver
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, you should consult the dealer or an experienced radio/television technician for additional suggestions. You may find the following booklet prepared by the Federal Communications Commission helpful: *How to Identify and Resolve Radio-TV Interference Problems*.

This booklet is available from the US Government Printing Office, Washington, DC 20402, Stock No. 004-000-00345-4.

Warning

This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

TRS-80TM Model III

Operation and BASIC Language Reference Manual

Radio Shack[®]

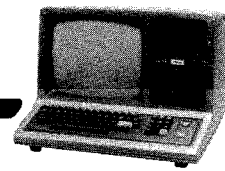
 A DIVISION OF TANDY CORPORATION
FORT WORTH, TEXAS 76102

TRS-80 Model III Operation and BASIC Language Reference Manual: ©1980 Tandy Corporation, Fort Worth, Texas 76102 U.S.A. All Rights Reserved.

Reproduction or use, without express written permission from Tandy Corporation, of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the use of the information obtained herein.

Model III System Software: ©1980 Tandy Corporation and Microsoft. All Rights Reserved.

The system software in the Model III microcomputer is retained in a read-only memory (ROM) format. All portions of this system software, whether in the ROM format or other source code form format, and the ROM circuitry, are copyrighted and are the proprietary and trade secret information of Tandy Corporation and Microsoft. Use, reproduction or publication of any portion of this material without the prior written authorization by Tandy Corporation is strictly prohibited.



To Our Customers. . .

The TRS-80[®] Model III Computer is a very powerful tool for business, home and recreation. Twenty years ago, this capability would have cost hundreds of times as much as your Model III cost, and would have taken up an entire room.

In spite of its power and internal complexity, the Model III can be quite simple to operate. In fact, **you** can determine just how “technical” a machine you want it to be.

At the simplest level of operation, you can use Radio Shack prepared cassette programs. All you will need to know is how to load and run a cassette program, and how to operate the cassette recorder. If this is where you want to start, read Chapters 1 through 6 of the Operation Section. You may also want to read about CLOAD and SYSTEM in Chapter 2 of the Language Section.

If you want to write your own programs and you are a beginner, read Chapters 1 through 6 of the Operation Section, then start reading the book, *Getting Started with TRS-80 BASIC*. That, plus several other Radio Shack books, can guide you to becoming a programmer in BASIC and Z-80 language (“machine code”).

If you already know BASIC, and especially if you have experience on a TRS-80 Model I, read the entire Operation Section of this manual, as well as the Appendix which compares the Model I and Model III. The Model III has many unique features and some very important differences. A few minutes spent before you press **(ENTER)** could save you hours later.

About This Manual

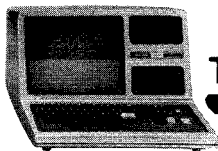
This manual contains operating instructions (Section 1) and a description of Model III BASIC (Section 2 and Appendix). It is arranged for easy reference, whether you are seeking simple or technical information. Page numbering starts over at the beginning of each chapter, and chapter numbering starts over at the beginning of each section. There is a comprehensive Index at the end of this book.

If you are a beginner, don't worry about the technical parts in the Operation Section. The beginning of each chapter is for you. (When you get to the POKE statements, you can skip ahead to the next chapter...) You don't need to read past Chapter 6. Then, when you learn simple BASIC programming, you can return and try out all the “goodies” packed into your Model III.

Very Important Note

Before you even plug in your Model III, read Chapters 2 and 3 in the Operation Section—no matter how much you think you know. This applies whether you have a cassette- or disk-based system.

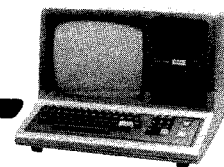
Remember, when all else fails, read the instructions!



Contents

Operation Section

A Brief Description of the Computer	1/1-3
Installation	2/1-3
Operation	3/1-9
Power-On <input type="checkbox"/> RESET Switch <input type="checkbox"/> Power-Off <input type="checkbox"/> Start-Up Dialog <input type="checkbox"/> Modes of Operation <input type="checkbox"/> Sample Session	
Using the Keyboard	4/1-3
Capitals and Lowercase <input type="checkbox"/> Special Keys <input type="checkbox"/> Control Codes	
Using the Video Display	5/1-5
Character Size <input type="checkbox"/> Cursor <input type="checkbox"/> Scroll Protection <input type="checkbox"/> Text <input type="checkbox"/> Graphics <input type="checkbox"/> Space Compression <input type="checkbox"/> Special Characters	
Using the Cassette Interface	6/1-6
Cassette Transfer Speed <input type="checkbox"/> Loading Errors <input type="checkbox"/> Saving a BASIC Program on Tape <input type="checkbox"/> Loading a BASIC Program from Tape <input type="checkbox"/> How to Search for a Program <input type="checkbox"/> Loading a SYSTEM Tape <input type="checkbox"/> Searching for a Program	
Using a Line Printer	7/1-6
Line Printer vs Video Display Output <input type="checkbox"/> Printer Control Features <input type="checkbox"/> Print Screen Function	
Using the RS-232-C Interface	8/1-8
What is an Interface? <input type="checkbox"/> Using the Model III as a Terminal <input type="checkbox"/> Programming the RS-232-C	
Routing Input/Output	9/1-3
To Route from One Device to Another <input type="checkbox"/> Routing Multiple Devices	
Real-Time Clock	10/1-3
To Set the Clock <input type="checkbox"/> To Read the Clock <input type="checkbox"/> To Display the Clock	
Input/Output Initialization	11/1-1
Technical Information	12/1-26
To Protect High RAM <input type="checkbox"/> ROM Subroutines <input type="checkbox"/> Memory Map <input type="checkbox"/> Summary of Important ROM Addresses <input type="checkbox"/> Summary of Important RAM Addresses	
Troubleshooting and Maintenance	13/1-3
Symptom/Cure Table <input type="checkbox"/> AC Power Sources <input type="checkbox"/> Maintenance	
Specifications	14/1-3
Power Supply <input type="checkbox"/> Microprocessor <input type="checkbox"/> RS-232-C Interface <input type="checkbox"/> Parallel (Printer) Interface <input type="checkbox"/> Cassette Interface	



BASIC Language Section

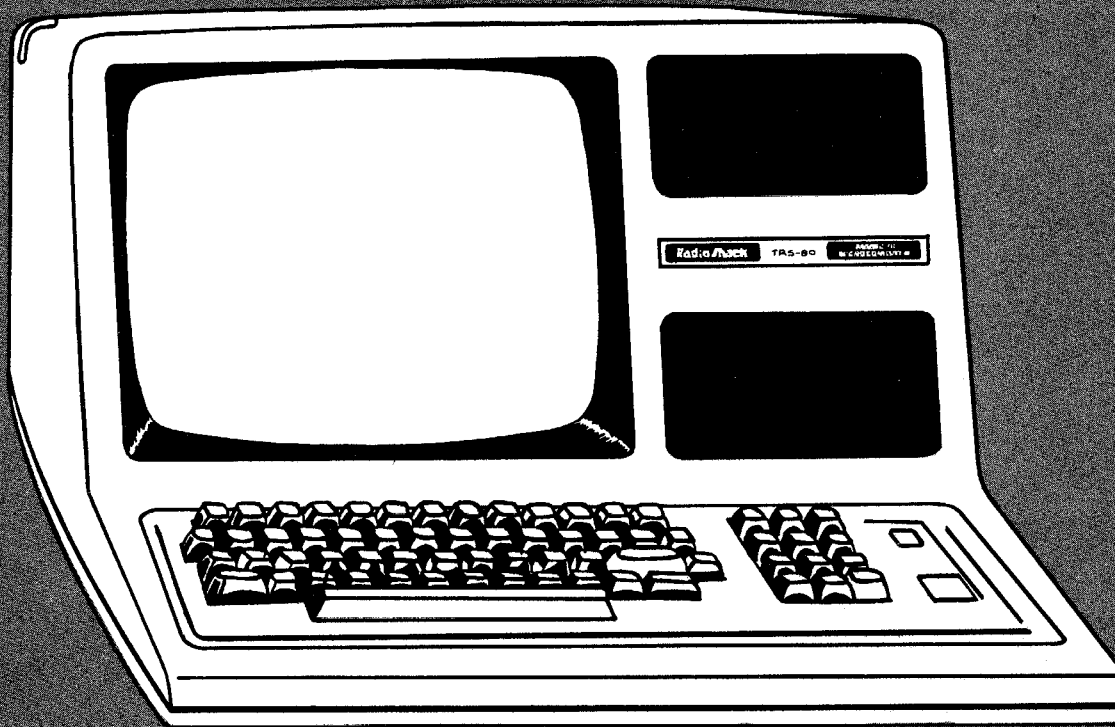
BASIC Concepts	1/1-30
Commands	2/1-7
Input-Output Statements	3/1-13
Program Statements	4/1-15
Strings	5/1-9
Arrays	6/1-6
Arithmetic Functions	7/1-5
Special Features	8/1-10
Editing	9/1-7

Appendices

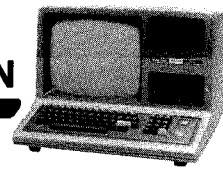
Model III Summary	A/1-18
Special Characters and Abbreviations <input type="checkbox"/> Commands <input type="checkbox"/> State- ments <input type="checkbox"/> Functions <input type="checkbox"/> Reserved Words <input type="checkbox"/> Program Limits <input type="checkbox"/> Memory Use <input type="checkbox"/> Accuracy <input type="checkbox"/>	
Error Codes	B/1-3
TRS-80 Model III Character Codes	C/1-9
Keyboard/Display Characters <input type="checkbox"/> Graphics <input type="checkbox"/> Special Charac- ters <input type="checkbox"/> Video Display Worksheet <input type="checkbox"/>	
Internal Codes for BASIC Keywords	D/1-2
Derived Functions	E/1-2
Base Conversions	F/1-4
Model I to Model III Program Conversion Hints	G/1-2
Glossary	H/1-3
RS-232-C Technical Information	I/1-4

Index

For Warranty and Customer Information, see the back cover and inside back cover.



Section 1: Operation



1 / A Brief Description

The Radio Shack TRS-80[™] Model III is a ROM-based computer system consisting of:

- A 12-inch screen to display results and other information
- A 65-key console keyboard for inputting programs and data to the Computer
- A Z-80 Microprocessor, the “brains” of the system
- A Real-Time Clock
- Read Only Memory (ROM) containing the Model III BASIC Language (fully compatible with most Model I BASIC programs)
- Random Access Memory (RAM) for storage of programs and data while the Computer is on (amount is expandable from “16K” to “48K”, optional extra)
- A Cassette Interface for long-term storage of programs and data (requires a separate cassette recorder, optional/extra)
- A Printer Interface for hard-copy output of programs and data (requires a separate line printer, optional/extra)
- Expansion area for upgrading to a disk-based system (optional/extra)
- Expansion area for an RS-232-C serial communications interface (optional/extra)

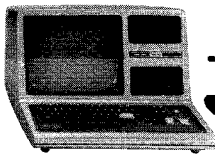
All these components are contained in a single molded case, and all are powered via one power cord.

Video Display Screen

Displayable characters include the standard 96 text-characters with the upper and lowercase alphabet; 64 graphics characters; and 160 special TRS-80 characters. In addition, there are numerous control and space-compression characters. Some of the character sets can be switched in and out by BASIC and other programs.

Keyboard

The keyboard allows entry of all the standard text and control characters. It also includes a 12-key section for convenient numeric entry. From the keyboard, you can select either all-capitals or upper and lowercase entry. The **(BREAK)** key is designed to return control to you during any operation, including cassette input/output or line printer output. Every key has an auto-repeat feature.



Z-80 Microprocessor

This is the central processing unit—where all the “thinking” is done. In the Model III, the microprocessor operates at a speed of over two million cycles per second.

Read Only Memory (ROM)

This is where the Computer’s built-in programs are stored, including the TRS-80 BASIC language. TRS-80 BASIC is fully compatible with the Level II language used in Model I TRS-80’s. Each time you power-on the Computer, this ROM program takes charge of the microprocessor, enabling you to type in simple BASIC-language instructions.

The Model III contains a “14K” ROM, meaning it contains $14 * 1024 = 14336$ characters (“bytes”) of permanently programmed memory.

Random Access Memory (RAM)

This is where your programs and results are stored while the Computer is on. It is erased when you turn the Computer off.

The Model III can be equipped with 16K, 32K or 48K of RAM (1K = 1024 bytes).

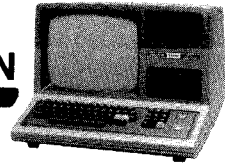
Peripherals

These are devices you can add to your Computer to increase its usefulness in programming and data storage. The Model III contains the necessary “interfaces” to simplify the addition of many peripherals.

Cassette

For long-term storage of programs and data, simply connect a cassette recorder to the Computer, and save the information on tape.

For program storage, you may select either High or Low transfer rates (use Low for compatibility with Model I, High for faster saves and loads).



Printer

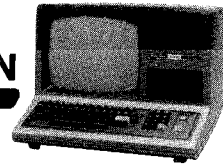
You may connect any Radio Shack "parallel interface" printer to the Model III; this will give you "hard-copy" capability for program listings, reports, mailing lists, invoices, etc.

Other Enhancements

The Model III contains space for a mini-disk controller and one or two mini-disk drive units. The Computer will accommodate one or two external drive units as well.

With a one-, two-, three- or four-drive system, you will be able to store and retrieve programs and data both quickly and reliably. Your Computer will then be under the control of TRSDOS™, the powerful Radio Shack Disk Operating System.

You can also add an internal RS-232-C serial interface. This will allow your computer to communicate with an RS-232-C equipped computer, serial line printer or other serial device.



2 / Installation

Carefully unpack the Computer. Remove all packing material and save it in case you ever need to transport the Computer. Be sure to locate all cables, papers, etc., that may be included in the shipping carton.

Place the Computer on the surface where you'll be using it. An appropriate power source should be nearby, so that no extension cord will be required.

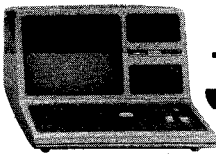
Do not connect the Computer to the AC power source yet.

Connection of Peripherals

Before connecting any peripherals (for example, line printer and cassette recorder), make sure the Computer and the peripheral devices are turned off.

Connect all peripherals to the appropriate jacks on the bottom and rear of the Computer. Refer to Figure 1 for location of connection points. For interconnections between cables and peripherals, refer to the Owner's Manual supplied with the peripheral device.

Note: All cables should exit to the rear of the unit so that no binding occurs.



TRS-80 MODEL III

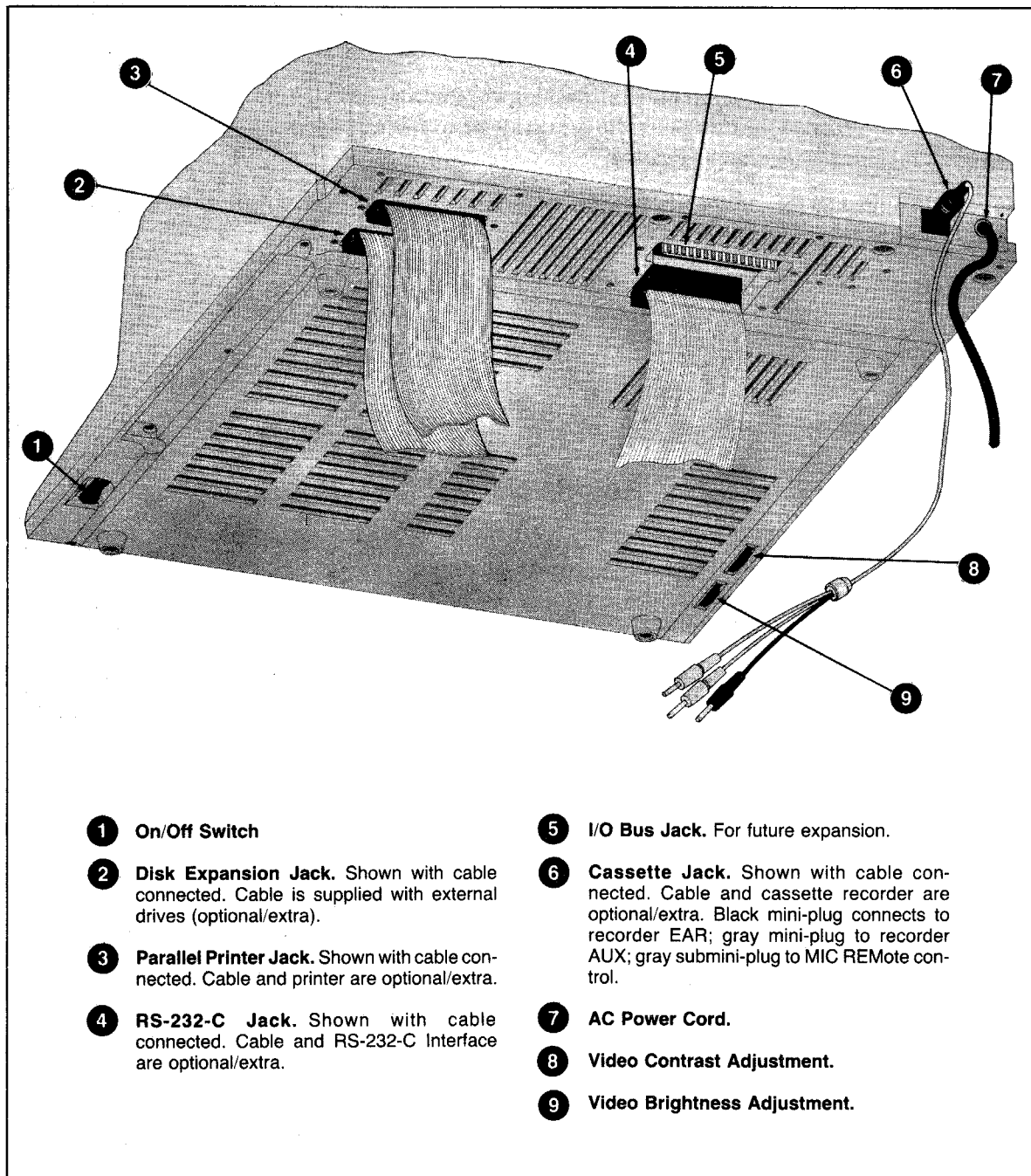
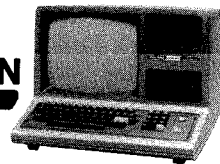


Figure 1. Connection of peripherals and location of controls.



Connection of a Cassette Recorder

The following instructions use the CTR-80A recorder (Radio Shack Catalog Number 26-1206) as an example. If you use a different recorder, connection and operation may vary.

Note: You do not need to connect the Cassette Recorder unless you plan to record programs or to load taped programs into the TRS-80.

A TRS-80 to Cassette Recorder connection cable is included with the CTR-80A; we suggest that you use this specially designed cable.

1. Connect the short cable (DIN plug on one end and three plugs on the other) to the **TAPE** jack on the back of the Computer. **Be sure you get the plug to mate correctly.**
2. The three plugs on the other end of this cable are for connecting to the recorder.
3. A. Connect the **black plug** into the EAR jack on the side of the recorder. This connection provides the output signal from the recorder to TRS-80 (for loading Tape programs into TRS-80).
B. Connect the larger **gray plug** into the AUX jack on the recorder. This connection provides the recording signal to record programs from the TRS-80 onto the tape.

Leave the AUX plug in whether you are recording or playing back cassette data.

- C. Connect the smaller gray plug into the smaller MIC jack on the recorder. This allows the TRS-80 to automatically control the recorder motor (turn tape motion on and off for recording and playing tapes.)

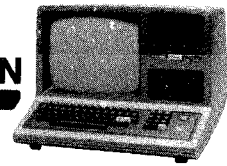
Note: Do not plug a remote microphone or a dummy plug into the larger MIC jack.

Connection to an AC Power Source

Make sure the Computer and all peripherals are **off**.

The AC Power Cord exits from the rear of the Computer. Connect it and all peripherals to an appropriate power source. Power requirements for Radio Shack products are specified on the units and in the Owner's Manual Specifications.

For convenience, you may connect all components to a single "power strip" such as Radio Shack's 26-1451 Line Filter. This will allow you to turn on the entire system with a single switch. Take care not to exceed the current capacity of the power strip.



3 / Operation

Power-On

The following instructions explain how to start up and use the Model III as a **ROM-based system only**.

If you have a Disk System and are going to load TRSDOS, follow the power-up instructions given in the Model III Disk System Owner's Manual. If you have a Disk System but you are not going to load TRSDOS, read the instructions later in this chapter.

The Computer and all peripherals must be **off**.

First turn on all peripherals, then turn on the Computer. (If you have all the components connected to a power strip, just turn on the power strip.)

After a few seconds, the following message should appear on the Video Display:

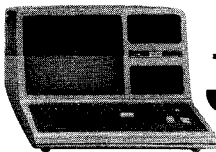
Cass?

The meaning of this message will be explained later.

If the message does not appear:

- A. The Video Display may need Brightness or Contrast adjustment. See Figure 1 for location of these controls.
- B. If the message still doesn't appear, then turn off the entire system, recheck all connections, and try again. For further assistance, see "Troubleshooting and Maintenance."

Do not turn any peripherals off while the Computer is in use; to do so could cause abnormal operation (the Computer could restart or "hang up", requiring you to reset or turn the system off and on again).



TRS-80 MODEL III

RESET

RESET is the orange-colored button at the upper right corner of the keyboard. To “start over” at the power-on message, you do not have to turn the unit off and on again. Pressing the RESET button will have the same effect.

Note: Resetting the Computer does not erase the contents of RAM. However, the BASIC language interpreter will start over, thus “losing” any program or data you had in memory.

To interrupt a program or operation **without** losing your BASIC program and data, hold down the **BREAK** key.

Power-Off

First turn off the Computer, then all other peripherals.

If you turn the Computer off for any reason, leave it off for at least 15 seconds before turning it back on again. The Computer’s power supply needs this time to discharge its stored energy before starting up again.

Whenever you turn off the Computer, all programs and data are erased. So be sure to save your information (e.g., on cassette) before turning off the Computer.

Start-Up Dialog

When you turn on or reset the Computer, it asks you two questions. First:

Cass?

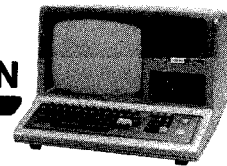
This question lets you determine the rate at which programs and data will be transferred to and from cassette. You can select either Low (500 baud) or High (1500 baud). Type

L

for Low, or

H

for High.



If you press **(ENTER)** without typing anything, High will be used.

For further details, see "Using the Cassette Interface."

Next the Computer will ask:

Memory Size?

This question lets you set an upper limit to the RAM which will be used to store and execute your BASIC programs. Simply press **(ENTER)** in response to this question. This tells the Computer to make the full amount of RAM available for use by your BASIC program.

Advanced programmers may want to reserve some memory for a machine-language ("Z-80") program or subroutine. Instructions for doing this are included in the "Technical Information" chapter.

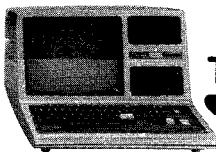
After you respond to the "Memory Size" question, BASIC will start with this message:

```
Radio Shack Model III Basic  
(c) '80 Tandy  
READY  
>
```

The Computer is now ready for use.

Special Instructions for Disk System Owners Using Model III without TRSDOS

If you have a disk drive and disk controller installed, hold down the **(BREAK)** key whenever you turn on or reset the Computer. This tells the Computer that you are not going to use the disk capability.



Modes of Operation

BASIC has four modes of operation:

- Immediate mode—for typing in program lines and immediate lines
- Execute mode—for execution of programs and immediate lines
- Edit mode—for editing program and immediate lines
- System mode—for loading machine-language tapes and for transferring control to machine-language programs

Immediate Mode

Whenever you enter the immediate mode, BASIC displays a header and a special prompt:

```
READY          (header)
> ■           (prompt followed by blinking block "cursor")
```

While you are in the immediate mode, BASIC will display the prompt at the beginning of the current logical line (the line you are typing in).

In the immediate mode, BASIC does not take your input until you complete the logical line by pressing **ENTER**. This is called "line input", as opposed to "character input".

Interpretation of an Input Line

BASIC always ignores leading spaces in the line—it jumps ahead to the first non-space character. If this character **is not** a digit, BASIC treats the line as an immediate line. If it *is* a digit, BASIC treats the line as a program line.

For example:

```
PRINT "THE TIME IS"; TIME$ ENTER
```

BASIC takes this as an immediate line.

If you type:

```
10 PRINT "THE TIME IS"; TIME$ ENTER
```

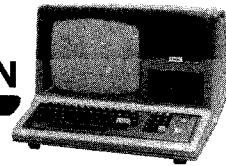
BASIC takes this as a program line.

Immediate Line

An immediate line consists of one or more statements separated by colons. The line is executed as soon as you press **ENTER**. For example:

```
CLS: PRINT "THE SQUARE ROOT OF 2 IS"; SQR(2)
```

is an immediate line. When you press **ENTER**, BASIC executes it.



Program Line

A program line consists of a line number in the range [0,65529], followed by one or more statements separated by colons. When you press **(ENTER)**, the line is stored in the program text area of memory, along with any other lines you have entered this way. The program is not executed until you type RUN or another execute command. For example:

```
100 CLS: PRINT "THE SQUARE ROOT OF 2 IS"; SQR(2)
```

is a program line. When you press **(ENTER)**, BASIC stores it in the program text area. To execute it, type:

```
RUN (ENTER)
```

Special Keys in the Immediate Mode

(?) = PRINT The question mark can stand for the commonly used keyword PRINT. For example, the immediate line:

```
? "HELLO."
```

is the same as the immediate line:

```
PRINT "HELLO."
```

Note: L? does *not* mean LPRINT.

This abbreviation can be used in a program, too.

(.) The period can stand for "current program line", i.e., the last program line entered or edited. The period can be used in most places where a line number would normally appear. For example, the immediate line:

```
LIST.
```

tells BASIC to list the current program line.

(') The single-quote tells BASIC to ignore the rest of the logical line. It is an abbreviation for the BASIC keyword REM. When used in a multi-statement line, it does not have to be preceded by a colon. For example, when you type in the line:

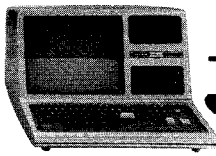
```
PRINT 1 + 1; '2 + 2
```

BASIC will print the sum 1 + 1 but not 2 + 2.

This abbreviation can be used in a program, too.

(SHIFT) (D) (C)
S P

Causes the Computer to print the Display contents to the line printer, if available. Press **(BREAK)** to interrupt this operation. This key sequence works in the other modes too.



Execute Mode

Whenever BASIC is executing statements (immediate lines or programs) it is in the execute mode. In this mode, the contents of the Video Display are under program control.

Special Keys in Execute Mode

SHIFT @ Pauses execution. Press any key to continue.

BREAK Terminates execution and returns you to the command mode.

Edit Mode

BASIC includes a line editor for correcting program lines. To edit a program line, type in the command:

EDIT *line number*

where *line number* specifies the desired line.

When the editor is working on a program line, it displays the number of the line being edited.

In the edit mode, the Keyboard input is character-oriented, rather than line-oriented. That is, BASIC takes characters as soon as they are typed in—without waiting for you to press **ENTER**.

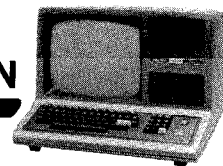
See the chapter on editing (Section 2) for details.

System Mode

In this mode, you can load and execute machine-language programs. By “machine-language”, we mean the set of machine instructions recognized by your Computer’s Z-80 microprocessor. In this manual, we will usually call it “Z-80” programming, in contrast to BASIC programming.

You don’t have to understand the Z-80 language to use some of the programs available. For example, several Radio Shack games are written in Z-80 code rather than in BASIC. To load such programs from tape, you use the System Mode.

Z-80 programming opens up whole new worlds of possibilities, but it is somewhat more demanding than BASIC programming.



The Technical Information chapter in this manual is written for those who are familiar with the Z-80 instruction set and other fundamental machine concepts. If you would like to explore these subjects, read:

TRS-80 Assembly Language Programming, by William Barden, Jr. Radio Shack Catalog Number 62-2006.

Although the book was originally written for the TRS-80 Model I, it applies almost exactly to the Model III as well.

For further details, see “Cassette Interface” in this Operation Section, and SYSTEM in the Language Section.

Sample Session

This section will give you a step-by-step example of what’s needed to type in a program and run it. We will be showing you the Computer/operator dialog exactly as it appears on the Display. If you have never used a computer keyboard before, read **Using the Keyboard** before trying this sample session.

You don’t need to know BASIC programming to go through this session—it is just an exerciser. If you are curious about the words used in this program, look them up on the Quick Reference Card supplied with your Computer, or in the Index of this manual.

Special Notation Used in this Dialog

BOLDFACE MATERIAL

Provided by the Computer—you don’t type it in.

ENTER

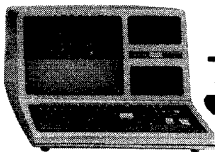
Means “Press the **ENTER** key.”

SHIFT **0**

This tells you to use the upper/lower case—caps only switch. You do this by pressing **SHIFT** and **0** together.

→

This means “press the **→** key” to skip over to the next eight-column boundary. We usually do this just for visual effect.



TRS-80 MODEL III

Answering the Start-Up Questions

Reset the Computer. Then follow this session.

Cass? (ENTER)

Memory Size? (ENTER)

Radio Shack Model III Basic

(c) '80 Tandy

READY

> ■

The blinking block after ">" is the "cursor". It tells you where the next character you type will be displayed.

Now continue:

>NEW (ENTER)

READY

>AUTO (ENTER)

10 CLS (ENTER)

20 PRINT "HI—I'M YOUR TRS-80 MICROCOMPUTER!" (ENTER)

30 PRINT "(SHIFT) @ What makes me so smart?" (SHIFT) @ (ENTER)

40 PRINT "(SHIFT) @ Millions of these:" (SHIFT) @ (ENTER)

50 PRINT CHR\$(21) (ENTER)

60 FOR I = 1 TO 256 (ENTER)

70 (←) PRINT CHR\$(253); CHR\$(254); (ENTER)

80 NEXT I (ENTER)

90 PRINT CHR\$(21) (ENTER)

100 END (ENTER)

110 (BREAK)

READY

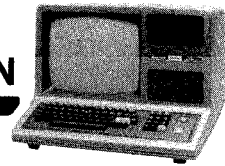
> ■

Now the program is in memory. To look at it, type:

>LIST (ENTER)

It should look like this:

```
10 CLS
20 PRINT "HI! I'M YOUR TRS-80 MICROCOMPUTER!"
30 PRINT "What makes me so smart?"
40 PRINT "Millions of these:"
50 PRINT CHR$(21)
60 FOR I = 1 TO 256
70     PRINT CHR$(253); CHR$(254);
80 NEXT I
90 PRINT CHR$(21)
100 END
```



Check each line. Don't worry about spacing; however, if anything else is different, simply re-type the incorrect line. For example, suppose you mistakenly type in line 90 like this:

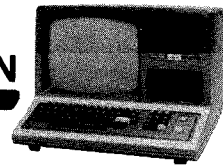
```
90 PRINT CHR$(201)
```

To correct it, simply type:

```
>90 PRINT CHR$(21) ENTER  
> ■
```

When everything is correct, you can run the program by typing:

```
>RUN ENTER
```



4 / Using the Keyboard

The keyboard allows entry of all the standard text and control characters. As with ordinary typewriters, use **(SHIFT)** to enter the upper symbol on those keys containing two symbols. For example, to enter a "!", press **(SHIFT)** **(1)**.

Capitals and Lower Case **(SHIFT)** **(0)**

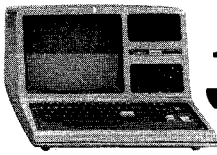
The A-Z keys can produce either upper or lowercase characters. There are two modes of operation: CAPS, in which the A-Z keys always produce capital letters; and ULC (upper/lowercase), in which the A-Z keys produce lowercase unless you press **(SHIFT)**.

When you start the Computer, the keyboard is in the CAPS mode. To switch to ULC, press **(SHIFT)** **(0)**. To switch back, press **(SHIFT)** **(0)** again. **(SHIFT)** **(0)** is a "toggle": each time you press it, you switch from one mode to the other.

Special Keys

Certain keys have special functions in BASIC. Rather than accepting them as keyboard data, BASIC performs the specified function.

Key	Function
(←)	Backspaces and erases the last character typed.
(→)	Tabs over to the next eight-column boundary.
(SHIFT) (←)	Starts over at the beginning of the line.
(SHIFT) (→)	Converts to 32 characters/line.
(SHIFT) @	Pauses program execution. Press any key to continue.
(ENTER)	Enters the line. BASIC will not interpret a line until you press (ENTER) .
(CLEAR)	Cancels the current line, erases the display, converts to 64 characters/line, and positions the cursor to the upper left corner ("home").



TRS-80 MODEL III

Special Keys, continued.

BREAK

Interrupts the current program or operation and prepares the Computer for another keyboard command. Use to cancel a cassette or line printer operation, or to break out of a BASIC program.



Activates the Print Screen function, copies the contents of the Screen to the Printer. Press **BREAK** to terminate this function and return to the immediate mode.

Other Features

Every key has a repeat feature: when you hold a key down for approximately one second, that key begins producing a stream of characters.

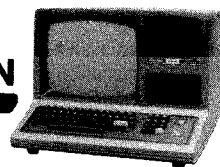
The keyboard includes a 12-key section for convenient numeric entry. Each of these keys is equivalent to the matching key on the standard keyboard section.

Control Codes*

* If you are unfamiliar with the concept of character codes, see the ASCII entry in the Glossary (Appendix). Also see the table of character codes in the Appendix.

You can produce 32 special control characters (ASCII Codes 0-31) from the Keyboard. For example,

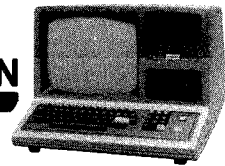
Key	ASCII Name	Code
	Backspace	8
	Tab	9
	Line Feed	10
ENTER	Carriage Return	13



You are not limited to these specially labeled keys. A special two-key combination allows the regular text keys to create additional control characters. Use this procedure:

1. Hold down **SHIFT**
2. Hold down **↓**
3. While holding down **SHIFT** and **↓**, press the desired character. For example:
SHIFT ↓ C = "Control C" = Code # 3.

For a complete list of keyboard characters available, see the Appendix.



5 / Using the Video Display

Character Size

There are 16 lines on the display, and two character sizes: normal (64 characters per line—"cpl"), and double-size, or 32 cpl.

The Computer starts in the 64 cpl mode. To change to 32 cpl, press **(SHIFT)←** in the immediate mode or execute the BASIC statement:

```
PRINT CHR$(23)
```

To return to 64 cpl, press **(CLEAR)** in the command mode, or execute the BASIC statement:

```
CLS
```

Cursor

The cursor indicates the current display position. When you start BASIC, the cursor is a blinking block. You can change the cursor character and you can make it solid (non-blinking).

Memory location 16412 contains the blink/non-blink status. When it contains a zero, a blinking cursor will be used. When it contains a non-zero value, a non-blinking cursor will be used.

For example, to make a solid cursor, execute the BASIC statement:

```
POKE 16412, 1
```

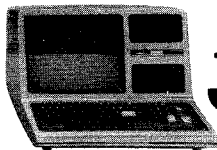
To make a blinking cursor, execute the BASIC statement:

```
POKE 16412, 0
```

Memory location 16419 contains the ASCII code of the cursor character. When you start BASIC, this address contains 176. To change the cursor, use the POKE statement. For example,

```
POKE 16419, 63
```

changes the cursor to a "?", since 63 is the ASCII code for a question-mark.



TRS-80 MODEL III

You can select any ASCII code from zero to 255.

To restore the cursor to its original character, execute this BASIC statement:

```
POKE 16419, 176
```

To turn the cursor on in the execute mode, execute the statement

```
PRINT CHR$(14)
```

To turn it off, use

```
PRINT CHR$(15)
```

Scroll Protection

Display "scrolling" occurs when the Computer moves all the text up one line to make room for a new line on the bottom row of the Display. When scrolling occurs, the top line on the Display is erased from the Display.

The Model III will let you protect from scrolling up to seven lines on the top of the Display. For example, suppose you are printing a table. You can put the column headings in a scroll protect area, so they will not be lost when scrolling takes place.

Memory location 16916 controls the size of the scroll protect area. A zero in this one-byte location means no lines are protected. A one means one line (the top line) is protected. And so forth.

For example, to protect the top four lines from scrolling, execute the BASIC statement:

```
POKE 16916, 4
```

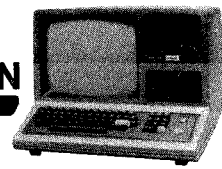
To restore the display to its original condition (no scroll-protect), execute the BASIC statement:

```
POKE 16916, 0
```

If you store a value greater than seven in this address, the Computer interprets the value in modulo eight. That is, the number is divided by eight and the remainder is used.

The following program demonstrates the scroll-protect feature:

```
10 CLS: POKE 16916, 3          'PROTECT TOP 3 LINES
20 PRINT "THESE TOP THREE LINES WILL NOT BE SCROLLED"
30 PRINT "BUT THE REST OF THE SCREEN WILL."
40 PRINT "-----"
50 FOR I = 1 TO 100
60 PRINT "THIS LINE IS IN THE NON-PROTECTED AREA SO WILL SCROLL"
70 NEXT I
80 POKE 16916, 0              'REMOVE SCROLL PROTECTION
```



Text Characters


The Model III Display can produce the standard ASCII text characters, including the upper and lowercase alphabet.

All text characters are created on an eight-by-eight matrix for excellent definition.

The following BASIC program will display all 96 text codes and characters:

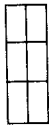
```
10 CLS
20 FOR I = 32 TO 127
30   PRINT @ (I-32) * 8, I; CHR$(I);
40 NEXT I
```

Many of these characters can be keyed in directly from the keyboard; others can only be generated by reference to their ASCII codes.

Note: The  key is echoed on the display as [instead of as an up-arrow. This is because Model III produces standard ASCII characters on its display. However, if the program calls for an up-arrow, the left-bracket will serve the same purpose.

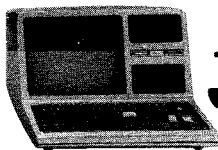
Graphics Characters

The Model III Display has 64 graphics characters, consisting of all possible on-off combinations in a two-by-three matrix:



The graphics characters are produced by codes 128 through 191. The following program will display them all:

```
10 CLS
20 FOR I = 128 TO 191
30   PRINT @ (I-128) * 8, I; CHR$(I);
40 NEXT I
```



Space Compression Characters

When you start BASIC, characters 192 through 255 are defined as space compression codes: 192 generates zero spaces; 193, one space; and so forth, up to 255, which generates 63 spaces.

These codes are useful for storing Video Display text in a minimal amount of memory. For example, the following line contains 55 characters (superior numbers indicate the number of blank spaces between letters):

<i>21 spaces</i>	<i>18 spaces</i>	
NAME	ADDRESS	PHONE

There are two sequences of blanks containing a total of 39 characters. By replacing the two space-sequences with two compression codes, we can save $39 - 2 = 37$ characters.

When the data is displayed, the space compression codes will be "expanded" into the appropriate number of spaces.

The following BASIC program illustrates this example:

```

5  CLS
10 POKE 16526, 105           'LSB OF $INITIO ENTRY ADDRESS
20 POKE 16527, 0            'MSB
30 X =USR(0)                 'CALL $INITIO
40 CLEAR 100
50 A$ = "NAME" + CHR$(192+21) + "ADDRESS" + CHR$(192+18) +
  "PHONE"
60 PRINT "THE LENGTH OF THE STRING IS"; LEN(A$)
70 PRINT "HERE IT IS:"
80 PRINT A$

```

Special Characters

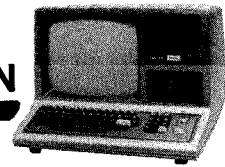
The Model III also features 96 special characters. The first 32 may be displayed by POKEing the appropriate code into video RAM (addresses 15360 to 16383); the remaining 64 may be displayed via the PRINT statement.

This program will display the first 32:

```

10 CLS
20 FOR I = 0 TO 31
30   POKE 15360 + I * 16, I
40 NEXT I
50 PRINT @ 640, " ";

```



The remaining 64 must first be "switched in" and then may be displayed via PRINT. Codes 192 through 255 normally function as space compression codes; however, a software switch will activate the special character set. The statement:

```
PRINT CHR$(21)
```

switches back and forth between space compression and special characters.

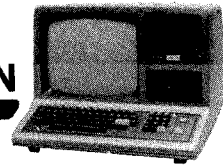
Another software switch selects an alternate set of special characters (Japanese Kana characters). Each time you execute the statement

```
PRINT CHR$(22)
```

the active/inactive sets are swapped.

The following program will switch in the special characters and display both sets of them.

```
5 CLS
10 POKE 16526, 105           'LSB OF $INITIO ENTRY ADDRESS
20 POKE 16527, 0            'MSB
30 X =USR(0)                 'CALL $INITIO
40 PRINT CHR$(21)           'SWITCH IN SPECIAL CHARACTERS
50 INPUT "PRESS <ENTER> TO SEE SPECIAL CHARACTERS"; X
60 FOR I = 192 TO 255
70 PRINT CHR$(I);
80 NEXT I
90 PRINT
100 INPUT "PRESS <ENTER> TO SWITCH TO ALTERNATE SET"; X
110 PRINT CHR$(22);         'SWITCH IN ALTERNATE SET
120 INPUT "PRESS <ENTER> TO RETURN TO NORMAL AND END"; X
130 PRINT CHR$(22); CHR$(21)
```



6 / Using the Cassette Interface

Model III's built-in cassette interface allows you to store data and programs with a cassette recorder such as Radio Shack's CTR-80A, Catalog Number 26-1206.

Connect the recorder to the Computer according to Figure 1 in this manual; for further connection instructions, refer to the cassette recorder owner's manual.

Cassette Transfer Speed

As explained previously, you select either Low or High cassette speed when you start BASIC.

If you want to load Model I Level II programs, you must select Low.

(The actual speed for Low is 500 baud, which is approximately 63 characters per second; for High, 1500 baud, or 190 characters per second. For short programs, you won't notice a three-to-one difference in loading times, due to the "overhead" required by any taped data. However, for longer programs, the difference in loading/saving times will approximate three-to-one.)

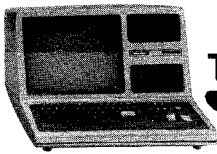
You do not have to restart BASIC to change the cassette speed. This speed is determined by the contents of memory address 16913. When this one-byte location contains zero, Low speed (500 baud) is used; when it contains any non-zero value, High speed (1500 baud) is used.

For example, to select 500 baud, execute the BASIC statement:

```
POKE 16913, 0
```

To select 1500 baud, execute the BASIC statement:

```
POKE 16913, 1
```

Loading Errors

There are three messages that may appear in the upper right of the Display during a tape input operation. They tell you that the tape operation was unsuccessful and needs to be repeated.

Message	Meaning
C*	Checksum Error during loading of a SYSTEM tape
D*	Data Error during loading of a BASIC program
BK	You pressed (BREAK) and cancelled the operation

The first two errors may be caused by an incorrect volume setting. Adjust the volume and try again. If you still have problems, recheck the cassette recorder connections. Another possible cause is dirty recorder heads. Clean the heads as explained in the cassette owner's manual. If none of this helps, the data on the tape may have been destroyed by static electricity or some other cause.

Saving a BASIC Program on Tape

When you want a long-term copy of a BASIC program (one that won't have to be typed in again), simply save it on tape with the `CSAVE` command.

The program should be in memory. Be sure you have selected the desired cassette transfer speed (500 or 1500 baud). In general, you should use 1500 baud, since it is faster and requires less tape.

1. Insert a blank cassette into the recorder (use Radio Shack's leaderless tape for best results).
2. Prepare the recorder to RECORD.
3. Type :

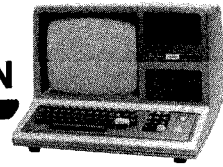
`CSAVE "P" (ENTER)`

The Computer will save the program on tape.

When the process is completed, the Computer will display:

```
READY  
>■
```

In this example, we used "P" as the file name; you can choose any single character except a double-quote. Enclose the character in double-quotes as shown in our example.



It is a good idea to save the program at least twice, preferably on separate cassettes. That way, if one cassette is lost or erased, you have an extra copy.

When you want to load the program in later, you can specify the file name, in which case BASIC will search for that file name; or you can omit the file name, in which case BASIC will load the first program on the tape.

Loading a BASIC Program from Tape

Be sure the Computer's cassette speed matches that of the recorded program (the speed at which it was CSAVED).

1. Prepare your recorder to PLAY the recorded cassette. Adjust the volume to the level recommended for 500 or 1500 baud. See Figure 2 on the next page.
2. Type:

CLOAD **(ENTER)**

The Computer will load the first program on the tape. While the program is loading, two asterisks will appear on the upper right of the Display. The one on the right will blink after every 64th character of data is received.

When the program is loaded, the Computer will display the message:

READY
>■

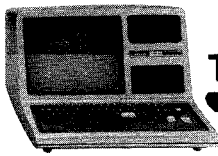
3. Type:

LIST **(ENTER)**

to list the program you have just loaded (just for verification).

4. You may now run the program by typing:

RUN **(ENTER)**



How to Search for a Program

If the tape contains different programs on the same side, you can make the Computer search through them until it reaches the desired program. To do this, just specify the name of the program. For example, if the program is named "P", then type in this command:

CLOAD "P" (ENTER)

While the Computer is skipping a non-matching program, it will display the file name of that program.

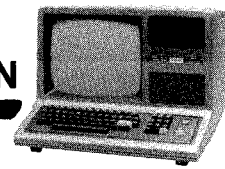
Note: If the program you named is not on the tape, the Computer will continue to wait for it, even after the tape has run out. Hold down the (BREAK) key until the Computer returns with the message:

READY

> ■

Recorder Model	User-Generated	Pre-Recorded From Radio Shack
CTR-80, 80A	5-7	4-6

Figure 2. Recommended levels for loading programs from tape.



Loading a SYSTEM Tape

In addition to BASIC programs, you may load machine-language programs from tape. Such programs are stored in a different format on the tape; we call them SYSTEM tapes. Radio Shack sells several machine-language programs on cassette, for example, Micromusic and Editor/Assembler.

You can also create your own SYSTEM tapes, using the Editor/Assembler Package.

Before loading the tape, be sure the Computer's cassette speed matches that of the recorded program.

1. Prepare your recorder to PLAY the recorded cassette. Adjust the volume to the level recommended in Figure 2.

2. Type:
SYSTEM **(ENTER)**

The Computer will display the monitor mode prompt:

*?

3. Type in the program's file name. For example, if the program is named EDTASM, you would type:

EDTASM **(ENTER)**

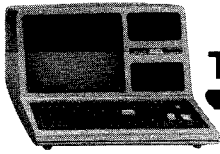
The Computer will load the program. While the program is loading, two asterisks will appear on the upper right of the Display. The one on the right will flash after every 64th character of data is received.

4. When the Computer has loaded the program, it will display another monitor prompt:

*?

What you do next depends on the program you have just loaded.

- A. If you want to load another program, then prepare the next cassette tape and repeat Step 3.
- B. If you want to return to BASIC, then press **(BREAK)**.
- C. If you want to run the machine-language program you just loaded, then type in a slash symbol "/" followed by the "entry address" and press **(ENTER)**, or simply type in the "/" and press **(ENTER)**. Specific instructions will be provided with the SYSTEM tape.



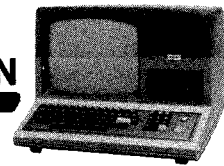
TRS-80 MODEL III

For example, to start the program at address 32000, type:

*? / 32000 **(ENTER)**

To start the program at the address specified by the SYSTEM tape, type:

*? / **(ENTER)**



7 / Using A Line Printer

Any Radio Shack ‘parallel interface’ printer may be connected to the Model III. There are some differences in printer functions available, so check in the printer owner’s manual for these details.

Line Printer vs Video Display

Output

Output to the line printer is similar to display output; in fact, for the two major display output operations, there are two matching line printer output operations:

Video Display	Line Printer
PRINT	LPRINT
LIST	LLIST

These are described in the BASIC Language Section of this manual.

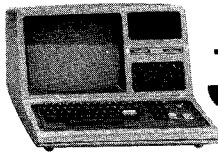
When you try to output information to the printer, the Computer will first see if a printer is connected and ready to accept the data. If it is not, the Computer will simply wait until the printer is available. During this time, you will not be able to type in instructions from the keyboard.

To regain keyboard control in this situation, hold down the **(BREAK)** key until the Computer displays

```
READY  
>
```

Certain of the Video Display features are not available on the printer. For example:

- The graphics and special character sets cannot be output to the printer. However, your printer may have its own special characters or ‘graphics’. Check in the owner’s manual.
- The CLS and PRINT @ statements have no line-printer counterparts.



Printer Control Features

Output to a printer involves several variables:

- Maximum line width (How many print columns are there?)
- Page length (How many print lines are on a page?)
- Printer status (Is the printer connected and ready to receive data and print it?)

In this section, we will explain how to set up the Model III to control all these variables.

Setting the Maximum Line Length

In Model III BASIC, you can preset the maximum line length. If a line exceeds the preset length, the Computer will automatically insert an end of line (carriage return) so that the rest of the line will be output on a new line. The following paragraphs explain why you may want to do this.

One important difference between display output and printer output is the maximum line length. (A "line" is a stream of data characters terminated by a carriage return **(ENTER)**.)

The Model III Display has a maximum line length of 64 characters. If you PRINT a line longer than this, the Computer simply "wraps around" to the beginning of the next line.

Printers have a maximum line length, too, but this length differs for various models. The response to an overflow (longer than maximum-length) line also varies. Some models wrap around to the next line automatically. Others may lose the extra data, and may begin abnormal operation when the line is too long.

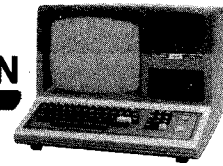
Another consideration is paper width. Suppose your paper is only wide enough to hold 80 characters—but the printer will accept lines of up to 132 characters. In this case, if you send a line longer than 80 characters, the printer will print part of the information past the edge of the paper.

How to Set the Line Length

Memory address 16427 contains a value equal to the maximum line length less two. For example, to set the maximum line length to 64, execute the BASIC statement:

```
POKE 16427, 62
```

Since the Display is 64 characters per line (cpl), this setting will make line printer output match Video Display output.



When address 16427 contains a value of 255, the maximum line length feature is disabled. No matter how long the line is, the Computer will not insert carriage returns in it. Remember, though, some printers automatically do this when the line exceeds a specified length.

When you start BASIC, address 16427 contains a value of 255, so the maximum line length function is disabled.

Page Controls

In many printer applications, you want to control the number of lines that are printed on a page. For example, in printing forms or reports, when a given number of lines have been printed, you want to advance the paper to the top of the next page.

Model III BASIC has several features to help you do this. It keeps track of the following information:

Data	Memory Address
Page size: number of lines per page plus one. Initialized to 67 = 66 + 1.	16424
Line count: number of lines (carriage returns) already printed plus one. Initialized to one.	16425

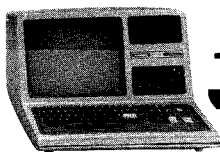
Most printers output six lines per inch; therefore standard 11" paper allows 66 lines, which matches BASIC's initialization value.

To change the maximum lines/page setting, store the desired number of lines plus one in 16424. For example, if your paper contains 88 lines per page, then execute this BASIC statement:

```
POKE 16424, 89
```

When you start the Computer, position the paper to the top of the page ("top of form"). That way BASIC's initial page information is correct. Each time BASIC outputs a line (i.e., a carriage return), the line count is incremented.

Note: If your printer's maximum line-length is shorter than BASIC's maximum line length, the printer will insert carriage returns that BASIC isn't allowing for. Therefore BASIC's line count will not be accurate.



TRS-80 MODEL III

To prevent this from happening, make sure BASIC's maximum line length (stored in address 16427) is no greater than that of your printer. You can find your printer's maximum line length in the printer owner's manual.

To do an automatic top of form (advancing the paper to the top of the next page), print the ASCII "Form Feed" code, decimal 12. For example, execute the BASIC statement:

```
LPRINT CHR$(12)
```

The paper will advance by the following amount:

$$\text{Top of Form} = \text{Max. lines/page} - \text{Lines already printed}$$

Each time you print a form feed, CHR\$(12), BASIC resets the line count automatically.

Sometimes you may want to reset the line count, for example, after manually advancing the paper to the top of form. To do this, store a one in 16425:

```
POKE 16425, 1
```

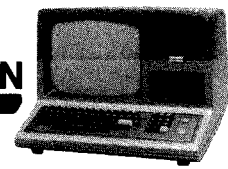
Checking the Printer Status

Unlike the Video Display, the printer is not always available. It may be disconnected, off-line, out-of-paper, and so forth. In such cases, when you try printer output, the Computer will wait until the printer becomes available. It will appear to be "locked up". To regain keyboard control (and cancel the printer operation), press **BREAK**.

Suppose you have a program which uses printer output. If a printer is not available, you don't want the Computer to stop and wait for it to become available. Instead, you may want to print a message like "PRINTER UNAVAILABLE" and stop.

To accomplish this, you need to check the printer status. The status is stored in address 14312. AND this value with 240. The result should equal 48. If it doesn't, that means the Printer is unavailable for some reason, and printer output is not possible. For example, your program could execute these statements:

```
100 ST% = PEEK(14312) AND 240
120 IF ST% < > 48 THEN PRINT "PRINTER UNAVAILABLE.": STOP
130 PRINT "PRINTER IS AVAILABLE"
```



Print Screen Function

Model III has a very handy feature to give you a "snapshot" of whatever is on the Display. It will work whenever the Computer is scanning the keyboard (BASIC's Immediate, Execute, Edit and System Modes). It does not work during cassette, printer or serial I/O.

When you want to copy the Display contents to the printer, simply press:



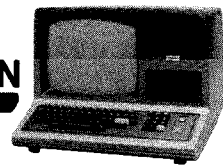
together. The Computer will stop what it's doing and print the screen.

The Computer will print the entire display, blanks and all. If you are only interested in printing the top portion of the display, press **BREAK** when those lines have been printed.

If a printer is not available, the Computer will wait until it becomes available or until you press **BREAK**.

If the Display contains special characters or graphics characters, they will be displayed as periods.

Note: You can also activate the Print-Screen function via the BASIC USR function. See \$PRSCRN in the Technical Information chapter.



8 / Using the RS-232-C Interface

What is an Interface?

It's a generalized means of communication between your TRS-80 and some external device, providing the necessary conventions regarding data-identification, transmission rates, send-receive sequences, error-checking techniques, etc. However, an Interface does **not** provide the programming necessary to **use** any particular TRS-80/ external device system.

For example, having the Interface installed does **not** automatically enable you to send BASIC programs from one TRS-80 to another; to output to a line printer via the Interface; etc. Such applications require "driver programs" which must be custom-designed for the equipment you intend to use.

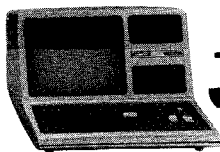
The Radio Shack RS-232-C Interface is designed to meet the EIA standards. However, we cannot guarantee that it will work with all so-called "RS-232-C compatible" devices. Nor do we commit ourselves to provide engineering and programming support for such applications, or other special custom-use situations.

We do, however, guarantee that our Interface will function correctly with all our own RS-232-C equipment.

The term RS-232-C refers to a specific EIA (Electronics Industries Association) standard which defines a widely accepted method for interfacing data terminal equipment with data communications equipment. The RS-232-C Interface is by far the most universally used standard for interfacing data processing equipment. Most video terminals, modems, card readers, line printers, mini-microcomputers, etc., utilize the RS-232-C standard for data interchange between devices.

Adding the RS-232-C to your Model III TRS-80 opens up a whole new world of compatibility. The Computer can then be programmed to communicate with a serial printer, telephone modem, serial display terminal — almost any RS 232-C device.

Note: The following information applies only if your Model III TRS-80 is equipped with the RS-232-C Interface.



Using the Model III as a Terminal

Probably the most common use of the RS-232-C interface will be to allow the Model III to act like a "terminal" to another "host" computer. In this application, whatever you type on the keyboard is sent via RS-232-C to the other host computer, and whatever the host computer sends to you is displayed on your screen.

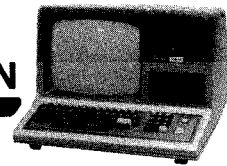
Before going into the details of RS-232-C operation, we'll show you a BASIC program that sets up a **simplified** terminal operation.

1. Make sure the RS-232-C characteristics are set to match those of the host computer. If they are not, then change them, as explained later in this chapter.

Note: For this BASIC Program, you must use a baud rate of 110. An equivalent Z-80 program could use any baud rate.

2. Connect the Model III to the host computer via the RS-232-C. You will need a telephone interface (modem) or other means of communication.
3. Type in and run the following BASIC program (you do not need to type in the comments (material that starts with a single quote). The program displays characters received via the RS-232-C, and sends characters you type in. It is for demonstration only, and is not meant to function as a practical terminal. Notice there are *no spaces* between the " " in line 160.

```
5 DEFINT A-Z          ' INTEGER VARIABLE FOR SPEED
10 POKE 16890, 0      ' DON'T WAIT FOR SERIAL I/O
15 POKE 16888, (2*16)+2 ' TX/RCV AT BAUD RATE 110
17 U1 = 16526        ' LSB OF USR CALL ADDRESS
18 U2 = 16527        ' MSB OF USR CALL ADDRESS
20 POKE U1, 90       ' SET UP USR CALL, LSB
30 POKE U2, 0        '          MSB
40 X = USR(0)        ' CALL $RSINIT
50 RCV = 80          ' LSB OF $RSRCV
60 TX = 85           ' LSB OF $RSTX
70 CI = 16872        ' CHARACTER INPUT BUFFER
80 CO = 16880        ' CHARACTER OUTPUT BUFFER
90 ' CHECK FOR SERIAL INPUT
100 POKE U1, RCV     ' SET UP USR CALL TO $RSRCV
110 X = USR(0)       ' CALL $RSRCV
120 C$ = CHR$(PEEK(CI)) ' LOOK AT INPUT BUFFER
130 PRINT C$;       ' IF C = 0, NOTHING HAPPENS
140 ' CHECK FOR KEYBOARD INPUT
150 C$ = INKEY$
160 IF C$ = "" THEN 100 ' NO KEY, SO GO CHECK SERIAL
165 PRINT C$;       ' DELETE THIS LINE IF HOST PROGRAM
166                ' HAS AN ECHO FEATURE
170 POKE CO, ASC(C$) ' PUT CHAR. INTO OUTPUT BUFFER
180 POKE U1, TX      ' SET UP USR CALL TO $RSTX
190 X = USR(0)      ' CALL $RSTX
200 GOTO 100        ' GO CHECK SERIAL INPUT
```



Programming the RS-232-C Interface

In this section, we will treat the RS-232-C just like any other input/output device, and will explain how your BASIC program can use it. In Technical Information, we explain how to use it in a machine-language ("Z-80") program.

For details about the RS-232-C signal conventions and theory of operation, see the Appendix.

Selecting the RS-232-C Characteristics

Before using the RS-232-C interface to communicate with another device, you must be sure your RS-232-C is set up to match the requirements of the other device.

So start by getting the following information about the other device. In the right column, we list typical values used.

Characteristic	Typical Values Used
Baud Rate	110, 150, 300, 600, 1200, 2400, 4800, 9600
Word Length (bits)	5, 6, 7, 8
Parity	Even, Odd, None
Stop Bits	1, 2

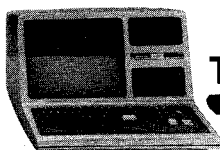
When you start the Computer, the RS-232-C is initialized to the following "default characteristics":

Baud Rate	300
Word Length (bits)	8
Parity	None
Stop Bits	1

In addition, the RS-232-C is initialized to wait for completion of character I/O before returning. That is, if you attempt to receive a character, the Computer will wait until a character is received; it will never return to you without a character.

Similarly, if you attempt to send a character, the Computer will wait until the receiving device is able to accept the character.

To regain control of the Computer during a wait, hold **BREAK** until READY returns.



I/O to the RS-232-C Interface

If the default settings are correct, you are ready to begin serial I/O. To change any of the settings, you need to re-initialize the RS-232-C interface. See "To Change the RS-232-C Characteristics".

There are two ROM subroutines for serial I/O (both were used in the simple terminal program):

\$RSTX	Send a character
\$RSRCV	Receive a character

Both subroutines are simple to use from BASIC via the USR function.

To Send a Character

1. The Computer should be connected to the serial device.
2. Define a USR call to \$RSTX (address 85) by executing these BASIC statements:

```
POKE 16526, 85  
POKE 16527, 0
```

3. Send the character by storing the ASCII code in memory location 16880. Suppose A\$ contains the character. Then execute this statement:

```
POKE 16880, ASC(A$)
```

4. Make the USR call with a dummy argument:

```
X = USR(0)
```

If the Computer is using the Don't Wait procedure, then control will return to BASIC even if the character was not sent. If the Computer is using the Wait procedure, control will return to BASIC after the character is sent.

5. Repeat steps 3 and 4 until all the data has been sent.

To Receive a Character

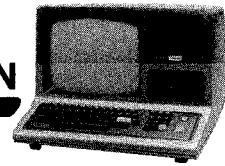
1. The Computer should be connected to the serial device.
2. Define a USR call to \$RSRCV (address 50) by executing these BASIC statements:

```
POKE 16526, 50  
POKE 16527, 0
```

3. Get the character by making the USR call with a dummy argument. For example:

```
X = USR(0)
```

Upon return from the subroutine, USR returns the ASCII code of the character received in memory location 16872. A zero indicates no value was received.



If the Computer is using the Don't Wait procedure, then control will return to BASIC even if no character was received. If the Computer is using the Wait procedure, control will return to BASIC after a character is received. Press **(BREAK)** to interrupt a WAIT and regain keyboard control of the Computer.

4. To make this character available to BASIC, execute a BASIC statement like:

```
A$ = CHR$(PEEK(16872))
```

which stores the string value in A\$. Remember, if A\$ = CHR\$(0), then no character was received.

5. Repeat Steps 3 and 4 until you are through receiving data.

To Change the RS-232-C Characteristics

If the TRS-80's default characteristics do not match the requirements of the other device, you can change some or all of them by using ("calling") an initialization subroutine that is stored in ROM.

Before calling \$RSINIT, you must store the desired characteristics in certain RAM locations:

Address	Contents
16888	Transmit/Receive Baud Rate Code
16889	Parity/Word Length/Stop Bit Code
16890	Wait/Don't-Wait Switch

Transmit/Receive Baud Rate Code

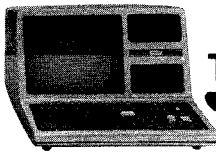
The TRS-80 RS-232-C allows you to receive and transmit at different rates. For most applications, the rates will need to be the same.

Instead of storing the actual baud rate, you store a code for the value, taken from the table below. You select the appropriate codes for send and receive rates, and then "pack" them into memory address 16888 as follows:

$$\text{Send/Receive Code} = (\text{Send Code} * 16) + \text{Receive Code}$$

For example, suppose we want to send and receive at 110 baud. Using the table on the next page, we find that the code for 110 baud is 2. So:

$$\text{Send/Receive Code} = (2 * 16) + 2 = 34$$



TRS-80 MODEL III

In technical terms, we are storing the send-rate code in the most significant four bits ('nibble') of 16888, and the receive-code in the least significant nibble.

Baud-Rate Codes

Desired Baud Rate	Error (%)	Baud Rate Code
50	0	0
75	0	1
110	0	2
134.5	0.016	3
150	0	4
300	0	5
600	0	6
1200	0	7
1800	0	8
2000	0.253	9
2400	0	10
3600	0	11
4800	0	12
7200	0	13
9600	0	14
19200	3.125	15

Parity/Word Length/Stop-Bit Code

You pack all of this information into one byte, using the following formula:

$$\text{Code} = (\text{Parityselect} * 128) + (\text{Word} * 32) + (\text{Stop} * 16) + (\text{Parityonoff} * 8) + (\text{Transmit} * 4) + (\text{DTR} * 2) + \text{RTS}$$

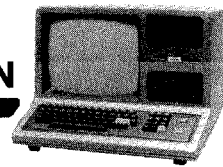
where:

Parityselect = 0 for odd parity
= 1 for even parity

Word = 0 for 5-bit words
= 1 for 6-bit words
= 2 for 7-bit words
= 3 for 8-bit words

Stop = 0 for 1 stop-bit
= 1 for 2 stop-bit

Parityonoff = 0 to enable parity
= 1 to disable parity



- Transmit = 0 to disable the transmitter
 = 1 to enable the transmitter
- DTR = 0 to set Data Terminal Ready signal low
 = 1 to set Data Terminal Ready signal high
- RTS = 0 to set Request to Send signal low
 = 1 to set Request to Send signal high

For example, to select 7-bit words, even parity, two stop-bits, transmit-enable, DTR high and RTS high, calculate the code this way:

$$\text{Code} = (1 * 128) + (2 * 32) + (1 * 16) + (0 * 8) + (1 * 4) + (1 * 2) + (1 * 1) = 215$$

For additional information on how to determine the appropriate code characteristics, read \$RSINIT in the Technical Information Chapter and see Appendix I.

Wait/Don't-Wait Switch

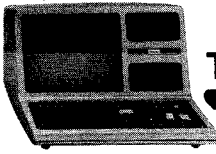
The TRS-80 lets you choose either Wait or Don't-Wait serial I/O.

When you select Wait I/O, the TRS-80 will not return from a serial I/O call until the operation is successful (i.e., a character is transmitted or received). Pressing **(BREAK)** will return control to your program.

When you select Don't-Wait I/O, the TRS-80 will return from a serial I/O call even if the operation was not successful (i.e., no character was transmitted or received).

The contents of memory location 16890 determines which procedure is used:

Contents of 16890	Procedure Used
Zero	Don't-Wait
Non-Zero	Wait



Calling \$RSINIT from BASIC

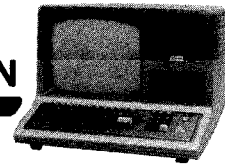
Store (POKE) the desired values into the RS-232-C control addresses (16888-16890). If any of the default characteristics are already correct, leave those addresses unchanged.

If you need to change the parity/word length/stop-bit code, see \$RSINIT in the Technical Information chapter. Once you have calculated the desired codes for baud rate, parity/word length/stop-bits and Wait/Don't-Wait, you are ready to call \$RSINIT.

Execute the following BASIC statements to define aUSR call to \$RSINIT:

```
POKE 16526, 90  
POKE 16527, 0  
X = USR(0)
```

When the last statement has been executed, the RS-232-C is initialized.



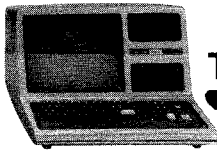
9 / Routing Input/Output

Model III lets you route I/O from one device to another. This gives your programs more versatility.

For example, suppose you have a program that outputs to the Video Display. Now suppose you want all display output to go to the printer. You can accomplish this without changing the program at all, using the route capability. The source device (in our example, the display) will then be logically equivalent to the destination device (printer) until you re-initialize the I/O drivers with \$INITIO (described later).

Here are the devices that may be routed:

Device	System Abbreviation
Keyboard	KI
Display	DO
Printer	PR
RS-232-C	
Send	RO
Receive	RI



To Route from One Device to Another

Note: To actually try out the next four steps, you must have printer connected to your Computer. If not, just read through the example.

1. Store the Source Device Abbreviation in memory locations 16930-16931. For example, to store DO (display) as the source device, execute the BASIC statements:

```
POKE 16930, ASC("D")
POKE 16931, ASC("O")
```

2. Store the Destination Device Abbreviation in memory locations 16928-16929. For example, to store PR (printer) as the destination device, execute the BASIC statements:

```
POKE 16928, ASC("P")
POKE 16929, ASC("R")
```

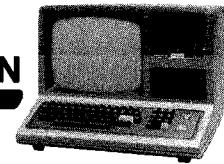
3. Set up aUSR call to \$ROUTE (address 108). For example, execute the BASIC statements;

```
POKE 16526, 108
POKE 16527, 0
```

4. Make aUSR call to \$ROUTE with a dummy argument. For example, execute the BASIC statements:

```
X = USR(0)
```

Upon completion of Step 4, the route is completed. Now everything you send to the display will be sent to the printer instead.



Routing Multiple Devices

You can change two or more of the I/O routes. To do this, you perform the routing Steps 1 through 4 once for each change you wish to make. However, to get the desired result, you must do the changes **in the correct order!** If you use one device as the **source** of a route, you should not later on use the same device as a destination. Here's why:

After you route device A to device B, device A is now logically equivalent to device B. Therefore:

- (1) Route A to B
- (2) Route C to A

Does not allow C to output to device A. Output to C will actually transfer to B, just as if you had executed these steps:

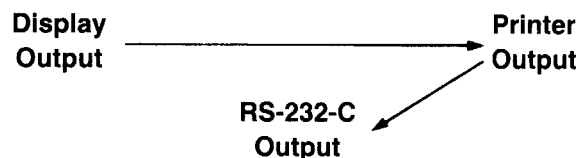
- (1) Route A to B
- (2) Route C to B

On the other hand:

- (1) Route C to A
- (2) Route A to B

Does allow device C to output to device A and device A to output to device B.

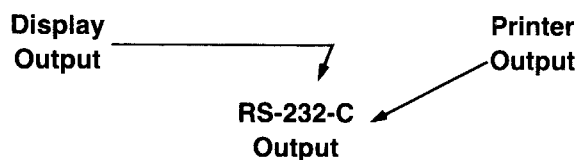
For example, suppose you want to route display output to the printer, and printer output to the RS-232-C. Here's a diagram of what you want to accomplish:



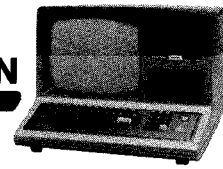
Display output goes to the Printer, and Printer output goes to the RS-232-C. All other I/O routes are unchanged. Note that Display output does **not** get carried forward from the Printer to the RS-232-C. To accomplish the routing pictured above, use this sequence:

1. Route DO to PR
2. Route PR to RO

If you mistakenly do the steps in reverse order, you will get this result:



In this case, Display output is "carried forward" from the printer to the RS-232-C. It does not output to the printer.



10 / Real-Time Clock

The Model III contains a real-time clock. It is always running, except during cassette and disk I/O and during certain other operations.

The clock keeps the following information in memory:

Abbrev.	Range of Values		Memory Location
MO	Month	01 - 12	16924
DA	Day	01 - 31	16923
YR	Year	00 - 99	16922
HR	Hour	00 - 23	16921
MN	Min.	00 - 59	16920
SS	Sec.	00 - 59	16919

The clock includes the logic for 28, 30 and 31-day months. It does not recognize leap years.

When you start the Computer, the clock is set to all zeroes:

00/00/00 00:00:00

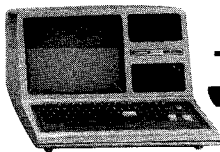
To Set the Clock

Simply store the appropriate data in the memory addresses given above. You may do this by running the following program:

```

10 DEFINT A-Z
20 DIM TM(5)
30 CL = 16924
40 PRINT "INPUT 6 VALUES: MO, DA, YR, HR, MN, SS"
50 INPUT TM(0), TM(1), TM(2), TM(3), TM(4), TM(5)
60 FOR I = 0 TO 5
70     POKE CL - I, TM(I)
80 NEXT I
90 PRINT "CLOCK IS SET"
100 END

```



To Read the Clock

The Model III includes a built-in BASIC function, TIME\$, to get the time in a 17-byte string. For example, execute the BASIC statement:

```
PRINT TIME$
```

To display the time.

To Display the Clock in Real-Time

You can turn on a continuously updated clock display. The current time (not the date) will be displayed in columns 57 - 64, regardless of what mode the BASIC is in: Immediate, Execute, Edit, or System. **As long as the clock is running**, it will be updated on the display.

To enable the clock display, call the ROM subroutine \$CLKON at address 664. To disable it, call the ROM subroutine \$CLKOFF at 673.

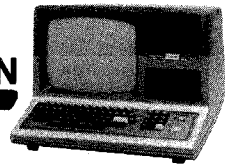
The following BASIC program shows how to turn the display on and off. Each time you want to switch it on or off, run the program.

Note: To calculate the most significant and least significant bytes of a decimal number, use this formula:

$$\begin{aligned} \text{MSB} &= \text{integer portion of } (\text{number}/256) \\ \text{LSB} &= \text{number} - (\text{MSB} * 256) \end{aligned}$$

For example, decimal address 661 can be broken down this way:

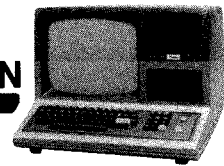
$$\begin{aligned} \text{MSB} &= \text{integer portion of } (661/256) = 2 \\ \text{LSB} &= 661 - (2 * 256) = 152 \end{aligned}$$



Sample Program

```
5 CLS
10 DEFINT A-Z
20 EN = 152: DI = 161      'LSB OF $CLKON/$CLKOFF
30 PRINT "<E> NABLE CLOCK DISPLAY"
40 PRINT "<D> ISABLE CLOCK DISPLAY"
50 INPUT A$
60 IF A$ = "E" THEN SW = EN: GOTO 100
70 IF A$ = "D" THEN SW = DI: GOTO 100
80 GOTO 30
100 POKE 16526, SW        'SET UP USR CALL
110 POKE 16527, 2        'MSB IS SAME FOR BOTH CALLS
120 X = USR(0)           'CALL USR SUBROUTINE
130 END
```

For further information about the real-time clock, see \$CLKON and \$CLKOFF in the Technical Information chapter.



11 / Input/Output Initialization

Whenever you start or reset the Computer, the input/output routines (“I/O drivers”) are initialized to their default values (as explained in the following chapters). For example, the Video Display is initialized to have a blinking cursor.

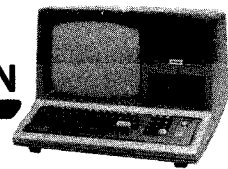
As described in the previous chapters, there are ways for you to alter these default characteristics via a BASIC or Z-80 program. Because of this feature, it is important to have a means of resetting the I/O drivers to their default conditions.

Model III has a ROM subroutine to re-initialize all I/O drivers to their default values. We call it \$INITIO.

The following BASIC program shows how to use \$INITIO.

```
10 POKE 16526, 105           'LSB OF $INITIO ENTRY ADDRESS
20 POKE 16527, 0             'MSB
30 X =USR(0)                 'CALL $INITIO
```

Run this program whenever you want to restore the I/O drivers to their initial characteristics.



12/Technical Information

This section is intended for Z-80 programmers and BASIC programmers who are familiar with binary and hexadecimal arithmetic and hardware concepts like bit and byte. Its purpose is to allow you to take full advantage of the Model III TRS-80.

If you want to understand and use the system on this level, but do not have the background, we suggest you read:

TRS-80 Assembly Language Programming
by William Barden, Jr.
Radio Shack Catalog Number 62-2006

This one book will get you off to a good start. It was written for the Model I TRS-80, but almost all of it applies to the Model III as well.

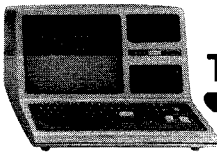
To Protect High RAM

In many applications, you will want to interface a BASIC program and a Z-80 routine. In such cases, you need to protect enough high RAM to accommodate your Z-80 routine. Otherwise, BASIC will use all RAM available for storage and execution of the BASIC program.

During the start-up dialog, you have the option of protecting high RAM via the Memory Size Question. If you simply press **(ENTER)** to this question, BASIC will use all available RAM.

To protect RAM, type in the "limit address" in decimal form, and then press **(ENTER)**. The limit address is the highest memory address you want BASIC to use. Addresses above this value will not be affected by BASIC.

For example, if you type: "32667 **(ENTER)**", BASIC will not use any memory above 32667. It **will** use 32667 and all lower-numbered memory locations.



ROM Subroutines

The Model III BASIC ROM contains many subroutines that can be called by a Z-80 program; many of these can be called by a BASIC program via the USR function. Each subroutine will be described in the format given below.

Important Note

Some of these ROM addresses or calling procedures may change in later releases of the Model III ROM. We suggest you design your programs to minimize the difficulty of adjusting to these possible changes. (Use EQUates for all ROM calls; modularize all uses of ROM routines; etcetera.)

1. \$NAME — Entry address

2. Function Summary

3. Description of function

4. Entry Conditions

5. Exit Conditions

6. Sample Program

Notes:

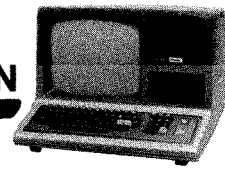
1. The subroutine name is only for convenient reference. It is not recognized by the Computer. The \$- prefix reminds you that it is a convenience name only.

The entry address is given in decimal/hexadecimal form. (The hexadecimal address will be given in this form: X'0000'.) This is the address you use in a Z-80 CALL. BASIC programmers store this address in the USR definition address (16526-16527).

4, 5. Entry and exit conditions are given for Z-80 programs. If a Z-80 register is not mentioned here, then you can assume it is unchanged by the subroutine.

6. Sample Program fragments are given in Z-80 Assembly Language and, where appropriate, in BASIC.

Here are the subroutines, arranged according to function. In the following pages, they are arranged alphabetically.



System Control

\$CLKON	Clock-display on
\$CLKOFF	Clock-display off
\$DATE	Get today's date
\$DELAY	Delay for a specified interval
\$INITIO	Initialize all I/O drivers
\$READY	Jump to Model III "Ready"
\$RESET	Reset Computer
\$ROUTE	Change I/O device routing
\$SETCAS	Prompt user to set cassette baud rate
\$TIME	Get the time

Cassette I/O

\$CSHIN	Cassette on, search for leader and sync byte
\$CSIN	Input a byte
\$CSOFF	Turn off cassette drive
\$CSHWR	Cassette on, Write leader and sync byte
\$CSOUT	Write a byte to cassette

Keyboard Input

\$KBCHAR	Get a character if available
\$KBWAIT	Wait for a character
\$KBLINE	Wait for a line
\$KBBRK	Check for BREAK key only

Printer Output

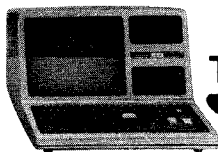
\$PRCHAR	Print a character
\$PRSCN	Print entire screen contents

RS-232-C I/O

\$RSINIT	Initialization
\$RSRCV	Receive a character
\$RSTX	Send a character

Video Display Output

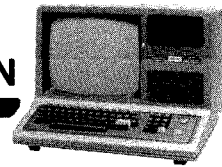
\$VDCHAR	Display a character
\$VDCLS	Clear the screen
\$VDLINE	Display a line



TRS-80 MODEL III

```
00001
00002 ; MODEL III ROM CALLS - DEMONSTRATION PROGRAM
00003 ;
00004 ; CREATED 07/07/80
00005 ; UPDATED 07/08/80
00006 ;
00007 ; TO DEMONSTRATE, JUMP TO THE APPROPRIATE ENTRY
00008 ; POINT. EACH DEMO ENDS WITH A JUMP TO BASIC 'READY'
00009 ;
00010 ;
00011 RESET EQU 0000H
00012 KBCHAR EQU 002BH
00013 VDCHAR EQU 0033H
00014 PRCHAR EQU 003BH
00015 KBLINE EQU 0040H
00016 KBWAIT EQU 0047H
00017 RSRCV EQU 0050H
00018 RSTX EQU 0055H
00019 RSINI1 EQU 005AH
00020 DELAY EQU 0060H
00021 INITIO EQU 0069H
00022 ROUTE EQU 006CH
0109 00023 VDCLS EQU 0109H
01D9 00024 PRSCN EQU 01D9H
01F8 00025 CSOFF EQU 01F8H
021B 00026 VDLINE EQU 021BH
0235 00027 CSIN EQU 0235H
0264 00028 CSOUT EQU 0264H
0287 00029 CSHWR EQU 0287H
028D 00030 KBRK EQU 028DH
0296 00031 CSHIN EQU 0296H
0298 00032 CLKON EQU 0298H
02A1 00033 CLKOFF EQU 02A1H
3042 00034 SETCAS EQU 3042H
1A19 00035 READY EQU 1A19H
3033 00036 DATE EQU 3033H
3036 00037 TIME EQU 3036H
37EB 00038 PRSTAT EQU 37EBH
00039 ;-----
8000 00040 ORG 8000H
00041 ;
```

Note: This Z-80 assembly language listing is continued under the ROM call entries for **Sample Z-80 Programming**.



\$CLKOFF — 673/X'02A1'

Disable the Clock Display

Entry Conditions

None

Exit Conditions

A is altered. All other registers are unchanged.

Sample Z-80 Programming

```

0000      CDA102      00042 : TURN OFF CLOCK
0003      C3191A      00043      CALL      CLKOFF
                        00044      JP        READY

```

Sample BASIC Programming

```

100 POKE 16526,161: POKE 16527,2      'LSB/MSB
110 X =USR(0)                          'DUMMY ARGUMENT

```

\$CLKON — 664/X'0298'

Enable the Clock Display

Entry Conditions

None

Exit Conditions

A is altered. All other registers are unchanged.

Sample Z-80 Programming

```

0006      CD9802      00045 : TURN ON CLOCK
0009      C3191A      00046      CALL      CLKON
                        00047      JP        READY

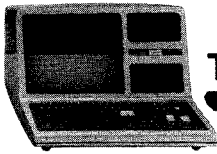
```

Sample BASIC Programming

```

100 POKE 16526,152: POKE 16527,2      'LSB/MSB
110 X =USR(0)                          'DUMMY ARGUMENT

```



\$CSHIN — 662/X'0296'

Search for Cassette Header and Sync Byte

Each cassette "record" begins with a header consisting of a leader sequence and synchronization byte. \$CSHIN turns on the cassette drive and begins searching for this header information. The subroutine returns to the calling program after the sync-byte has been read.

Entry Conditions

None

Exit Conditions

A is altered. All other registers are unchanged.

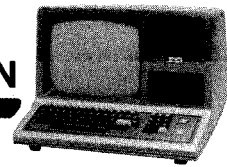
Sample Z-80 Programming

The following program reads the tape created by the \$CSHWR sample program.

```

00048 ; READ A MESSAGE FROM TAPE & STOP ON CAR-RET'N
800C CDC901 00049 CALL VDCLS CLEAR SCREEN
800F 3E0D 00050 LD A,0DH
8011 CD3300 00051 CALL VDCHAR SKIP A LINE
8014 CD4230 00052 CALL SETCAS LET USER SELECT BAUD RATE
8017 213B80 00053 LD HL,MSG0 (HL)=CASSETTE PROMPT
801A CD1B02 00054 CALL VDLINE
801D CD4900 00055 CALL KBWAIT WAIT FOR ANY KEY
8020 216280 00056 LD HL,TXT (HL)=256-BYTE BUFFER
8023 CD9602 00057 CALL CSHIN FIND START OF RECORD
8026 CD3502 00058 LOOP CALL CSIN INPUT A BYTE
8029 77 00059 LD (HL),A STORE IT
802A 23 00060 INC HL POINT TO NXT LOC.
802B FE0D 00061 CP 0DH WAS LAST BYTE=CAR-RET'N?
802D 20F7 00062 JR NZ,LOOP IF NO, GET NXT BYTE
802F CDF801 00063 CALL CSOFF IF YES, TURN OFF CASSETTE
8032 216280 00064 LD HL,TXT DISPLAY THE MESSAGE
8035 CD1B02 00065 CALL VDLINE
8038 C3191A 00066 JP READY AND QUIT
803B 50 00067 MSG0 DEFM 'PREPARE TAPE TO PLAY AND PRESS ANY KEY'
8061 0D 00068 DEFB 0DH
8062 0062 00069 TXT DEFS 256 STORAGE FOR TAPED MESSAGEE

```



\$CSIN — 565/X'0235'

Input a Byte

After completion of \$CSHIN, use \$CSIN to begin inputting data, one byte at a time.

Note: You must call \$CSIN often enough to keep up with the baud rate (either 500 or 1500 baud).

Entry Conditions

None

Exit Conditions

A = Data byte

Sample Z-80 Programming

See \$CSHIN.

\$CSHWR — 647/X'0287'

Write Leader and Sync Byte

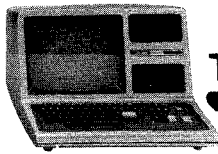
Each cassette "record" begins with a header consisting of a leader sequence and a synchronization byte. \$CSHWR turns on the cassette and writes out this header.

Entry Conditions

None

Exit Conditions

A is altered.



Sample Z-80 Programming

		00070	;INPUT A KEYBOARD MESSAGE AND WRITE IT TO CASSETTE			
8162	CDC901	00071	CALL	VDCLS		
8165	3E0D	00072	LOOP1	LD	A,0DH	CARRIAGE RETURN
8167	CD3300	00073	CALL	VDCHAR		SKIP TO NEXT DISPLAY LINE
816A	21A081	00074	LD	HL,MSG1		PROMPT MESSAGE
816D	CD1B02	00075	CALL	VDLINE		DISPLAY IT
8170	21EA81	00076	LD	HL,TXT1		256-BYTE BUFFER
8173	06FF	00077	LD	B,255		MAX OF 255 CHARACTERS
8175	CD4000	00078	CALL	KBLINE		GET A LINE FROM KB
8178	38EB	00079	JR	C,LOOP1		LOOP IF <BREAK> WAS PRESSED
817A	3E0D	00080	LD	A,0DH		
817C	CD3300	00081	CALL	VDCHAR		SKIP A LINE
817F	CD4230	00082	CALL	SETCAS		LET USER SELECT BAUD RATE
8182	21B381	00083	LD	HL,MSG2		CASSETTE PROMPT
8185	CD1B02	00084	CALL	VDLINE		
8188	CD4900	00085	CALL	KBWAIT		WAIT UNTIL A KEY IS PRESSED
818B	CD8702	00086	CALL	CSHWR		WRITE CASSETTE HEADER
818E	21EA81	00087	LD	HL,TXT1		(HL)=MESSAGE
8191	7E	00088	LOOP2	LD	A,(HL)	A=ASCII BYTE
8192	23	00089	INC	HL		POINT TO NEXT BYTE
8193	CD6402	00090	CALL	CSOUT		WRITE LAST BYTE TO TAPE
8196	FE0D	00091	CP	0DH		WAS IT A CARRIAGE RETURN?
8198	20F7	00092	JR	NZ,LOOP2		IF NO, THEN GET NEXT BYTE
819A	CDF801	00093	CALL	CSOFF		IF YES, TURN OFF CASSETTE
819D	C3191A	00094	JP	READY		
81A0	54	00095	MSG1	DEFM	'TYPE IN A MESSAGE'	
81B2	0D	00096	DEFB	0DH		
81B3	4D	00097	MSG2	DEFM	'MESSAGE STORED. PRESS ANY KEY WHEN READY TO RECORD...'	
81E9	0D	00098	DEFB	0DH		END OF LINE
81EA		00099	TXT1	DEFS	256	

For a program to read the tape in, see \$CSHIN.

\$CSOFF — 504/X'01F8'

Turn Off Cassette

After writing data to cassette, call this subroutine to turn off the cassette drive.

Entry Conditions

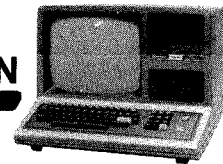
None

Exit Conditions

None

Sample Z-80 Programming

See \$CSHWR.



\$CSOUT — 612/X'0264'

Output a Byte to Cassette

After writing the header with \$CSHWR, use \$CSOUT to write the data, one byte at a time.

Note: You must call \$CSOUT often enough to keep up with the baud rate (either 500 or 1500 baud).

Entry Conditions

A = Data byte.

Exit Conditions

None

Sample Z-80 Programming

See \$CSHWR.

\$DATE — 12339/X'3033'

Get Today's Date

Entry Conditions

(HL) = Eight-byte output buffer

Exit Conditions

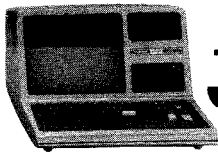
(HL) = Date in this format:

MO/DA/YR

All other registers are altered.

Sample Z-80 Programming

82EA	210883	00100 ; GET TODAY'S DATE & TIME	
82ED	CD3330	00101 LD HL,TXT2	8-BYTE BUFFER
82F0	21FFB2	00102 CALL DATE	
82F3	CD3630	00103 LD HL,TXT3	8-BYTE BUFFER
82F6	21FFB2	00104 CALL TIME	
82F9	CD1B02	00105 LD HL,TXT3	(HL)=TIME/DATE MSG.
82FC	C3191A	00106 CALL VDLIN	DISPLAY TIME/DATE
82FF		00107 JP READY	
8307	20	00108 TXT3 DEFS 8	TIME GOES HERE
8308		00109 DEFB 20H	ASCII SPACE
8310	00	00110 TXT2 DEFS 8	DATE GOES HERE
		00111 DEFB 0DH	END OF LINE



\$DELAY — 96/X'0060'

Delay for a Specified Interval

This is a general-purpose routine to be used whenever you want to pause before continuing with a program.

Entry Conditions

BC = Delay multiplier. Actual delay will be:
2.46 + (14.8 * BC) microseconds
When BC = 0000, 65536 is used. This is the maximum delay (about one second).

Exit Conditions

BC and A are altered.

Sample Z-80 Programming

3E20		00112	;	SHOW ALL DISPLAY CHARACTERS, WITH DELAY AFTER EACH	
8311	CD6900	00113	CENTER	EQU	3E20H
8314	CDC901	00114		CALL	INITIO
8317	3E00	00115		CALL	VDCLS
8319	01FF7F	00116		LD	A,0H
831C	32203E	00117		LD	BC,7FFFH
831F	F5	00118	LOOP3	LD	(CENTER),A
8320	C5	00119		PUSH	AF
8321	CD6000	00120		PUSH	BC
8324	C1	00121		CALL	DELAY
8325	F1	00122		POP	BC
8326	3C	00123		POP	AF
8327	20F3	00124		INC	A
8329	C3191A	00125		JR	NZ,LOOP3
		00126		JP	READY
					RESTORE ALL I/O DRIVERS
					FIRST CLEAR SCREEN
					SET 1/2-SEC DELAY FACTOR
					WRITE CHARACTER TO VIDEO
					SAVE LAST CHAR. CODE
					AND DELAY FACTOR
					NEXT CHAR CODE
					IF NOT ZERO, DISPLAY IT
					ELSE END

\$INITIO — 105/X'0069'

Initialize All I/O Drivers

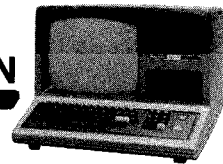
Call \$INITIO to restore all I/O drivers to their initial default conditions, including I/O routes.

Entry Conditions

None

Exit Conditions

All registers are altered.



Sample Z-80 Programming

See SDELAY.

Sample BASIC Programming

```
10 POKE 16526,105: POKE 16527,0  
20 X =USR(0)
```

```
?LSB/MSB  
?DUMMY ARGUMENT
```

\$KBCHAR — 43/X'002B'

Get a Keyboard Character if Available

This subroutine checks the keyboard for a character. The character (if any) is not displayed.

Entry Conditions

None

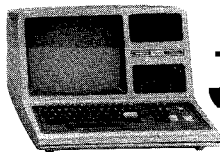
Exit Conditions

A=ASCII Character. IF A=0, no character was available.

DE is altered.

Sample Z-80 Programming

See \$RSINIT.



\$KBLINE — 64/X'0040'

Wait for a Line from the Keyboard

This routine gets a full line from the Keyboard. The line is terminated by a carriage return (X'0D') or **BREAK** (X'01'). Characters typed are echoed to the display.

Entry Conditions

B = Maximum length of line. When this many characters are typed, no more will be allowed except for **ENTER** or **BREAK**

(HL) = Storage buffer. Length should be B + 1.

Exit Conditions

C Status = **BREAK** was the terminator.

B = Number of characters entered.

(HL) = Line from keyboard, followed by terminating character.

DE is altered.

Sample Z-80 Programming

See \$CSHWR.

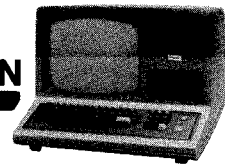
\$KBWAIT — 73/X'0049'

Wait for a Keyboard Character

This routine scans the keyboard until a key is pressed. If **BREAK** is pressed, it will be returned in A like any other key. The character typed is not echoed to the Display.

Entry Conditions

None



Exit Conditions

A = Keyboard character

DE is altered.

Sample Z-80 Programming

See \$CSHWR.

\$KBBRK — 653/X'028D'

Check for **BREAK** Key Only

This is a fast key scan for the **BREAK** key only. Use it when you want to minimize keyboard scan time without totally locking out the keyboard.

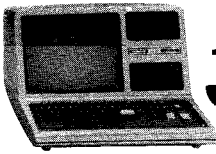
Entry Conditions

None

Exit Conditions

NZ Status = **BREAK** was pressed

A is altered.



\$PRCHAR — 59/X'003B'

Output a Character to the Printer

\$PRCHAR waits until the Printer is available or until **BREAK** is pressed. If **BREAK** is pressed, \$PRCHAR returns to caller.

Entry Conditions

A = ASCII character

Exit Conditions

DE is altered.

Sample Z-80 Programming

```

00148 ; PRINTER DEMO
8356      216583      00149      LD      HL,TXT4      (HL)=SAMPLE TEXT
8359      7E          00150 LOOP5    LD      A,(HL)      GET CHAR. INTO A
835A      23          00151      INC     HL          POINT TO NEXT CHAR
835B      CD3B00      00152      CALL   PRCHAR     PRINT CHAR IN A
835E      FE0D        00153      CP      0DH       WAS IT A CARRIAGE RETURN?
8360      20F7        00154      JR     NZ,LOOP5   IF NO, GET NEXT CHAR.
8362      C3191A      00155      JP     READY      IF YES, QUIT
8365      54          00156 TXT4    DEFB  'THIS SENTENCE WILL BE PRINTED'
8382      0D          00157      DEFB  0DH
402D      00158      END
000000 ASSEMBLY ERRORS

```

\$PRSCN — 473/X'01D9'

Print Entire Screen Contents

This routine copies all 1024 characters from the screen to the printer. If the printer is unavailable, it waits until the printer becomes available. If **BREAK** is pressed, \$PRSCN returns to the caller.

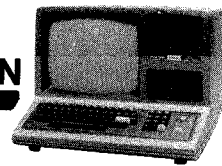
Entry Conditions

None

Exit Conditions

All registers are altered.





\$READY — 6681/X'1A19'

Jump to Model III BASIC "Ready"

To exit from a machine-language program into BASIC's immediate mode, jump to \$READY (don't call it).

Entry Conditions

None

Exit Conditions

None

Sample Z-80 Programming

See SCSHIN.

\$RESET — 0/X'0000'

Jump to RESET

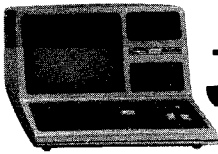
Jump to this address to re-initialize the entire system starting at the "Cass?" prompt. If a disk controller is present, the Computer will attempt to load TRSDOS. To prevent this from happening, the operator must hold down **(BREAK)** before this jump is executed.

Entry Conditions

None

Exit Conditions

None



\$ROUTE — 108/X'006C'

Change I/O Device Routing

Entry Conditions

(X'4222') = Two-byte source device ASCII abbreviation: {KI,DO,RI,RO,PR}

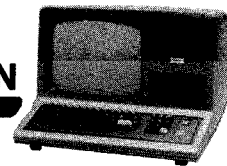
(X'4220') = Two-byte destination device ASCII abbreviation. Same set as above.

Exit Conditions

DE is altered.

Sample Programming.

See Chapter 9 in this section.



\$RSINIT — 90/X'005A'

Initialize the RS-232-C Interface

When you start the Computer, the RS-232-C interface is initialized to the following characteristics:

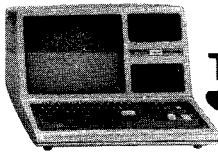
- Send/Receive Baud Rate: 300
- Word length: 8
- Parity: None
- Stop-Bits: One
- Wait for completion of character I/O

To change any of these, you must call \$RSINIT.

Entry Conditions

- (16888) = Send/Receive Baud Rate Code:
Most significant four bits = send rate
Least significant four bits = receive rate
See the table of baud rate codes in Chapter 8.
- (16890) = Wait/Don't Wait Switch
Zero = "Don't Wait"
Non-Zero = "Wait"
- (16889) = RS-232-C Characteristics Switch:

Bits	Meaning	Bits	Meaning
7	Parity: 1 = Even 0 = Odd	3	Transmit On/Off 0 = Disable 1 = Enable
6,5	Word Length: 00 = 5 Bits 01 = 6 Bits 10 = 7 Bits 11 = 8 Bits	2	Data Terminal Ready 0 = No 1 = Yes
5	Stop Bits: 0 = 1 Bit 1 = 2 Bits	1	Request To Send 0 = No 1 = Yes
4	Parity On/Off 0 = Parity 1 = No Parity		



TRS-80 MODEL III

Exit Conditions

DE is altered.

Sample Z-80 Program

```

                                00127 ; TERMINAL PROGRAM FOR DEMO OF RS-232-C CALLS; $KBCHAR AND $VDCHAR
                                00128 ;
                                00129 ; ASSUME 16888 & 16889 CONTAIN THE PROPER INITIALIZATION VALUES
                                00130 ;
0320    AF                00131    XOR    A                ZERO A TO SELECT "DON'T WAIT"
0320    32FA41           00132    LD     (16890),A
0330    CD5A00           00133    CALL  RSINIT
0333    CDC901           00134    CALL  VDCLS
0336    CD2E00           00135    KEYIN  CALL  KBCHAR        CHECK KEYBOARD
0339    FE00             00136    CP     0
033E    2806            00137    JR     Z,RSIN        IF NOTHING, CHECK RS232
033D    CD3300           00138    CALL  VDCHAR        SELF-ECHO
0340    CD5500           00139    CALL  RSTX         SEND IT TO RS232
0343    21E841           00140    RSIN  LD     HL,16872  (HL)=CHAR.INPUT BUFFER
0346    CD5000           00141    CALL  RSRVC        CHECK FOR RS232 INPUT
0349    7E              00142    LD     A,(HL)      GET BUFFER CONTENTS
034A    FE00            00143    CP     0
034C    28E8            00144    JR     Z,KEYIN     IF NOTHING, CHECK KB
034E    CD3300           00145    CALL  VDCHAR        ELSE DISPLAY IT
0351    1BE3            00146    JR     KEYIN       CHECK KB
0353    C3191A           00147    JP     READY       RETURN TO BASIC

```

\$RSRCV — 80/X'0050'

Receive a Character from the RS-232-C Interface

If RS-232-C Wait is enabled, this routine waits for a character to be received, or until **BREAK** is pressed.

If Wait is not enabled, it returns whether or not a character is received.

Entry Conditions

None

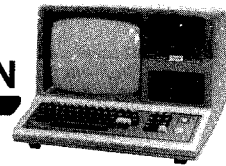
Exit Conditions

(16872) = Character received. Zero indicates no character.

DE is altered.

Sample Z-80 Programming

See \$RSINIT.



\$RSTX — 85/X'0055'

Transmit a Character to the RS-232-C Interface

If RS-232-C Wait is enabled, this routine waits until the character is transmitted or until **(BREAK)** is pressed.

If Wait is not enabled, it returns whether or not a character is transmitted.

Entry Conditions

A = Character

Exit Conditions

Z Status = No character sent

DE is altered.

Sample Z-80 Programming

See \$RSINIT.

\$SETCAS — 12354/X'3042'

Prompt User to Set Cassette Baud Rate

This call repeats the first question in the Model III start-up dialog. It displays the prompt:

Cass?

on the next line of the display, and waits for the operator to type "H" (high—1500 baud) or "L" (low—500) or **(ENTER)** (default to high).

Upon return from the call, the cassette rate is set accordingly.

Entry Conditions

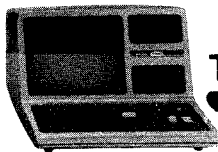
None

Exit Conditions

All registers are altered.

Sample Z-80 Programming

See \$CSHWR.



\$TIME — 12342/X'3036'

Get the Time

Entry Conditions

(HL) = Eight-byte output buffer

Exit Conditions

(HL) = Time in this format:
HR:MN:SS

All other registers are altered.

Sample Z-80 Programming

See \$DATE.

\$VDCHAR — 51/X'0033'

Display a Character

This subroutine displays a character at the current cursor location.

Entry Conditions

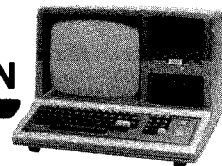
A = ASCII character

Exit Conditions

DE is altered.

Sample Z-80 Programming

See \$DELAY.



\$VDCLS — 457/X'01C9'

Clear the Video Display Screen

Entry Conditions

None

Exit Conditions

All registers are altered.

Sample Z-80 Program

See SCSHWR.

\$VDLINE — 539/X'021B'

Display a Line

This subroutine displays a line. The line **must** be terminated with an ASCII ETX (X'03') or carriage return (X'0D'). If the terminator is a carriage return, it will be printed; if it is an ETX, it will not be printed. This allows VDLINE to position the cursor to the beginning of the next line or leave it at the position after the last text character.

Entry Conditions

(HL) = Output text, terminated by X'03' or X'0D'.

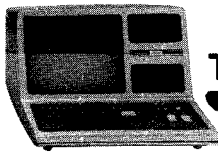
Exit Conditions

(HL) = First character after the terminator.

DE is altered.

Sample Z-80 Programming

See SCSHWR.



BREAK Processing

The **BREAK** key is intercepted during keyboard scan operations. The Computer transfers control to a three-byte jump vector in RAM (hex values: C3 lsb msb). For special applications, you may change the jump vector addresses to allow your own program to handle the **BREAK** key.

The keyscan **BREAK** jump vector is located at 16396 (X'400C').

Register contents on entry to the jump vector

DE = Modified by the Computer

(SP) = The return address of the interrupted program. That is, a RET will transfer control to the point at which the program was interrupted.

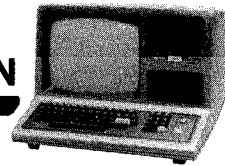
Sample BASIC Programming

Run this BASIC program to disable **BREAK**.

```
10 POKE 16396,175          ' 175 = Z-80 "XOR A" CODE
20 POKE 16397,201         ' 201 = Z-80 "RET" CODE
```

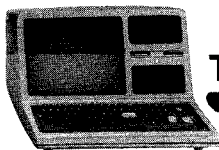
Run this BASIC program to enable the **BREAK** key.

```
10 POKE 16396,201         ' Z-80 "RET" CODE
```



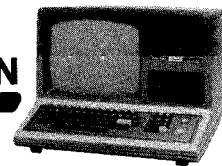
Memory Map

Decimal Address	Contents	Hexadecimal Address
0	12 K ROM Model III BASIC	0
12288	2 K ROM for System Use	3000
14336	Keyboard Matrix	3800
15360	Memory-Mapped Video Display: Upper left corner = 15360 + 0. Lower right corner = 15360 + 1023.	3C00
16384	Reserved for System Use	4000
17129	User Memory For Program and Data	42E9
32767 49151 65535	"16K RAM" ends here. "32K RAM" ends here. "48K RAM" ends here.	7FFF BFFF FFFF



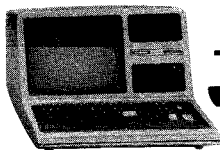
Summary of Important ROM Addresses

Address		Contents	Function
Dec	Hex		
0	0000	\$RESET	System reset
43	002B	\$KBCHAR	Check for keyboard character
51	0033	\$VDCHAR	Display a character
59	003B	\$PRCHAR	Print a character
64	0040	\$KBLINE	Wait for a keyboard line
73	0049	\$KBWAIT	Wait for a keyboard character
80	0050	\$RSRCV	Receive character from RS-232-C
85	0055	\$RSTX	Transmit character to RS-232-C
90	005A	\$RSINIT	Initialize RS-232-C
96	0060	\$DELAY	Delay for a specified time
105	0069	\$INITIO	Initialize all I/O drivers
108	006C	\$ROUTE	Route I/O
457	01C9	\$VDCLS	Clear the screen
473	01D9	\$PRSCN	Print screen contents
504	01F8	\$CSOFF	Turn off cassette
539	021B	\$VDLINE	Display a line
565	0235	\$CSIN	Input a cassette byte
612	0264	\$CSOUT	Output a cassette byte
647	0287	\$CSHWR	Write the cassette header
653	028D	\$KBBRK	Check for (BREAK) key only
662	0296	\$CSHIN	Read the cassette header
664	0298	\$CLKON	Turn on the clock display
673	02A1	\$CLKOFF	Turn off the clock display
6681	1A19	\$READY	Jump to BASIC "Ready"
12339	3033	\$DATE	Get the date
12342	3036	\$TIME	Get the time
12354	3042	\$SETCAS	Set cassette baud rate
14312	37E8	\$PRSTAT	Printer status (Read Only) "Go" only if: Bit 7 = 0 "NOT BUSY" Bit 6 = 0 "NOT OUT OF PAPER" Bit 5 = 1 "DEVICE SELECT" Bit 4 = 1 "NOT PRINTER FAULT" Bits 3,2,1 and 0 are not used.



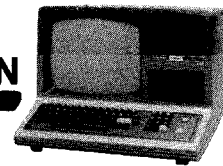
Summary of Important RAM Addresses

Address		Contents	Initial Contents
Dec	Hex		
16396	400C	(BREAK) Jump Vector Keyboard scan operations Three bytes	C9 xx xx
16409	4019	Caps Lock Switch 0 = "Upper and Lower Case" Not 0 = "Caps Only"	"Caps"
16412	401C	Cursor Blink Switch 0 = "Blink" Non-Zero = "No-Blink"	"Blink"
16416	4020	Cursor Address Two bytes: LSB, MSB	N/A
16419	4023	Cursor Character ASCII Code 32 — 255	176
16424	4028	Maximum Lines/Page plus one	67
16425	4029	Number of lines printed plus one	1
16427	402B	Line Printer Max. Line length less two. 255 = "No Maximum"	"No Max"
16872	41E8	\$RSRCV Input Buffer One byte	0
16880	41F0	\$RSTX Output Buffer One byte	0
16888	41F8	\$RSINIT Baud Rate Code TX Code = Most Sig. Nibble RCV Code = Least Sig. Nibble	85
16889	41F9	\$RSINIT Parity/Word Length/ Stop-Bit Code	108
16890	41FA	\$RSINIT WAIT Switch 0 = "Don't Wait" Non-Zero = "Wait"	"Wait"
16913	4211	Cassette Baud Rate Switch 0 = 500 Baud Non-Zero = 1500 Baud	N/A



TRS-80 MODEL III

Address		Contents	Initial Contents
Dec	Hex		
16916	4214	Video Display Scroll Protect From 0 to 7. Greater values are interpreted in modulo 8	0
16919	4217	Time-Date Six binary bytes: SS MM HH YY DD MM	00:00:00 00/00/00
16928	4220	\$ROUTE Destination Device Two-byte I/O designator	N/A
16930	4222	\$ROUTE Source Device Two-byte I/O designator	N/A

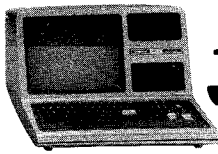


13 / Troubleshooting And Maintenance

If you have problems operating your TRS-80, please check the following table of symptoms and cures. It's also possible that you have not followed the instructions correctly.

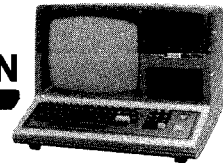
If you can't solve the problem, take the unit in to your local Radio Shack. We'll have it fixed and returned to you ASAP!

Symptom	Possible Cause. Cure.
The Cass? message does not appear when you turn on the Computer.	<ol style="list-style-type: none">1. No AC power. Check power cord connection to Computer and all peripherals.2. Incorrect power-up sequence.3. Peripheral device (e.g., printer) is not connected properly. Recheck connection.4. Disk system. To operate without a TRSDOS diskette, hold down BREAK while you reset or power on.5. Video Display needs adjustment. Check Brightness and Contrast controls.



TRS-80 MODEL III

Symptom	Possible Cause. Cure.
Can't get a cassette program to load.	<ol style="list-style-type: none">1. Improper cassette connection. Check connection instructions in cassette owner's manual.2. Cassette load speed does not match the speed of the recorded tape. Model I Level II BASIC programs are always Low (500 baud). Model III programs may be either High (1500) or Low.3. Incorrect volume setting. Try another volume setting.4. Information on tape may have been garbled due to static electricity discharge, magnetic field, or tape deterioration. Try to load duplicate copy, if available.
Computer "hangs up" during normal operation, requiring reset or power-off/on	<ol style="list-style-type: none">1. Fluctuations in the AC power supply. See AC Power Sources, below.2. Defective or improperly installed connector. Check all connection cables to see that they are securely attached and that they are not frayed or broken.3. Programming. Re-check the program.



AC Power Sources

Computers are sensitive to fluctuations in the power supply at the wall socket. This is rarely a problem unless you are operating in the vicinity of heavy electrical machinery. The power source may also be unstable if some appliance or office machine in the vicinity has a defective switch which arcs when turned on or off.

Your Model III TRS-80 is equipped with a specially designed, built-in AC line filter. It should eliminate the effects of ordinary power-line fluctuations.

However, if the fluctuations are severe, you may need to take some or all of the following steps:

- Install bypass or isolation devices in the problem-causing devices
- Fix or replace any defective (arcing) switches
- Install a separate power-line for the Computer
- Install a special line filter designed for computers and other sensitive electronic equipment

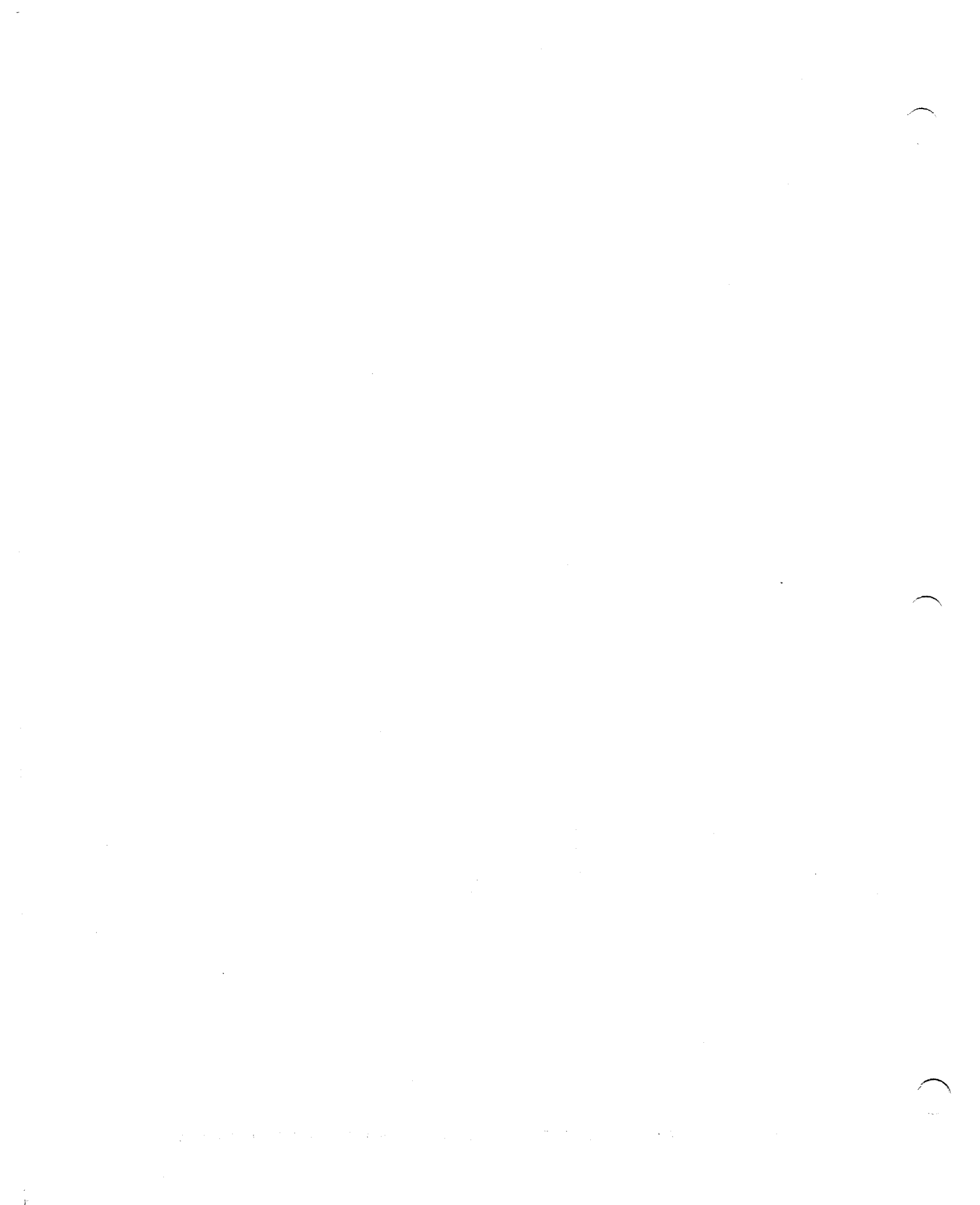
Power line problems are rare and many times can be prevented by proper choice of installation location. The more complex the system and the more serious the application, the more consideration you should give to providing an ideal power source for your Computer.

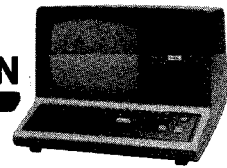
Maintenance

Your Computer requires little maintenance. It's a good idea to keep it clean and free of dust build-up. This is especially important for the keyboard. Radio Shack sells a custom-designed Model III dust cover you may find helpful.

If you need to clean the Computer case, use a damp, lint-free cloth.

The peripheral devices (cassette recorder, line printer, etc.) may require more maintenance. Check the owner's manual for each peripheral in your system.





14 / Specifications

AC Power Supply

This applies to non-disk systems only. For disk systems, see the Disk System Owner's Manual.

Power Requirements	105 - 130 VAC, 60 Hz (240 VAC, 50 Hz Australian) (220 VAC, 50 Hz European)
Current Drain	0.83 Amps RMS

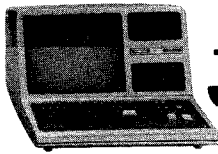
Microprocessor

Type	Z-80
Clock Rate	2.02752 MHz

RS-232-C Interface

Standard RS-232-C Signal		Pin #
PG	Protective Ground	1
TD	Transmit Data	2
RD	Receive Data	3
RTS	Request To Send	4
CTS	Clear To Send	5
DSR	Data Set Ready	6
SG	Signal Ground	7
CD	Carrier Detect	8
DTR	Data Terminal Ready	20
RI	Ring Indicator	22
STD*	Secondary Transmit Data	14
SUN*	Secondary Unassigned	18
SRTS*	Secondary Request To Send	19

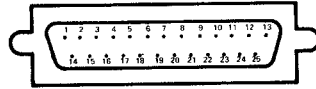
***Note:** These signals are not used for the secondary functions, but are reserved for future use.



TRS-80 MODEL III

RS-232-C Pin Location

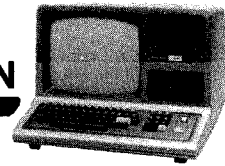
Looking from the outside at the RS-232-C jack on the Model III Computer:



Parallel Printer Interface

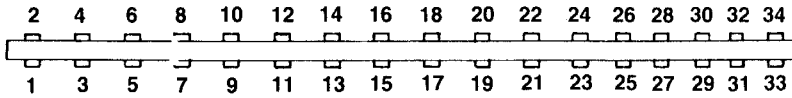
Signal	Function	Pin #
STROBE*	1.5 μ S pulse to clock the data from processor to printer	1
DATA 0	Bit 0 (lsb) of output data byte	3
DATA 1	Bit 1 of output data byte	5
DATA 2	Bit 2 of output data byte	7
DATA 3	Bit 3 of output data byte	9
DATA 4	Bit 4 of output data byte	11
DATA 5	Bit 5 of output data byte	13
DATA 6	Bit 6 of output data byte	15
DATA 7	Bit 7 (msb) of output data byte	17
BUSY	Input to Computer from Printer, high indicates busy	21
PAPER	Input to Computer from Printer, high	23
EMPTY	Indicates no paper — if Printer doesn't provide this, signal is forced low	
SELECT	Input to Computer from Printer, high indicates device selected	25
FAULT*	Input to Computer from Printer, low indicates fault (paper empty, light detect, deselect, etc.)	28
GROUND	Common signal ground	2,4,6,8,10 12,14,16,18, 20,22,24,27, 31,33,34
NC	Not connected or not used	26,29,30,32

*These signals are active-low.



Printer Pin Location

Looking from the bottom rear at the printer card-edge connector as in Figure 1 on 2/2:



Cassette Interface

Suggested Input Level for Playback from Recorder

1 to 5 Volts peak-to-peak at a minimum impedance of 220 Ohms

Typical Computer Output Level to Recorder

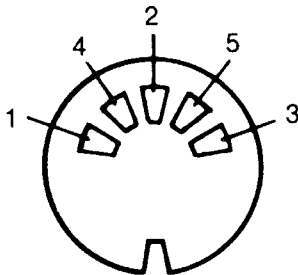
800 mV peak-to-peak at 1 K Ohm

Remote On/Off Switching Capability

0.5 A maximum at 6 VDC

Cassette Jack Pin Location

Looking at the outside of the cassette jack on the Computer:



- 1. Remote Control
- 2. Signal Ground
- 3. Remote Control
- 4. Input from Recorder's Earphone Jack
- 5. Output to Recorder's Aux or Mic Jack