ST80D SMART TERMINAL

by Lance Micklus

ST-80D SMART TERMINAL INSTRUCTIONS

by Lance Micklus

Your TRS-80 is about to become one of the smartest terminals in town! But it's going to take a little work on your part to help it figure things out, because the computer on the other end of the line could be **anything** and each computer seems to have its own special way of doing thins.

What we're going to do here is tell you about ST-80D. Between these instructions and the User's Guide for the computer you'll be talking to on the other end of the line, you should be up and running in just a few minutes.

To begin , let's briefly discuss the ST-80D program. It was written specifically for Disk Drive TRS-80 computers, and since it is in machine language, it's FAST. Finally, (as you may have guessed) it is an advanced version of the ST-80 program for non-disk TRS-80 users.

In addition to being able to talk to a timesharing computer, you'll be able to transfer files to or from the computer on the other end of the phone very easily. You will also be able to customize ST-80D to your specific timesharing environment.

To run the program, you're going to need at least one disk drive and 16K* of memory. Also, you must have a Radio Shack RS-232-C board in your expansion interface. Finally, you will need a modem.

The program disk contains the ST-80D program under the file name ST80D/CMD. The disk includes Radio Shack TRSDOS** for the convenience of single drive users. It is assumed that you've purchased your own TRSDOS disk from Radio Shack, with its instruction manual, and have acquired an average amount of skill running DOS and DISK BASIC***.

*32K is required for TCONV/BAS

**Copyright Radio Shack

***Copyright Microsoft, Inc.

LOADING THE PROGRAM

To load the program, you must be at the DOS level. Just type:

ST80D **ENTER**

and the program will load. Later in these instructions you will learn how to make your own translation tables. To load ST-80D with your own translation table, type the following command:

ST80D filespec **ENTER**

where filespec is the name of your translation table. Be sure you leave at least one blank space between the command file name, ST80D, and the translation table name.

HINT: Use the file extension /TBL in the name of your translation table, and name translation table after the name of your account. Then you can type:

ST80D SIGMA6 **ENTER**

This will run ST-80D using the translation table stored in the file SIGMA6/TBL. Remember, if no file extension is included in the filespec for the translation table, then the extension /TBL is assumed. This works the same way the /CMD is assumed to program files.

INITIALIZING

When the program begins, ST-80D will identify itself. It will then ask if you want to change the switch settings on your RS-232-C board. Press **N** to run with the current settings on the switch pack. If you press **Y**, ST-80D will begin to ask you a series of questions. These questions let you change the settings on the switch pack directly from the keyboard.

For each question asked, you may press the key for the setting you wish to use, or you may just hit **ENTER**. If you hit **ENTER** as you reply, the switch settings for that particular setting will be used.

THE KEYBOARD

The TRS-80 doesn't have a full set of terminal type keys. To get around this problem, some of the keys have been reprogrammed by ST-80D to perform different functions.

To test out the program, set your modem to HALF DUPLEX. That lets the output of the keyboard get sent back into the TRS-80 as input data to be displayed on the screen. Remember, what you see on the screen is what the modem is sending back to you ... NOT NECESSARILY WHAT THE TRS-80 IS SENDING OUT.

Now for those special keys:

BREAK This is an easy one. To send a break, just press the **BREAK** key.

ESC This is one of the missing keys. Since you don't need a clear key for terminal operation, press the **CLEAR** key to send an ESC code. (Pronounce this as **escape**)

RUB This most clearly approximates the normal action of the **left arrow key** , so press the left arrow key to send a **RUB** character. Normally, this is an ASC DELETE X'7F'. Using TCONV/BAS, you can change the byte sent by this key if you need something different.

CONTROL To send a control character, press the **up arrow key** while pressing any of the letter keys on the keyboard. In addition to this, there are ten user-definable control keys which cen be set up under TCONV/BAS. There are the keys 0 through 9 when the up arrow key is pressed simultaneously.

REPEAT This is a little fancy, but if you hold the **right arrow key** down while pressing any other key, ST-80D will repeat the charcter. (Nice if you need a line of dashes or something)

COMMAND This is a special function of ST-80D. If you press the **shift key** and the **up arrow key**, and **one of the letter keys**, you can issue special control commands to ST-80D. Nothing is sent...this only tells ST-80D you want it to do something.

NOTE: Be sure you understand the difference between the CONTROL and COMMAND keys. The **up arrow key** is a control key **unshifted** but a command key **shifted**. And, it must be held down at the same time as one of the other keys.

FILE I/O

Now for the part you've been waiting for: sending and receiving disk files with the outside world. Here's the way it works. ST-80D maintains a buffer which starts at the end of the program and goes to the end of memory. ST-80D doesn't memory-protect. It grabs all the memory it can find. If you have machine language programs in high memory, you might best consider them lost.

The memory buffer can load data from either disk or the serial in port, and can send data to the disk or out the serial port. Let's take an example:

suppose we want to call a Sigma/6 computer and send it a file called ZAP.

First we load in ST-80D and initialize. Now press the **shift up arrow** and the **G** key (for GET). ST-80D will ask for the filespec. Enter it. (NOTE: the left arrow key fixes mistakes). Next, ST-80D will ask for the speed. (More on this later). Just press the **3** key ... you don't need to hit **ENTER**. ST-80D will find the file on your disk and load it into memory. If you don't have enough memory, a MEMORY FAULT error will occur.

Now sign on to the Sigma/6 in the normal manner. On the Sigma/6, the command is BUILD ZAP. Sigma prompts me with a "1.000". Sigma now waits for input. Now press the **shift up arrow O** key (for output). ST-80D will immediately start to dump out its memory buffer. This continues until the memory buffer is empty, or the **right arrow** key is pressed.

The speed of the transmission may be increased or decreased by pressing the **up arrow** or **down arrow** keys. It may be temporarily halted by holding down the space bar. Don't send the data out too fast ... you may overload the Sigma/6 input buffer. Remember — timesharing systems are set up to get data typed by people. Not many people type at 360 w/p/m!

Getting data from the Sigma/6 is a little easier. Just press the **shift up arrow** and **C** key. The data you now receive will load into the buffer, which automatically clears itself. Blinking squares at the top of the screen indicate memory loading is occuring. To close the memory buffer and terminate the memory loading operation, press the **shift up arrow X** key. To save the memory buffer on your disk drives, press **shift up arrow F** (for file). ST-80D will now prompt for the filespec, then dump.

Now let's get down to some details. The memory buffer automatically clears whenever new data is dumped INTO it. You can dump data out of it several times, so if you blow it, you can redump again. That way, a disk-full error doesn't totally spoil your day. Just put in a clean disk and redump.

When sending data to the serial port from the memory buffer, only the above-mentioned keys will work. The rest of the keyboard is dead until the transfer is either aborted or completed. When receiving data into the memory buffer from the serial port, the keyboard remains active.

Timesharing computers often send things that you don't need in your user files. ST-80D does some pre-formatting for you. All delete characters X'7F' are removed. Control characters are translated according to the rules for the printer translation table. Be sure your printer translation table is set up correctly when receiving data from a timesharing computer. This would include deletion of line feeds, bells, etc.

User files sent to the time sharing computer must be ASCII files. No translation is done to them by ST-80D.

Machine language programs and BASIC programs stored in compressed format are not in ASCII files. BASIC programs can be saved in ASCII format, though. The DOS manual gives details, but essentially, you follow the quote at the end of the filespec with a comma, then the letter **A**.

When you're sending out a file, and are asked for the transmission speed, remember this: you're setting a counter. Before a byte is sent, ST-80D zeroes a second counter and starts counting up until the two counters are the same. Then it sends the byte. So, the larger the number you type in, the slower the transmission speed.

To remember what the up and down arrows do to modify transmission speed, think: **SPEED UP** and **SLOW DOWN**.

HOW MUCH MEMORY DO I NEED TO SEND A FILE?

In general, a 2-ganule file requires only 16K of memory. A 15-ganule file needs 32K, and a 28-ganule file must have a full 48K.

There are situations where bigger files may fit into less memory. ST-80D stores spaces in compressed format. Any data files with a lot of spaces in them can be compressed. (This feature works full time)

With text file, it's not unusual for memory requirements to shrink to two-thirds the amount normally needed.

AUTO PILOT

Some timesharing computers send control characters to monitor the automatic transmission of data. ST-80D supports this feature with two special codes, DC1 and DC3. While transmitting data from its memory buffer, if ST-80D receives a X'13' (DC3) byte, it will temporarily halt transmission until it receives a X'11' (DC1) byte.

If you're custom-building a program on your timesharing computer to send data to ST-80D, there are two other device codes you should know about: DC2 and DC4. When ST-80D receives a X'12' (DC2) byte, it opens the memory buffer just as if the **shift up arrow** and **C** keys had been pressed from the keyboard. A X'14' (DC4) byte closes the memory buffer the same as if the **shift up arrow** and **X** keys had been pressed from the keyboard.

If you don't use DC2 and DC4, you may not be able to avoid extra data at the end of the transmission (caused by the timesharing computer prompting for the next command before you can manually close the memory buffer).

OTHER ST-80D COMMANDS

In addition to the commands mentioned already which transfer files to and from a timesharing computer, ST-80D responds to several other keyboard sequences. Remember, you must enter the command by holding down the **shift up arrow** keys at the same time as you press one of the following letters:

B Turns on the auto line feed. If the timesharing system you're using doesn't send a line feed after each return character, use this feature to make the video display skip down to the next line.

D Turns the auto line feed off. ST-80D normally operates with the auto line feed off.

E Exits and returns you to DOS

I Reinitializes the serial in board. This lets you change baud rate without reloading

L Lets you run with an upper/lower case keyboard. In this mode, pressing the shift key and sending an alphabetical letter causes a lower case letter to be sent. Pressing the **shift down arrow** keys flip flops the keyboard to work like a typewriter, where lower case letters are sent unshifted and capitals shifted. It also makes the cursor turn into the underline character. Press **shift down arrow** again and the cursor returns to a block with lower case letters sent with the shift key pressed.

M Sends the auto logon message out the serial port

P Turns on the printer feature. The parallel port printer will now print all incoming data

R Equivalent to a CMD"R" in Disk BASIC

S Turns off the printer feature. Data will now only appear on the screen

T Equivalent to a CMD"T" in Disk BASIC

U Locks out all lower case letters from the keyboard. Only upper case letters can be sent. ST-80D initially comes up this way.

Z Zeroes the system clock. If you turn on the clock display from DOS and zero the clock when you logon, the clock acts like a timer and lets you know how long you've been connected.

TCONV/BAS

This highly self-prompting Disk BASIC program is designed to creat new translation tables for the ST-80D program to use.

You must have at least 32K of RAM to run the program. Because of the large amount of string space used to run the program, **it is normal for it to hang up for several seconds** while BASIC regathers additional string space.

Translation tables are a way to convert data from one system to another when some incompatibility exists. There are various ways to set up translation tables. ST-80D uses two tables, each of which works a little differently from the other.

An example best explains how these translation tables may be used. To clear the screen and home the cursor, ST-80D wants to receive a X'0C' byte. Let's say the timesharing system you're using sends out a X'1A' byte to clear the screen. To resolve this incompatibility, the translation table can be set up to convert all X'1A' bytes to X'0C' bytes.

Besides control characters, some timesharing systems use ESC codes. For example, an ESC character followed by the letter **H** might mean home the cursor. ST-80D's translation table can also be set up to handle this type of format.

These translation tables are very powerful tools. They are, in fact, what makes the ST-80D program so smart. The workings of these tables are explained in the TCONV/BAS program by typing EXPLAIN.

TCONV/BAS also lets you program ten of the keyboard keys any way you wish. You can change the byte sent out by the **RUB** key, **left arrow**, if the X'7F' byte is not compatible with your system.

Finally, you can set the auto logon message. This automatically sends your name, account and password to the host computer when you log on to the system.

The translation table TABLE/TBL on your disk, is set up exactly the same way as the default translation table which is built into the ST-80D program. By using the TCONV/BAS program, you can analyze how the default table is set up.

HINT: The use of a translation table is optional. But you will probably want to customize ST-80D to fit your specific application.

GRAPHICS CHARACTERS

You can create graphic characters on the screen using ST-80D. First, send ST-80D and ESC character (X'1B'), then a 7-bit character. ST-80D will turn on bit 7 (the most significant bit) of the second character to make it into a graphics character.

It will take some experimantation to use this feature. You will find the TABLE command in the TCONV/BAS program helpful in determining which character to send after the ESC byte.

MAKING YOUR TRANSLATION TABLE PERMANENT

If you frequently use one translation table, you may want to install it permanently into the ST-80D program so you don't have to keep loading it from disk.

First load ST-80D with your own translation table. After initializing, immediately return to DOS. Now type TAPEDISK. Enter the next commands exactly as shown. **Spaces are significant.**

```
F    ST80D/CMD:0   6000   6FFF   6000          ENTER
E
```

You now have ST-80D installed with your own translation table permanently in place, and no longer need to specify that translation table when you load the program.

UTILITY PROGRAMS

There are several programs on your disk which are utility in nature. All of the machine language programs, which have the "CMD" extension in them, have one additional feature for single-disk users. After loading, they test the **ENTER** key. If it is pressed, these programs will wait until the key is released. This gives you time to change disks. Upon release of the **ENTER** key, they will then execute. If you have more than one disk drive, you probably will not need to use feature.

FTYPE

This little program takes the guess work out of trying to figure out if a file is ASC or BINARY. Remember, only ASC files can be transmitted. To run it, just type from DOS

FTYPE filespec

where filespec is the name of the file in question. FTYPE will read the file and tell you what it is.

For our purposes, an ASC file is one which contains no nulls or 8-bit characters. Please note, however, that control characters are ASC. Which means whatever is on the receiving end must be able to handle imbedded control characters correctly.

BTA

Any binary files you might have would probably be of little use to anything other than another TRS-80. BTA is a program which makes it possible for two TRS-80 computers to exchange a binary file (such as a machine language program).

What BTA does is convert the binary file to an ASC file which is a hex dump. Thus, the recommended file extension for such intermediary files is "HEX".

From DOS, type the following command:

BTA filespec1 filespec2

where filespec1 is the name of the binary file, and filespec2 is the new file you wish to create (the ASC hex dump of filespec1)

This newly-created file, called filespec2, is totally in ASC format, and may be sent via ST-80D's file exchange features.

ATB

This is the program that gets your machine language programs back to binary. From DOS, type:

ATB filespec1 filespec2

where filespec1 is a hex dump of a binary file, and filespec2 is the binary file you wish to create.

LET'S TAKE AN EXAMPLE

Suppose you have a program called WHITE which is stored on disk under the file name WHITE/CMD. A friend with a TRS-80 and the ST-80D package wants you to send it to him by telephone. From DOS you type the following:

BTA WHITE/CMD ZIP

This now converts your program file to ASC hex code. List the file ZIP and convince yourself. Now you call your friend and both of you load up your ST-80D programs. Since the TRS-80's don't echo, you will both have to run in half duplex. One of you must be able to run your modem in the **answer mode**.

From ST-80D you hold the **shift up arrow** and **G** keys down. Tell ST-80D to get the file ZIP. Since ST-80D can swallow data as sent, set the transmission speed to 0. Meanwhile, your friend presses his **shift up arrow** and **C** keys to tell his ST-80D program to store all incoming data. Assuming you are both ready, you press the **shift up arrow** and **O** keys.

Now the action begins! Your TRS-80 starts sending the hex dump and the other starts saving it. When the transmission is complete, your friend presses the **shift up arrow** and **X** keys to tell his ST-80D program to terminate data storage.

Your friend now presses the **shift up arrow** and **F** keys, and tells his ST-80D program to store the data out to his drive under the file name ZIP. Now your friend exits his ST-80D program and returns to DOS. He then types the following:

ATB ZIP WHITE/CMD

The ATB program now reads the ZIP file and creates the binary file WHITE/CMD, your program.

Did it work? Besides trying it, there is one way to verify that two TRS-80 files are the same; get a checksum of each file.

This is the job of the CKSUM program. From DOS type:

CKSUM filespec

where filespec is the name of the file whose checksum you want. If both files are the same, then their checksums will be the same. If both files have different checksums, then there is something different about them.

Checksums can be made on any type of file, ASC or binary. You will find this a handy little program.

WHAT TO DO WHEN THINGS GO WRONG

The world is full of bugs and gremlins who are out to spoil your day. And, if they don't get you, Murphy's law of computer errors will!

Besides line noise on the telephone, the timesharing computer will always crash at the worst possible moment. And then there's the message that suddenly appears on your screen (and in your data file) reminding all users to "PLEASE DELETE ALL UNNECESSARY FILES".

The fact of the matter is some data files can be fixed up to eliminate any errors. This is the function of the BASIC program NUMBER/BAS.

Run the program and give it the filespec of the data file with the errors. Answer the next question "LN". NUMBER/BAS will now read your file and turn it into a BASIC program you can load into memory and edit. In fact, after it's done putting line numbers into your file, it loads it into memory. Just type LIST.

What kind of a BASIC program? One that is made up entirely of REMARKS, because each line begins

with an apostrophe. Hopefully, you can edit out the errors. When you're done, save the correct version like a BASIC program but in ASC format. In other words, stick a comma and the letter **A** after the filespec.

Now run NUMBER/BAS again. The filespec is the corrected file. Answer the next question NLN,to remove the line numbers. NUMBER/BAS will create an ASC data file again.

That should fix things up. If the data error occured in a hex dump file (created by BTA/CMD), the line number ATB displayed on the screen when it died is where it found the error. That gives you a good start as to where to look.

TEST RUNS

ST-80D was tested out on a variety of timesharing systems at different times during its development. Here are some of the things we found:

On a HARRIS/7 system, using the VULCAN monitor, we couldn't send data from a file to the HARRIS because the HARRIS stopped taking data from the TRS-80 after every line so it could store the line out on disk. By the time it came back to use, a couple of characters had been missed. There was no big problem receiving data from the HARRIS, however, and as far as the terminal operation went, everything was great.

On a SIGMA/6, using the CPV monitor, we had good luck sending and receiving data. The SIGMA/6 is our old hang out, so the system was very familiar to us. The SIGMA/6 had no trouble receiving data sent by the TRS-80, provided the transmission speed was set down to a point where the SIGMA could just keep up.

On a DECSYSTEM 20, running the TOPS 20 monitor, our tests were impressive. The DEC uses the DC1 and DC3 codes. As a result, data could be sent at the fastest possible rate, which was not as fast as the TRS-80 would go. It seemed that when we opened the throttle full blast, the DEC could send the DC3 code fast enough to cut the TRS-80 off. Consequently, some data got lost. Running just a little slower, the DEC took the data in as fast as it could, stopping the TRS-80 as needed. It was like watching a man and his dog.

All three systems were located in Burlington, Vermont. The SIGMA/6 and HARRIS/7 belong to the University of Vermont and the DECSYSTEM 20 is a commercial system run by Interactive Computing of Vermont (ICOV).

In addition to these three systems, an advance release was also run on a HP2000 and a CDC system at the University of Virginia. Exact details are unavailable, but excellent results were obtained using the program.

Early versions of ST-80D were also tested on the Dartmouth Timesharing System and an IBM 370. Unfortunately, these systems became inaccessable to us before the final version of ST-80D was ready for testing.

OTHER THINGS.....

ST-80D was designed to be used with Radio Shack printers. The use of the printer at speeds above 1200 baud is not recommended, as the printer can't keep up with incoming data.

All disk write operations by ST-80D are sequential and verified. You don't have to turn the VERIFY (ON) mode on.

The software is no better than the hardware. You can expect a lot of trouble if:

A. Youre using the least expensive memory chips on the market

B. You use the least expensive disks on the market

C. You don't keep your machine in top operating condition

TRSDOS 2.1 BUGS

There are several bugs in DOS 2.1 which affect ST-80D. Rather than attempt to patch these up and risk an incompatible procedure with future releases of DOS, it was decided to live with the current version and work around the problems.

The first is with the **BREAK** key. So long as interrupts are enabled, DOS will test the **BREAK** key every 25 milliseconds, regardless of what else is happening. When it is pressed and and interrupt occurs, DOS goes to DEBUG to find out what to do. Under certain conditions, DEBUG gets lost and you'll end up rather suddenly in Level II BASIC.

There are two ways around that problem. One is to type DIR from the DOS level just prior to runnint ST-80D. This seems to set the interrupt routine straight on what it ought to do. The second solution is to disable the interrupts from Disk BASIC or by using the **shift up arrow** and **T** keyboard sequence.

The second problem is a bit more subtle. The TAPEDISK command doesn't make correct entries in the directory. This error doesn't bother the machine language loading routine but it does upset other routines

reading this type of data. Likewise, Disk BASIC also doesn't make correct entries when creating RANDOM files. You will note, for example, that if you run a CKSUM on any of ST-80D's translation tables the result is always 00. To get around both of these problems, use the DOS COPY command to make a second copy of these files. The copies will be okay.

Note that Disk BASIC sequential files are all right. These include BASIC program files, and **sequentially created** data files, i.e., OPEN"O", 2, "ZIP". You should also be aware that the DOS copy command adds one more sector to the file. This could have a significant impact if you're using the end of file indicators IF EOF(2) or LOF(2) to tell you when to stop reading your data.

ST-80D COMMAND TABLE

ST-80D responds to the following keys when they are pressed **simultaneously** with the **shift** and **up arrow** keys:

B—Auto line feed on
C—Serial:in to memory buffer, start
D—Auto line feed off
E—Exit ST-80D, return to DOS
F—Dump memory buffer to TRS-80 disks
G—Dump file from TRS-80 disks to memory buffer
I—Reset serial board switch settings
L—Allow lower case letters from the keyboard
M—Transmit auto logon message
O—Transmit memory buffer to the serial:out port
P—Line printer on
R—CMD"R"
S—Turn line printer off
T—CMD"T"
U—Allow only upper case letters to be sent from keyboard
X—Serial:in to memory buffer, stop
Z—Zero TRS-80 system clock

ST-80D FLOWCHART

Process a byte received
from the serial port .....