80 microcomputing T.M.
80 PINE STREET,
PETERBOROUGH, NH 03458

August 4, 1982

Dennis Cardinal
1810 Hibiscus Ct. S.
Oldsmar FL 33557

Re: Article Proposal

Dear Dennis:

Thank you for your recent correspondence concerning an article proposal you have for 80 Micro. Yes, we are interested. Why not send it in for review and possible publication

Thank you for your interest in our magazine and your article idea.

Sincerely,
Jake Commander

Jake Commander
Technical Editor
80 Micro

Dennis Cardinale
1810 Hibiscus Ct. S.
Oldsmar, FL 33557
August 28, 1982

Dear Jake Commander,

I recently sent an article proposal to you and have received your reply. I was very excited that you were interested in my program named TRSTOS. Enclosed is an 11 page manuscript, four pages of tables and figures, the program listing, a cassette tape, and a self addressed stamped envelope.

Side one of the tape has the program TRSTOS so you can test it immediately. Follow the directions in the manuscript under "Using TRSTOS" and you can't go wrong. side two has the source code which is compatable with Radio Shack's Cassette Editor/Assembler. Both recordings are repeated twice just in case you have problems.

Please use the SASE to notify me that you have received this package and if you are going to use my program.

I'm Hoping you will like this program and that I will see it printed in a future issue of 80 Micro.

Sincerely yours,

Dennis Cardinale

Dennis Cardinale

TRSTOS

By Dennis Cardinale
1810 Hibiscus Ct. S.
Oldsmar, FL 33557

Armed with only a Model I, Level II, 16K, tape based computer system, I sometimes get jealous of those rich Model III owners with TRSDOS 2.3's disk BASIC and it's souped up commands. I longed for a DOS-like program for tape to appear on the market and watched all mail order companies, but to no avail. I eventually decided that we tape users were forgotten and that no company was about to make my life easier, so I attempted to write my own. My result was a program called TRSTOS (Tape Operating System) and it went well above some of the aspects of most disk BASICs. It's features are listed below:

* An upper/lower case driver.
* <SHIFT> 0 to toggle upper/lower case.
* A screen print routine accessable via USR(0) or <SHIFT> <down arrow> 0.
* An error printer which prints all errors in full.
* An error locater which prints the bad line and puts a question mark at the error.
* Six line manipulating keys (LMK's):
  Period -- list the current line.
  Comma -- edit the current line.
  UP arrow -- list the previous line.

Down arrow -- list the next line.

<SHIFT> up arrow -- list the first line.

<SHIFT> down arrow -- list the last line.

* Eight abbreviations used in the immediate mode:

A -- Auto.

C -- Cont.

D -- Delete.

E -- Edit.

L -- List.

N -- New.

R -- Run.

S -- System.

* Three base conversions:

&H -- Hexadecimal to Decimal.

&O -- Octal to Decimal.

&B -- Binary to Decimal (Yes! Binary).

* Tab extender to let you tab more than sixty-three spaces.

This entire program can be typed in on a 16K Editor/Assembler. In it's present form (see program listing) it may be used on a Model I with an upper/lower case kit installed, but with some modification it can be used on an upper case Model I or a Model III.

A bit (binary) about me

Dennis Cardinale

Before I go on, let me tell you a little about myself. I am fifteen years old and have been programming for about three years. I learned entirely by myself from reading books and magazines. I wrote my first few programs in BASIC before I ever touched a computer.

My first experience with an 80 was in a Radio Shack store where they had just gotten a Color Computer and had it on display. For about six months after that, I was in the Shack just about every day and on Saturdays I'd stay there about six hours. Eventually, I got to know all of the salesmen there and became a VWK (Very Well Known) customer.

The computer never ceased to amaze me and I was determined to learn all I could about it. Now I know BASIC, Assembly, PASCAL, and FORTRAN. I also know Color BASIC and some extended Color BASIC.

Theory of operation

This program is based on two areas of the communications region in RAM: the disk BASIC exits and the Device Control Blocks (DCB's).

Disk BASIC exits are CALLs to the communications region from strategic locations in ROM. This is meant to give BASIC expandability with disks, although it can be modified by tape. When Level II is initialized without disks, these exits are set to Hex C9's (RET opcode) so that the system doesn't go down if

one is called. Also, when a disk basic token, such as &, is encountered, control is sent to the proper exit for that operation. A listing of the disk BASIC exits used in this program is provided in table 1. A note of caution: do not modify these addresses unless you know exactly what you're doing. If you do, you may have to power off resulting in loss of any program or data currently in memory.

The second part of the communications region that is used is the Device Control Blocks. There are three of these: one for the video, one for the keyboard, and one for the printer. The DCB's hold important information such as the driver address (the address of the routine that scans the keyboard or prints a character), the cursor location, and certain other data.

TRSTOS modifies two of these DCB's: the video DCB to print upper/lower case characters and the keyboard routine to accept <SHIFT> 0 for shift lock toggle, <SHIFT> <down arrow> @ for the screen print routine, and to invert the shift effect on upper/lower case characters.

In the following paragraphs is a disection of each part of the program. Note that any feature may be removed by removing the part of the program that contains that feature. Also, you must delete the corresponding ORiGin statement and it's JumP or DEFW.

Video driver

The video driver is located in lines 1800 - 1880 and is perhaps the simplest part of the program. When Level II uses it's video driver, it converts all lower case to upper case thus prohibiting lower case. In my routine, all characters are printed unchanged so that lower case can now be printed.

Keyboard driver

There were a number of things that needed to be changed in the keyboard routine. The first is that in BASIC's routine, <SHIFT> produced lower case characters. This is exactly opposite of what should happen. My routine check to see if <SHIFT> is pressed and if it is it will return an upper case character. If it isn't, then it returns a lower case character.

The second thing that needed to be changed was <SHIFT> 0. if <SHIFT> 0 is detected, then control is sent to the LOCK routine in lines 1380 - 1420. This routine will set the SHLOCK flag to 0 if upper case or 255 (Hex 0FFH) if upper/lower case.

A third change was making the computer check for <SHIFT> <down arrow> @. If these keys are pressed together, control is sent to the PRSCRN routine in lines 4680 - 4960. This will send the screen to the line printer and return with a zero in in the A register. If the printer is not ready, it will wait until it is or until <BREAK> is pressed. If <BREAK> is pressed then a one is returned in the A register.

Error Printer and locater

The error printing and locating routine is in lines 2610 - 3340. The first thing the routine does is look up the address of the proper error message in the error table (lines 1140 - 1360) and print the error with the PR2 routine in lines 6160 - 6240. This routine allows the error message to be printed in upper/lower case. Next, the error routine will see if the error occured in the immediate mode. If it did, then control is passed to the ready prompt. Otherwise, it will print IN nnnn where nnnn is the line number in which the error occurred. Then it will skip a line and print the line number and a space. DE is now loaded with the line number and a search is made at the FNDLIN routine. A listing of the subroutines used in this program is provided in table 2. When the line is found, HL will contain the address of the line and DE is loaded with the address of the error. The line is printed until HL equals DE and a question mark is displayed. The rest of the line is printed and control is returned to the user.

Line manipulating keys

The LMK decoder is in lines 3350 - 4670. This routine is called from BASIC at the first letter of the input routine. The routine first makes sure that it was CALLed from the READY prompt. If it wasn't, control is returned to ROM's input routine. If it was, the computer waits for the first character.

If the first key pressed is not a LMK then control is returned to the input routine. Otherwise, control is sent to the proper location for that key. As soon as control is passed to the right location, the routine CLEAR will be called. This will clear the stack and set the cursor one line up. I did this so that subsequent pressings of a LMK will produce a neat, listing-like display. Then the program will find the correct line to list or edit and send control to the LIST or EDIT routine.

Abbreviations

The abbreviation routine is in lines 5050 - 5690. This routine is called from the tokenizing routine and is meant for disk BASIC to be able to add extra token, if needed. I use it for a different purpose, of course, to detect and accept abbreviations. The routine first looks at the line to see if the first letter is a valid abbreviation. Then it makes sure it is not a variable by looking for an equals sign. If all is well, DE is loaded with the execution address and pushed onto the stack so BASIC can jump to it from the RST 10H subroutine (see table 3 for a listing of ROM subroutines used in this program).

Base conversions

The base conversion routine, which is in lines 5700 - 6140, is called when BASIC encounters an & token. The routine first makes sure that an 'H', 'O', or 'B' follows the &. If it

doesn't, then a syntax error is generated. If it does, then the location COUNT is loaded with a one for binary, a three for octal, or a four for hexadecimal. This number tells the MULT routine to multiply HL by two, eight, or sixteen, respectively. The program uses the following formula for finding the decimal equivilent:

(((D1 * m) + D2) * m) + Dn

where D1 through Dn represent the digits one through n and m represents the multiplier.

The process continues for all digits until a non-hex digit is found or HL > 65535. If the latter happens, an illegal function call error is generated and control is returned to BASIC. Otherwise, Work Register Area 1 (WRA1 where intermediate results are stored) is loaded with HL and the mode flag is set to two, which specifies an integer.

Tab extender

Of all the articles I've seen to make Model I tab to columns larger than sixty-three, they were all clumsy BASIC programs that had to be merged, renumbered, etc. My tab routine actually lets you type something like PRINT TAB(75) or any number up to 255. In the BASIC tab routine, the value is ANDed with 3FH (63 decimal) and then calls my routine. I re-evaluate the value and let the computer accept any number up to 255.

Using TRSTOS

Dennis Cardinale

Type in the program on a 16K or larger Editor/Assembler. You cannot enter the program using TBUG or DBUG or any other debugging utility because of the multiple origin statements. Assemble the program using the name TRSTOS and be sure to check for errors since this is a very long program. Save the source code (unless you feel like retyping if you made a mistake) and type B <ENTER>. Press enter for memory size and type SYSTEM. At the prompt, type TRSTOS and the program will load. At the prompt again, type a slash ("/") and press enter. The screen will clear and "MEMORY SIZE?" will appear. If you want to save memory for a program high in memory, type in the appropriate memory size or just press enter.

You are now in TRSTOS! Try it out. Type "A" and press enter. You should now be in the AUTO mode. Type in a program with an error. Type in R for run and the error should be displayed along with the bad line and the location of the error. If it was a syntax error, you should be in the EDIT mode. If you are, hit <ENTER>. Now test the LMK's. Press the arrow keys with or without <SHIFT> and the corresponding lines should be displayed. Press the period and it should list the current line. Press the comma and you should be able to edit the current line. Now press enter and type <SHIFT> 0. Type a letter and it should be lower case. type <SHIFT> and a letter and it should be upper case. Turn on your printer and type <SHIFT> <down arrow> @. The screen should be sent to the printer. Type LPRINT TAB(75) &H1234 and 4660 should be printed at column 75. Also try &O and &B.

If one or more features don't work, go back to your Editor/Assembler and find the error in the section that is giving you problems. Since this program is modularized, it should be easy to fix. If all's well, you have a successful TRSTOS tape operating system. A note to all new ROM owners ("MEM SIZE?" ROM): the keyboard routine may not work without modification because the keyboard debounce routine is slightly different from the old ROMs.

Use on other systems

If you have a Model III or a Model I with upper case only, this program can be easily converted for your system. For Mod III owners, you don't need the upper/lower case driver or keyboard driver, so delete these. You can also take out the screen print routine. Remember to remove the JumPs and DEFWs. The error routine, LMK's and abbreviations should work on your computer.

For upper case Mod I owners, take out the video routine, but not the keyboard routine. you need this for the screen print routine. Also, remove lines 2490 and 2500 and the lock routine. This removes the <SHIFT> 0 capability.

Expanding the Program

Since this program is modularized, it is extremely easy to expand. Just type in ORiGin statement at the beginning with the

address of the exit you want to Patch and then a JP or a DEFW to your routine's address. Then type in the addition, but be sure lines 6420 - 6470 always come last. If you think your addition is outstanding, I'd like to hear about it. If you have an idea, but you do not know how to go about writing it, send me a self addressed stamped envelope and I'll see what I can do.

Well, I know that as soon as you saw this article, you jumped for joy and exclaimed, "This is it! This is just what I've been waiting for!" Don't feel embarrassed, though, I do the same thing when I see a good Program. Also, If you have a question, comment, complaint, or Problem, drop me a SASE and I'll be happy to reply.

Table 1. Disk BASIC Exits used in this Program

401EH	Address of video driver.
4016H	Address of Keyboard driver.
41A5H	Exit from error routine.
41AFH	Exit from keyboard input routine.
41B2H	Exit from statement tokenizing routine.
4194H	Exit from evaluation routine - & processing.

Table 2. Software routines in this program

PRINT Save DE and print line pointed to by HL.

PRCHAR Save DE and print character in A register.

LOCK Toggle upper/lower case.

FNDLIN Find address of line in DE and set flags.

VIDEO Video driver routine.

KEYBRD Keyboard driver routine.

ERROR Error printing and locating routine.

PR2 Print error message in upper/lower case.

ERR Print a space and a question mark (" ?").

ERR1 Print just a question mark ("?").

KEY Expand token in A to proper keyword.

INPUT LMK Processing routine.

CLEAR Clear stack and position cursor one line up.

CURLIN Find current line number.

LIST List line to video.

EDI Edit current line.

DN list next line.

AUP list first line.

ADN list last line.

UP List previous line.

PRSCRN Send entire screen to line printer.

BREAK Scan for <BREAK> key.

ABBRV Abbreviation recognition routine.

CONV Base conversion routine.

MUL Multiply routine for base conversion.

INIT System initialization when TRSTOS is loaded.

Dennis Cardinale

Table 3. ROM routines used in this Program

10H Examine next character pointed to by HL and set flags.

18H Compare HL to DE and set flags.

3BH Print character in A register.

32AH Print character in A register to video.

44BH Continuation of ROM keyboard driver routine.

0FA7H Print IN nnnn with nnnn in HL.

0FAFH Change HL to ASCII and send to video.

05D1H Get Printer status.

1650H Address of token listing in ROM.

1A19H Return to READY Prompt.

28A7H Print line pointed to by HL.

Figure 1. Memory map while using TRSTOS