NORTHERN BYTES

Volume 6 Number 1

GREETINGS! Welcome to a brand new volume of NORTHERN BYTES! If you missed any of the eight issues of Volume 5, they are all still available at a price of $2.00 each (if you live outside the U.S. and Canada, and wish airmail delivery, add $1.00 per issue). I hope that the quality of Volume 6 will surpass that of Volume 5.

I can always tell when winter is here, for two reasons: One is that fluffy cold white stuff that starts piling up outside, and the other is the avalanches - no, not of the fluffy white stuff, of the mail that starts pouring in! Seems that everyone rediscovers their computers when the chill winds of winter drives us all indoors. So, please, I ask again that you not be too disappointed if you write us a letter and don't get a personal reply. NORTHERN BYTES is pretty much a one-man operation, and I don't intend to spend all my free time chained to a computer. If you feel you must get in touch with me, I'd prefer you call me at (906) 632-3248, with the understanding that if I'm not home and you leave a message for me to call you back, I will make that call collect. If you do write and you live in the U.S.A., you MUST enclose a self-addressed stamped envelope if you wish a reply (Canadians enclose a Canadian quarter instead of a stamp, elsewhere in the world, don't worry about it, I'll cover the first round).

Folks, my memory (the stuff in my head, not in my TRS-80) is terrible sometimes. So, if I promised you something and didn't deliver, PLEASE don't be shy about dropping me a postcard and reminding me! Also, if you've sent an article in for NORTHERN BYTES, please check your mailing label to see that you were properly credited with six issues. And a reminder - if we've reprinted an article you wrote for another newsletter, you need only send us your name and address to get six issues of NORTHERN BYTES free.

And one more reminder (if everyone will read this, it will save me from having to answer a ton of mail): There are basically three ways to get NORTHERN BYTES on a regular basis: 1) A copy of the current issue is sent FREE with any order from The Alternate Source. Or, if you do not wish to order, you can be added to the TAS "active" mailing list and get the current issue of NORTHERN BYTES for $2.00. 2) If you write an article for NORTHERN BYTES, you get a six-issue "subscription" free. 3) If you have a VISA or MasterCard and you want to be absolutely sure of getting each issue as soon as it's printed, you may send us your credit card number and expiration date, and request that we automatically send you each new issue (and charge it to your credit card) as it becomes available. We don't encourage this, but will do it as a convenience for our readers. The cost is still $2.00 per issue in the U.S. or Canada, or for surface mail elsewhere ($3.00 for overseas airmail).

This issue of NORTHERN BYTES contains two often-requested features. One is an index of the articles in Volume 5, Numbers 1-8. The other is a list of TRS-80 user groups that we are aware of. Unlike some other publications, we actually have sent out postcards to clubs we haven't heard from in a while, to see if they're still around. This is actually our "exchange newsletter list", so most of these groups are currently publishing a newsletter (a few of the groups are having difficulty getting one out regularly, but that's to be expected in any group of volunteer organizations).

Question: Do any of you folks have any information on long distance telephone services that can be accessed from anywhere in the country, possibly via an (800) number? I think such animals exist, but have been unable to locate any of them. The problem is that here in Sault Ste. Marie, we're about 150 miles from even a medium-sized city (not counting Sault Ste. Marie, Ontario, Canada, which is about one mile away as the crow flies), and none of the big long distance carriers seem to be in any big hurry to extend their service up here (even though we're on Michigan Bell and have electronic switching). You'd think that if nothing else, a WATS line reseller would be interested in putting a switch up here, but so far, no such luck. Now that MCI Mail is discontinuing their free (800) number access (see story elsewhere in this issue), such a service could prove to be more valuable than in the past.

If my comments this time sound a bit rambling, it's because it just seems like there aren't enough hours in the day to get everything done. I'd like to put together a couple more Public Domain disks, but just haven't had the time yet. The next one we release will have a public domain Bulletin Board System on it, for those of you that want to try out being a SYSOP.

That brings up one more thought. Would anyone out there be interested in a Bulletin Board System sponsored by Northern Bytes and/or The Alternate Source? We've been thinking about it for a while, but haven't been able to make the commitment of time and/or finances. There are a lot of "hidden" costs involved in running such a system. If we did go to the trouble to set up such a system, would anyone access it? Suppose we charged for the service but made it available via a WATS line? At this point, it's just a thought, but we're open to suggestions (especially those that would allow us to serve the greatest number of users at the least cost for everyone involved).

Something I've envisioned is a network of BBS's across the country that would operate as local BBS's during the day and evening hours, but late at night would be called (twice, at predetermined times) by a "headquarters" BBS (which presumably would have access to an outbound WATS line or other cheap long distance service) to exchange messages destined for other BBS's in the system. Thus a user in New York could leave a message on his local BBS for a user in California, and late at night his local BBS would upload that message to the "headquarters" BBS (wherever it might be) which in turn would download it to the BBS in California. It might even be possible to allocate each participating system so much "public message" space, that would then become part of the common message base on all of the BBS's the next day.

The "headquarters" BBS would call each of the participating systems twice each night. First, it would call to receive uploads from the system; then it would sort all private messages as to the proper destination BBS, and construct the public message base from the public messages received (perhaps sorting them by time entered, or by subject categories); and finally it would call each system back to deliver the private messages for that system, and the public message base. If the system were to grow, there might even be the possibility of having "sub-headquarters" systems, that would collect and distribute messages within a region, and pass along to the "headquarters" computer messages destined for outside the local region. A regional "sub-headquarters" computer could service, for example, all the BBS's accessible over a "band one" WATS line (only the immediately surrounding states), thus ultimately cutting the phone charges at the "headquarters" BBS.

Participating local BBS's would be expected to contribute financially toward the operation of the "headquarters" system(s) (especially the telephone toll charges). In addition, compatible software packages would have to be designed to permit the operation described above (one package for the "headquarters" and a compatible one for the local systems). Finally, it would be understood that this would be a "hobbyist" system, not intended for business traffic or to replace the existing electronic mail services (not that messages would be "censored" to prohibit business transactions, but to make it understood that all messages are transmitted at the user's risk, and that delivery of messages is NOT guaranteed).

Anybody want to write the software for this? Besides helping your fellow computerists, it could turn out to be a profitable project, since each BBS on the system would have to purchase the necessary software.

The above idea is just a suggestion. I am, in effect, "running it up the flagpole to see if anyone salutes it". If you feel you have a better proposal, feel free to send it in. Who knows? Perhaps we can help design the next generation of bulletin board systems, and bring electronic communication within the reach of everyone!

THE EXTERMINATOR

Folks, when you get a new copy of NORTHERN BYTES, I encourage you to check out "THE EXTERMINATOR" right away, and then (if you have the original issues of NORTHERN BYTES) go back and correct the BUGS in your original copies, or at least make a note in the margin to the effect that a correction appears in issue xx. That way, should you decide to enter a program, patch, or whatever at a later date, you won't accidentally enter a "buggy" version. And, if you happen to be a newsletter editor, you won't accidentally reprint the bugs!

Anyway, it's too cold for BUGS in Northern Michigan at this time of the year, so John Hallgren sent me a few from sunny Florida! The first one goes all the way back to my SETDATE article that appeared in Volume 5, Number 2, page 7. Near the bottom of column one on page seven, I gave a patch written by John that disables the DATE and TIME prompts when you boot up a copy of TRSDOS 1.3. Trouble is, it didn't work, and that's because I only published half of the patch! Here's how the paragraph SHOULD have read:

> TRSDOS 1.3 - Sorry, nothing as simple as a SYSTEM command setting here. According to John Hallgren, who is probably as much of an expert on TRSDOS (in its various forms) as anyone, the patches to skip the date/time prompt on power-up are as follows:

```
PATCH *0 (ADD=4EB1,FIND=06,CHG=17)
PATCH *0 (ADD=4ED4,FIND=3A81,CHG=184D)
```

For those of you who recall that I published a different patch to correct this problem in "THE EXTERMINATOR" column in Volume 5, Number 4, John informs me that the patch printed there also works but in a different way (and changes exactly the same number of bytes!). The difference is that John's patches will set the date to 00/00/00, while the other patch simply skips the date entry but does not affect whatever may already stored in the date field (which is usually NOT a valid date!). Of course, if you are using SETDATE, that will put a valid date into the date field so if you have already applied the other patch, you don't have to change it. But for the record, John's patch would have worked if I had printed the whole thing (and is definitely the preferred patch to use because it gets rid of any garbage in the date field, even when SETDATE is not used)! The problem occurred because of a misplaced period (indicating that the end of the patch was after the first line) in the newsletter from which I reprinted the patch.

Unfortunately, that's not the only BUG in one of John's programs. On page 5 of Volume 5, Number 4 I managed to let not one, but TWO typos slip by unnoticed (I've GOT to learn not to proofread at 2 A.M.!). The purpose of the program in question was to allow a user of TRSDOS 2.7DD to do a DIR of a non-system disk in drive zero by using a new (X) parameter. If you tried it and it didn't work, don't blame John, it was a pair of typos in NORTHERN BYTES!

First of all, near the end of the paragraph that ends about one-third of the way down page 5 in column two, we left out some text, which sort of destroyed the meaning of the sentence in question. Here's how it SHOULD have read, with the omitted text underlined:

> ... If "X" was specified, I then need to establish intercepts at the standard exit points so that the prompt for the SYSTEM disk can be issued. This is done in lines 360-430.

In the original article, another paragraph followed the one corrected above. Then, about halfway down the page, the actual patches were printed, and the second patch line was wrong! It should have read as follows:

```
PATCH *21 (A=7,B=083,F=41 20 20 20 20 20 5C, C=58 20 20 20 20 20 62)
```

In the above line, the portion printed in error is underlined. I apologize to John and to all of our readers for these errors!

Before I go, one more BUG to wipe out. On page 15 of Volume 5, Number 6, I reprinted a couple of NEWDOS/80 zaps from the CINTUG newsletter. The one labelled "ZAP 086" (found about two-thirds of the way down in column one) is slightly wrong. The first line of the zap should begin at SYS16/SYS,02,E8 (not EB). And, the second line says "change all zeroes to ..." However, the "02" at the end of that zap is ALREADY THERE, so "all zeroes" is not quite correct. I have applied this zap (starting at byte "E8") to a copy of Model III NEWDOS/80, and it seems to work fine!

LETTERS DEPARTMENT

Dear Jack:

Please forward this note to Arne Rohde if you can. I couldn't find any address in Volume 5, Number 7. Thanks.

Dear Arne:

Here is a patch to modify the Model I version of M-ZAL release 3 to accept upper and lower case text from the keyboard. Your keyboard must have an upper/lower case modification installed, and your DOS must accept lower case.

As suggested in the manual, copy the module TXEDIT/CMD to a working disk using the filespec TXBASE/CMD. Then make these patches:

```
     change ABAFH from 1BH to 7BH            (1)
            ABB2H   "  16H  " 6CH            (2)
            AD6BH   "  30H  " 28H            (3)
            AEE1H   "  5BH  " 7BH            (4).
```

In DUMPing the editor back to disk you need to know that the module starts at AB00H, ends at BD8CH, and its entry point is AB0FH. The patched editor may be relocated in RAM using the same procedure as for the original editor.

Change (3) allows lower case input in the edit mode, preserving the up-arrow for scrolling. Change (4) allows lower case input in the line-insert mode. With these changes, the original character insert/delete commands (shift-I/shift-D) now serve only for text entry. I decided to change the character delete to shift-down-arrow-D. Change (1) takes care of this. The logical character insert command would be shift-down-arrow-I, but the keyboard confuses this with the right-arrow used for cursor movement. As a long-time SCRIPSIT user, I modified the character insert command to shift-down-arrow-S through change (2).

Control codes (00H-1FH) from the keyboard are XORed with 7FH prior to a search for a match on the lookup table beginning at ABAFH. The original character delete command was 64H, or lower case d. Thus the unpatched table value at ABAFH was 64H XOR 7FH = 1BH. The original character insert command was 69H, or lower case i. The table value at ABB2H was 69H XOR 7FH = 16H. Shift-down-arrow-D produces 04H, and shift-down-arrow-S gives 13H. XOR these with 7FH to get the values I used in changes (1) and (2).

Line commands and extended commands must be invoked using upper case keystrokes. For example, you will get an error message if you press CLEAR followed by a lower case i to enter the line insert mode.

Yours truly, Craig Carlyle

[Editor's note: Thanks, Craig, on behalf of Arne and other other NORTHERN BYTES readers that may have had need for these patches. I have forwarded your letter to Arne at his new address in New Zealand.]

Dear Jack,

In reading the documentation for TRSDOS 6.01.02 I get rather confused about the functions of Set, Route, Link, and Filter. Their approach of heavy description and light examples should be just the reverse. If they would start out with many examples and then explain the difference between the examples it would be much easier to follow.

I would like to see some examples (many) of how these various commands may be used.

Enjoy your paper very much.

- B. Jim Smith, Saint John's High School, Shrewsbury,
Massachusetts 01545

[Jim, I don't use TRSDOS 6 any more than I can possibly help it, so I'm afraid I can't be of much help here. I'm sure that some of our readers are familiar with TRSDOS 6, and perhaps one of them would like to write something up to assist the rest of us. In any case, I've printed your full address so that folks can contact you directly, should someone wish to go that route. Let us know if you find some better documentation on these TRSDOS 6 features!]

Dear Jack,

... I have a few questions for you (or Northern Bytes):

If I save a BASIC program in ASCII and I wish to read the file in SuperScripsit (with the A option), I will get an END OF FILE ENCOUNTERED error. So now read the file with Scripsit and save it again in ASCII. Then SuperScripsit can read the whole file. Is there a patch for SuperScripsit so it will work properly?

I am trying to figure out some NEWDOS routines. Most of the routines need the PDRIVE setting and the SYSTEM setting. Do you know where exactly in memory (which address) those parameters are stored?

- Paul Fransen, Havikhoek 48, 3201 SH Spijkenisse, HOLLAND

[Unfortunately, I don't use SuperScripsit, so I passed that question along to Greg Small, and he came up with the following:

"SuperScripsit is expecting a 00H byte as a terminator. This IS what Scripsit uses but is NOT what BASIC uses during an ASCII save. So, loading the file into Scripsit and resaving it will allow SuperScripsit to now work on the program.

"The solution is simple. Using Newdos/80's CREATE command, enter the following:

```
CREATE ZEROBYTE/TXT LRL=1 REC=1 <ENTER>
```

This will create a one byte file that is, interestingly enough, a 00H byte. To fix a file that SuperScripsit didn't like simply use the following:

```
APPEND ZEROBYTE/TXT filespec <ENTER>
```

which will fix the file for SuperScripsit use."

As for the NEWDOS/80 PDRIVE tables, they are stored in memory starting at 4371H in the Model I or at 4291H in the Model III. Starting at that address is a 10 byte table containing the PDRIVE specification for Drive 0 in coded form. The next ten bytes, starting at 437BH (Mod I) or 429BH (Mod III) are the table for Drive 1, the next ten bytes for Drive 2 and so on. As for the SYSTEM command, I could be wrong about this, but I think that executing the SYSTEM command (in such a way that SYSTEM flags are changed) actually modifies portions of the DOS itself – in other words, I don't think the SYSTEM table is stored in memory as such during DOS operation. One of the articles I've been planning to reprint from another newsletter describes how the SYSTEM flags are stored on a NEWDOS/80 disk, and I have sent you a copy of that article. I hope it helps!]

[This letter was forwarded to me by Charley at TAS!]

... I'm interested in computer chess programs and literature. I have disassembled Microchess 1.5 and Sargons I & II but am having troubles understanding the subroutines. If you know of anyone with similar interests I would appreciate your passing along my name.

– Ben Ottmer, 130 Forest Drive, Lakewood, New Jersey 08701

[Ben, see the next letter!]

Dear Mr. Decker:

I saw my first copy of Northern Bytes when I ordered some software from The Alternate Source for my Model 4 TRS-80. I notice your readers seem to be spending a lot of energy trying to break the protection on Super Utility Plus. If they can do that, perhaps they, or you, can help me with a related project. I have a copy of the Model III Sargon II, purchased from Hayden Software. It is on a disk with its own operating system, and I would like to convert it into a CMD file able to run under LDOS 5.1.3. This is not to make copies of it; I can copy the whole disk using Trakcess, so I don't need a CMD version to make backups. But I have various utilities like Logical System's LSFEDII disk editor which only work on CMD files, and I would like to be able to use them on all my programs.

– Gary W. Shanafelt, 2128 1/2 S. 18th Street, Abilene, Texas 79605

[Gary, I hope the Super Utility thing was a passing fad. I think the reason so many people were motivated to try and break the protection was that one of the many uses for Super Utility was to be able to copy "protected" disks, which it would do in many cases, but it was specifically designed to not be copyable itself! Apparently a lot of folks felt that since Super Utility could be used to "pirate" other software, SU itself ought to be a bit more vulnerable. Besides that, Super Utility author Kim Watt seemed to spend an awful lot of time and effort changing the protection scheme (it was rumored that at one point the protection was changed after each 100 copies of SU were produced), so I think that with some folks cracking the protection became a challenge, sort of like solving an adventure-type game. In any case, I do NOT want NORTHERN BYTES to become a forum for exchanging methods of cracking protected software, but I can sympathize with your desire to make a CMD file of your program (I have always hated protection schemes that use an entire disk to store a single program!).]

[Another letter forwarded to me by Charley at TAS!]

... Is anyone working on how to copy COPYCAT3? It looks like a real challenge. My efforts so far have failed. The program seems to make some sense out of the trash in tracks 2-8. Would be interested in anything you can send about this subject.

– Edward O. Noble, P.O. Box 759, Mesilla Park, New Mexico 88047

[Good Grief! Here we go again?! Tony Domigan has commented that Copycat 3 has a variation of Super Utility's loader encryption, but that's all I know about it. Readers – any help?]

MODEL III SELF-BOOTING DISK USING NEWDOS/80

by Leonard Yates

(with a little help from Gary Bryce)

[This article has an interesting heritage. We ran an article here in NORTHERN BYTES (Volume 5, Number 4) entitled "CREATE A SELF-BOOTING DISKETTE USING NEWDOS/80, which was written by Joachim Kelterbaum of Essen, West Germany. Gary Bryce of Sydney, Australia got hold of it, discovered it wouldn't work quite right for him, so he made some corrections and published them in the SYDTRUG NEWS. We reprinted those corrections in "The Exterminator" column in NORTHERN BYTES Volume 5, Number 6. Then Leonard Yates read the article in the SYDTRUG NEWS, discovered that further corrections were needed for the method to work with Model III NEWDOS/80 diskettes, and wrote the article that follows. This article is also reprinted from the SYDTRUG NEWS (the newsletter of the Sydney TRS-80 Users Group, P.O. Box 297, Padstow, N.S.W. 2211, AUSTRALIA.]

Back in the August 1984 issue of SYDTRUG NEWS, Gary presented an article on creating self-booting disks using NEWDOS/80. The Model I mods worked well but, as he said at the end of the item, had not been verified on the Model III. Well, to get a Model III data disk to self-boot, you'll have to proceed as follows (I'll make this complete so you don't have to refer to the original article):

1. FORMAT a data disk and copy the /CMD file to be made self-booting to it. If it's the only file on the disk (apart from BOOT/SYS and DIR/SYS), chances are it will have only one extent (as revealed by DIR dn1 A). If it has more than one extent, it's probably more than 32 granules long (Super Utility 3.2 is the longest program I've been able to self-boot).

2. Using the DFS option of SUPERZAP, note the DRS of relative sector 0 of the /CMD file on the data disk. Next, using the DD option of SUPERZAP, and responding with the drive number and DRS noted above, record the TRS and TRK (in HEX) of the /CMD file on the data disk.

3. Using the DFS option again, zap FRS 1 of BOOT/SYS (of the data disk) as follows:

Byte 05 – TRS of /CMD file (will be 05H if
it's the only USR file on the disk, and
hence won't need to be changed).
Byte 3E – change from C8H to C9H.

Note: If the /CMD file is the only USR file on the data disk, TRS will be 05H and TRK will be 00H. BOOT/SYS is normally set up to load SYS0/SYS which always occupies from sector 05 of track 0; for a non-system (data) disk, the first USR file will occupy this space. Note also that on NEWDOS/80 system and data disks for the Model III, sector 1 of BOOT/SYS is a duplicate of sector 0 (required for booting on the Model III). These changes are made to sector 1 only. Further, if your /CMD program is the only USR file on the data disk, the only zap needed is to byte 3E of FRS 1 of BOOT/SYS.

4. Mount the self-booting data disk on drive 0 and press RESET. The disk will boot and the /CMD file will load and execute.

CHRISTIAN COMPUTER USERS ASSOCIATION

This is the first organization I've heard of that offers support to persons using computers in Christian-oriented applications, AND that has a TRS-80 Special Interest Group. The TRS-80 SIG has even published two issues of its own newsletter (edited by Bob & Gail Grommes). The purpose of the TRS-80 SIG is "To serve members of the Christian Computer User's Association who own a TRS-80 Model III, 4, 4P or compatibles so that their use of these computers as a tool of ministry is more informed and effective." They also have a library of public domain software of interest to Christians using TRS-80 computers, Models III, 4 or 4P. For more information, drop a line to The Christian Computer User's Association, ATTN: TRS-80 S.I.G., 1145 Alexander Street S.E., Grand Rapids, Michigan 49507.

DOSPLUS MODELA/III LOAD PATCH

by Dave Bower

As any DOSPLUS 3.5/Model 4P user has already found out, the Model 4P will not automatically load the Model III ROM image from a DOSPLUS 3.5 diskette. I am going to show how to patch the DOSPLUS 3.5 BOOT/SYS file so that the 4P can locate the MODELA/III file. And while we're in there we'll also put the 4P into high speed everytime we boot.

The Model 4P has no trouble discerning a Model III diskette (of any kind) from a Model 4 diskette. The problem is finding the directory on non-TRSDOS and non-LDOS diskettes. The 4P looks at the third byte (relative byte 2) of the second sector (relative sector 1) for the location of the directory track. DOSPLUS stores this information in the third byte of sector 00.

The solution is to put a directory byte where the 4P expects to find it. And if you change that one byte, the 4P will load the ROM image from a DOSPLUS diskette. BUT (and here's the kicker) when the 4P then proceeds to boot that DOSPLUS diskette it crashes (big time) because you have thrown a monkey wrench in the BOOT file. That's right, our directory locater is right smack in the middle of executable code.

So we move the first eight bytes elsewhere in the sector, add a RETurn (C9) to the end of it, and call it as a subroutine. And we'll also put the code to put the 4P into the high speed mode in that subroutine. Why 8 bytes? Well, 2 bytes before the directory byte are wasted, the directory byte, 2 bytes for padding so we don't leap out of the middle of an instruction and 3 bytes for a CALL to our subroutine. And don't forget that the computer is still going to execute this code so the first three bytes, including the directory byte, have to be executable code, that doesn't affect the program.

There is not a lot of room to work with. The ROM (or in our case the ROM image) loads the second sector of a Model III diskette to boot the system. That is 256 bytes. To make room I did away with the disk error routine (14 bytes) and the "DISK ERROR!" message (11 bytes). I relocated a subroutine into the old error routine so that I could put my patch into one continuous area (mainly because I didn't have room for two calls (3 bytes each) and two returns (1 byte each) If I broke it up). The error routine is for a fatal error and hangs the system, so I did the same in seven bytes (vice 25). And finally I had to change two references to the subroutine I relocated.

Use Super Utility (or a similar zapping program) to patch the second sector (sector 01) of the diskette. All numbers are given in HEX and the locations are relative locations in the sector (i.e. the fist byte is 00 and the last byte is FF). If your old code doesn't match my old code, DON'T DO IT, IT WON'T WORK! I don't know if there is more than one version of DOSPLUS BOOT/SYS on the street, but if I come across any others I'll provide the patch, if possible.

Once you have patched and tested a diskette to your satisfaction you can copy that BOOT/SYS to your other DOSPLUS diskettes. Since it has an access level of 7 (no access by anybody, for any reason, under any circumstances) you have to remove the passwords from the two diskettes using Super Utility. This patch should have no effect on the performance of this diskette in a Model III. And this technique should be adaptable to most any other DOS. You need about 9 free bytes minimum and you can usually get this by cannibalizing an error routine. One last note, any DOSPLUS configuration file made on the Model III (or 4P in the slow mode) will put the 4P back in the slow mode, even after this boot.

Here is the patch:

```
==> ** is the actual location of the directory.

Rel. Location   00
Old Byte        3E 3C 32 49 40 21 DD 43
New Byte        00 3E ** 3E C3 CD EB 43

Rel. Location   71
Old Byte        21 F2 43 06 0B 7E 23
New Byte        3E C5 CD 33 00 18 F9

Rel. Location   78
Old Byte        CD 33 00 10 F9 18 FE
New Byte        D3 F0 06 0E 10 FE C9

Rel. Location   C0
Old Byte        EB
New Byte        78

Rel. Location   CA
Old Byte        EB
New Byte        78

Rel. Location   EB
Old Byte        D3 F0 06 0E 10 FE C9 44 49 53
New Byte        3A 10 42 CB F7 32 10 42 D3 EC

Rel. Location   F5
Old Byte        4B 20 45 52 52 4F 52 21 00
New Byte        3E C3 32 49 40 21 DD 43 C9
```

Disclaimer: Install this at your own risk. I make no promises to its suitability or reliability on your system. Which is a polite way of saying that if after installing these patches your computer calls up the banks computers and transfers all your funds to my Swiss account, sorry 'bout that. Must have been a typo somewhere.

MODEL III MOTHERBOARDS AND OTHER SURPLUS BARGAINS

Attention hardware hackers! How would you like to buy the guts of a TRS-80 Model III for only $35 ($29 each in quantities of 10 or more, or as low as $19 each if you buy 100)? They're being offered by Arnold Company, 214 Hill Lane, Red Oak, Texas 75154 (telephone (214) 576-2291). What's the catch? Well, the boards were pulled from Model III's returned to Tandy for service. Tandy replaced the board rather than fixing it, and to make sure they didn't see these defective boards again, they punched a hole in them (usually small and usually only damaging some of the traces). Arnold Company claims that none of these boards are "basket cases" and if you are not satisfied, you may return the boards within ten days for a full refund. The boards come without RAM or ROM, but most have the character generator and Z-80 CPU. If you order 20 or more boards, they will send at least one with the full ROM while they last. They will also include schematics, a parts list, and trace photo with each order.

This company bare 68000 CPU boards (a "TRS-80 Mod 2 to 16 upgrade board") for $39, a "Bare Mod 16 CPU board only" for $19, and a "TRS-80 Mod 3 bare circuit board with documentation" for $19). A Model 2 power supply is $39.95. And, if you bought some "bare" disk drives and would like a case to put them in, a "dual 5 1/4 case only for 2 std size drives (almond)" is $14.50 plus $2 additional shipping per case. If you also need the power supply for the drives and want to build your own, a "bare dual P.S. board for above case with easy to follow assembly instructions" is $9. They also have many other goodies (such as new TI99/4A keyboards and new TI99 RF Modulators for $9 each, 10 at $8 each, 100 for $4 each; and Hi-Tek numeric keypads for $6.95, or $5 each for 10 or more). U.P.S. ground shipping is $3.00 for the first 10 items, 30 cents for each additional item after that.

The ad we received will be running in the February issue of the Computer Shopper, so if you want any of the TRS-80 boards, you'd best order now. If you do, tell them you saw this mention in NORTHERN BYTES!

A TIP FOR MODEL 4 & 4P HACKERS

If you have access to the January, 1985 issue of 80-Micro, don't overlook Hardin Brothers' review entitled "This is TRSDOS 6.2" beginning on page 158, especially the box entitled "Notes from The Source" on page 160. There are a few goodies here, including how to reprogram the function keys (the lookup table for their values is stored from 0918H to 091DH), how to find the date storage area, and how to activate the KILL command in addition to the REMOVE command - it's a one byte patch, as follows:

PATCH SYS1/SYS.LSIDOS (X'2054'="K")

These apply to TRSDOS 6.2 only, not 6.1 or earlier. The main article is a review of Logical Systems, Incorporated's published source code for TRSDOS 6.2, which, to quote Hardin, "aren't inexpensive; at $99 apiece, they may have established a record as the most expensive paperback books ever published."

FOR SALE

One 80 track single-sided MPI disk drive with case and power supply, recently adjusted and aligned by The Alternate Source, asking $150. Also one working Radio Shack Telephone Interface II 300 baud acoustic MODEM, asking $50. Call (517) 351-7738 and ask for Mark.

ACCESSING THE VIDEO ON THE MODEL 3 AND 4

by Laurie Shields

[Laurie Shields is the author of Zen, a Z-80 Editor-Assembler program for the TRS-80 Models I/III/4, that is distributed in North America by The Alternate Source. Laurie resides in Chesterfield, England.]

At a recent club meeting I was asked to produce a solution to 'PEEK'ing and 'POKE'ing the Model 4's screen. After the initial reaction of 'impossible', I recalled one of the SVC calls in the DOS that could copy the video memory to an accessible part of user RAM and also copy it back. With a little bit of machine code it should be possible to set up two USR's using these SVC's, and thus provide the long lost utility that no TRS-80 programmer should be without. The first problem was testing the logic, and as I only had a Model I to work with we ended up with versions for both the Models I/III and the Model 4, but don't discard the Model I/III routine as unnecessary as there are some hidden benefits.

In order to use the technique a block of memory equal to the size of the video RAM is needed, and the simplest way of allocating part of user memory for this purpose is to 'DIM'ension an integer array. Finding the size couldn't be simpler - since each integer requires two bytes of memory the whole Model I screen (1024 bytes) would need 512 integers. To avoid any wasted memory we will use the zero element in the array to store the first and second bytes of the screen, so the dimension statement will be: DIM SC%(511).

The machine code routine itself will need about 22 bytes of memory and we again have a number of choices: POKEing into reserved memory, packed strings, adding another 11 elements to the array for storing the screen, etc.; but the neatest way would be to set up another smaller integer array just big enough to hold the program. Let's call this one US%(n), where n will be decided after writing the routine.

On the Model I/III the screen is easily accessed, being memory mapped from 3C00H to 3FFFH inclusive. To copy the contents to or from our array we only need to set up the HL and DE registers as pointers, load BC with the number of bytes to move (1024 decimal), and execute an LDIR instruction. To keep the USR calls simple we will include the parameter 0 or 1 to indicate whether the the data moves 'Out' or 'In' relative to the array. Since a block move USR routine might come in useful for other purposes, we will make sure that the locations within the array for these source, destination and byte-count values are such that they can be easily modified from BASIC.

Now for the assembler part, this was written with Zen, the layout of the listing being slightly different from what you may be used to:

```
 1                   ;Screen/s
 2                   USRIN:   EQU  0A7FH    ; Model I/III ROM routine
 3                   VIDEO:   EQU  3C00H
 4
 5                            ORG  0A000H   ; could be anywhere
 6                            LOAD $        ; load code into memory
 7
 8 A000 CD7F0A       USR:     CALL USRIN    ; get USR parameter in HL
 9 A003 7D                    LD   A,L      ; put into A ignoring H
10 A004 B7                    OR   A        ; check if 0 or 1
11 A005 210000                LD   HL,$-$   ; get VARPTR(SC%(0)) into HL
12 A008 00                    NOP           ; padding
13 A009 11003C                LD   DE,VIDEO ; screen location into DE
14 A00C 00                    NOP           ; padding
15 A00D 010004                LD   BC,1024  ; prepare to move 1024 bytes
16 A010 2801                  JR   Z,MOVE   ; if move Out then skip
17 A012 EB                    EX   DE,HL    ; switch registers for move In
18 A013 ED80         MOVE:    LDIR          ; move data/screen contents
19 A015 C9                    RET
20                            END
```

For those who have never used machine code programs stored in integer arrays, the following diagram should make everything crystal clear. The memory addresses are shown as an offset from the first byte of array US%(10) found from VARPTR(US%(0)).

Memory Address	Decimal Value	Array Subscript	Array Value	Z80 Code	Instruction Mnemonic
+ 0	205	0	32717	CD	CALL USR Parameter to HL
+ 1	127			7F	
+ 2	10	1	32010	0A	
+ 3	125			7D	LD A,L
+ 4	183	2	8631	B7	OR A
+ 5	33			21	LD HL, VARPTR of US%(0) to be supplied
+ 6	0	3	0	0	
+ 7	0			0	
+ 8	0	4	4352	0	NOP padding
+ 9	17			11	LD DE, Video Screen Address
+ 10	0	5	15360	00	
+ 11	60			3C	
+ 12	0	6	256	00	NOP padding
+ 13	1			01	LD BC, 1024 Byte Count
+ 14	0	7	1024	00	
+ 15	4			04	
+ 16	40	8	296	28	JR Z, to MOVE
+ 17	1			01	
+ 18	235	9	-4629	EB	EX DE,HL
+ 19	237			ED	MOVE:LDIR 2 byte instr.
+ 20	176	10	-13904	B0	
+ 21	201			C9	RET

The tricky part is line 11, where we will load the HL register with the start of the integer array used for storing the screen. Since this array could be located virtually anywhere in memory this value can only be supplied at run-time. Conveniently, however, this address is stored in the 6th and 7th bytes of US%(n), thus forming US%(3). To put the address of first byte of SC%(0) there we use the BASIC statement:

```
US%(3) = VARPTR( SC%(0))
```

As the assembler has usefully put the object code into memory, we can use a little BASIC program to PEEK the bytes out in decimal format for inclusion into a DATA statement. We may as well resolve the problem of integers bigger than 32767 right from the start, so the program is:

```
100 'Getcode
110 A=&HA000
120 INPUT"Number of bytes ";B
130 FOR N = 0 TO B STEP 2
140 L = PEEK( A + N ): H = PEEK( A + N + 1 ): C = 256 * H + L
150 IF C > 32767 THEN C = C - 65536
160 PRINT C; ",": NEXT
```

Which produces this output:

```
 32717, 32010, 8631, 0, 4352, 15360, 256, 1024, 296, -4629, -13904,
```

The BASIC program incorporating the above as a DATA statement to test the logic on the Model I, was:

```
100 'Stest/Bas
110 DIM X%, US%, US%(10), SC%(511)
120 FOR N% = 0 TO 10: READ US%(N%): NEXT
```

```
130 DATA 32717, 32010, 8631, 0, 4352, 15360, 256, 1024, 296, -4629,
-13904
140 END
300 US% = 1: GOTO 320 '  Save screen     'I'nto the array
310 US% = 0:          '  Restore screen  'O'ut of the array
320 US%( 3 ) = VARPTR( SC%( 0 )):
   DEFUSR = VARPTR( US%( 0 )):
   X% = USR( US% ):
   RETURN
```

Run the program and after READY is displayed key GOSUB 300 , saving the screen contents; key CLS followed by GOSUB 310 and the screen is restored. The video is now very easily transferred back and forth to the array SC%, without any need for setting memory size or loading machine language utilities. Even better, the whole of the screen can be written off to disk or tape as a file, or as part of a file containing a number of different video displays. Such a technique could be used in a menu driven program where two or three different menus are used during different stages of the program's execution, and rather than clearing the screen and printing the menu each time the USR routine would give an immediate refresh. The time taken on reading the data from disk into the integer arrays during program initialisation would hardly be noticed.

However, when I started to write such a program to demonstrate the idea we came across the rather protracted procedure invented by Microsoft for transferring numeric quantities to and from file buffers. To do the job according to the book we would have had to write FIELD statements covering 128 integer elements with the associated 128 CVI, RSET and MKI statements. NEWDOS80 programmers would have it a lot easier using "FF" files and Igels, but even with that superb operating system the process was rather slow compared with the speed at which the data was moved to and from the screen. If only we had a foolproof way of finding out whereabouts in memory Disk BASIC had allocated the file buffer, then we could use our own block move USR and save messing about with FIELD, CVI, MKI etc.

In the end finding the buffer turned out a lot easier after a bit of thought. Simply FIELD a one byte string, 1 AS UB$, and then ask BASIC for the VARPTR(UB$). The value returned is the address of a 3 byte block of data giving the length of the string, the least and then the most significant bytes of the address of the string itself. These latter two values are the ones we need so the address of the file buffer is given by:

```
140 OPEN "R", 1, "SCREEN/DAT":
   FIELD 1, 1 AS UB$:
   UB% = 256 * PEEK( VARPTR( UB$ ) + 2) + PEEK( VARPTR( UB$ ) + 1)
```

Armed with this information and, remembering that the buffer is only 256 bytes, we can write a routine to send the contents of the array to disk as follows:

i) set the indicator US% = 0 for output from array
ii) set the element US%(5) = the file buffer address
iii) set the element US%(7) = 256; the size of the block to be moved
iv) re-define the USR address in case BASIC has moved the array
v) set up a FOR/NEXT loop count of 0 to 3
vi) set the element US%(3) = appropriate position in array US%(n)
vii) activate the USR to block move 1/4 of array to file buffer
ix) PUT the file buffer to the appropriate disk file record
x) Loop until executed four times.

To input from disk it is only necessary to do the reads before the block moves, with the appropriate indicator in US%. Combining the disk I/O and the USR sections together generated a program that would very successfully execute block moves in any possible combination. What was lacking, however, was a routine that enabled the video screen to be edited before being saved. The simple one line word processor:

10 IN$=INKEY$: IF IN$="" THEN 10 ELSE PRINT IN$;: GOTO 10
showed signs of promise, and a bit of experimenting indicated that the shifted-arrow keys performed a lot better than the unshifted. Note that on the Model III [and on some of the very last Model I's to be manufactured -editor] it is necessary to press SHIFT, down arrow & letter Z to generate the SHIFT-down arrow of the Model I. So the next stage was to test for CHR$(91), CHR$(10), CHR$(9) or CHR$(8) and change the values to 27, 26, 25 or 24 as appropriate. Controlling the cursor location by manipulating the values in &H4020/21 to the top, bottom, right or left hand edges of the screen in response to shifted arrow-keys if they were pressed ended up with a simple screen editor to cover all our needs. The shifted '@' is used to toggle a graphics mode whereby 64 is added to characters for printing, and one last problem, discovered later, was the need to inhibit printing in the bottom right hand corner of the screen, as this forces the ROM to scroll the screen up one line and ruin the display.

The final version of the Model I/III demonstration program, DSCREEN/BAS, is shown in the program listing below. The main features have already been described but a more detailed explanation of some of the finer points may be required.

Firstly a listing of all the variables and their use:

C4$	Cursor on	Chr$(14) Used by the screen editor.
C5$	Cursor off	Chr$(15) " " " " "
CU%	Cursor pos.	&H4020 stored as variable for speed.
GR%	Graphic switch	Used to generate graphics characters.
IN$	Inkey string	All keyboard input.
N%	Loop counter	Used during disk I/O as sector count.
SC%(511)	Screen array	Integer array to store screen data.
SE$	Editor string	Used by editor checking input for cursor keys.
UB$	Fielded var.	Used to find file buffer.
UB%	Buffer addr.	Address of file buffer.
UN%	Screen Number	Counter for screens in file Screen/Dat.
US%	USR parameter	Sets Into or Out from array SC%
US%(10)	USR array	Integer array storing machine code USR.
X%	Dummy	Used to activate USR routine.

Taking the program in sections:

Lines	Function
100 - 140	Program initialisation.
150	Generating the character string SE$. The screen editor checks all input characters against this string for validity and deciding appropriate action.
160 - 180	Printing the menu.
190	Switch on the cursor, call the INKEY$ subroutine, switch off the cursor, test for validity and Goto as required. The +1 forces a Goto 190 if non-valid key pressed.
191 - 198	Calls to subroutines as required ending with Goto menu.
199 - 204	Option 9, Display Disk Screens allows backstepping to a previous screen, editing any screen whilst on display and premature exit with the '@' key if required.
210 - 230	Standard routines.
300 - 320	Block move of screen as described earlier but video address and byte count values updated in case of alteration by the disk I/O section.
330 - 370	Disk I/O section. As described earlier with the variable US% being used to test whether Read from or Write to disk.
400 - 540	The screen editor, each part is explained.
400	Entry point for a completely fresh screen.
410	Edit screen entry point. Switch on cursor and get character.
420	Test input key for control code and if not then key must be character for printing and drop through to next line.
430	If cursor position is lower right hand corner then 410. Otherwise if in graphics mode add 64 before printing.
440	Exit from editor. Switch off cursor and reset graphics.
450	Clear key, Chr$(31), used to clear to end of line.
460 - 490	Change unshifted arrow keys to shifted.
500 - 530	Moving cursor to screen limits in response to the shifted arrow keys.
540	Switching the value in GR%, this line is in effect: IF GR% = 0 THEN GR% = -1: GOTO140 ELSE GR% = 0: GOTO140

Run the program and what you end up with is limited solely by your imagination. The potential is there to create any combination of text and graphics on multiple screens and store them safely on disk for re-use in other programs, with a stripped down version of the USR calls to read from disk, store in arrays and put on the screen without any delay whatsoever. So far the routine has proved very useful during discussions with commercial clients in creating displays for stock control, order entry, customer detail files and the like, where the ultimate user has the chance of seeing how his data will appear on the screen when the final package is presented.

[The program listing follows, but first, a word of advice from Laurie's letter which accompanied this article: "If you are using TRSDOS 6.X never use line-feeds for formatting in multi-statement lines (as in DSCREEN/BAS) and save to disk in ASCII format. This is what happens: As BASIC is writing an ASCII output file it finds an 0AH character and decides in its infinite wisdom that there ought to be an 0DH with it just for old time's sake. Now you can guess the problem on reading back - the 0DH is an end-of-line marker and the rest of the line following it cannot be loaded by BASIC so everything stops." So, don't save this program to disk using the ',A' option, and expect BASIC to read it back in! Here's the listing!]

```
100 'Dscreen/bas
110 DIM IN$, C4$, SE$, N%, US%, X%, UN%, UB%, C5$, UB$, CU%, GR
%, US%(10), SC%(511)
120 FOR N%=0TO10: READ US%(N%): NEXT
130 DATA 32717,32010,8631,0,4352,15360,256,1024,296,-4629,-13904
135 CLS:PRINT:LINEINPUT"Filespec for screen data (SCREEN/DAT)
: ";IN$:IFIN$="" THEN IN$="SCREEN/DAT"
140 OPEN"R",1,IN$:
  FIELD1,1ASUB$:
  UB%=256*PEEK(VARPTR(UB$)+2)+PEEK(VARPTR(UB$)+1)
150 CU%=&H4020:C4$=CHR$(14):C5$=CHR$(15):
  SE$=CHR$(13)+CHR$(31)+CHR$(91)+CHR$(10)+CHR$(9)+CHR$(8)+C
HR$(27)+CHR$(26)+CHR$(25)+CHR$(24)+CHR$(96)
160 CLS:
  PRINTTAB(15)"Screen to Disk Utility":
  PRINTTAB(15)"======================="
170 PRINT"
<1>  Screen to Memory        <2>  Memory to Screen
<3>  Disk to Memory         <4>  Memory to Disk
<5>  Screen to Disk        <6>  Disk to Screen
<7>  Edit Memory           <8>  Edit Disk"
180 PRINT"<9>  Display Disk Screens    <A>  Memory to printer
<B>  Disk to printer        <@>  Finish":PRINT:
  PRINT"Select  ? ";
190 PRINTC4$;: GOSUB230: PRINTC5$;:
  ON INSTR("123456789AaBb@",IN$)+1 GOTO 190,191,192,193,194,1
95,196,197,198,199,205,205,206,206,210
191 GOSUB400: GOSUB300: GOTO160
192 GOSUB310: GOSUB230: GOTO160
193 GOSUB220: GOSUB330: GOTO160
194 GOSUB220: GOSUB340: GOTO160
195 GOSUB220: GOSUB400: GOSUB300: GOSUB340: GOTO160
196 GOSUB220: GOSUB330: GOSUB310: GOSUB230: GOTO160
197 GOSUB310: GOSUB410: GOSUB300: GOTO160
198 GOSUB220: GOSUB330: GOSUB310: GOSUB410: GOSUB300: GOS
UB340: GOTO160
199 FOR UN%=0 TO LOF(1)/4 -1
200 GOSUB330: GOSUB310: PRINT@60,UN%;: GOSUB230
201 IF IN$="[" THEN UN%=UN%-1: IF UN%>0 THEN 200 ELSE UN%=
0: GOTO200
202 IF IN$="@" THEN UN%=LOF(1)/4-1: GOTO204
203 IF IN$="E" OR IN$="e" THEN GOSUB310: GOSUB410: GOSUB300
: GOSUB340
204 NEXT: GOTO160
205 GOSUB550: GOTO160
206 GOSUB220: GOSUB330: GOSUB550: GOTO160
210 CLOSE: END
220 PRINT: INPUT"Screen save number ";UN%: RETURN
230 IN$=INKEY$: IFIN$="" THEN 230 ELSE RETURN
300 US%=1: GOTO320:    ' Screen to array
310 US%=0:             ' Array to screen
320 US%(5)=15360: US%(7)=1024: US%(3)=VARPTR(SC%(0)):
  DEFUSR=VARPTR(US%(0)): X%=USR(US%): RETURN
330 US%=1: GOTO350:    ' Disk to array
340 US%=0:             ' Array to disk
350 US%(5)=UB%: US%(7)=256: DEFUSR=VARPTR(US%(0))
360 FOR N%=0TO3:
     US%(3)=VARPTR(SC%(N%*128)):
     IFUS%=1 THEN GET1,1+N%+4*UN%: X%=USR(US%)
     ELSE X%=USR(US%): PUT1,1+N%+4*UN%
370 NEXT: RETURN
400 CLS:              ' Screen Editor
410 PRINTC4$;: GOSUB230
420 ON INSTR(SE$,IN$) GOTO 440,450,460,470,480,490,500,510,520,5
30,540
430 IF PEEK(CU%)=255 AND PEEK(CU%+1)=63 THEN410
  ELSE IF NOT GR% THEN PRINT IN$;:   GOTO410
     ELSE PRINT CHR$(ASC(IN$)+64);: GOTO410
440 PRINTC5$;: GR%=0: RETURN
450 PRINTCHR$(30);: GOTO410
460 PRINTCHR$(27);: GOTO410
470 PRINTCHR$(26);: GOTO410
480 PRINTCHR$(25);: GOTO410
490 PRINTCHR$(24);: GOTO410
500 PRINTC5$;:
   POKE CU%,PEEK(CU%)AND63: POKE CU%+1,&H3C:
   GOTO410
510 PRINTC5$;:
   POKE CU%,PEEK(CU%)OR192: POKE CU%+1,&H3F:
   GOTO410
520 PRINTC5$;:
   POKE CU%,PEEK(CU%)OR63:
   GOTO410
530 PRINTC5$;:
   POKE CU%,PEEK(CU%)AND192:
   GOTO410
540 GR%=NOT GR%: GOTO410
550 FOR N%=0 TO 15:
     FOR X%=0 TO 31:
        GR%=SC%(N%*32+X%):
        LPRINT CHR$(GR% AND 255);:
        GR%=GR%/256:
        GR%=GR%-256*(GR%<0):
        LPRINT CHR$(GR%);:
     NEXT: LPRINT:
   NEXT: LPRINT " ": RETURN
```

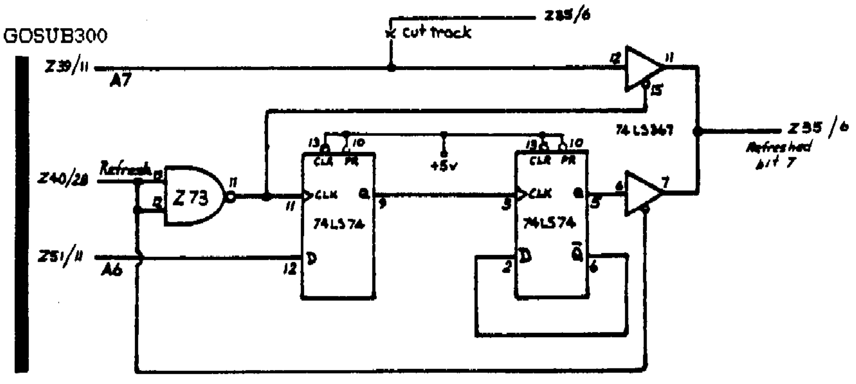
RANDOM I/O

by Dave Kennedy

[Reprinted from SYDTRUG NEWS, the newsletter of the Sydney TRS-80 Users Group (P.O. Box 297, Padstow, N.S.W. 2211, Australia).]

While Errol Rosser went jaunting off to Adelaide recently, I was attempting to fit a bank of T.I. dynamic RAMs to a Model I. The only problem I was to discover (after buying the chips - admittedly at an excellent price) was that these chips require a full 256 cycle Refresh (unlike most others which only need a 128 cycle Refresh). Now the Z-80 only provides a seven bit refresh count (128 cycle) from the Refresh register; this appeared to make my RAMs useless to me until I came up with the circuit below, which will give a full 256 cycle count, by gating bit 6 back onto the address bus as bit 7 at the end of every second 128 bit cycle.

The connections marked on the circuit apply to the 64K DRAM modification by Errol Rosser which appeared in the May 1984 issue of SYDTRUG NEWS [and was reprinted in NORTHERN BYTES Volume 5, Number 8].



CHANGE SUPERSCRIPSIT'S CURSOR CHARACTER

This information is supplied by Gil Spencer of Spit Junction, New South Wales, Australia. SuperScripsit (version 1.2.08) uses address 7E0AH for storage of the cursor character. To re-define the cursor character, the bytes to change are 9016H in SCR17/CTL and 8F25H in SCR16/CTL. Change these bytes from 8FH to whichever character you wish to use as the cursor character. If you are using SUPERZAP, the locations to change are as follows:

SCR16/CTL, FRS 4, BYTE 0A
SCR17/CTL, FRS 2, BYTE F3

Also, if you don't like using the delta symbol as a substitute for two spaces, the patches to eliminate this particular "feature" can be found in the October, 1984 "Reader Exchange" column in <u>80-Micro</u> (page 29, letter from James Reed).

MCI MAIL CHANGES

Regular readers of NORTHERN BYTES are aware that we have suggested that all personal computer users sign up for MCI Mail because it was such a good deal - no charges unless you actually sent mail. Unfortunately, the management of MCI Mail has apparently decided that they need to increase their revenues, so starting January 1, 1985, the free ride is over. The HELP NEWS file of November 19, 1984 broke the bad news. It stated:

> MCI Mail is announcing new prices and discounts that will become effective on January 1, 1985. This new pricing structure is designed to encourage you to take full advantage of MCI Mail in your business communications.
>
> All users will be billed an annual "mailbox" fee of $18. Since you are already a registered, you will not be billed until your anniversary date. New subscribers will be billed upon registration, but will be credited for the entire $18 against their first two full months of usage.
>
> All users must then choose one of the following subscription packages:
>
> INDIVIDUAL
>
> For your $18 annual fee, you enjoy the full line of services now available to a Basic user and the discount prices for Broadcast messages (see below).
>
> EXECUTIVE 50
>
> If you expect to bill at least $50/month, you may want to select this package and take advantage of the 5% discount on MCI Mail. Also included are all the features of Advanced service -- FREE! ($120 value).
>
> EXECUTIVE 250
>
> If you expect to bill at least $250/month, you will appreciate the "Executive 250" package. Your company will qualify for a 10% discount on MCI Mail and Free Advanced service for ALL company users (at their individual option).

In addition to the yearly registration fee, some changes in the pricing of MCI Mail services will occur on January 1, 1985. A new category of MCI Mail "Instant" letter will be introduced. This is the "note" of fewer than 500 characters, which can be sent for only 45 cents. On the other hand, the price of Overnight Letters and Four Hour Letters will increase to $8 and $30, respectively, but up to six pages may now be sent for the initial charge. Volume discounts will be available for "broadcast" messages sent to 100 or more recipients.

The biggest blow of all to many users - and the one that will probably cause many small users to drop MCI Mail - is a new connect time charge of 15 cents per minute when MCI Mail is accessed via the (800) WATS number. Keep in mind that at present MCI Mail is accessible through a local number in the following cities only: Atlanta, GA; Baltimore, MD; Boston, MA; Buffalo, NY; Chicago, IL; Cincinnati, OH; Clearwater, FL; Cleveland, OH; Columbus, OH; Dallas, TX; Denver, CO; Detroit, MI; Fort Worth, TX; Hackensack, NJ; Hartford, CT; Honolulu, HI; Houston, TX; Indianapolis, IN; Jacksonville, FL; Kansas City, MO; Largo, FL; Long Island (Garden City Area), NY; Los Angeles, CA; Memphis, TN; Miami, FL; Milwaukee, WI; Minneapolis, MN; Naperville, IL; Newark, NJ; New York City, NY; Oakbrook, IL; Oakland, CA; Philadelphia, Pa; Phoenix, AZ; Pittsburgh, PA; Rochester, NY; Sacramento, CA; San Diego, CA; San Francisco, CA; San Jose, CA; Santa Ana, CA; Seattle, WA; Sherman Oaks, CA; Stamford, CT; St. Louis, MO; St. Petersburg, FL; Tampa, FL; Washington, DC; White Plains, NY. If you don't live in or near one of these cities, you'll either have to pay for the toll call to the nearest city listed, or pay the 15 cents per minute charge for the use of the WATS line. No, Sault Ste. Marie isn't anywhere near Detroit, and no, I won't be checking my MCI Mailbox as regularly as I used to.

It's worth noting that 15 cents per minute is equal to $9.00 per hour. It's possible to get regular long distance service for less than that in many parts of the country, so it might be cheaper for many users to just dial direct into the nearest "local access" city. Even AT&T's "Reach Out America" plan offers long distance service for less than $9.00 per hour, once you get past the first hour of service (which costs $10.00).

The news is not all bad, of course, because there are other changes starting January 1, and some of them really can be considered improvements to the system. For example, when you send an "Instant" letter to someone, you never know if they will read it right away, or if they will let it sit in their "electronic mailbox" for a couple of weeks (perhaps because they don't want to pay the toll charges to check their mail on a daily basis). If you want to spend $1.00 extra, an MCI "Mail Alert" Operator will call the recipient(s) to tell them a message has been delivered to their INBOX. Another often-requested feature that is finally being offered is a printed MCI Mail Directory (described as "A complete, alphabetical listing of MCI Mail users by name") that will be sent free of charge to all MCI Mail subscribers.

The most exciting change, however, is the new international capabilities of MCI Mail. As HELP NEWS states:

> Shortly after the first of the year, people in over 40 countries outside the United States will be able to access MCI Mail through their country's public packet switching network. This means instant global communications for your company! And, Overnight and Four Hour courier delivery to U.S. recipients from subscribers throughout the world, at the usual low MCI Mail rates!
>
> In addition to signing up for an MCI Mailbox, the subscriber will need an account with his local PTT (Postal, Telephone and Telegraph authority). An account which provides access to MCI International/Western Union International host 4759 will be required. This account will be billed by the PTT. The MCI Mail charges will be billed separately by MCI Mail.
>
> Early next year, MCI Mail will also introduce postal and overnight courier delivery throughout the world -- direct from your PC.

A couple of regular contributors to NORTHERN BYTES (Greg Small in Canada and Tony Domigan in Australia) have been participating in the test phase of the global service. I find it kind of exciting to be able to communicate with someone on the other side of the globe on an almost immediate basis (for example, it used to take a week for an airmail letter sent by me to reach Tony, and another week to get his reply. Now we can communicate back and forth several times in the same day, if necessary, for less than the cost of an international telephone call!).

But how does an overseas user get onto MCI Mail, and how much does it cost? Let's use Tony's experience as an example. He first had to contact Mr. William Byrne at MCI Mail [Mail address: 2000 M. Street N.W., Suite 300, Washington, D.C. 20036. Telephone: (202) 463-3281. Telex: 6502194251, answerback 6502194251 MCI] to get a test account set up with MCI Mail. Then he had to get an account with a data network in Australia. As Tony relates, "I applied to OTC in Australia for Midas membership [Midas is a public data network in Australia, in some respects similar to Telenet or Tymnet in the U.S.A.]. I had to specify who I wanted to transmit to, and provide evidence of my credit history. With regard to the information on the distant host, they were really only concerned with whether it was in North America or Europe. By changing the prefix of my login code I can change my destination from North America to Europe. OTC passed my application last month and promptly lost it-or some such excuse-causing a delay in my MCI connection."

Tony further states that the Midas network charges him $12 per hour of connect time, and 60 cents per kilocharacter (1,000 characters transmitted or received) with a minimum of one minute (20 cents) and one kilocharacter (60 cents) per connection (these rates are in Australian dollars, of course, and in mid-December one Australian dollar was equivalent to about 84 cents U.S.). In addition, when he is out of the test phase, Tony will have to pay the normal charge to send MCI Mail ($1.00 per "instant" letter). He states, "The rate is quite reasonable. The only time I come near the minimum charge is when I connect around 3:00 A.M. to 5:30 A.M EST (North America) - I connect to the Host but cannot get further. I assume MCI must shutdown for maintenance during that period(?)" (I have also had the experience of calling during the wee hours of the morning and getting no answer, especially on weekends, so I suspect that Tony's assumption is correct).

Tony's difficulties in getting signed up with Midas hopefully are not typical of what future users will encounter. I would hope that any remaining bugs in the signup procedure have been ironed out by now.

Greg Small in Toronto has been using a two-step procedure to get onto MCI Mail - he first uses the Bell Datapac network to access the Tymnet network in the U.S., and then accesses MCI Mail through Tymnet. One problem he encountered was that the XON and XOFF flow control codes sent by MCI Mail never seemed to reach him, so if he attempted to upload text to MCI Mail, portions of the text file would be lost. Neither Bell Datapac nor MCI Mail could figure out what was wrong. That left one possibility, and a call to Tymnet confirmed that something there had not been set up properly, hence the codes were not being passed through (hopefully, they are now). In any case, when the system is fully operational, Greg anticipates being able to directly access MCI Mail from Bell Datapac, without using Tymnet as an intermediate link. I suspect that the problem in Canada is that unlike the Australian Midas network, Bell Datapac will not directly bill the end user for connect time used - rather, they bill the host computer, and the host must in turn bill the Datapac charges back to the end user. My guess is that MCI Mail's software is not yet set up to handle this kind of billing of network charges, so they are using Tymnet as an intermediary during the test phase, because Tymnet is already connected with Bell Datapac (that's one way that Canadians access Compuserve and similar services in the U.S.) and already has the required billing capability.

One problem that has yet to be resolved at this writing is that both Greg and Tony seem to have trouble with backspace (Control-H) characters. They tend to get entered into the text, rather than just deleting the previously-entered character. U.S. users do not have this problem, so it appears there may still be a few bugs in the interfacing of the networks.

By the time you read this, both Greg and Tony should have their own personal accounts. But, if you can't locate them on the system (under user names GSMALL and TDOMIGAN, of course) and want to try sending a message to their "test account" addresses, the user ID numbers are 248-6778 for Greg, and 248-6780 for Tony. I emphasize again that by the time you read this, Greg and Tony will probably have their own accounts and may not even be able to access the test accounts anymore.

The bottom line of all this is that although MCI Mail is introducing some exciting new features, they also seem to be concentrating on wooing the high-volume business user, at the expense of the individual user. I find the tactic of offering basically "free access" to the system for the past year to get users "hooked" a bit offensive. I do not object to the $18 annual fee (although it will probably inhibit some potential users from signing up), but I do strenuously object to the 15 cents per minute charge for use of the WATS lines. It seems as though a communications giant like MCI should be able to offer local access to MCI Mail in at least all of the cities where MCI long distance phone service is available! It would be nice if MCI Mail could make some concession for individual users that don't choose to live in the big cities, and perhaps offer free WATS line access during late-night hours and on weekends.

I apologize to those of you that signed up for MCI Mail at my urging and now feel that the new rates are a bad deal, but of course I had no way of knowing that they were coming. If you feel the same way I do about the new charges (especially the WATS line access charge), I suggest you drop an electronic letter to MCI Mail and let them know how you feel (and whether the new charges have caused you to curtail your use of the system in any way). On the other hand, I will probably stay with MCI Mail for at least another year, to facilitate receiving articles and other communications from those who find it convenient to use that medium.

POST SCRIPT: I dropped a line to Alan Abrahamson (editor of the Voice of the '80) recently and happened to mention my displeasure with the new MCI Mail rates. I found his reply interesting: "After your P/R and a tweak from Humble Dave in Tennessee, I signed up. Of course this was a NO CHARGE ballgame at that time. Now that they have us, I guess they can rub it in. I have no local access number [in Fairfield County, Connecticut] and NEW YORK would cost me more than the 15 cents per minute that the 800 line will cost in January. I really don't know what I will do, probably just pay the bills like a sucker.

"You mentioned EasyLink from Western Union. I had an interesting experience there. I was offered a 60 day free trial. I said OK. It took 3-4 weeks to get the manuals. The day after I got the manuals and spent a while reading them, I tried to access the database. I could not. So I called 800 WU-CARES number for assistance. I was told that I had been discharged from the service. This was day 30 of service and I had never accessed the system before. I wrote a scorching letter to the service representative telling her that if Western Union wanted to do business with the public, that they had best get their act together. I never received even the courtesy of a form letter reply. I think that they had no answer for this foulup." (End of quote.)

So, I guess I'm still looking fo the "ideal" electronic mail service for personal computer users, especially for those of us that happen to live outside the major metropolitan areas! I guess all of this goes to prove the old saying, "There Ain't No Such Thing As A Free Lunch" ...

* * * * * * * * * * *

ANOTHER FREE SERVICE BITES THE DUST

We mentioned in a previous issue of NORTHERN BYTES a "FREE" electronic information service operated by ONLINE TELECOMPUTING, INC. and accessible by an 800 number from anywhere in the continental U.S. Well, by the time you read this (or shortly thereafter) they will be requiring an annual registration fee of $20 per user. The WATS line access will be maintained, but since ONLINE is basically a "product information service" (translation: online advertising) you might decide you have better ways to spend $20.

ONLINE TELECOMMUNICATIONS also plans to start a new service called "ONLINE PLUS" in the near future, which will offer such things as nationwide electronic mail, special interest group bulletin boards, and public domain program downloads. Access will be through TELENET and WATS lines, but there will be a charge of $6.25 per hour over TELENET and $13.75 per hour over WATS (good grief! That's even worse than the $9.00 per hour that MCI Mail charges!). Oh, and there's a $49.95 registration and setup fee, which includes the first hour of usage (if you're still interested, call (404) 998-7776 for more information).

I guess if you live in a rural area, it doesn't pay to own a MODEM unless you're independently wealthy!

MEMDISK AND MODEL 4 SUPERSCRIPSIT by Tom Price
(Compuserve 70015,232)

[Reprinted from the Marin County TRS-80 Users Group Newsletter.]

A number of Model 4/4P owners have asked for the configuration which will allow them to run Model 4 SuperScripsit most efficiently from MEMDISK. Herewith is the setup I use on my 4P:

(1) Make up a 'template' data diskette containing the following files:

System files 0, 1, 2, 3, 4, 6, 7, 8, 10, and 12
SuperScripsit files SCR16/CTL, SCR17/CTL, SCR32/CTL and SCR33/CTL

The above SuperScripsit overlays are the only ones that are accessed with any regularity during normal document preparation and editing.

(2) After activating the MEMDISK as Drive :2, BACKUP your template diskette to the MEMDISK.

(3) Make the MEMDISK the system disk and write protect it with:

```
SYSTEM (SYSTEM=2)
SYSTEM (DRIVE=0,WP)
```

Now you can put a data diskette containing all of the SuperScripsit files, less the ones on the MEMDISK, into one floppy drive and use the other floppy drive for documents or for the dictionary. This setup greatly enhances the speed of SuperScripsit, since most of its required accesses are coming from the MEMDISK.

EXPANSION UNIT FOR PMC-80/SYSTEM-80/VIDEO GENIE USERS

I hope to have more information on this next issue, but for now I'll just mention that an expansion unit consisting of a single-density disk controller and printer interface is available from Mr. Errol Rosser, 6 Sixth Avenue, Condell Park, Sydney, N.S.W. 2200, AUSTRALIA. The kit (actually a complete board that is installed into the keyboard unit) sells for $A150 (150 Australian dollars) and Errol estimates that shipping to outside of Australia would probably cost an additional $A10. The unit can be installed in 2-3 hours by a rank novice, and any of the commonly available double density adapters can be used with the disk controller if double density operation is desired. Errol hopes to offer a similar unit for Model I TRS-80 users in the near future, this will fit into a disk drive case (along with the drives) and will sell for about $A190. Interested readers may contact Mr. Rosser at the above address, or telephone him at (02) 709-7646 (this is his home number. U.S.A. & Canada dial 011+61+2+709-7646, and remember the time difference - when it's noon in the U.S., it's the middle of the night in Australia!)

MAGIC MATH PLUS

by Dr. Michael W. Ecker

In Volume 5 Number 7 of Northern Bytes a method of drawing lines on a TRS-80 was presented. Essentially what was done there was an attempt at understanding the concept of the slope of a line. In effect, the author re-created the concept - re-inventing the wheel, you might say.

The line program here does the following: prompts for input of the coordinates of the two points; checks to ensure that both points have permissible coordinates (so that a function command error will not result from attempting to illuminate a pixel not on the screen); computes the slope of the line determined by the two points; depending on whether the line is vertical (slope undefined case) or not (the usual situation), finds the equation of the line; and lastly lights up pixels corresponding to points which satisfy the equation.

There is one other conditional test made. What it does is check for lines for which the line rises or falls "faster" than it moves left or right. If that happens, the line could have gaps (thanks Jack!), so it becomes desirable to use the slope to modify the pixel-by-pixel advancement in steps of no more than one in the y-direction, instead of in the x-direction as is the case when the line is not so steeply sloped.

Mathematically, the equation of the line obtained uses the point - slope formula, the slope being m (M in the program) and the point being chosen arbitrarily as the first one, (X1,Y1). Do not use parentheses when you input the coordinates, but do use a comma.

I will answer reader questions relating to the program or other items of a mathematical / recreational nature, provided you enclose a self-addressed stamped envelope. Write me directly:

Dr. Michael W. Ecker
129 Carol Drive
Clarks Summit, Pennsylvania 18411
(717) 586-2784

```
10 CLS
20 PRINT "THIS PROGRAM WILL DRAW A LINE BETWEEN ANY
TWO POINTS"
30 PRINT "THAT YOU INPUT. THE FIRST (X) COORDINATE MUST
BE AN"
40 PRINT "INTEGER BETWEEN 0 AND 127, INCLUSIVE; THE
SECOND ONE"
50 PRINT "(Y) MUST BE BETWEEN 0 AND 47."
60 PRINT:PRINT:PRINT:INPUT "ENTER TO CONTINUE";QQ$:CLS
70 FOR J=1 TO 2
80 PRINT "X COORDINATE, Y COORDINATE FOR POINT
NUMEER";J
90 INPUT X(J),Y(J)
100 IF X(J)<0 OR X(J)>127 OR Y(J)<0 OR Y(J)>47 THEN 80
110 NEXT J
120 CLS: IF X(1)=X(2) THEN 300: REM VERTICAL LINE CHECK
130 M=(Y(2)-Y(1))/(X(2)-X(1)): REM COMPUTE SLOPE (STEEPNESS)
140 IF ABS(M)>1 THEN 400
150 FOR X=X(1) TO X(2) STEP SGN(X(2)-X(1))
160 Y=M*(X-X(1))+Y(1)
170 SET(X,Y)
180 NEXT X
190 FOR DL=1 TO 2000:NEXT:CLS:GOTO 70
300 FOR Y=Y(1) TO Y(2) STEP SGN(Y(2)-Y(1))
310 SET (X(1),Y)
320 NEXT Y
330 FOR DL=1 TO 2000:NEXT:CLS:GOTO 70
400 FOR X=X(1) TO X(2) STEP SGN(X(2)-X(1))/(2*ABS(M))
410 Y=M*(X-X(1))+Y(1)
420 SET(X,Y)
430 NEXT X
440 FOR DL=1 TO 2000:NEXT:CLS:GOTO 70
```

Dr. Michael W. Ecker is an Associate Professor of Mathematics and Computer Science at the University of Scranton in Scranton, Pennsylvania, as well as a contributing editor / computer magazine columnist for Popular Computing, Soft Sector and others, as well as recently for Byte. In addition to mathematics and computer recreations, he does software reviews and writes software. His company, Recreational Mathemagical Software, offers Magic Math Plus to TRS-80 Model 3,4,4P users with disk systems. It features a self-booting disk, licensed disk operating system, and four menus among the 40 programs. Offerings include Loan Amortization and Compound Interest, Super-Blackjack, The Game of N, Fibonacci Numbers, and many, many more. Cost is $37.50 ($36 plus $1.50 S&H) payable by check made out to Recreational Mathemagical Software. For a full catalog, send a self-addressed stamped envelope with your request to the above address. Upcoming issues of Northern Bytes will continue to feature / serialize some of the programs from that collection.]

RANDOM I/O

by Errol Rosser

[This article is excerpted from Mr. Rosser's column which appears in SYDTRUG NEWS, the newsletter of the Sydney TRS-80 Users Group (P.O. Box 297, Padstow, N.S.W. 2211, Australia).]

Firstly, Barrie Algood sends some alterations to the JKL overlay for NEWDOS, which was set up for an EPSON type printer with 480 dots per line in low resolution [This program originally appeared in NORTHERN BYTES Volume 5, Number 4, pages 6-8 -editor]. These modifications convert JKLSYS/SRC to print correctly on the BMC, CP, FAX, and EMTEK type printers which use 630 dots per line in low-res mode.

Modifications for JKLSYS/SRC to work on EPSON look-alikes. Printers such as the CP-80 & BMC-80 have a 7 dot wide character set built into the printer. The computer uses 6 dot wide characters and 6 dot wide graphics. The EPSON also uses 6 dot wide characters which makes it quite easy to use the computer's graphics & the printer's alphas. Now if we use 6 dot graphics coupled with 7 dot alphas, we end up with alphas & graphics running together (not very nice). The solution then is to send the printer 7 dot wide graphics which can be achieved by sending 14 dot graphics & put the printer in hi-res graphics mode. Problem solved? ...No! Still a slight problem which is "fixed" by sending 16 dot wide graphics in hi-res mode. The other mods are needed to align printing start position & differences in line feeds. Regards, Baz.

The changes are:

```
00240           LD      A,8DH
00260           LD      A,8DH
00730           LD      B,60
00740  INDENT   LD      A,0
00750           CALL    PRT
01200           LD      B,4
01700           LD      A,3
01821           LD      A,7FH
01822           CALL    PRT
01823           XOR     A
01880           LD      B,4
02090           LD      A,'A'
02270           LD      A,'L'
02290           LD      A,60
02310           LD      A,4
02360           LD      B,8
```

and delete lines 02120 to 02160 inclusive.

David Kennedy has modified the source code of JKLSYS/SRC to allow the program to be used with LDOS. The program may even be SYSGENed and therefore saved in the configuration file as long as the KI/DVR is activated before the JKL program. The complete modified source code including a full re-location table is too long to include in the Newsletter but can be downloaded from the group BBS (CLUB-80).

[Note to NORTHERN BYTES readers in Australia - the number is (02)332-2494. Readers in North America, forget it unless you happen to have a MODEM that uses the CCITT protocol (not the Bell protocol commonly used in the U.S. and Canada).]

A note on generating /SYS files for use under NEWDOS/80 version 2.0. When the /SYS files are copied with a format 6 COPY command (CBF with FMT) the DOS tries to put the files the same distance from the directory as they were on the source disc (see Apparat's ZAP 65 pages 13-31). This will only affect the copy process if you try to copy to a disc which has its directory closer to the start or end of that disc. The error that occurs is GAT OVERFLOW.

And to finish off for this month, a couple of patches to LDOS from Gary Bryce. First a patch to disable PASSWORD checking during any file operation:

```
PATCH SYS2/SYS.RS0LT0FF:0 (D02,19=18)
```

Next to prevent clearing of the screen between pages of a DIR display.

```
PATCH SYS6/SYS.RS0LT0FF:0 (D08,E1=00 00 00)
```

Finally a ZAP to NEWDOS from Geza Dujmovich,to stop clearing of the screen when going to BASIC.

```
BASIC/CMD 16,19 change 64 1C 1F 44 to 64 00 00 44
```

OVERSEAS EXPERIENCE - HARDWARE/SOFTWARE

by "Computer Nut"

(Part 2 of a series)

Continuing on from the previous installment we come to the subject of modifications which are a combination of hardware and software.

Many non-english languages have alphabets with letters which are not found in the so-called Roman alphabet. Some languages use the same letters, but with accents on some of the letters, such as accent grave, accent aigu, circumflex and cedilla. For serious work on microcomputers it is necessary to be able to enter, display and print these characters.

Unfortunately, to save a few cents on extra keys, Tandy have persisted in leaving off the extra keys which normally are found on an ASCII keyboard. The spaces for them are left in the key strobe containing the letters X, Y and Z, at least on the Model I/III and 4. These keys contain some of the accent characters required for some languages.

The TRS-80 Model 4 is produced in Europe, by Matra in France. To cater for the needs of some European languages, these computers may be fitted with special keyboards, with up to 5 extra keys. Two of them are positioned on either side of the space bar, and the three others are squeezed in at the right hand end of the alphabetic keys. Some of the present keys are moved around to cater for special national keyboard layouts.

This is the part that can be done in hardware. But to make use of the extra keys, some software modifications are also necessary.

When operated in Model III mode, the keyboard input software is taken from the ROMs installed in the computer. Allowing accents on alphabetic keys requires some extra software, and unfortunately also some RAM for storing various information. The ROMs installed in the French manufactured Model 4's are obviously different from the American version, and to make room for the extra code some of the routines have been moved around. This becomes obvious when trying to run some of the operating systems available for the Model III.

The changes are made in ROM 3 where the keyboard input code and tables reside, and apparently also in ROM 1. Some of the printer ready test code has been moved (or perhaps removed) and the standard version of LDOS relies on this code. TRSDOS seems to work without problems, as does most of Newdos/80.

Programs which perform their own keyboard input will give unexpected results, because some of the characters have been moved around. The legends on the shifted numeric keys do not correspond to those on the American keyboards, and some of the other keys have been moved around and given new legends.

So how does the code generate letters with accents? First you press the accent you want, then the letter you want accented. The accent is stored in memory, and when the letter key is pressed it looks up a code in a table and gives a new code with the most significant bit set. The character generator ROM has been modified so that these new characters are displayed correctly on the screen. This is the method employed for the new keyboard input routines in the ROM, and probably also in the European versions of TRSDOS 6. However, many programs will reject these accented letters and will not accept them as alphabetic characters.

How can this dilemma be solved?

There seems to be no obvious simple solution. It is probably not feasible to reeducate American programmers so that they realise the problems involved. And if this is not possible for the relatively simple problems of the European languages, then the problems with special alphabets such as Greek, Arabic, Japanese, and others must be almost insurmountable.

Another possible solution would be to buy only European computers, but the large British computer industry seems to have the same limited vision in its computer development. Also the software industry in Europe at the moment does not seem capable of catering for all software needs, or at least there is still a strong demand for American software such as operating systems and the integrated software packages. Development of software for sale requires either a large user base, or few users willing to pay large sums for the right software.

The present solution seems to be the development of specialised software on a local basis, and the modification of American software packages to cater for the general software market. Conversion can be expensive, and the market relatively small, so software prices are typically 2 to 3 times as high as in the US.

My suggestion to software developers in the US is: If you are writing software which you expect to be able to sell internationally, then if possible allow for entry of all, or almost all, the characters in the expanded character set. The ASCII control characters from 0 to 1FH are still used as control characters, but all values from 20H to 0FFH are potential input characters for data values.

"Computer Nut"

TRS-80 GRAPHICS ON THE EPSON FX-80 PRINTER

by Nate Salsbury

I got started in computers via an interest in weaving which I still have (but have had to abandon because of all the F U N one has in debugging programs!)

In the craft of weaving, there is a useful technique called "drawdowns" which allows the weaver to make a graphical picture of his weaving project without threading a loom. This involves filling in lots of little boxes on graph paper. Two TRS-80 pixels side by side make a perfect square and I bought my first Model I for the fun of trying to write a program to make drawdowns on a computer rather than by hand. (I was successful and now sell the program to other weavers - have sold over 100 to date.)

Why am I telling you this? In order to get hardcopy, I had to buy an Epson MX 80 - the only machine at the time that supported the TRS-80 graphic characters. I wrote my program to interface to that machine.

All went well until the FX 80 appeared on the scene and Epson left out the TRS-80 characters in its standard ROM set! They DID provide an option to 'create' your own characters however, so I set about writing a program to "download" the TRS-80 graphics into the FX 80 (so that I could sell my program to owners of THAT machine.)

Once you wade through the syntax problems of the poor English translation of the manual, it isn't too tough to do, as you'll see. To spare one or more of your readers the agony of doing the same study, I'm offering 'FX80' to your journal and, if you guys like it, you're welcome to put it on a TAS Public Domain disk.

Best regards,
Nate (known to several of his children as The Great) Salsbury

```
1 '    FX-80 DRIVER TO OBTAIN TRS-80 GRAPHIC CHARACTERS
2 '    Prepared by:  Salsbury Associates Inc.
                     610 Madam Moore's Lane
                     New Bern, NC  28560
3 '
4 '  Given to Northern Bytes for publication as a Public
   Domain program  October 3, 1984
5 '
6
'**************************************************************
7 'THE FX 80 DIP switch 1-4 must be OFF for this program to work
8
'**************************************************************
9 '
10 CLS: DEFINT A-Z: GOTO 150
20 '
30 '   Subroutine to define special download characters
35 '
40 FOR C = 1 TO 3
50 LPRINT CHR$(CH);CHR$(0);
60 NEXT C: RETURN
70 '
80 '   Subroutine to select special download character
90 '
100 LPRINT CHR$(27); "&"; CHR$(0);  'Select download character
110 LPRINT CHR$(DC);CHR$(DC);       'DC = ASCII Character
120 LPRINT CHR$(139);: RETURN       'CHR$(139) is the attribute
149 '
150 CLS: PRINT @ 512,"": INPUT "Turn on the printer and then hit
<ENTER> ....";Z
160 CLS: PRINT@ 517, "Now we're setting up the printer for TRS-80
graphics"
170 '
180 '   From FX-80 instruction manual page 3-43
190 '   Copy ROM to download CG
200 '
210 LPRINT CHR$(27); ":"; CHR$(0); CHR$(0); CHR$(0);
220 '
1280 '  Set up ASCII 160 for graphic 128
1281 DC = 160: GOSUB 100
1282 CH = 0: GOSUB 40   : GOSUB 40
```

```
1283 '
1290 ' Set up ASCII 161 for graphic 129
1291 DC = 161: GOSUB 100
1292 CH = 224: GOSUB 40   : CH = 0: GOSUB 40
1293 '
1300 ' Set up ASCII 162 for graphic 130
1301 DC = 162: GOSUB 100
1302 CH = 0: GOSUB40   : CH = 224: GOSUB 40
1303 '
1310 ' Set up ASCII 163 for graphic 131
1311 DC = 163: GOSUB 100
1312 CH = 224: GOSUB 40   : GOSUB 40
1313 '
1320 ' Set up ASCII 164 for graphic 132
1321 DC = 164: GOSUB 100
1322 CH = 24: GOSUB 40   : CH = 0: GOSUB 40
1323 '
1330 ' Set up ASCII 165 for graphic 133
1331 DC = 165: GOSUB 100
1332 CH = 248: GOSUB 40   : CH = 0: GOSUB 40
1333 '
1340 ' Set up ASCII 166 for graphic 134
1341 DC = 166: GOSUB 100
1342 CH = 24: GOSUB 40   : CH = 224: GOSUB 40
1343 '
1350 ' Set up ASCII 167 for graphic 135
1351 DC = 167: GOSUB 100
1352 CH = 248: GOSUB 40   : CH = 224: GOSUB 40
1353 '
1360 ' Set up ASCII 168 for graphic 136
1361 DC = 168: GOSUB 100
1362 CH = 0: GOSUB 40   : CH = 24: GOSUB 40
1363 '
1370 ' Set up ASCII 169 for graphic 137
1371 DC = 169: GOSUB 100
1372 CH = 224: GOSUB 40   : CH = 24: GOSUB 40
1373 '
1380 ' Set up ASCII 170 for graphic 138
1381 DC = 170: GOSUB 100
1382 CH = 0: GOSUB 40   : CH = 248: GOSUB 40
1383 '
1390 ' Set up ASCII 171 for graphic 139
1391 DC = 171: GOSUB 100
1392 CH = 224: GOSUB 40   : CH = 248: GOSUB 40
1393 '
1400 ' Set up ASCII 172 for graphic 140
1401 DC = 172: GOSUB 100
1402 CH = 24: GOSUB 40   : GOSUB 40
1403 '
1410 ' Set up ASCII 173 for graphic 141
1411 DC = 173: GOSUB 100
1412 CH = 248: GOSUB 40   : CH = 24: GOSUB 40
1413 '
1420 ' Set up ASCII 174 for graphic 142
1421 DC = 174: GOSUB 100
1422 CH = 24: GOSUB 40   : CH = 248: GOSUB 40
1423 '
1430 ' Set up ASCII 175 for graphic 143
1431 DC = 175: GOSUB 100
1432 CH = 248: GOSUB 40   : GOSUB 40
1433 '
1440 ' Set up ASCII 176 for graphic 144
1441 DC = 176: GOSUB 100
1442 CH =7: GOSUB 40   : CH = 0: GOSUB 40
1443 '
1450 ' Set up ASCII 177 for graphic 145
1451 DC = 177: GOSUB 100
1452 CH = 231: GOSUB 40   : CH = 0: GOSUB 40
1453 '
1460 ' Set up ASCII 178 for graphic 146
1461 DC = 178: GOSUB 100
1462 CH = 7: GOSUB 40   : CH = 224: GOSUB 40
1463 '
1470 ' Set up ASCII 179 for graphic 147
1471 DC = 179: GOSUB 100
1472 CH =231: GOSUB 40   : CH = 224: GOSUB 40
1473 '
1480 ' Set up ASCII 180 for graphic 148
1481 DC = 180: GOSUB 100
1482 CH = 31: GOSUB 40   : CH = 0: GOSUB 40
1483 '
1490 ' Set up ASCII 181 for graphic 149
1491 DC = 181: GOSUB 100
1492 CH = 255: GOSUB 40   : CH = 0: GOSUB 40
1493 '
1500 ' Set up ASCII 182 for graphic 150
1501 DC = 182: GOSUB 100
1502 CH = 31: GOSUB 40   : CH = 224: GOSUB 40
1503 '
1510 ' Set up ASCII 183 for graphic 151
1511 DC = 183: GOSUB 100
1512 CH = 255: GOSUB 40   : CH = 224: GOSUB 40
1513 '
1520 ' Set up ASCII 184 for graphic 152
1521 DC = 184: GOSUB 100
1522 CH = 7: GOSUB 40   : CH = 24: GOSUB 40
1523 '
1530 ' Set up ASCII 185 for graphic 153
1531 DC = 185: GOSUB 100
1532 CH = 231: GOSUB 40   : CH = 24: GOSUB 40
1533 '
1540 ' Set up ASCII 186 for graphic 154
1541 DC = 186: GOSUB 100
1542 CH = 7: GOSUB 40   : CH = 248: GOSUB 40
1543 '
1550 ' Set up ASCII 187 for graphic 155
1551 DC = 187: GOSUB 100
1552 CH = 231: GOSUB 40   : CH = 248: GOSUB 40
1553 '
1560 ' Set up ASCII 188 for graphic 156
1561 DC =188: GOSUB 100
1562 CH = 31: GOSUB 40   : CH = 24: GOSUB 40
1563 '
1570 ' Set up ASCII 189 for graphic 157
1571 DC = 189: GOSUB 100
1572 CH = 255: GOSUB 40   : CH = 24: GOSUB 40
1573 '
1580 ' Set up ASCII 190 for graphic 158
1581 DC = 190: GOSUB 100
1582 CH = 31: GOSUB 40   : CH = 248: GOSUB 40
1583 '
1590 ' Set up ASCII 191 for graphic 159
1591 DC = 191: GOSUB 100
1592 CH = 255: GOSUB 40   : CH = 248: GOSUB 40
1593 '
1600 ' Set up ASCII 192 for graphic 160
1601 DC = 192: GOSUB 100
1602 CH = 0: GOSUB 40   : CH = 7: GOSUB 40
1603 '
1610 ' Set up ASCII 193 for graphic 161
1611 DC = 193: GOSUB 100
1612 CH = 224: GOSUB 40   : CH = 7: GOSUB 40
1613 '
1620 ' Set up ASCII 194 for graphic 162
1621 DC = 194: GOSUB 100
1622 CH = 0: GOSUB 40   : CH = 231: GOSUB 40
1623 '
1630 '  Set up ASCII 195 for graphic 163
1631 DC = 195: GOSUB 100
1632 CH = 224: GOSUB 40   : CH = 231: GOSUB 40
1633 '
1640 ' Set up ASCII 196 for graphic 164
1641 DC = 196: GOSUB 100
1642 CH = 24: GOSUB 40   : CH = 7: GOSUB 40
1643 '
1650 ' Set up ASCII 197 for graphic 165
1651 DC = 197: GOSUB 100
1652 CH = 248:GOSUB 40   : CH = 7: GOSUB 40
1653 '
1660 ' Set up ASCII 198 for graphic 166
1661 DC = 198: GOSUB 100
1662 CH = 24: GOSUB 40   : CH = 231: GOSUB 40
1663 '
1670 ' Set up ASCII 199 for graphic 167
1671 DC = 199: GOSUB 100
1672 CH = 248: GOSUB 40   : CH = 231: GOSUB 40
1673 '
1680 ' Set up ASCII 200 for graphic 168
1681 DC = 200: GOSUB 100
1682 CH = 0: GOSUB 40   : CH = 31: GOSUB 40
1683 '
1690 ' Set up ASCII 201 for graphic 169
1691 DC = 201: GOSUB 100
1692 CH = 224: GOSUB 40   : CH = 31: GOSUB 40
1693 '
1700 ' Set up ASCII 202 for graphic 170
1701 DC = 202: GOSUB 100
1702 CH = 0: GOSUB 40   : CH = 255: GOSUB 40
1703 '
1710 ' Set up ASCII 203 for graphic 171
1711 DC = 203: GOSUB 100
1712 CH = 224: GOSUB 40   : CH = 255: GOSUB 40
1713 '
1720 ' Set up Ascii 204 for graphic 172
1721 DC = 204: GOSUB 100
1722 CH = 24: GOSUB 40   : CH = 31: GOSUB 40
1723 '
1730 ' Set up ASCII 205 for graphic 173
1731 DC = 205: GOSUB 100
1732 CH = 248: GOSUB 40   : CHA = 31: GOSUB 40
1733 '
1740 ' Set up ASCII 206 for graphic 174
1741 DC = 206: GOSUB 100
1742 CH = 24: GOSUB 40   : CH = 255: GOSUB 40
1743 '
1750 ' Set up ASCII 207 for graphic 175
1751 DC = 207: GOSUB 100
1752 CH = 248: GOSUB 40   : CH = 255: GOSUB 40
1753 '
1760 ' Set up ASCII 208 for graphic 176
1761 DC = 208: GOSUB 100
1762 CH = 7: GOSUB 40   : GOSUB 40
1763 '
1770 ' Set up ASCII 209 for graphic 177
1771 DC = 209: GOSUB 100
1772 CH = 231: GOSUB 40   : CH = 7: GOSUB 40
1773 '
1780 ' Set up ASCII 210 for graphic 178
1781 DC = 210: GOSUB 100
1782 CH = 7: GOSUB 40   : CH = 231: GOSUB 40
1783 '
1790 ' Set up ASCII 211 for graphic 179
1791 DC = 211: GOSUB 100
1792 CH = 231: GOSUB 40   : GOSUB 40
1793 '
1800 ' Set up ASCII 212 for graphic 180
1801 DC = 212: GOSUB 100
1802 CH = 31: GOSUB 40   : CH = 7: GOSUB 40
1803 '
1810 ' Set up ASCII 213 for graphic 181
1811 DC = 213: GOSUB 100
1812 CH = 255: GOSUB 40   : CH = 7: GOSUB 40
1813 '
1820 ' Set up ASCII 214 for graphic 182
1821 DC = 214: GOSUB 100
1822 CH = 31: GOSUB 40   : CH = 231: GOSUB 40
1823 '
1830 ' Set up ASCII 215 for graphic 183
1831 DC = 215: GOSUB 100
1832 CH = 255: GOSUB 40   : CH = 231: GOSUB 40
1833 '
1840 ' Set up ASCII 216 for graphic 184
1841 DC = 216: GOSUB 100
1842 CH = 7: GOSUB 40   : CH = 31: GOSUB 40
1843 '
1850 ' Set up ASCII 217 for graphic 185
1851 DC = 217: GOSUB 100
1852 CH = 231: GOSUB 40   : CH = 31: GOSUB 40
1853 '
1860 ' Set up ASCII 218 for graphic 186
1861 DC = 218: GOSUB 100
1862 CH = 7: GOSUB 40   : CH = 255: GOSUB 40
1863 '
1870 ' Set up ASCII 219 for graphic 187
1871 DC = 219: GOSUB 100
1872 CH = 231: GOSUB 40   : CH = 255: GOSUB 40
1873 '
1880 ' Set up ASCII 220 for graphic 188
1881 DC = 220: GOSUB 100
1882 CH = 31: GOSUB 40   : GOSUB 40
1883 '
1890 ' Set up ASCII 221 for graphic 189
1891 DC = 221: GOSUB 100
```

```
1892 CH = 255: GOSUB 40   : CH = 31: GOSUB 40
1893 '
1900 '  Set up ASCII 222 for graphic 190
1901 DC = 222: GOSUB 100
1902 CH = 31: GOSUB 40   : CH = 255: GOSUB 40
1903 '
1910   Set up ASCII 223 for graphic 191
1911 DC = 223: GOSUB 100
1912 CH = 255: GOSUB 40   : GOSUB 40
1913 '
2000 '  Select download character set
2010 '
2020 LPRINT  CHR$(27); "%"; CHR$(1); CHR$(0);
2030 '
2040 '  Set up for 8 dot vertical spacing
2050 '
2060 LPRINT CHR$(27); "A"; CHR$(8);: LPRINT CHR$(15)
2070 CLS: PRINT @ 325, "Your FX-80 is now ready to print TRS-80
graphics.
"
2080 PRINT: PRINT "NOTE: To return your printer to 'normal', turn
it OFF
    and then ON after finishing your graphic printing.
"
49999 END
```

BASIC PROGRAM FUNCTIONS YOU CAN USE

The following functions were adapted from an article by Jehangir Gazdar, which originally appeared in the Capitol PC Monitor and was later reprinted in the ABC Micro Newsletter. Although the functions were originally written for use with the IBM PC, they seem to work on the TRS-80 as well.

The first group of functions are used when you are printing a sentence which includes a numeric variable, and you want to make sure you have the correct plural ending. For example, your program might say something like "X new record(s) have been added" where X is a number from zero through ???. Surely one would think that the computer should be able to tell whether to print "record" or "records", depending on the number of records added (one, or other than one). Well, if line 130 in the sample program is included near the start of your program, you need only rewrite your PRINT statement in this form:

PRINT X "new record" FNA$(X) " have been added"

In the above example, FNA$ will return a null string if X=1, or the one-character string "s" if X<>1. Note that the variable name in the above line does not have to be "X", it could be any valid variable name.

The function in line 150 (FNB$) works exactly the same way except that a null string will be returned if the variable used equals either 0 or 1. The function in line 170 (FNC$) is used when you need a plural "es" ending rather than a simple "s".

But what if you also need to include the words "is" or "are" in your sentence? That's where the functions in lines 220 and 240 come in. The function in line 220 (FND$) returns the word "is" for a value of one, while the function in line 240 (FNE$) returns "is" for a value of either zero or one. As an example, the above sample line could be rewritten this way:

PRINT X "new record" FNA$(X) " " FND$(X) " to be added"

Note that in this example, a space character must be inserted between the two functions. If X=1, the above line would be printed as " 1 new record is to be added" while if X=2, the line would be printed as " 2 new records are to be added".

You may have noticed that a space character is printed before and after the numeric variable in the above example. The space prior to the numeric variable would have been occupied by a minus sign ("-") if the value of the variable had been negative. In many cases, you may want to extract just the number, without any leading or trailing spaces, and the function in line 290 (FNF$) performs this function. The functions in lines 310 and 330 do exactly the same thing for all non-zero values, but if the variable contains zero, either the word "no" (if function FNG$ in line 310 is used) or the word "zero" (if function FNH$ in line 330 is used) will be substituted. If our sample line were rewritten as follows:

PRINT FNF$(X) " new record" FNA$(X) " " FND$(X) " to be added"

then the leading space prior to a positive value of X would be eliminated, but the minus sign would not be chopped off if X<0. In this example, a value of zero would be printed as "0", but if FNG$ or FNH$ were used instead of FNF$ on the above line, a value of zero would be printed as "no" or "zero" respectively.

Of course, it's possible to achieve the same effects by using other BASIC programming techniques, but most of them will involve the use of multiple IF-THEN statements and may occupy several program lines. The above functions are much easier to implement in many cases.

Please note that this article presumes that the reader has at least some understanding of user-defined functions in BASIC. If this is not the case, you may wish to blow the dust off your BASIC programming manual (assuming you can find it) and read up on the subject. The main thing to remember is that the variable "N" used in the functions below is a "dummy" variable, you may use any valid variable name when you invoke the function (just as I used "X" in the above examples). This is what makes user-defined functions so valuable - you can use the same function over and over without having to first reassign variables (as you might have to do if you used a subroutine to accomplish the same purpose). I hope these functions are of some use to you!

```
100 'PLURAL ENDINGS FUNCTIONS
    110 '
    120 'Return null if N=1, else "s"
    130 DEFFNA$(N)=MID$("s",1-(N=1),1)
    140 'Return null if N=0 or N=1, else "s"
    150 DEFFNB$(N)=MID$("s",1-(N=0 OR N=1),1)
    160 'Return null if N=1, else "es"
    170 DEFFNC$(N)=MID$("es",1-2*(N=1),2)
    180 '
    190 'SELECT IS/ARE
    200 '
    210 'Return "is" if N=1, else "are"
    220 DEFFND$(N)=MID$("areis",1+(N=1)*-3,3)
    230 'Return "is" if N=0 or N=1, else "are"
    240 DEFFNE$(N)=MID$("areis",1+(N=0 OR N=1)*-3,3)
    250 '
    260 'NUMERIC STRING HANDLING
    270 '
    280 'Return digit string with no leading/trailing spaces
    290 DEFFNF$(N)=MID$(STR$(N),2+(N<0))
    300 'Same as above but return "no" if N=0
    310 DEFFNG$(N)=MID$(STR$(N)+"no",2+(N<0)-((LEN(STR$(N))-
    1)* (N=0)),LEN(STR$(N))+(N>0))
    320 'Same as above but return "zero" if N=0
    330 DEFFNH$(N)=MID$(STR$(N)+"zero",2+(N<0)-
    ((LEN(STR$(N))-1)* (N=0)),LEN(STR$(N))+(N>0)-2*(N=0))
```

TIP FOR VIDEO-GENIE [ALSO PMC-80, SYSTEM-80]

by H. Delahaye

Translated from Dutch to English by Paul Fransen

[Reprinted from REMARKS, the publication of the TRS-80 Gebruikers Vereniging (TRS-80 Users Group) in The Netherlands.]

The printer with the Video Genie is port-addressed. That's why you can't get a hardcopy with the Tandy EDTASM program. You can fix this by making the following changes:

Load the system tape. Don't start the program but press BREAK. Then change the following bytes:

```
POKE 19002,253                         HEX:  4A3A  FD
POKE 19029,253                               4A55  FD
POKE 19082,0                                 4A8A  00
POKE 19083,219                               4A8B  DB
POKE 19084,253                               4A8C  FD
```

Then start again by entering SYSTEM. When you see the "*?" enter "/" and ENTER. With the right DEBUG-type program you can save the changed program to tape.

TRS-80 AMATEUR RADIO NETS

Rich Thayer, W9LFO of Greenview, Illinois passed along the information that there are amateur radio nets devoted to the TRS-80 Models I/III/4 at 1:00 P.M. on Sundays at 14342 KHz. and at 4:00 P.M. at 7293 KHz. Times given are Central Standard Time. (Sounds interesting, wish I had a decent shortwave receiver so I could at least listen in!).

SCREEN BLANKING ROUTINE FOR USE WITH NEWDOS/80

by Carl Ford, Jr.

[The following routine was sent to us by Mr. Ford, who writes, "Here is an improved version of the screen blanking routine I sent you several months ago. The major change is that any video printing which uses the video DCB vector to the video driver will restore/maintain the video display, as will any use of the keyboard."

Following you will find 1) The EDTASM source code for the BLANK program, 2) The suggested /JCL file used to set HIMEM and run the BASIC demo program, and 3) The BASIC demo program itself. Please note that the program as is works with the Model III and Model 4 (in the Mod III mode) only. If the address 447BH in line 1810 is changed to 4410H, the program will run on the Model I, but the cursor character remains on the video display during periods when the rest of the screen is blanked. If anyone feels the urge to fix this problem and create a working Model I version, how about sending us the modifications and we'll pass them along!]

```
00100 ;
00110 ;IF YOU HAVE THE HABIT (OR NECESSITY) OF LEAVING YOUR
00120 ;MODEL III TURNED ON FOR LONG PERIODS OF TIME WHEN NO-ONE
00130 ;IS ACTIVELY TYPING ANYTHING IN (AS IN MANY BUSINESS SIT-
00140 ;UATIONS) EVENTUALLY THE SCREEN WILL SHOW "BURNED-IN"
00150 ;SPOTS FROM HAVING THE SAME CHARACTERS CONTINUOUSLY
00160 ;DISPLAYED IN THE SAME LOCATION. THIS WAS NOT A PROBLEM
00170 ;WITH THE MODEL I, AS YOU COULD JUST TURN OFF THE VIDEO
00180 ;AND LET THE CPU CONTINUE PROCESSING (OR JUST WAITING)
00190 ;WITHOUT THE VIDEO DISPLAY. HAVING THE COMPUTER SITTING
00200 ;THERE DISPLAYING TANTALIZING INFORMATION IS ALSO A
00210 ;POTENTIAL SECURITY RISK AND INVITES "PLAYING".
00220 ;
00230 ;AFTER WAITING 3 YEARS FOR SOMEONE ELSE TO PUBLISH OR
00240 ;ADVERTISE A SCREEN-BLANKING ROUTINE FOR THE MOD III, I
00250 ;FINALLY GAVE UP AND DID IT MYSELF.
00260 ;
00270 ;THIS ROUTINE PATCHES INTO THE ND-80 INTERRUPT QUEUE AND
00280 ;IS CALLED EVERY 255TH INTERRUPT CYCLE (ABOUT 6 SECONDS
00290 ;AT 3.4 MHZ PROCESSOR SPEED). WHEN CALLED, IT DECREMENTS
00300 ;AN INTERNAL TIMER ('TIME') AND IF 'TIME' IS NOT 0,
00310 ;CONTINUES WITH THE INTERRUPT QUEUE. IF 'TIME' EQUALS 0:
00320 ;   (1) SCREEN IS SAVED TO A 1024 BYTE BUFFER
00330 ;   (2) SCREEN IS BLANKED OUT
00340 ;   (3) CURSOR CHARACTER IS SAVED AND CURSOR IS SET TO
00350 ;       BLANK SPACE, SO RUNNING PROGRAM WON'T PUT CURSOR
00360 ;       BACK ONTO SCREEN.
00370 ;   (4) CHARACTER UNDER CURSOR IS SAVED AND SET TO BLANK
00380 ;      ('CURALT' IS LOCATION DEFINED FOR STORAGE OF CHAR-
00390 ;      ACTER UNDER CURSOR).
00400 ;   (5)'BLANK' IS REMOVED FROM INTERRUPT QUEUE (TURNED
00410 ;       OFF) SO IT WON'T DESTROY SAVED SCREEN BY OVER-
00420 ;       WRITING THE BUFFER WITH THE BLANKED-OUT SCREEN.
00430 ;   (6) RETURN TO INTERRUPTED PROGRAM, SO IT CAN CONTINUE
00440 ;      NORMAL PROCESSING WITH THE SCREEN BLANKED OUT.
00450 ;
00460 ;THERE IS A SECOND ROUTINE ('KEYS') PATCHED INTO THE KEY-
00470 ;BOARD SCANNING, WHICH CONTINUOUSLY MONITORS THE KEYBOARD
00480 ;FOR ANY KEYPRESS. WHENEVER THERE IS A KEYPRESS IT TAKES
00490 ;ONE OF TWO ACTIONS:
00500 ;   (1) IF THE VIDEO HAS NOT BEEN BLANKED, IT RESETS
00510 ;       'TIME' BACK TO THE ORIGINAL COUNT (5 MINUTES?) AND
00520 ;       JUMPS TO THE REGULAR ND-80 KEYBOARD ROUTINE.
00530 ;   (2) IF THE VIDEO IS ALREADY BLANKED, A KEYPRESS
00540 ;       INDICATES THAT THE OPERATOR IS BACK AND WILL NEED
00550 ;       TO SEE WHAT IS GOING ON. THE ROUTINE THEN RESTORES
00560 ;       THE SCREEN, CURSOR AND ALTERNATE CURSOR CHARACTER;
00570 ;       PUTS 'BLANK' BACK INTO THE INTERRUPT QUEUE (REAC-
00580 ;       TIVATES IT FOR NEXT USE); FLUSHES THE KEYBOARD
00590 ;       INPUT AND RETURNS TO THE RUNNING PROGRAM.
00600 ;
00610 ;THE THIRD ROUTINE('PRINT') IS PATCHED INTO THE THE VIDEO
00620 ;DCB VECTOR. WHENEVER THE VIDEO DRIVER ROUTINE IS CALLED
00630 ;BY A RUNNING PROGRAM, IT CHECKS THE SCREEN STATE AND
00640 ;TAKES THE SAME ACTIONS AS THE 'KEYS' ROUTINE ABOVE.
00650 ;
00660 ;THE PROGRAM ALSO CONTAINS A SUBROUTINE, 'DEACT', WHICH
00670 ;CAN BE CALLED FROM ASSEMBLER OR BASIC PROGRAMS TO
00680 ;DEACTIVATE 'BLANK' WHEN NECESSARY.
00690 ;
00700 ;USE:
00710 ;  (1) SET 'ORG' TO AN APPROPRIATE LOCATION FOR YOUR
00720 ;      MEMORY CONFIGURATION. "BLANK" NEEDS 174 BYTES
00730 ;      FOR ITSELF AND 1024 BYTES FOR A SCREEN BUFFER, AND
00740 ;      CAN BE ASSEMBLED TO ANY LOCATION IN MEMORY -- IT
00750 ;      CANNOT BE RELOCATED WITHOUT REASSEMBLY OR USE OF A
00760 ;      RELOCATING LOADER.
00770 ;  (2) CHECK THE 'EQU' LIST AND SET 'TIME' AND 'BUFFER'
00780 ;      VALUES TO SUIT YOUR NEEDS.
00790 ;  (3) ASSEMBLE AS 'BLANK/CMD'. THE APPARAT VERSION OF
00800 ;     ENHANCED 'EDTASM' HAS BEEN USED HERE. 'BLANK/CMD' ON
00810 ;     THIS DISK IS THE ALREADY ASSEMBLED VERSION OF
00820 ;     'BLANK/ASM', USING THE PARAMETERS GIVEN.
00830 ;  (4) SET 'HIMEM' TO THE NECESSARY LEVEL TO ALLOW ROOM
00840 ;     FOR THIS PROGRAM, THE SCREEN BUFFER AND ANY OTHER
00850 ;     ROUTINES YOU NEED IN HIGH MEMORY. AS ASSEMBLED HERE
00860 ;     'HIMEM' MUST BE SET TO 64334. VIDEO BUFFER MAY BE
00870 ;     USED BY OTHER PROGRAMS, AS LONG AS YOU MANAGE IT SO
00880 ;     'BLANK' DOESN'T DESTROY A SCREEN YOU NEED FOR OTHER
00890 ;     PURPOSES.
00900 ;  (5) USE THE DOS COMMAND "BLANK" OR "BLANK/CMD"
00910 ;     SOMEWHERE IN YOUR INITIALIZATION PROCEDURE. THIS
00920 ;     WILL LOAD 'BLANK', JUMP TO 'ENTRY' TO ACTIVATE IT
00930 ;     AND RETURN TO THE CALLING COMMAND. THIS CAN EASILY
00940 ;     BE DONE AS PART OF A 'JCL' FILE CALLED BY 'AUTO'.
00950 ;  (6) AFTER ACTIVATION, AT ANY TIME WHEN THE SET PERIOD
00960 ;     (ABOUT 2 MIN.) PASSES WITHOUT A KEY BEING DEPRESSED,
00970 ;     THE SCREEN WILL BE BLANKED.
00980 ;  (7) WHENEVER SCREEN IS BLANKED, IT MAY BE RESTORED
00990 ;     SIMPLY BY TOUCHING ANY KEY. 'SHIFT' IS BEST --
01000 ;     THE ROUTINE EXECUTES SO QUICKLY THAT ANY KEY
01010 ;     PRESSED WILL PROBABLY STILL BE DOWN WHEN THE
01020 ;     SYSTEM EXECUTES IT'S NEXT KEYBOARD SCAN AND WILL BE
01030 ;     RETURNED TO THE INTERRUPTED PROGRAM. 'SHIFT' RETURNS
01040 ;     AN ASCII '0', AS IF NO KEY WERE DEPRESSED.
01050 ;  (8) TO DEACTIVATE 'BLANK' FOR PROGRAMS THAT HAVE A
01060 ;     CONFLICT WITH IT'S MEMORY USAGE, JUST USE THE BASIC
01070 ;     STATEMENTS 'DEFUSR0=XXXXH:I=USR0(0)' (SET 'XXXX' TO
01080 ;     THE ADDRESS GIVEN FOR 'DEACT' IN YOUR ASSEMBLER'S
01090 ;     SYMBOL TABLE, OR IN THE LISTING).
01100 ;
01110 ;KNOWN LIMITATIONS ARE:
01120 ;  (1) FUNCTIONS ONLY UNDER NEWDOS-80 AT PRESENT, USING
01130 ;     THE "ENQUEUE A USER TIMER INTERRUPT ROUTINE" AT
01140 ;     4410H. COULD BE PATCHED DIRECTLY INTO INTERRUPT
01150 ;     SERVICE VECTOR OF OTHER OPERATING SYSTEMS, BUT ALL
01160 ;     TIMING WOULD HAVE TO BE IMPLEMENTED INTERNALLY.
01170 ;  (2) EXECUTING PROGRAMS WHICH DO NOT USE THE KEYBOARD
01180 ;     AND/OR VIDEO DCB VECTORS FOR PRINTING OR KEYBOARD
01190 ;     MONITORING (USE PEEK/POKE OR DRIVERS NOT PATCHED
01200 ;     INTO THE THE DCB'S) WILL NOT RESTORE THE SCREEN
01210 ;     BEFORE PROCESSING KEYBOARD OR VIDEO REQUESTS.
01220 ;        FOR PROGRAMS WHICH PATCH THEIR OWN DRIVER(S) INTO
01230 ;     THE DCB(S), YOU MUST FIRST DEACTIVATE 'BLANK' (IF
01240 ;     IT IS ACTIVE), THEN ACTIVATE THE APPLICATION PROGRAM
01250 ;     (TO GET IT'S VECTORS INTO THE DCB'S), AND LASTLY
01260 ;     REACTIVATE 'BLANK' BY JUMPING TO 'ENTRY'. THIS WILL
01270 ;     INSERT THE 'BLANK' ROUTINE(S) AHEAD OF THE OTHER
01280 ;     PROGRAM'S KEYBOARD/VIDEO DRIVERS.
01290 ;  (3) THERE ARE CONFLICTS WITH SOME MACHINE-LANGUAGE
01300 ;    PROGRAMS THAT MUST OCCUPY CERTAIN AREAS IN MEMORY.
01310 ;    THIS CAN BE CORRECTED BY RELOCATING 'BLANK' AND THE
01320 ;    SCREEN BUFFER TO SOME UNUSED AREA. IN PRACTICE, THIS
01330 ;    IS TOO MUCH TROUBLE, AND I JUST DEACTIVATE 'BLANK'
01340 ;    BEFORE LOADING THESE PROGRAMS. PROGRAMS WHICH RESPECT
01350 ;    THE 'HIMEM' SETTING WILL FUNCTION OK, UNLESS THEY
01360 ;    HAVE TROUBLE WITH THE FOLLOWING TWO PROBLEMS.
01370 ;  (4) 'BLANK' SAVES AND RESTORES THE CURSOR CHARACTER
01380 ;    AND THE CHARACTER UNDER THE CURSOR, USING THE RAM
01390 ;    STORAGE AREAS DESIGNATED FOR THIS PURPOSE IN THE
01400 ;    VIDEO DCB. ANY PROGRAM THAT UTILIZES INTERNAL STORAGE
01410 ;    FOR THESE CHARACTERS MAY NOT HAVE THEM RESTORED
01420 ;    PROPERLY. I'VE SOLVED THIS PROBLEM IN SEVERAL
01430 ;    PROGRAMS BY PATCHING THEM TO USE THE SAME AREAS
01440 ;    (IN THE DCB) FOR THIS STORAGE.
01450 ;  (5) 'BLANK' STAYS ACTIVE CONTINUOUSLY (UNLESS DEACT-
01460 ;     IVATED) AND IS REFERENCED ON EVERY INTERRUPT CYCLE.
01470 ;     IF THE RIGHT ROUTINE IS NOT AT THE RIGHT ADDRESS AT
01480 ;     EVERY INTERRUPT, IT WILL BOMB OR HANGUP. THIS MEANS
01490 ;     THAT YOU CANNOT OVERWRITE OR EVEN RELOAD 'BLANK'
```

```
              01500 ;    ITSELF, UNLESS YOU FIRST DEACTIVATE IT. IN MANY SIT-
              01510 ;    UATIONS A RESET ("COLD" OR "WARM") IS THE MOST
              01520 ;    PRACTICAL WAY TO REINITIALIZE THE COMPUTER.
              01530 ;
              01540 ;THIS PROGRAM HAS BEEN WRITTEN TO MEET MY PARTICULAR
              01550 ;NEEDS. THERE ARE A NUMBER OF MODIFICATIONS THAT YOU
              01560 ;MIGHT CONSIDER TO TAILOR IT TO YOUR NEEDS. WE
              01570 ;USE IT WITH A COMPLEX OF OTHER UTILITY ROUTINES THAT
              01580 ;ARE ALSO KEPT IN HIGH MEMORY AT NEARLY ALL TIMES. IF
              01590 ;USING 'BLANK' AS A SELF-CONTAINED ROUTINE, YOU MIGHT
              01600 ;ALSO MODIFY THE INITIALIZATION ROUTINE TO SET 'HIMEM'
              01610 ;AUTOMATICALLY TO THE FIRST BYTE BEYOND THE INITIAL-
              01620 ;IZATION ROUTINE, THUS RECOVERING A FEW BYTES AND
              01630 ;ELIMINATING THE NEED TO SET 'HIMEM' SEPARATELY.
              01640 ;
              01650 ;
              01660 ;*****************************************************
              01670 ;*                                                   *
              01680 ;*                   'BLANK/CMD'                     *
              01690 ;*                                                   *
              01700 ;*       ROUTINE TO BLANK SCREEN AFTER SET PERIOD    *
              01710 ;*                                                   *
              01720 ;* CARL FORD, 256 N VIRGINIA AVE, PENNS GROVE, NJ 08069 *
              01730 ;*                 (609)-299-0200                    *
              01740 ;*                                                   *
              01750 ;*                LAST REVISED - 11/12/84            *
              01760 ;*****************************************************
              01770 ;
              01780 ;
              01790 ;
FB4E          01800         ORG     64334           ;TO TOP-OF-MEMORY
447B          01810 ENQINT  EQU     447BH   ;INTERRUPT ENQUE ROUTINE
4413          01820 DEQINT  EQU     4413H   ;INTERRUPT DEQUE ROUTINE
000A          01830 TIME    EQU     10      ;# OF 6 SECOND PERIODS
3C00          01840 VIDEO   EQU     15360   ;VIDEO BEGINS
0400          01850 SIZ001  EQU     1024    ;LENGTH OF SCREEN
4023          01860 CURCHR  EQU     16419   ;CURSOR CHARACTER STORAGE
4022          01870 CURALT  EQU     16418   ;CHARACTER UNDER CURSOR STORAGE
0020          01880 BLANK   EQU     32      ;ASCII BLANK
4015          01890 KEYDCB  EQU     4015H   ;KEYBOARD DEVICE CONTROL BLOCK
401D          01900 VIDDCB  EQU     401DH   ;VIDEO DEVICE CONTROL BLOCK
38FF          01910 KEYTST  EQU     38FFH   ;KEY-PRESSED TEST LOCATION
              01920 ;
              01930 ;
              01940 ;         *** INITIALIZE ROUTINES ***
              01950 ;
              01960 ;*** INSERT KEYBOARD & PRINT MONITORS INTO KEYBOARD QUE
FB4E 2A1640   01970 ENTRY   LD      HL,(KEYDCB+1)   ;GET CURRENT KEYBD VECTOR
FB51 22C1FB   01980         LD      (KEYJMP+1),HL   ;PUT AT END OF ROUTINE
FB54 21B6FB   01990         LD      HL,KEYS         ;KEYBD PROCESSING ADDRESS
FB57 221640   02000         LD      (KEYDCB+1),HL   ;PUT INTO DCB VECTOR
FB5A 2A1E40   02010         LD      HL,(VIDDCB+1)   ;GET CURRENT VIDEO VECTOR
FB5D 22C9FB   02020         LD      (VIDJMP+1),HL   ;PUT AT CONTINUATION
FB60 21C3FB   02030         LD      HL,PRINT        ;RESTORE SCREEN VECTOR
FB63 221E40   02040         LD      (VIDDCB+1),HL   ;PUT INTO DCB VECTOR
              02050 ;
              02060 ;*** PUT "BLANK" INTO INTERRUPT QUE
FB66 116CFB   02070         LD      DE,INTRTN       ;INTERRUPT RTN ADDRESS
FB69 CD7B44   02080         CALL    ENQINT          ;PUT INTO INTERRUPT QUE
              02090 ;
              02100 ;         *** INTERRUPT PROCESSING ROUTINE ***
              02110 ;
              02120 ;*** DECREMENT TIMER & CHECK FOR 0
FB6C 00       02130 INTRTN  DEFB    0               ;WORK AREA
FB6D 00       02140         DEFB    0               ;WORK AREA
FB6E FF       02150         DEFB    255             ;# OF 6 SEC INTERVALS
FB6F FF       02160         DEFB    255             ;LENGTH OF 1ST INTERVAL
FB70 F5       02170         PUSH    AF              ;SAVE REGISTER
FB71 3AFBFB   02180         LD      A,(TIMER)       ;GET CURRENT TIMER COUNT
FB74 3D       02190         DEC     A               ;SHORTEN IT BY 1
FB75 32FBFB   02200         LD      (TIMER),A       ;SAVE IT
FB78 2033     02210         JR      NZ,EXIT         ;TIMER NOT 0 YET
              02220 ;
              02230 ;*** SAVE SCREEN
FB7A D9       02240         EXX                     ;SAVE REGISTERS
FB7B 11FCFB   02250         LD      DE,BUF001       ;BEGINNING OF BUFFER
FB7E 21003C   02260         LD      HL,VIDEO        ;BEGINNING OF SCREEN
FB81 010004   02270         LD      BC,SIZ001               ;BYTES TO MOVE
FB84 EDB0     02280         LDIR                    ;MOVE IT
              02290 ;
              02300 ;*** BLANK SCREEN
```

```
FB86 21003C   02310         LD      HL,VIDEO        ;BEGINNING OF VIDEO
FB89 3620     02320         LD      (HL),BLANK      ;BLANK 1ST BYTE
FB8B 11013C   02330         LD      DE,VIDEO+1      ;NEXT LOCATION
FB8E 01FF03   02340         LD      BC,SIZ001-1     ;AREA TO BLANK
FB91 EDB0     02350         LDIR                    ;FILL VIDEO
              02360 ;
              02370 ;*** SAVE & BLANK OUT CURSOR & ALTERNATE
FB93 3A2340   02380         LD      A,(CURCHR)      ;GET CURSOR CHARACTER
FB96 32F9FB   02390         LD      (CURSOR),A      ;STORE CURSOR
FB99 212340   02400         LD      HL,CURCHR       ;CURSOR STORAGE
FB9C 3620     02410         LD      (HL),BLANK      ;BLANK IT
FB9E 3A2240   02420         LD      A,(CURALT)      ;GET CHAR UNDER CURSOR
FBA1 32FAFB   02430         LD      (CURSOR+1),A    ;STORE IT
FBA4 212240   02440         LD      HL,CURALT       ;ADDRESS
FBA7 3620     02450         LD      (HL),BLANK      ;BLANK IT
              02460 ;
              02470 ;*** RETURN TO INTERRUPTED PROGRAM
              02480 ;
FBA9 CDAFFB   02490         CALL    DEACT           ;DISABLE 'BLANK'
FBAC D9       02500         EXX                     ;RESTORE REGISTERS
FBAD F1       02510 EXIT    POP     AF              ;RESTORE REGISTER
FBAE C9       02520         RET
              02530 ;
              02540 ;         *** REMOVE "BLANK" FROM INTERRUPT QUE ***
              02550 ;FOR USE BY PROGRAMS THAT NEED TO DISABLE "BLANK"
              02560 ;
FBAF 116CFB   02570 DEACT   LD      DE,INTRTN       ;VECTOR TO ROM ROUTINE
FBB2 CD1344   02580         CALL    DEQINT          ;REMOVE FROM QUE
FBB5 C9       02590         RET                     ;BACK TO CALLING PROGRAM
              02600 ;
              02610 ;         *** ROUTINE FOR MONITORING KEYBOARD ***
              02620 ;
FBB6 F5       02630 KEYS    PUSH    AF              ;SAVE REGISTER
FBB7 3AFF38   02640         LD      A,(KEYTST)      ;GET TEST LOCATION
FBBA FE00     02650         CP      0               ;KEY PRESSED
FBBC C4CBFB   02660         CALL    NZ,RESTOR       ;RESTORE SCREEN, ETC
FBBF F1       02670         POP     AF              ;RESTORE REGISTER
FBC0 C30000   02680 KEYJMP  JP      0               ;TO KEYBOARD DRIVER
              02690 ;
              02700 ;         *** ROUTINE FOR INTERCEPTING VIDEO ***
              02710 ;
FBC3 F5       02720 PRINT   PUSH    AF              ;SAVE REGISTER
FBC4 CDCBFB   02730         CALL    RESTOR          ;RESTORE SCREEN
FBC7 F1       02740         POP     AF              ;RESTORE REGISTER
FBC8 C30000   02750 VIDJMP  JP      0               ;TO PRINT DRIVER
              02760 ;
              02770 ;         *** CHECK SCREEN & RESTORE IF BLANKED ***
              02780 ;
FBCB D9       02790 RESTOR  EXX                     ;SAVE REGISTERS
FBCC 3AFBFB   02800         LD      A,(TIMER)       ;CHECK TIME LEFT
FBCF FE00     02810         CP      0               ;SCREEN ALREADY BLANKED?
FBD1 201D     02820         JR      NZ,RESET        ;IF NOT, GO
              02830 ;
FBD3 11003C   02840         LD      DE,VIDEO        ;BEGINNING OF VIDEO
FBD6 21FCFB   02850         LD      HL,BUF001       ;BEGINNING OF BUFFER
FBD9 010004   02860         LD      BC,SIZ001       ;BYTES TO MOVE
FBDC EDB0     02870         LDIR                    ;MOVE IT
              02880 ;
              02890 ;*** RESTORE CURSOR & ALTERNATE
FBDE 3AF9FB   02900         LD      A,(CURSOR)      ;GET CURSOR CHARACTER
FBE1 322340   02910         LD      (CURCHR),A      ;PUT IT BACK
FBE4 3AFAFB   02920         LD      A,(CURSOR+1)    ;GET CHAR UNDER CURSOR
FBE7 322240   02930         LD      (CURALT),A      ;RESTORE IT
              02940 ;
              02950 ;*** PUT "BLANK" BACK INTO INTERRUPT ROUTINE
FBEA 116CFB   02960         LD      DE,INTRTN       ;ROUTINES ADDRESS
FBED CD7B44   02970         CALL    ENQINT          ;RESTORE TO INTERUPT QUE
              02980 ;
              02990 ;*** RECYCLE TIMER & RETURN TO INTERRUPTED PROGRAM
FBF0 3E0A     03000 RESET   LD      A,TIME          ;GET DELAY
FBF2 32FBFB   03010         LD      (TIMER),A       ;RESET DELAY COUNTER
FBF5 D9       03020         EXX                     ;RESTORE REGISTERS
FBF6 3E00     03030         LD      A,0             ;CANCEL KEYPRESS
FBF8 C9       03040         RET                     ;BACK TO CALLING ROUTINE
              03050 ;
              03060 ;***SET STORAGE SPACE
FBF9 0000     03070 CURSOR  DEFW    0       ;STORAGE FOR CURSOR & ALTERNATE
FBFB 0A       03080 TIMER   DEFB    TIME    ;TIMING COUNTER
0400          03090 BUF001  DEFS    1024    ;VIDEO STORAGE BUFFER
FB4E          03100         END     ENTRY   ;AUTO EXECUTE FROM "CMD"
00000 TOTAL ERRORS
```

```
BLANK   0020   BUF001  FBFC   CURALT  4022   CURCHR  4023   CURSOR  FBF9
DEACT   FBAF   DEQINT  4413   ENQINT  447B   ENTRY   FB4E   EXIT    FBAD
INTRTN  FB6C   KEYDCB  4015   KEYJMP  FBC0   KEYS    FB86   KEYTST  38FF
PRINT   FBC3   RESET   FBF0   RESTOR  FBCB   SIZ001  0400   TIME    000A
TIMER   FBFB   VIDDCB  401D   VIDEO   3C00   VIDJMP  FBC8
```

[This is the contents of the START/JCL file:]

```
HIMEM,FB4EH
BLANK
BASIC,RUN"DEMO"
```

[This is the BASIC "DEMO" program:]

```
10 CLS
20 PRINT"
  This is a demonstration of 'BLANK' screen-blanking routine.
Just wait about 2 minutes and the screen will be cleared to
protect the video from burned-in characters.
30 PRINT"
  To return the blanked-out screen to it's previous condition,
just touch any key -- <SHIFT> is preferred.
40 PRINT"
  Any keyboard or printing to the video activity will delay
the screen-blanking for an additional 2 minute period. The
delay period may be adjusted to suit your preference.
100 GOTO100
```

ASK ALAN

by Alan Abrahamson

[The following article is reprinted from the August and October, 1984 issues of the Voice of the '80 newsletter, of which Alan is the editor.]

Being well known as somewhat of a deviot, this month's column shall deviate from the normal Q & A format. I have for some time now been doing lots of work with text editing, or word processing as some may know it better. I also have had the occasion to use my text from the word processor within my BASIC program. This type of situation was normally used as a sequential ASCII file, saved in ASCII from the word processor, and then used from within the BASIC program as text for directions or instructions within the program. I believe that I have explained the method of doing this in a prior column.

Well this time the "end" was different and the "means" to achieve this "end" was entirely different. I needed a method to create text. Well that was easy and straight forward using my trusty word processor program. But now I wanted to take that sequential ASCII data and transform it into a random access file so that I could select any "block" (record) of text from the file at will. I mused the problem for a bit and came up with the following program to accomplish the task.

The program is heavily commented and should not be difficult for any of you to decipher. I don't know if others need this capability. I do know that it was useful to me to create help screens with my word processor, block the text into records of 256 bytes (4 lines of text of 64 characters per line included in each record), and then use this utility to move the sequential data into a random record access file. I could now select any record in the file with a standard random file OPEN and GET. On to the program.......

SEQTORAN/UTL

```
1 REMarkable SEQTORAN/UTL by Alan Abrahamson (05/30/84)
   Takes text file from word processor. Each line must end with
   (C/R) & blocks it into a random file record, 4 lines per each 256
   byte record.
10 CLEAR 1000 : CLS
   'Clear string space, wipe off screen.
12 LINEINPUT"Text file name ";F1$ : LINEINPUT"Help file name
   ";F2$
   'Get word processor filespec & Help file filespec.
20 OPEN"I",1,F1$
   'Open Text file in buffer 1.
30 OPEN"R",2,F2$
   'Open Help file in buffer 2.
35 FORX=0TO3 : FIELD2, (64*X) AS DU$, 64 AS B$(X) : NEXT
   'Field buffer for 4 screen lines of 64 characters each.
36 FORX=0TO3
   'Start loop
38 IF EOF(1) THEN CLOSE : END
   'Check for EOF and end if true.
40 LINEINPUT#1,A$
   'Read sequential record to Carriage Return.
42 LSET B$(X) = A$ : NEXT
   'Set sequential line into buffer variable.
67 FORX = 0 TO 3 : PRINT B$(X); : NEXT : PRINT
   'Show what you are going to file to disk.
70 PUT 2, X1 + 1
   'Put record to disk. X1 will equal 0 initially.
80 X1 = X1 + 1
   'Increment X1
90 GOTO35
   'Go do it again at 35.
60000 SAVE"SEQTORAN/UTL
   'Alan's SAVE line
```

Now that you have the random help file created, it is a simple task to display that data from within your BASIC program. I shall show you, with the following code, one way to do this.

DISPLAY/BAS

```
10 CLEAR 10000 : CLS
   'Make some string room, clear CRT
20 OPEN"R",1,"HELPFILE"
   'Open your HELPFILE created with SEQTORAN/UTL
30 FIELD 1, 128 AS A$, 128 AS B$
   'Field 2 variables to receive text
40 LINEINPUT "Which record shall I get for you? "; RN$
   'Prompt for help file record number
50 RN = VAL(RN$) : GET 1, RN : PRINT A$ : PRINT B$ : GOTO 40
   'RN=Record number, GET record, Print buffer data out of A$ and
   B$ and then go back to 40 for next prompt.
```

So far I have presented to you a method of transferring a sequential text (word processor) file to a random access file. I am now going to show you a way to use this help screen data in a practical manner within your own BASIC program. The demonstration program below will accomplish this task.

Each line of code is liberally commented and if you key in the code, prepare a text file saved in ASCII with you word processor, and use the SEQTORND/UTL printed above you will really get the meaning of this demonstration. The technique of using the dummy variables to block move the data using "LSET" is about 10 times the speed of a similar "PRINT" statement. The LSET command appears to do a Z-80 block move (LDIR,LDDR) between the VARPTR's (pointers) of the variables involved. This technique can be used in any screen data manipulation routine to move data from point A to point B. Remember that the LSET routine physically moves the data from one place to another, it is NOT just a pointer swap.

The key ingredient in using this method is to establish the dummy or real variables, prior to the LSET. It is not necessary to OPEN any files in using LSET, but you cannot use LSET on a variable that has not yet been initialized. Lines 18 and 19 set up the dummy variable (DU$) and the holding variable (U$) so that after line 37 moves the pointers, lines 38, 41 and 43 can LSET (move) the data.

Enjoy!!!

HELP/DEM

```
1 REMarkable HELP/DEM program by Alan B. Abrahamson
   (06/01/84)
   Setup the HELP file with word processor, save in ASCII with
   format of 4 lines each ending with C/R for each help record. Now
   run SEQTORAN/UTL.
10 CLEAR 765 : CLS : DEFINTA-Z
   'Make string space, define all variables as integers
12 LINEINPUT "Filename of help file ";HF$ : IF HF$="" THEN
   HF$="HELP/DAT" : PRINT HF$ : OPEN"R",1,HF$ : CLS ELSE
   OPEN"R",1,HF$ : CLS
   'Prompt for filespec, open file then clear screen.
13 FIELD1,255 AS A$
   'Field random buffer (255 bytes)
15 TRASH$="This is the trash that may have appeared on the last
   four lines of the screen. This stuff must be replaced back on the
   screen   after the help message has been shown. The idea is to
   capture   the screen, store it and then put it back after."
16 PRINT@768,TRASH$; : TRASH$=""
   'Print data then null string.
17 POKE16383,191
```

```
   'Sample data in 15 (255 bytes), place a bit of data in last screen
   location for this test.
18 DU$=" "
   'set up dummy string for screen dump
19 U$=STRING$(255,32)
   'Set up U$ to hold screen data.
20 PRINT@64,CHR$(30); : LINEINPUT"Help screen number, ''END''
   to exit  ";HS$ : GOSUB600 : HS=VAL(HS$) : IF HS<1 OR HS>10
   THEN 20
   'Prompt for screen, 600 checks for exit, check for valid screen
   number.
30 GET1,HS
   'Get 1 Help screen record.
31 PRINT@1021,RIGHT$(STR$(HS),2);
   'Print help screen number for proof of new screen data.
37 B1$=CHR$(PEEK(16383)) : POKE VARPTR(DU$),255 : POKE
   VARPTR(DU$)+1,0 : POKE VARPTR(DU$)+2,63
   'Put last screen byte in B1$, POKE DU$ length to 255, POKE DU$
   with screen address (256*63+0 = 16128)
38 LSETU$=DU$
   'Physically move screen data to U$ in memory.
41 LSETDU$=A$
   'Print Help screen from buffer to screen.
42 IN$=INKEY$ : IF IN$="" THEN 42
   'Hold screen with inkey$ for key press.
43 LSET DU$=U$ : POKE 16383,ASC(B1$)
   'Print U$ back to screen and replace last byte stored in B1$
50 GOTO20
   'Go get another help screen number.
600 IF HS$="End" OR HS$="END" OR HS$="end" THEN CLOSE : END
   ELSE RETURN
   'Have clean way to exit the program and close file with word
   ''END''.
60000 SAVE"HELP/DEM
   'Alan's save line.
```

INTERRUPTS IN THE TRS-80

by Warwick Sands

[This article is reprinted from the Adelaide (South Australia) MICRO-USER News. It is specifically applicable to the Model I, but some of the principles involved may be of interest to users of other Models as well.]

Firstly let's examine the interrupt mechanism inherent in the Z80. There are three kinds of interrupt modes in the Z80: IM0, IM1, IM2. The Z80 defaults to IM0 at power-up, since this allows 8080 code to run on the Z80 without modification.

I don't want to become too involved in the hardware side of things but one or two points do need to be explained. I hope I can do so simply and lucidly.

Interrupts are initiated by some external device forcing the voltage on pin 16 of the Z80 to 0 volts. At the end of every instruction the Z80 checks to see if the voltage on pin 16 has gone to 0 volts. If this has occurred, even momentarily, and interrupts are enabled then several things occur. Firstly the Z80 disables all further interrupts. Secondly the Z80 acknowledges the interrupt by pulling two other lines to 0 volts. This may be detected by the interrupting device. What happens then depends on which interrupt mode is active.

If in IM0, then the Z80 expects an instruction to be placed upon the DATA bus by the calling device. This can be any instruction but normally it is a CALL or RST instruction.

IM1 expects no action on the part of the interrupting device. Rather the Z80 forces a call to 38H. The code at 38H handles the servicing of the interrupt. If necessary it determines which device has caused the interrupt and jumps to the relevant code. This is the interrupt mode used in the TRS-80 and SYSTEM-80.

IM2 is the most powerful interrupt mode. In this mode the interrupting device puts on the DATA bus the lower 8 bits of the address of a table. The upper 8 bits are stored in the the I register in the Z80. The Z80 then retrieves from this table the address of the routine which handles the interrupting device and jumps to it. It is a pity it can't be used, but TANDY didn't want to install the 25 cents worth of integrated circuit required to allow the Z80 to receive this data from the interrupting device!

There is also the non-maskable interrupt (NMI). The NMI causes a jump to 66H. In the TRS-80 the code at 66H checks if an expansion interface is connected. If so, then the program jumps to 0H which boots the disk, otherwise program goes to BASIC ready, via 6CCH. The non-maskable interrupt is initiated by the RESET button on the '80.

Let's now examine how the maskable interrupt works on the '80. Firstly under Level II. If there is no expansion interface in the system, an interrupt is never initiated therefore no problems will ever occur. If there is an expansion interface in the system interrupts will be generated every 25 milliseconds.

However the very first instruction in the ROM is a DI (Disable Interrupts) and since the Z80 only responds to interrupts at the end of an instruction, the interrupt will never be acknowledged until the interrupts are once more enabled with the Enable Interrupts (EI) instruction. Since there is no EI instruction in the ROM, under normal circumstances an EI instruction is never encountered.

If you write a machine code routine that runs under Level II and you do have an EI instruction what happens? If you don't have an expansion interface, no worries because no interrupts are generated. If you do have an expansion interface, you have an interrupt occurring every 25 milliseconds. Now what? Note that the interrupt mode has defaulted to IM0 and since there is no code in the ROM to change it to IM1, the CPU will expect an instruction on the data bus. What instruction will it get? Since no data is supplied on the data bus, the instruction will be perceived as 0FFH which is the RST 38H instruction.

The code at 38H forces a jump to 4012H. According to all the 'reliable' commentaries on the '80, the code should read:

```
        DI          ;disable interrupts
        RET         ;continue with previous task
```

This is what one would expect to be there. If there is no interrupt handler present, acknowledge no future interrupts and return to the caller. However a bug crept in for it actually reads:

```
        EI          ;enable interrupts
        RET         ;continue with previous task
```

Oh well, you might say, at the worst it means we execute an extra six or so instructions every 25 milliseconds. Unfortunately, it is not as simple as that. When the hardware generates the 25 millisecond interrupt, it permanently puts 0 volts on pin 16, the interrupt line. The only way this 0 volts can be removed is by reading memory location 37E0H (this resets a flip-flop). So the sequence of events turns out to go something like this:

The interrupt is generated. The processor ends up at the code at 4012H. This enables the interrupt. Since pin 16 is still 0 volts, the processor immediately ends up back at 4012H. And you have your typical computer hangup.

Three possible solutions for the machine language programmer in the Level II environment spring to mind. Firstly don't have any EI instructions in the machine language routine. Secondly, change the code at 4012H to what it should have been. Thirdly, write a routine that uses the interrupt mechanism and make certain that you read 37E0H.

If you're working under any DOS there are no problems. The DOS will set the interrupt mode to IM1. In TRSDOS 2.3 it resides at 4E01H in SYS0. The DOS interrupt handler will read 37E0H and reset the interrupt.

Another small point of interest. When you execute a CMD"T" the hardware interrupt isn't turned off. The interrupt line is still pulled low every 25 milliseconds, the CMD"T" routine simply causes the routine to skip any interrupt processing. But 37E0H is still read, allowing pin 16 of the Z80 to return 5 volts.

I still can't understand why everyone says NEWDOS/80 is so complicated, I learned to drive it in about a day and a half. Oh well, to each his own.

FREE DATA BASE DIRECTORY

[Reprinted from the Tuscon PC Users Group newsletter by way of the Long Island Computer Association, Inc. Newsletter]

Telenet is making available a free 100-page directory that describes over 300 on-line data bases, as well as 140 data base vendors and computer service companies who use Telenet services. A separate section identifies dozens of companies who offer business applications software in such areas as cash management, data base management, and financial analysis.

For your free copy, contact:

GTE Telenet
8229 Boone Blvd.
Vienna, Virginia 21180
(800) 835-3638

NEWDOS/80 TIPS FROM LAURIE SHIELDS

Laurie has passed along a couple of tips for NEWDOS/80 users that I think are worthy of being passed along. First of all, he reports a BUG in NEWDOS/80 version 2, in that if you exit BASIC with a CMD"S=DosCommand" with files open, BASIC will not close them as it does with CMD"S", LOAD, RUN, etc. Second, Laurie reports that he's discovered "the magic of NEWDOS/80's /./5 format in their JCL files. What it does is to descend one level of DOS activity in the execution of a DO file." Unimpressed? Well, consider that you can go into MINI-NEWDOS/80 from within any application program and DO a JCL file. If that JCL file has the /./5 code within it, it will drop back into the application program (in effect it will do an MDRET) and the remainder of the JCL file will be intercepted by the application program as keyboard input (assuming, of course, that the application program uses the DOS keyboard driver). In some cases this feature could save lots of repetitive typing, since the JCL file could contain a frequently used series of commands. Granted, this is one of those features that maybe 95% of our readers will never use, but the other 5% may be very glad that I passed the info along!

PROMPT

Program by Joop van Dam

Translated from Dutch to English by Paul Fransen

[This program is reprinted from REMARKS, the publication of the TRS-80 Gebruikers Vereniging (TRS-80 Users Group) in The Netherlands. It's not of great practical value, I admit, but April Fool's Day will be here before you know it...]

This program will let you create your own prompt (to replace BASIC "READY"). It works on a Model I, Level II or with DOS BASIC. Before you run the program, save it (see last statement in line 210). The machine language routine and the string (from INPUT) will be poked into unused RAM, so you don't have to set MEMSIZE. The input routine could be a simpler one, but this is an alternative.

```
100 CLS:PRINT@320,"Input a string with a maximum of 23 characters..:"
110 CLEAR60:PRINT:PRINTCHR$(138)STRING$(22,136)STRING$(22,24);
120 T$=STRING$(2,8)+CHR$(138)+CHR$(136)+CHR$(24)
130 FORX=0TO1STEP0:I$=INKEY$:IFI$=""THENNEXT
140 IFI$=CHR$(13)THENPRINTCHR$(8)CHR$(30);:GOTO190
150 IFI$=CHR$(8)ANDP$=""THENNEXT
160 IFI$=CHR$(8)THENPRINTT$;:P$=LEFT$(P$,LEN(P$)-1):P=P-1:NEXT
170 P$=P$+I$:PRINTCHR$(8)I$CHR$(138);
180 P=P+1:POKE16696+P,ASC(I$):IFLEN(P$)<23THENNEXT
190 POKE16697+P,13:POKE16698+P,0
200 FORX=0TO15:READD:POKEX+16571,D:NEXT
210 POKE16812,195:POKE16813,187:POKE16814,64:NEW
220 DATA205,248,1,205,249,32,33,57,65,205,167,40,225,195,43,26
230 'Program: Joop van Dam.
```

If the special prompt disappears then you can recover it by entering:

POKE 16812,195 : POKE 16813,187 : POKE 16814,64

This works fine, even after a re-Boot with another DOS!

CLUBS AND NEWSLETTERS

We have recently completed a purge of our mailing list, and while the list below is not 100% accurate, it's in a lot better shape than it was a few months ago. This list consists of user groups with which we exchange newsletters, a few user groups that are temporarily not publishing a newsletter, and a few newsletters not associated with a user group. The codes at the end of each listing indicate the orientation of the club or newsletter - (T) indicates TRS-80, (C) is TRS-80 Color Computer only, and (G) is used to indicate a group or newsletter that caters to all makes and models of computer, but includes at least some support for TRS-80 users.

If you spot an error, or a club or publication that has been discontinued, please drop us a postcard or letter and let us know. On the other hand, if your group isn't listed and you feel it should be, first make sure that if your group produces a newsletter you are sending exchange copies to NORTHERN BYTES at the correct address: c/o Jack Decker, 1804 West 18th Street, Lot # 155, Sault Ste. Marie, Michigan 49783 (sending them to The Alternate Source in Lansing DOES NOT COUNT!!!!!). Then drop us a note and we will try to determine why your group has not been included on the list.

This list is sorted within countries by zip or postal code order.

U.S.A.:

Western Massachusetts Computer Club, 59 East Canton Circle, Springfield, Massachusetts 01104 (G)
TRS-80 Users Group of Eastern Massachusetts, c/o Matthew W. Slate, 42 Oak Hill Road, Sudbury, Massachusetts 01776 (T)
Boston Computer Society, One Center Plaza, Boston, Massachusetts 02108 (G)
August J. Woerner, S.O.A.R. Network News, 4 Nathaniel Road, Barrington, Rhode Island 02806 (G)
Southern Maine TRS-80 Users Group, c/o Anthony T. Scarpelli, editor, 82 Wellington Road, Portland, Maine 04103 (T)
Connecticut Computer Society, Inc., 1199 Farmington Avenue, West Hartford, Connecticut 06107 (G)
Fairfield County TRS-80 Users Group, c/o Alan Abrahamson, 10 Richlee Road, Norwalk, Connecticut 06851 (T)
Central Jersey Computer Club, c/o Richard H. Williams, RD # 1, Box 147, Hopewell, New Jersey 08525 (G)
Garden State Color Computer Users' Group, c/o Edward Savage, Jr., 83 Archertown Road R.R. # 1, New Egypt, New Jersey 08533 (C)
BUG-80 Users Group, P.O. Box 62, Glen Gardner, New Jersey 08826 (T)
Amateur Computer Group of New Jersey, P.O. Box 319, South Bend Brook, New Jersey 08880-0319 (G)
New York Amateur Computer Club, Inc., P.O. Box 106, Church Street Station, New York, New York 10008 (G)
ABC Micro Newsletter, c/o Mike Hudak - ABC Television, 1345 Avenue of the Americas, 28th Floor, New York, New York 10019 (G)
Metro TRS-80 Users Group, c/o G.F. Mueden, 310 West 106th Street (15D), New York, New York 10025 (T)
Kings Byte Newsletter, c/o Morty Libowitz, 1063 East 84 Street, Brooklyn, New York 11236 (T)
Long Island Computer Association, P.O. Box 71, Hicksville, New York 11902 (G)
TRS-80 Users Group of Central New York, 113 Belle Terrace, North Syracuse, New York 13212 (T)
Rochester Area Microcomputer Society, P.O. Box 90808, Rochester, New York 14609 (G)
Rochester S-80 Computer Club, Inc., P.O. Box 15476, Rochester, New York 14615 (T)
Capitol Area TRS-80 Users Group, c/o Fairview Township Fire Hall, 340 Lewisberry Road, New Cumberland, Pennsylvania 17070 (T)
Philadelphia Area Computer Society, P.O. Box 312, LaSalle University, Philadelphia, Pennsylvania 19141 (G)
Association of Personal Computer Users, c/o Donald G. Chapman, 3911 Blackthorn Street, Chevy Chase, Maryland 20815 (G)
TCUG, Inc., P.O. Box 2826, Fairfax, Virginia 22031 (T)
Richmond TRS-80 Users Group, c/o Merle D. Kerby, 13419 Oak Lane, Midlothian, Virginia 23113 (T)
Greensboro Eighty Microcomputer Society (GEMS), P.O. Box 10014, Greensboro, North Carolina 27404 (T)
Midlands-80 Computer Club, P.O. Box 7594, Columbia, South Carolina 29202 (T)
Online Telecomputing, Inc., 1175 Grimes Bridge Road, Suite 199-CS4000, Roswell, Georgia 30075 (G)
CSRA Computer Society, Post Office Box 284, Augusta, Georgia 30903 (G)
Cursor Computer Club, c/o J.R. Foss & Associates, 3816 Saxon Drive, New Smyrna Beach, Florida 32069 (G)
Tallahassee Amateur Computer Society, Post Office Box 6716, Tallahassee, Florida 32314 (G)
Pensacola TRS-80 Users Club, Dr. Frank Biasco, President, 9759 Pickwood Drive, Pensacola, Florida 32504 (T)
Central Florida Computer Society, P.O. Box 8019, Maitland, Florida 32751 (G)
Tampa Bay TRS-80 Users Group (TBUG-80), c/o Tom Stiles, 1721 Greenlea Drive, Clearwater, Florida 33515 (T)
ELSE, P.O. Box 2142, Pinellas Park, Florida 33565 (T)
Central Alabama Microcomputer Society, P.O. Box 17021, Montgomery, Alabama 36117 (G)
Greater Gulf Coast Computer Club, c/o Gerald T. Regan, Spring Hill College, Mobile, Alabama 36608 (G)
Computer Club for the TRS-80 Users, Dept. of Chemistry, East Tennessee State University, Box 23350A, Johnson City, Tennessee 37614 (T)

HUB Computer Users Group, c/o Elvis B. Gates Jr., P.O. Box 3713, Hattiesburg, Mississippi 39401 (T)
Columbus & Central Ohio Color Computer Club, c/o Dave Fultz, 3648 Beulah Road Apt. C, Columbus, Ohio 43224-3133 (C)
Amateur Computer Society of Central Ohio, c/o John Cramer, P.O. Box 28606, Columbus, Ohio 43228 (G)
Toledo Area Tandy Special Interest Group, c/o John C. Stahl, Secretary, 3324 Oakway Drive, Toledo, Ohio 43614 (T)
The Color Computer Club, P.O. Box 478, Canfield, Ohio 44406-0478 (C)
Cincinnati TRS-80 Users Group (CINTUG), P.O. Box 399145, Cincinnati, Ohio 45239 (T)
Northeast Computer Club, P.O. Box 50252, Indianapolis, Indiana 46250-0405 (T)
Dearborn TRS-80 Users Group, P.O. Box 1942, Dearborn, Michigan 48121 (T)
Michigan Computer Consortium, P.O. Box 1302, East Lansing, Michigan 48823 (G)
Mid Michigan Microcomputer Group, c/o 1732 Cambria Drive # A-1, East Lansing, Michigan 48823 (G)
Educational, Recreational Computer Club, Box 325, Owosso, Michigan 48867 (G)
Central Michigan TRS-80 Users Group (CMTUG), c/o Dennis Hill, 410 Liberty Street, Lansing, Michigan 48906 (T)
Kalamazoo Area Computer Association, 7044 Leawood, Portage, Michigan 49081 (T)
Christian Computer Users Association, Inc. - TRS-80 SIG, c/o Bob Grommes, B & G Microsystems, 1733 Eastern S.E., Grand Rapids, Michigan 49507 (T)
Northern Bytes, c/o Jack Decker, 1804 West 18th Street, Lot # 155, Sault Ste. Marie, Michigan 49783 (T)
FISHNET, Jeffrey & Sandra Carlson, 1111 Fifth Avenue, North Fort Dodge, Iowa 50501 (G)
Milwaukee Area TRS-80 Users Group, P.O. Box 184, South Milwaukee, Wisconsin 53172-0184 (T)
Madison TRS-80 Users Group, 1004 East Johnson Street, Madison, Wisconsin 53703 (T)
Continental Computer Club, c/o John Witmer, 1035 Marshall Road, Northbrook, Illinois 60062 (G)
Northern Illinois Computer Owners League, c/o Charles C. Hornaday, 26 W 484 Butterfield Road, Wheaton, Illinois 60187 (T)
Chicago Area Computer Hobbyist Exchange, Box C-176, 323 South Franklin, # 804, Chicago, Illinois 60606 (G)
Great River Microcomputer Users Group, c/o L. Moeller, 1226 Daniel Court, Quincy, Illinois 62301 (T)
Kansas City TRS-80 Users Group, c/o Mary Youngblood, 300 N.W. 83rd Street, Kansas City, Missouri 64118 (T)
Religious Computer Users Group, Overland Park Christian Church, 7600 West 75th Street, Overland Park, Kansas 66204 (G)
Little Apple CoCo MUG, c/o L. Haverstein, 1915 College Heights Road, Manhattan, Kansas 66502 (C)
Omaha Area TRS-80 Users Group, c/o Dan Shelby, Chariman, 7318 South 33rd Street, Omaha, Nebraska 68147 (T)
LNW User Group Newsletter, Jay J. Hokanson, Editor, 4345 Manchester Road, Grand Island, Nebraska 68801 (T)
Tulsa Computer Society, Post Office Box 1133, Tulsa, Oklahoma 74101 (G)
Christian Computer Users Network, c/o Rev. Ed Deane, Highland Park United Methodist Church, 3300 Mockingbird Lane, Dallas, Texas 75205 (G)
Fred Maia W5YI, Editor, W5YI Report [an amateur radio newsletter], P.O. Box # 10101, Dallas, Texas 75207 (G)
Xcalibur Newsletter, Keene Computer Services, Box 13044, College Station, Texas 77840 (T)
Corpus Christi TRS-80 Users Group, c/o Patrick Michaud, 2201 Hickory Drive, Portland, Texas 78374 (T)
ZIN User Group, c/o Harold Thomas, Editor, 530 Belleview Drive, Corpus Christi, Texas 78412 (T)
Lubbock Computer Club, Rt. 1 Box 61, Lubbock, Texas 79401 (G)
Denver Area TRS-80 Association, c/o Mike Hastings, 16887 East Linvale Place, Aurora, Colorado 80013 (T)
Denver Amateur Computer Society, Box 477, Wheatridge, Colorado 80034-0477 (G)
Waterton Computer Club (TRS-80 SIG), c/o Lowell Simons, 1150 Ridge Road, Littleton, Colorado 80120 (T)
Computer Notes, c/o Bill White, 1863 Bitterroot Drive, Twin Falls, Idaho 83301 (T)
Tuscon Color Computer Users Group, Box 15186, Tuscon, Arizona 85708 (C)
Valley Computer Club, P.O. Box 6545, Burbank, California 91510-6545 (G)
North Orange County Computer Club, P.O. Box 3616, Orange, California 92665-0616 (G)
Cabrillo Computer Society, P.O. Box 3032, Camarillo, California 93010 (T)
Cybernaughts, c/o Matthew D. Lee, P.O. Drawer 1943, Santa Maria, California 93456 (T)
Marin County TRS-80 Users Group, 120 Lincoln Drive, Sausalito, California 94965 (T)
Jefferson State Computer Users Group, c/o C.B.C. Incorporated, 2355 Camp Baker Road, Medford, Oregon 97501 (G)
Northwest Computer Society, P.O. Box 4193, Seattle, Washington 98104 (G)
Snohomish Users Group, c/o J. Emge, P.O. Box 1323, Stanwood, Washington 98292 (T)

CANADA:
Micro-80 Computer Club of Ottawa, 150 Metcalfe Street Suite 409, Ottawa, Ontario, K2P 1P1 (T)
NEWDOS/80 International Users Group, c/o Greg Small, Box 607, Stouffville, Ontario, L0H 1L0 (T)
Mr. John Easton, Christian Computer/based Communications, 44 Delma Drive, Toronto, Ontario, M8W 4N6 (G)
Kitchener-Waterloo Microcomputer Club, E2-3354, Electrical Engineering Dept., University of Waterloo, Waterloo, Ontario, N2L 3G1 (T)
Winnipeg Micro-80 User's Group, c/o Don Rigg, 17 Bittersweet Bay, Winnipeg, Manitoba, R2J 2E5 (T)
Vancouver TRS-80 Users Group, #2101 - 3737 Bartlett Court, Burnaby, British Columbia, V3J 7E3 (T)
NEWDOS/80 User's Group, c/o Jack Douglas Feka, P.O. Box 1717, Victoria, British Columbia, V8W 2Y1 (T)

OUTSIDE THE U.S.A. AND CANADA:
Sydney TRS-80 Users Group (SYDTRUG), P.O. Box 297, Padstow, New South Wales 2211, AUSTRALIA (T)
TRS-80 System-80 Computer Group, c/o W.J. Allen, 16 Laver Street, Macgregor, Queensland 4109, AUSTRALIA (T)
Adelaide Micro-User News, 36 Sturt Street, Adelaide, South Australia 5000, AUSTRALIA (T)
L'Association Des Utilisateurs De TRS-80, 132, rue La Fayette, 75010 Paris, FRANCE (T)
TRS-80 Gebruikers Vereniging, Postbus 551, 2070 AN Santpoort-Nord, HOLLAND (T)
TRS-80 Gebruikersvereniging, Paul Fransen, Secretary, Havikhoek 48, 3201 SH Spijkenisse, HOLLAND (T)
N.Z. Microcomputer Club Inc., P.O. Box 6210, Auckland, NEW ZEALAND (G)
Microsoft Club, c/o Gabriel Dominguez, Magistrado M. Artime, 6; 15004 La Coruna, SPAIN (T)

NORTHERN BYTES INDEX FOR VOLUME 5

Notes on this index: 1) If an asterisk appears next to a name in the author's name field, it means that the person mentioned did not actually write the article, but did provide the pertinent information. 2) the issue and page number are given in the format 3/17 (in this example, the article would be found in NORTHERN BYTES Volume 5, Number 3, page 17). If the symbol */* appears, it means that a correction or follow-up article appears in THIS ISSUE (we could not give the page number since the layout for this issue had not been completed at the time this index was printed). 3) We have tried to index all corrections, follow-up articles, and "letters to the editor" dealing with a given article immediately following the listing for the original article. 4) SOME, but not all, "letters to the editor" have been indexed. The indexed letters usually contained useful information of some sort.

EDITORIAL

BUTTERFIELD, JIM SWAPPING AND SHARING
An essay on software swapping 8/8
DECKER, JACK IN SEARCH OF... 80-USER DIGEST
Whatever happened to... 1/1
DECKER, JACK PUBLIC DOMAIN SOFTWARE & BULLETIN BOARD SYSTEMS (EDITOR'S SOAPBOX REVISITED)
Startup of TAS Public Domain Software Library 1/1
FOLLOW UP - EDITORIAL COMMENTS 8/1
DECKER, JACK REACH OUT, REACH OUT AND CRUSH SOMEONE...
AT&T does it to us again 2/12

FEATURE ARTICLES

HARDWARE

REVIEWS

NYACC Newsletter PERSONAL SOFTWARE MAGAZINE
Why they never review a "bad" product 3/17
SMITH, WINFIELD * UN-DISK DEPARTMENT
"The Original Paper Porter" and ESF interests 2/19

SOFTWARE

ADAMS, JOHN C., JR. ASSIGN THOSE LUN'S
Help for Microsoft FORTRAN package users 5/19
BAKER, ROXTON A QUICK Z80 TABLE SEARCH ROUTINE
Z80 binary search algorithm 1/9
BECKER, LEE MOD4BASC/BAS
Converts Model III BASIC programs to Model 4 format 1/2
BERTSEKAS, DIMITRI P. BASICOMP
Version 2 of a BASIC compiler program 5/9
BOWER, DAVID H. LETTER TO THE EDITOR
Three one-liner programs for the Pocket Computers 6/5
BROTHERTON,MICHAEL WHERE
Shows exact location of error within BASIC line 7/6
CHRISTENSEN, WARD MODEM PROTOCOL DOCUMENTATION
Specifications for the popular MODEM protocol 3/18
COULTER, BILL EXPRESSION INPUT ROUTINE
Input BASIC expressions for evaluation by USR routine 5/22
COVILL, D.L. MODEM FILE TRANSFER PROTOCOL
More information on Christensen's protocol 3/18
DECKER, JACK MODEL 4 "KBMOD/ASM" PROGRAM
Adds screen edit feature to Mod 4 in Mod III mode 1/3
DECKER, JACK REROUTE/ASM - A PROGRAM FOR NEWDOS/80 V.2 USERS
Permits routing printer output to disk 1/4
DECKER, JACK SETDATE
The easy way to enter the date at system startup 2/7
THE EXTERMINATOR 4/1
THE EXTERMINATOR 5/2
LETTER TO THE EDITOR 6/4
LETTER TO THE EDITOR 7/2
THE EXTERMINATOR */*
DECKER, JACK UPPERCASE FUNCTION FOR MODELS I & III TRS-80
USR function converts mixed case string to uppercase 2/22
DECKER, JACK ADDING YOUR OWN /SYS OVERLAYS TO NEWDOS/80
Includes new code for "Improved Ampersand Function" 3/3
THE EXTERMINATOR 4/2
THE EXTERMINATOR 7/1
DECKER, JACK BINARY CLOCK
Machine language program demonstrates binary counting 5/2
DECKER, JACK MCI MAIL FILE TRANSFER PROGRAMS
Converts any file so it can be sent via MCI Mail 7/3
DOFFING, JIM MENU/CMD
Makes TRSDOS 1.3 menu-driven 7/9
DOMIGAN, TONY TRSDIR
Read TRSDOS 1.3 directory under NEWDOS/80 2/16
THE EXTERMINATOR 3/1
DOMIGAN, TONY ZAPDEBUG
Modifies TRSDOS 1.3 DEBUG to read disk sectors 4/14
DOMIGAN, TONY GETPROG
Use the Model 4P expansion memory in the Model III mode 7/19
EPP, LYMAN MOVING THE MODEL 4 ROM INTO RAM
Another way to to it 6/22
FELONG,C. & BROWN,K. A 6809 ASSEMBLER IN MICROSOFT BASIC
Assemble 6809 machine language on the Models I/III/4 3/6
FRANSEN, PAUL BOOT/BAS
Change the banner on a NEWDOS/80 system disk 7/21
HANLIN, T. & LEWIS, D. MERGELINE
Allows easy merging of BASIC program lines 6/12
KELTERBAUM,JOACHIM A NEW OVERLAY MODULE FOR NEWDOS/80 VERSION 2
JKL with TRS-80 graphics for Epson RX80/FX80 4/6
RANDOM I/O */*
KELTERBAUM,JOACHIM DIRSLOT/CMD
NEWDOS/80 utility puts file in specified directory slot 6/8
KELTERBAUM,JOACHIM INTERRPT/CMD
Save core image of running program to disk for later restart 6/16
KOEHLER, ROBERT B. LOWER CASE DESCENDERS FOR LINE PRINTER VII AND DMP-100
Software fix prints text with true descenders 6/13
MacSWAIN, GORDON MODEL 100 ASSEMBLY LANGUAGE FILE TRANSFER PROGRAM
Transfer files between Model 100 & Model I/III/4 4/12
McGLUMPHY & DECKER COMPDIR REVISITED
BASIC program compares two NEWDOS/80 directories 4/22
McGLUMPHY, DAVID R. ASCII FILE ENCODING/DECODING PROGRAM
Keep fact you may have herpes a secret from prying eyes... 5/17
McGLUMPHY, DAVID R. TELEPHONE DIALER PROGRAM
Let your computer do the dialing 5/20
THE EXTERMINATOR 7/2
LETTER TO THE EDITOR 8/2
McGLUMPHY, DAVID R. LINE REVERSAL PROGRAM
Reverses a BASIC string. 7/4
McGLUMPHY, DAVID R. DON'T GIVE ME THAT LINE
BASIC routine to draw lines on video display 7/4
THE EXTERMINATOR 8/1
McGLUMPHY, DAVID R. BASIC TELEPHONE DIALER PROGRAM
Another phone dialer, this one in BASIC 7/18
OWEN, DAVE MAINTENANCE TIPS
Program to test Mod III Floppy Disk Controller 4/16
ROHDE, ARNE DR - A DISK ROUTE PROGRAM FOR NEWDOS/80 VERSION 2
Expands NEWDOS/80's ROUTE command options 8/5
ROHDE, ARNE VCLIST - A VISICALC FORMULA LISTING PROGRAM
Lists to printer, disk file, or screen 8/15
SHIELDS, LAURIE VERFILE/CMD
Verifies a disk file (checks for bad sectors, etc.) 5/7
ZAJAC, RON NEWDOS/80 SCREEN DUMP TO DISK FILE PROGRAM
Uses "456" keys to dump "snapshot" of screen to disk 5/4

TIPS AND TECHNIQUES

ADELAIDE Micro-User COLOUR COMPUTER WOES
How NOT to sell CoCo programs outside North America 4/13
ASHLEY, KEN MODEL 100 HINT
Precede text with "@" to prevent lost characters 4/14
AUERBACH, ALEX A ZERO COST HARDWARE PERIPHERAL
Printing one-up labels with non-moveable pinfeeds 4/22
BARTO, R. NEWDOS/80 TIP
Change the default disk name from "NOTNAMED" 3/5
BENNETT, GLENN L. * JUMP TO 0000H TO REBOOT?
Use HALT instead to reset DD board 1/7
BOWER, DAVE SOME HELPFUL STUFF
Lots of information for Model 4 users 8/18
BOWER, DAVID H DOSPLUS HINTS
For users of both single- and double-sided diskettes 4/13
BROWN, BILL LINE EDIT FUNCTION & DEF FN
Disk BASIC line editor function 2/21
BRUMLEY, BOB * BOB BRUMLEY'S HACKER HINTS
Tips for Mod I, Mod 4, NEWDOS/80 users 4/14
COOK, MERRILL * A GREEN CRT FOR YOUR TRS-80
Maybe for as little as $40? 4/15
LETTER TO THE EDITOR 6/4
CRAMER, JOHN * CP/M PUBLIC DOMAIN DISKS
Details of how to get them from John 6/22
DECKER, JACK NOTES FROM ALL OVER
Includes Model 4 upgrade info 1/10
DECKER, JACK FIXING A "FINGERPRINTED" DISK
Remove a fingerprint from the shiny part 2/4
DECKER, JACK PAINT
Paints supposed to match Tandy Grey 2/4
DECKER, JACK TRSDOS 6.0 HINT (UNVERIFIED)
Is "LSIDOS" really the password? 2/6
DECKER, JACK VIDEO4 UPGRADE AVAILABLE FREE
If you got an early version! 3/2
DECKER, JACK MOVING MODEL 4 ROM INTO RAM
Put BASIC in RAM, then fix it up... 3/2
THE EXTERMINATOR 5/2
FOLLOW UP ARTICLE 6/22
DECKER, JACK 8080/8085 TO Z80 CONVERSION CHART
Convert mnemonics in assembly code listings 3/17
DECKER, JACK DEFEATING CALL WAITING
Don't let your MODEM calls be interrupted 4/16
DECKER, JACK CHRISTIAN-ORIENTED BULLETIN BOARD SYSTEMS
Four BBS's that have a Christian orientation 5/16
DUNHAM, HOPE LETTER TO THE EDITOR
Make TRSDOS 6.1.1 recognize more than two floppy drives 7/2
FILIPIAK, ROBERT THE OTHER TYDBYTE
Date routines-convert an 8 byte date string to 2 bytes 8/16

GREET, RAY FOR/NEXT LOOP TERMINATION
How and why to terminate FOR/NEXT loops properly 8/22
HALLDEN, DON * MODEL III DISK BASIC BUG
A bug in BASIC's FIELD statement 4/20
LETTER TO THE EDITOR 8/3
HANABURGH, DAVE HELP FOR A TIRED THUMB
Fix for the PLAY button on CTR-80 recorder 2/22
HESTER, VERN * ANOTHER BUG IN THE MODEL I/III BASIC ROM
An INC HL at 0EF2H should be a RST 10H 8/14
JONES, MORRIS MODEL III TRSDOS 1.3 BUG
Attention assembly language programmers... 2/10
THE EXTERMINATOR 4/2
KELTERBAUM,JOACHIM CREATE A SELF-BOOTING DISKETTE USING NEWDOS/80
Make your own self-booting disks 4/11
THE EXTERMINATOR 6/2
MODEL III SELF-BOOTING DISK USING NEWDOS/80 */*
LOGAN, LES * DISKETTE TIP
Why alignment disks don't have hub rings 3/17
MCTUG Newsletter MODEL 4 MISCELLANEOUS
Setting FORMS under TRSDOS 6 & MBASIC hints 2/20
MCTUG Newsletter MODEL 4P BOOT MODE SELECTION
How holding keys depressed affects the 4P boot 4/15
McGLUMPHY, DAVID R. LETTER TO THE EDITOR
Review of R/S calculator & NEWDOS/80 tips 8/2
NEWS RELEASE ONLINE TELECOMPUTING, INC.
Electronic information service available via (800) number 6/21
PEEK, BILL * BACKING UP A COLOR COMPUTER DISK USING SUPER UTILITY PLUS
You can do it on a Model I/III/4 6/15
PORRECA, TIM RETRIEVING KILLED NEWDOS/80 FILES
How to "unkill" a NEWDOS/80 file 2/15
ROHDE, ARNE LETTER TO THE EDITOR
Added commands for NEWDOS/80, EDAS 3.5 fixes and more 6/4
SALSBURY&WILLIAMS * A NEW MODEL 4 ROM?
First we'd heard. Unfortunately, it was true. 6/7
SMALL, GREG NEWDOS/80'S PARAMETER SCANNER
How to use it in your programs 4/8
LETTER TO THE EDITOR 8/3
SMITH, J. LETTER TO THE EDITOR
Eliminating "garbage collection" & Mod 4 added memory 4/20
SNIVELY, PAUL FORTH BULLETIN BOARD SYSTEMS
BBS support for Forth fans 4/8
SNIVELY, PAUL DETERMINING THE DOUBLER IN USE ON A TRS-80 MODEL I
How your software can tell what's in use 4/15
THE EXTERMINATOR 5/2
SNIVELY, PAUL LETTER TO THE EDITOR
Addresses of similar routines in Mod III vs. Mod 4 BASIC 6/2
TAVERNIER, CLAYTON COLOR COMPUTER AND MODEL I/III COMPATIBILITY
Copying CoCo disks on a Model I/III 1/8
THE EXTERMINATOR 3/1
TERRELL, WILLIAM F. READ IBM DISKETTES WITH SUPERZAP
How to make SUPERZAP read an IBM-PC diskette 4/20
LETTER TO THE EDITOR 6/4
VARIOUS AUTHORS MISCELLANEOUS MODEL 4 INFORMATION
Info on TRSDOS 6.0, MEMDISK, FORMAT, and BASIC 3/20
WILLIAMS, GORDON * NEWDOS/80 PDRIVE SETTINGS
To read LDOS and DOSPLUS disks 3/21
THE EXTERMINATOR 6/1

ZAPS AND PATCHES

BRATE, DON * TRSDOS 1.3 PATCH PROGRAM
Installs your choice of zaps to TRSDOS 1.3 6/19
THE EXTERMINATOR 8/1
CINTUG Newsletter OPTIONAL NEWDOS/80 VERSION 2.0 ZAPS
Zaps for PDRIVE, Series 1 EDTASM, and FORTRAN pkg 6/15
THE EXTERMINATOR */*
DECKER, JACK COMPDIR 2.0 BUGS
Corrections to old version of COMPDIR/BAS 2/2
DECKER, JACK NEWDOS/80 HINT
Reading a disk made in a "flippy" drive 2/6
DECKER, JACK MODEL III PRINTER DRIVER BUG
This one might get you if you LPRINT long lines! 2/10
DECKER, JACK ANOTHER VIDEO4 PATCH
Fix TRSDOS 1.3 directory for 24x80 display 3/2
DECKER, JACK TRS-80 ROM ROUTINES DOCUMENTED UPGRADE
If you got a bad copy, return it for a good one!! 3/5
DECKER, JACK ZAP TO DISABLE LDOS 5.1 PASSWORD CHECKING
(SYS2/SYS,2,19 from 28H to 18H) 3/15
DECKER, JACK ALDATE
Patch ALLWRITE'S ALK/CMD-puts date in DATE/TXT 7/15
DECKER, JACK NOTES ON THE NEW VERSION OF TASMON
Includes source code for new INSTALL/CMD program 8/13
DIME Newsletter VISICALC FOR THE MODEL I AND MODEL III
Patches to run version VC-150Y0-T83 under NEWDOS/80 2/21
DOMIGAN, TONY KBE PROGRAM ZAP
Fixes DVRIII64/CMD to work on Model 4 4/5
DOMIGAN, TONY TRSDOS 1.3B PATCHES
Install "R"epeat command similar to NEWDOS/80 4/15
DOMIGAN, TONY NEWDOS/80 VERSION 2 ZAPS
Various zaps for Models I/III/4 4/18
DOMIGAN, TONY MAKING AN AUTO-BOOT-LOADING NEWDOS/80 V2.0 SYSTEM DISK FOR THE MODEL 4P
So you don't have to load the MODELA/III file separately 7/14
DOMIGAN, TONY PATCHES FOR TAS PROGRAMS TO WORK ON THE MODEL 4P
Patches for KBE and the old version of TASMON 7/18
DOMIGAN, TONY VIDEO4 PATCH FOR THE MODEL 4P
Patches for a previous version of VIDEO4 7/20
DOMIGAN, TONY DATE/ASM
Patches Mod III NEWDOS/80 date format to DD/MM/YY 7/21
DRATCH, BOB PATCHES TO ALLOW SOME PROGRAMS TO WORK WITH THE MODEL III VERSION OF NEWDOS/80
FORTRAN package & Series I EDTASM 3/21
EPP, LYMAN NEWDOS/80 ZAP
Lets you boot up at 4 MHz speed on Model 4 4/18
HALLGREN, JOHN TRS-80 TIDBITS, TRASH, TREASURE, AND TRIVIA
Zaps for TRSDOS v. 1.3, 2.7DD, 2.8, & NEWDOS/80 v.1.0 4/3
THE EXTERMINATOR */*
HARRINGTON, NATHAN DEFEAT BACKUP PROTECTION ON TRSDOS 6.x DISKS
Remove the protection on some Radio Shack products 4/21
KRUMHOLZ, TRUMAN NEWDOS/80 VERSION 2 MODEL I PATCHES
A pair of zaps you might find handy 6/15
MCTUG Newsletter MODEL 4 MISCELLANEOUS
Patches to CLICK/FLT and COMM/CMD 2/20
MIKESELL, LESLIE * MODEM80 PATCH
Makes MODEM80 drop DTR and RTS before exit to DOS 6/18
McCLURE, DALE UNKILLING FILES IN TRSDOS 1.3
A one byte patch makes it possible 2/15
PEARSON, EDWARD A. MODEL III SCRIPSIT PATCH
Lets it read 1500 baud tapes 4/18
POORMAN, DAN PATCH FOR STRINGY FLOPPY TO TASMON
Lets TASMON 2.12 work with Exatron unit 2/19
PRICE, TOM PATCHES FOR TRSDOS 2.7 DD
Fixes some bugs in that operating system 2/10
SALSBURY, NATE A PATCH OF A PATCH
Convert cassette EDTASM+ to run on Mod III disk 4/9
LETTER TO THE EDITOR 6/4
SALSBURY, NATE MODEL III BASIC PATCH
Fixes a bug in TRSDOS 1.3 BASIC/CMD 4/11
SALSBURY, NATE TASMON PATCH FOR MODEL 4
Fixes old version of TASMON to work on Model 4 4/21
SCHULZE, MIKE * PATCHES FOR ALE RUNNING ON THE LNW-80
Enables CONTROL and defines ESCAPE as F1 8/14
SNIVELY, PAUL MULTIDOS 1.6 PATCHES
All kinds of good stuff for Multidos users 4/15
UNKNOWN MAX-80 PATCH FOR TASMON
Permits old version of TASMON to run on MAX-80 3/21
UNKNOWN MACRO-80 PATCH
Patches to make M80/CMD and L80/CMD run on Model III 8/4
VAN DAM, JOOP NEWDOS/80 MODEL I DISK ALARM
Zaps SYS0/SYS to catch unauthorized users of your DOS 6/19
VARIOUS AUTHORS MISCELLANEOUS MODEL 4 INFORMATION
Patches for TRSDOS 6.0 and COMM/CMD 3/20
WHITTAKER, JIM RSCOBOL PATCHES
Lets you use the program on other DOS's 8/17

SPECIAL LIMITED-TIME OFFER FROM THE ALTERNATE SOURCE!

NEW TOSHIBA P1340 PRINTER BREAKS THE PRICE BARRIER!

Until now, most of us have had to choose between ither a dot-matrix or a daisywheel printer to serve ur printing needs.

A DAISYWHEEL printer offers print quality that is indistinguishable from that of a typewriter, but is more expensive to operate (you must replace the daisywheel printing elements frequently) and is usually slower in operation. In addition, you cannot easily change type styles (for example, from a normal type style to italics) in mid-paragraph, because you must stop and physically change the print wheels to change type styles. And, of course, a daisywheel printer is incapable of printing graphics.

A DOT-MATRIX printer overcomes most or all of the disadvantages of a daisywheel printer, but has one big problem of its own: The print quality is not nearly as good! It is very easy to tell that the printed copy came from a computer and not from a typewriter.

Some folks purchase two printers (one daisywheel and one dot-matrix) to get the best of both worlds, but there's a better (and considerably less expensive) solution: The new TOSHIBA P1340 printer. The P1340 is a dot-matrix printer, but unlike most dot matrix printers which use only seven or nine pins to printer a character, the P1340 has a print head that contains 24 overlapping pins. This means that the average reader simply cannot tell the difference between printout produced by the P1340 and the output of a typewriter or daisywheel printer. Yet printout speed is up to 50% faster than a typical daisywheel printer. For very fast printing, a high-speed draft quality mode can be used. And you can print graphics using 0x180 dots per inch!

Advanced features include software selectable multiple fonts, pitches and line spacing, plus proportional spacing. Compressed print allows you to put the 132-column format on easily-handled letterhead size paper. Graphics are dot-addressable.

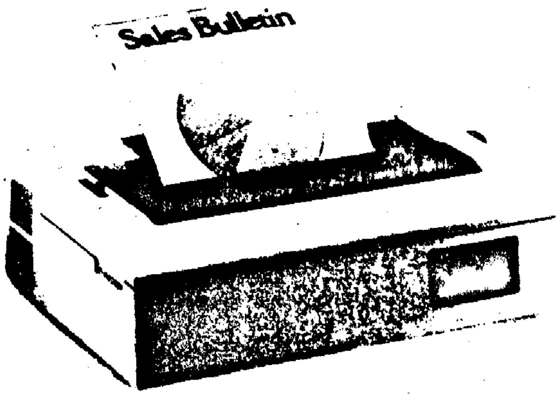
A printer is useless if your software won't run it, so the P1340 offers a "Qume SPRINT 5" emulation mode, making it compatible with most word-processing programs.

Compare the specifications below with any other printer with similar features (we'll bet you won't find one for under $1000), and you'll realize what a bargain this printer is at only $795 (that's $200 off the suggested list price!). User group discounts are available for quantity orders (contact us for details). But hurry - we must receive your order by 4:00 P.M. E.S.T. on Friday, January 25, 1985 in order to give you this printer at this special price.

One other note - if you need a wide carriage that will handle paper up to 15" wide, or would like to be able to "download" fonts of your own design to the printer, we also offer the TOSHIBA P1351, the "big brother" of the P1340. The P1351 is even faster than the P1340 and also has the "Qume SPRINT 5" emulation mode. The P1351 lists for $1895, but we offer it at $1545, a savings of $350. An automatic sheet feeder is available for the P1351 for $795 extra. Call or write for more information on the P1351.

THE ALTERNATE SOURCE
704 North Pennsylvania Avenue
Lansing, Michigan 489
Telephone: (517) 482-8270

```
                                                  P 1 3 4 0
For letter-qual   1.34     1.26     Q U A L I T Y
Toshiba P1340 prin 3.74    3.54      P R I N T E R
unique high densi 1.89     1.55    Letter Quality
dot-matrix print  6.97     6.35    High Speed Draft
lets you print cri                 Compressed Print at 16.7 CPI
letter  copy  at  2.45     2.16    180x180 Dots/in.2
draft copy at 144 5.75     5.02    10 Pitch - ABCDEF
with software-se   .68      .44    12 Pitch - ABCDEFGHI
fonts, you get to 8.88     7.62    Proportional-spaced
a variety of type 15.85   13.97     ◣Block Graphics◢
```



Specifications

Print Method	Impact wire dot matrix
Print Head	24 pin, overlapping Pin-replaceable for repair
Print Modes	Letter quality: 54 CPS Draft quality: 144 CPS (12 CPI) 120 CPS (10 CPI) Condensed print: 78 CPS (16.7 CPI)
Line Lengths:	132 CPL (16.7 CPI) 96 CPL (12 CPI) 80 CPL (10 CPI)
Character Set	96 ASCII character set (Std.) Letter-quality: 24 x 36 dot matrix (proportional spacing) Draft quality: 24 x 9 dot matrix Condensed Draft quality: 24 x 9 dot matrix
Horizontal Spacing	10 CPI, 12 CPI, Condensed 16.7 CPI, proportional spacing
Vertical Spacing	6 LPI, 8 LPI, etc.
Line Spacing Speed	37 LPS (for 6 LPI), etc.
Print Char. Code	ASCII code, extended
Paper	From 4.5" to 10" wide. Single sheet or continuous forms
Copies	Up to 3 parts, including original
Paper Feed	Friction feed and pin feed are standard
Ribbon	Cartridge ribbon: black
Operating Environment	Temperature: 41 to 104°F (5 to 40°C) Humidity: 20 to 85% RH
Power Requirement	115 VAC, 60 Hz
Dimensions (WxHxD)	16.5" x 5.9" x 11.6" (296 x 150 x 420 mm)
Weight	22 lbs (10 kg)
Interface	Parallel interface or serial RS-232C interface**

** PLEASE NOTE: We ship all printers with PARALLEL interface except when SERIAL interface is specifically requested

UPDATE 10:
A sneak preview...

Listen up, True Believer! The world's premier comics company has joined with the originator of Adventure games to bring you the awesome QUESTPROBE Series: an epic group of home-computer Adventures by Marvel Comics and Scott Adams in which YOU become the greatest Marvel Super Heroes (TM) characters. The TRS-80 version of the first QUESTPROBE is now ready for Model I, III and 4 in the III mode.

Much more than an arcade game and recommended by educators around the world, an Adventure allows you to move through a wonderous electronic world by entering English commands. In the HULK (Volume I of the series) you actually control the intellect of Bruce Banner and the savage might of ol' "Jade-Jaws" himself, the HULK (TM). Your objective is two-fold: to unlock the riddle of the Chief Examiner and...to survive! Each installment of the QUESTPROBE is a complete adventure. The HULK was written by Scott Adams with artwork by Mark Gruenwald and John Romita. The HULK is available immediately for only $29.95.

***

Our Public Domain Library is expanding. Public Domain Disk Number 5 (AKA "PD 5") is ready and includes many programs from past issues of Northern Bytes and a complete Bulletin Board system by Mike Bernstein, MINIBBS!

As with all past volumes, PD 5 is a 40-track, single density, double sided "flippy" diskette and sells for $10. Multiple copies (5 or more) are available for only $6 each. Programs include: DIRSLOT/ASM, /CMD and /DOC, DR/ASM, /CMD and /DOC, INTERRPT/ASM, /CMD and /DOC, MINIBBS2/BAS, /DOC, /JCL, /ASM and /CMD, MOVE/SRC, UCLIST/BAS, UCLIST/DOC, XMODEM30/CMD (part of MINIBBS), ALDATE/ASM and /DOC, BOOT/BAS, DIALER/BAS, FILTOMCI/BAS, MCIPGMS/DOC and MCITOFIL/BAS (for MCI mail users), LINE/BAS and /DOC and 9 other files.

Note to Model I owners: If you have the old Radio Shack 35-track drives, you may have problems reading the 40-track diskette. Before ordering, you should make arrangements for transferring from 40 to 35 track diskettes.

Back volumes 1-4 of the Public Domain Library are available, only $10 each.

***

Just when you thought that Model I and III were dead, behold a pho rises from the ashes. Models 4 and 4P are proving quite popular school systems, software developers and other persons who care to closely investigate the actual value of their computing dollar. Rumor has it that the 4 and 4P have already sold more machines than Models I and III combined. This means that the volume of software for these machines is increasing. Wouldn't it be nice if you could run these programs on your Model I and III?

You can. Dennis Allen, author of BAS34, BASANYO and BAS2000 (Soon to be BASGW, since it works for all Gee-Whiz BASICs) has produced one additional package (with some prompting from some die-hards here at TAS): BAS43. With this neat package, Model I and III users can convert ASCII BASIC programs written with Model 4 BASIC to run with their machine! While it is impossible for the machine to make all decisions necessary for this conversion, we take care of 90% of the gruntwork, including the conversion to 2 character variables (a special XLATE program is included in case you want to manually assign specific variable names), PRINT @, TAB, USING, EXPONENTIATION and more. A cross reference is created detailing all BASIC reserved words incompatible with the Model I/III BASIC, and the line number where they occur.

BAS43, with a 30+ page manual detailing what BAS43 will convert (and not convert) and many programming tips and suggestions, is available immediately for only $49.95.

Order from:

The Alternate Source
704 North Pennsylvania Avenue
Lansing, MI 48906
(517) 482-8270

NORTHERN BYTES

c/o Jack Decker
1804 West 18th Street
Lot # 155
Sault Ste. Marie, Michigan 49783
MCI Mail Address: 102-7413
Telex: 6501027413
(Answerback: 6501027413 MCI)

Bulk Mail
U.S. Postage Paid
Permit #815
Lansing, MI

POSTMASTER: If undeliverable return to:

The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906

To: