NORTHERN BYTES

Volume 6 Number 8

GREETINGS! Welcome to the final issue of Volume 6 of Northern Bytes. Will we begin a Volume 7? Yes, if you'll help support us. More on that later, but first this IMPORTANT ANNOUNCEMENT:

ATTENTION COMPUTER CLUBS AND USER GROUPS:

I am about to begin my more-or-less annual purge of our exchange newsletter mailing list. If your group is receiving NORTHERN BYTES on an exchange basis (this will be indicated by the words "NEWSLETTER EXCHANGE" at the top of your mailing label), please take a moment to check your group's mailing list (or to notify the person who maintains the mailing list for your group) to make sure that NORTHERN BYTES is on the list under the following address:

NORTHERN BYTES
c/o Jack Decker, editor [this line is optional]
1804 West 18th Street # 155
Sault Ste. Marie, Michigan 49783-1268

Please note that some groups still have us on the mailing list under an incorrect address (wrong street or lot number, etc.) while others are still addressing our copies to Microcomputer Users International, a computer club that ceased to exist about 2½ years ago. This causes no problems unless there is a substitute mail carrier, in which case it is anybody's guess as to whether we'll ever see your newsletter. And hardly anyone is using our nine-digit zip code yet, so for that reason alone you should pull up our entry in your mailing list file and check it against the above address. Don't just assume that we're on the list, mailing list files sometimes get corrupted by all sorts of strange forces (last year several groups expressed great surprise that we had somehow been removed from their mailing lists. They thought we were on their lists; they were WRONG!).

Also, the above address is the only acceptable address to use for exchange newsletters. If, by some chance, you've been sending your exchange newsletters to another address, it doesn't count!! They must be sent to the Sault Ste. Marie address.

The point is that if your group is NOT sending us copies of your newsletter (or if we are not receiving them due to an improper address), you will be dropped from the newsletter exchange list (meaning you won't get free copies of NORTHERN BYTES anymore) AND you will not be listed in the club/user group listing that I (hopefully) plan to publish in the next issue (or maybe the one following that, depending on the timing between this issue and the next). Therefore, if you discover that your group has not been sending us copies of your newsletter, it is imperative that you contact us IMMEDIATELY if you wish to remain on the exchange list. If your club or group is not publishing a newsletter at this time, please drop us a line anyway if you want to be included in the club/user group listing.

Last year I sent postcards to all the clubs we were about to drop from the list, but that took a lot of extra time and effort on my part, so I am NOT going to do it this year. What this means is that if you are reading your group's copy of this newsletter, it is incumbent upon YOU to check with whoever maintains your group's mailing list to make sure that they have seen this notice. If I don't hear from your group, don't come crying later if you fail to receive issues of NORTHERN BYTES and/or you miss being included in the listing. As the saying goes, "If you snooze, you lose!"

TOLL-FREE SIGNUP FOR NORTHERN BYTES

We've often said that we need more readers if NORTHERN BYTES is to continue publication, and that's still true. Many of the early sources of TRS-80 related information have dried up. Right now you can count the publications that offer any meaningful support for the TRS-80 Models I/III/4/4D/4P on one hand, and probably have a finger or two left over. We know that there are many thousands of TRS-80 owners still out there, and even discounting the non-technical types that wouldn't appreciate the content of NORTHERN BYTES, we feel that we should have many more readers than we currently have.

We've tried to make getting NORTHERN BYTES as risk-free as possible - instead of asking you to commit money for a "subscription" in advance, we bill each issue to your Visa or Mastercard as it is published and mailed to you. This system has worked fairly well for a few hundred readers, but we know that there are many more of you that probably have been meaning to send us your credit card number, but never got around to it. Well, now you can get on the mailing list as easy as picking up a telephone and dialing a toll-free number. The numbers to use are as follows:

Within Michigan: (800) 632-7818, extension 700
Elsewhere in U.S.A.: (800) 253-3200, extension 700

When the operator answers, give her the "extension" number and she will then take your order. You can also order merchandise from The Alternate Source using these numbers. PLEASE NOTE, however, that these numbers ring into a telephone answering service in Kalamazoo, Michigan. The operators there are not computer people. They can't answer technical questions, they just take orders. When you are ordering something with special considerations (for example, you want a certain product only if it runs under a specific DOS), you can leave this information as a short message to accompany your order. However, you must provide all necessary information, the operator will NOT prompt you for it. If you are ordering TAS merchandise, please be prepared to give your computer make and model, and the name of the Disk Operating System that you'll be using with the products you've ordered. And, be sure to have your credit card number and expiration date handy (at the rates we're paying for this service, we sure don't want to have them wait while you look for this information!).

If you need technical information about a product, you'll still have to call the main TAS number, (517) 482-8270 (on your nickel, of course).

Last summer I said that I would pull the plug on NORTHERN BYTES if the readership didn't increase. Well, it has increased, but only slightly. I'm hoping that the toll-free number will be the catalyst that will help us get on a profitable basis. If this doesn't work, you can expect the remaining days of NORTHERN BYTES to be short-lived. We've tried just about everything else (that we can afford, anyway).

NOTE TO OVERSEAS READERS:

If you live outside the U.S. or Canada, you may have experienced unexpected delays in receiving NORTHERN BYTES Volume 6, Numbers 5, 6 and 7. We outlined the reasons for this in our last issue. In any case, if you have failed to receive any of these issues, please let us know as soon as possible and we will send replacement copies. An airmail postcard will be adequate to inform us, you don't need to write a formal letter. We'd also like to know if you've experienced unusual delays in receiving the mailed copies. Our apologies for any inconvenience this may have caused.

MAIL TO THE EDITOR

I've often said this before, but it really pains me that I am unable to answer each piece of mail I've received individually. So, if you've sent me anything recently, be it a letter, disk, tape or whatever, I thank you, and apologize if you didn't get a personal reply. Unfortunately, I am still a relatively slow typist, and I just don't have 36 hours per day to answer correspondence. I hope that no one feels slighted. Sometimes I don't reply because I don't have an answer at hand, other times because the reply would need lengthy research that I do not have the time to do at the moment, and at other times it may be that

no reply really seems necessary. Also, there's such a thing as your letter just plain arriving on a bad day - such as a day when I'm away from home for a week or so. When I get back from a trip like that, I usually have a mountain of mail waiting, and each piece somehow seems less significant than usual.

If you REALLY want an answer to a letter you're sending (or want me to see the program or article you have submitted), the absolute WORST thing you can do is to send it c/o The Alternate Source (unless you are sending it to one of their electronic mailboxes on MCI Mail, Compuserve, or Delphi, which is actually a very good way to send letters or programs to me). There are several reasons for that, but probably the easiest to explain is that when I get back home after a visit to TAS, I usually have a whole box full of things to work on, PLUS I have all the mail that has accumulated here while I was gone. I guess I just somehow figure that if someone couldn't be bothered to send something directly to me, I'll look at it after I get through with the pile that has accumulated here. That usually takes me about four or five days (including processing time), during which that box of things from TAS somehow seems to dim in importance. If you're lucky, I may get to it a month or two later. On top of all that, if it's a disk I generally don't know all the circumstances surrounding its arrival (was there a letter or instructions with it that somehow got misplaced?), so I am a bit more cautious about it (read: disinclined to use it in NORTHERN BYTES or on a PD Library Disk).

I again apologize for not being able to answer every letter I receive. Some publications just say that they don't answer individual letters and never tell you why. I try to answer a few of them, and to let you know why I can't answer them all!

SPECIAL THANKS

You may notice that this issue contains several article reprints from other newsletters. Most of these are from Australia and came to us through the efforts of Tony Domigan and the editors of the newsletters involved. Unfortunately, Tony has been lured into the world of IBM clones and MS-DOS, so he expects to be doing less with his TRS-80 (in fact, by the time you read this, he will have probably sold his Model 4P). Tony's efforts have improved the content of NORTHERN BYTES in many ways, and for that I can only say, "Thank you, Tony." It's always sad to lose a reader to the Big Blue camp (and to what many TRS-80 users consider a more user-hostile Disk Operating System), but especially sad when that reader has contributed so much to our struggling efforts.

I sometimes wonder if the desertion from the TRS-80 camp isn't as much media hype as anything. There are many satisfied TRS-80 users out there, and the TRS-80 is a fine machine that runs some excellent software. In fact, for non-graphics-intensive work that does not require horrendous amounts of memory, I don't see how you can beat the TRS-80 Model 4. You might wonder what kind of computer I'd use if not the TRS-80. Well, I've never had any real motivation to work with other computers since my TRS-80's do just about everything I want them to do, but based on reviews I've read and reports I've heard from users, I'd probably consider the Commodore Amiga very strongly. But only if they lower the price, and get rid of some of those horrible commercials they've run on TV (which, mercifully, have not appeared very often!).

By the time you get this issue we'll probably be into the new year, so I'll just wish you all the very best for 1986. Maybe this will be the year they bring out the voice-operated typewriter interface. I keep hoping...

THE EXTERMINATOR

In the article "STANDARD SORT ROUTINES" by Ron Zajac, which appeared in Volume 6, Number 5, part of line 1070 of the SHELSORT demonstration program was missing. I was testing some zaps to Allwrite at the time, and didn't realize that my zaps were occasionally causing text to be dropped at the end of video lines. If you see anything else that appears a bit weird in that issue, that may be the cause. You know how we always tell people to test all zaps thoroughly before using them with files that contain valuable data? Well, I guess we should take our own advice!

Anyway, here's line 1070 as it should have appeared:

```
1070 GOSUB 1140: IF A(S)>A(S+M) THEN B=A(S): A(S)=A(S+M): A(S+M)=B:
IF S-M>=1 THEN S=S-M: GOTO 1070
```

Sorry for any inconvenience this may have caused, but of course it was a "computer error"...

LETTERS DEPARTMENT

Reminder: Persons sending letters intended for publication should send them on magnetic media or via Compuserve, Delphi, or MCI Mail (especially if longer than a couple of paragraphs). If you are NOT using Allwrite (or Newscript) and your word processor offers the option to save your file in ASCII format, please do so (especially if using SuperScripsit!). Your cooperation in this matter will help us to bring you a better newsletter!

Dear Jack,

The Shell sort discussed by Ron Zajac in Volume 6, No. 5 appears to be a Shell-Metzner sort. I also have no reference, but I was told in a class that this is a modification by Marlene Metzner of a sort by Donald Shell. One or both of these individuals may have worked for Pratt & Whitney Aircraft, since the sort reportedly was originally published in one of that company's in-house bulletins. In line 1070 some information is missing from the fifth statement. Perhaps it should have read ". IF S>M THEN S=S-M "?

Sincerely, Don Singer

[Oops! Thanks for discovering the bug, Don, and for passing along the information. Hopefully, you've already discovered the correction in THE EXTERMINATOR column above. Your guess was close, though!]

Dear Jack,

For the first time in several years I have tried using the RS-232 port on the Model 4. I was asked to write a program to collect all incoming data on an IBM-PC, using hardware handshaking to control the transfer speed. I thought I could use one of the TRS-80 terminal programs for transmission, but found that they did not use hardware handshaking. I found that Omniterm can make transmission dependent on hardware handshaking, but does not set the output if it cannot cope on reception. I would have thought the Model 4 programs would use interrupts to implement a receive buffer, being configurable for either XON/XOFF or hardware handshaking. Probably this is not necessary for 300 baud 'phone communications, but for direct connection at up to 9600 baud it would certainly be convenient. I'll probably have something soon which will fit my specific needs - it just surprised me that it wasn't readily available.

Kindest regards,

Arne Rohde, Box 82-211, Highland Park, Auckland, New Zealand

[There are many terminal programs around that work fine at 300 baud, but have real problems when operating at higher speeds. I don't have any answers on this one, but wonder if our readers have any recommendations?]

[Message received via Compuserve:]
Date: 16-Nov-85 00:50
From: Keith E. Davis, M.D. [72256,3155]
Subj: Users Group

Dear Jack,

I enjoy receiving Northern Bytes very much. I would like to ask your help in contacting other users of Adventure International's "Maxi Manager II" program.

I would like to form a Maxi Manager Users Group with a newsletter and part of a BBS. If any of your readers are interested, have them send a S.A.S.E. to me at:

Keith E. Davis, M.D.
Maxi Manager Users Group
P.O. Box 609
Shoshone, Idaho 83352

or message on CompuCenter IOWA at (319) 338-2750 (BBS) to DOC (my username). Thanks.

Keith E. Davis, M.D.

[Glad to help, Doc!]

[Message received via Compuserve:]
Date: 16-Nov-85 14:20
From: John J. Stein [74056,673]
Subj: Northern Bytes

Jack,

Thank you for the copy of Northern Bytes. I couldn't help noticing that every article was for NEWDOS/80 people. Is this a coincidence? I use LDOS. Reading about NEWDOS/80 doesn't help me.

Also, although you have a huge selection of public domain software, it seems that I already have most of what you offer.

I'm still undecided about subscribing. I like the layout and style, but I would like more information about programs and tips for LDOS users.

John J. Stein

[Believe it or not, we appreciate John's comments regarding the content of NORTHERN BYTES. Although we have not made a conscious effort to exclude articles relating to LDOS, it does seem to be true that the articles we receive tend to be more relevant to NEWDOS/80 than to any other DOS. I suppose that is due to the fact that NEWDOS/80 remains the most popular DOS for Model I/III users. We'd love to be able to carry more articles relating to DOSes other than NEWDOS, but can only publish what we receive. Occasionally we do reprint articles from other newsletters that help fill the void (such as a recent article on creating a double-sided booting disk under LDOS), but in most cases we prefer to use articles that have been sent directly to us (even if they first appeared in another newsletter - the main thing is that if we receive an article on disk or via electronic mail, it doesn't have to be re-typed!).

Almost all of the articles in NORTHERN BYTES have been contributed by our readers (that is, we don't pay for them like the big magazines do), but we do give a free six-issue "subscription" to NORTHERN BYTES to anyone that writes an article for us (provided that we actually use it, of course). If any of our readers feel that they have some expertise with LDOS (or any of the other DOSes), it could be worth some free issues of NORTHERN BYTES to you to share your knowledge!

Keep in mind that when you "subscribe" to NORTHERN BYTES, you're not paying for ANY issues in advance. Thus, you can try a couple of issues and if you decide that NORTHERN BYTES isn't helpful to you, you can cancel your "subscription" at any time just by dropping us a line to that effect, with no penalty whatsoever!]

Dear Jack,

After reading your comments concerning LDOS and TRSDOS 6's "sudden death" in the latest issue of Northern Bytes [Volume 6, Number 6], I could not escape the feeling of, "here we go again". I can't recall how many times you have erroneously referred to a "go to sleep" mode as a "sudden death" mode, but its been many. I also fail to discern why your brilliant readership has not set you straight on what causes that condition and what you can do to correct it; unless they have and your unmitigated affliction to NEWDOS has caused you to refrain from informing your readers as to the solution.

I'll be charitable and assume no one else has brought the solution to your attention. Thus, I leave it up to myself to educate you and your readers.

I am sure that the hackers out there (which really is your audience) recognize that LDOS had its roots in VTOS version 4. VTOS 4 as was VTOS 3 as was TRSDOS 2.2, 2.1, 2.0, and the infamous version 1.0 originated from the individual known to history as Randy V. Cook (I'll omit the slashed ohs).

The actual phenomenon of "sudden death" was prevalent in TRSDOS 1.0 and 2.x series and was in part caused by a capacitor in the expansion interface which released the drive select lead of the floppy prior to the completion of the data transfer. Thus, the system would lock up because the floppy would be disengaged and the disk controller would never time-out. Dennis Kitsz, well known author of Model I hardware fame, noted this. That should not be news to anyone.

When Randy brought out VTOS 3.0, he added a feature of using the SHIFT-BREAK key combination to reselect the last drive - in case a motor timeout occurred. The other thing which Randy did was change the points in the disk driver where the the interrupts were disabled. Under TRSDOS 2.x, interrupts were disabled very early in the controller handshake protocol. This had the result of making the interrupt clock inaccurate. Thus, refining the point at which the interrupts were disabled led to a more accurate clock.

When Randy released VTOS 4.0, things had to tighten up a bit more in the interrupt department since a type-ahead keyboard routine was now available. The interrupts were disabled only to the barest minimum in the disk driver. It was implemented that way to enable the capture of keystrokes while the disk drives churned away. What typist making use of typeahead is willing to stop typing when the drive activity light goes on? In order to gain a feature, one had to be given up. For the casual reader, here is where you really have to pay attention.

Because the interrupts were disabled in the disk driver only after the first byte was transferred to/from the controller, a possibility arose that an interrupt could occur between the transfer of that byte and the transfer of the second byte. It would take a considerable discussion of specifications of the Western Digital disk controller to understand why the disable interrupt could not be placed immediately preceding the transfer of the first byte. Suffice it to say that the timing between the transfer of the first and second byte is a variable; for proper operation of type ahead, the DI had to be where it was! Now when the interrupt servicing routines were off doing their thing, data was lost in the transfer between the host (the CPU) and the disk controller. The disk driver circumvented this obstacle by retrying the requested operation.

There are retries due to parity errors and there are retries due to lost data errors. Usually the parity errors are retried 5 or 10 times by the driver, then reported back as an I/O error to the caller. The VTOS 4.0 disk driver did an infinite retry of the "lost data" error since it was caused by the interrupted data transfer. Now the appearance of this problem was an occasional slightly longer time to do a data transfer - depending on the rotational speed of the floppy disk! It turns out that when a disk drive is precisely tuned to rotate at 300 rpm, it is perfectly synchronized to the interrupt clock of 25 Hz (Model I) or 30 Hz (Model III) or 60 Hz (Model 4). Thus, those folks who went out and adjusted their drives to rotate at exactly 300 rpm, found that the drives would more often than not, "go to sleep" when operating under VTOS. This phenomenon was carried through to LDOS and later to TRSDOS since both of these systems continued to support type-ahead.

Since Logical Systems has always told people of this phenomenon and always recommended the drives be aligned to 301 or 302 rpm, I fail to understand why people still hassle with this problem. Eventually, some "smart" drive manufacturer introduced a drive with a microprocessor controlled speed that was a rock solid 300 rpm. I believe it was that fiasco which forced Logical Systems to add the "SYSTEM (SMOOTH" function in TRSDOS 6.2. Such a function completely destroyed the ability to type while the floppy drives were doing data transfer; it did speed up the system by eliminating retries due to lost data errors. All "SYSTEM (SMOOTH" does is add a disable interrupt early in the disk driver.

The bottom line is that if you or any of your readers have a problem with a disk drive "going to sleep" using LDOS or TRSDOS 6 (I doubt there are still any VTOS users out there), I suggest that they just adjust the speed of their disk drives to 301-302 rpm. The problem will be solved.

Thanks for taking the time to read this - and I hope that you, Jack, have read it. You need the education it provides.

Respectfully, Roy Soltoff

[Roy, I do appreciate you taking the time to educate me. I had read something about this problem years ago, but had forgotten about it until I received your letter.

As for my "unmitigated affliction to NEWDOS" (sic), I think I have covered that point in my response to the previous letter, so I won't belabor it here.

However, in one way your response is typical of what I have come to expect from some LDOS users. Notice that here we have disk drives that are running at the "correct" speed - 300 RPM, right on the money. All other DOSes work just fine with the drives running at that speed. Only LDOS has a problem with drives running at the "correct" 300 RPM speed. The solution? Change your drive speed so it will work with LDOS! In other words, we would almost be led to believe that LDOS comes from somewhere on high and must therefore be perfect, and if any incompatibilities exist, all other software and hardware must be modified to work with LDOS! A disk drive manufacturer comes out with a drive that runs at a steady 300 RPM, and you refer to it as a "fiasco"! I suppose that the drive manufacturer really does deserve that negative evaluation - after all, how dare they introduce a drive that automatically runs at the correct speed, without first considering that it might not work properly with the almighty LDOS?!

I've noticed that same attitude in regard to software that runs just fine under all DOSes except LDOS. It can't possibly be a bug in LDOS (although it could be a "design feature", which is a bug that has been documented), so, Mr. Software Author, you'll just have to rewrite your program so that it works under LDOS, because after all, LDOS is the only real DOS anyway. Or so some LDOS users would have us believe.

I guess that I am used to having to "work around" the hardware limitations of my equipment, but perhaps the LDOS authors felt that they were above that effort. Granted that there are some hardware problems that can't be worked around (such as the lack of lowercase characters on the video display of the old, unmodified Model I TRS-80) that almost demand a hardware modification, no matter which operating system is used. But when you have a problem that a) occurs only under one DOS, and b) occurs when the disk drives are set to the "proper" speed, one wonders if it isn't just a bit arrogant to demand that users reset the speed of their disk drives just so they can run that one DOS. A typical programmer may not balk at making such an adjustment (though I know a few who would), but I'd hate to try and sell a program commercially that would run only under LDOS - I'm afraid most users wouldn't take kindly to being required to have their disk drives re-timed to use my program.

I fully understand the tradeoff involved here - the bottom line is that you can't have reliable type-ahead during disk drive operations without changing the drive speed a bit. However, the Model I version of ALLWRITE has typeahead (via the ALK driver) and it will not work properly during an automatic save to disk. So what? I simply stop typing while the save in in progress. Granted, sometimes it's a bit irritating, but I'd much rather be slightly irritated by having to stop typing for a few moments than very irritated by having the computer "go to sleep" on me with a buffer full of text.

So I guess it all boils down to what is most important to each individual DOS user. Personally, I had other dislikes about LDOS at the time I tried it (such as the gymnastics one had to go through to make a self-booting double-density system disk on the Model I, and such as the X'nnnn' format required for hexadecimal notation, whereas many other DOSes would accept the much simpler nnnnH format), so I decided to keep using a DOS that worked reliably for me and that I was comfortable using (NEWDOS/80). Now I am using NEWDOS/80 with Alan Johnstone's modifications, which allow me to directly read from and write to a TRSDOS 6 disk, so even the portability factor is not an incentive for me to use LDOS on my Model I. Others may believe that LDOS is the greatest thing since sliced bread, and indeed, it may be the best DOS for those people. I may prefer an four cylinder economy car while you may be more interested in high performance on the racetrack, and in that case it is clear that we would not both be happy driving the same vehicle. You may really appreciate the features of LDOS while I happen to prefer the benefits of using NEWDOS/80. That doesn't make either one of us wrong, it just means we have different opinions on what is important in a DOS.

Once again, I encourage our readers that have some expertise with LDOS to contribute some of that knowledge to the pages of NORTHERN BYTES. Who knows, you may point out a feature that I hadn't realized was available in LDOS, that might make me think about it in a lot more favorable light!]

Date: Wed Nov 27, 1985 6:29 pm EST
From: Paul Jaeger / MCI ID: 237-3942

TO: * Jack Decker / MCI ID: 102-7413
Subject: Floating point errors

Dear Jack,

Thanks for NB 6/6. I was especially interested in your article on floating point errors, so I though I would check my Fortran and Pascal as well. The problem is the same, with a few variations. I am including source files and sample output. The Alcor Pascal is better in its representation of the real numbers, but the integer still gets screwed up.

Microsoft Fortran (R/S #26-2219):

```
c bugtest/for

      a = 0.

      do 100 j = 1,20
         call doit(a,b,i,i1)
         write (1,10) a,b,i,i1
10       format (f11.8,2x,f11.8,2x,i5,2x,i5)
100   continue
      end

      subroutine doit(a,b,i,i1)
      a = a+.01
      b = a * 100.
      i = int(b)
      i1= ifix(b)
      return
      end
```

```
 .01000000   1.00000000      1      1
 .02000000   2.00000000      2      2
 .03000000   3.00000000      3      3
 .04000000   4.00000000      4      4
 .05000000   5.00000000      5      5
 .06000000   6.00000048      6      6
 .07000000   7.00000000      7      7
 .08000000   8.00000000      8      8
 .09000000   9.00000000      9      9
 .09999999   9.99999905      9      9
 .10999999  10.99999905     10     10
 .11999999  11.99999905     11     11
 .13000000  13.00000000     13     13
 .14000000  14.00000000     14     14
 .15000001  15.00000095     15     15
 .16000001  16.00000191     16     16
 .17000002  17.00000191     17     17
 .18000002  18.00000191     18     18
 .19000003  19.00000191     19     19
 .20000003  20.00000381     20     20
```

Alcor Pascal (R/S #26-2212):

```
program bugtest;

var   a,b:  real;
      i,i1,ctr:  integer;

procedure doit (var a,b:real; var i,i1:integer);
begin
      a := a + 0.01;
      b := a * 100;
      i := round(b);
      i1:= trunc(b);
end;

begin
      a := 0;
      for ctr := 1 to 20 do
         begin
            doit(a,b,i,i1);
            writeln(a:11:8,'  ',b:11:8,'  ',i:5,'  ',i1:5);
         end
end.
```

0.01000000	1.00000000	1	1
0.02000000	2.00000000	2	2
0.03000000	3.00000000	3	3
0.04000000	4.00000000	4	4
0.05000000	5.00000000	5	5
0.06000000	6.00000000	6	6
0.07000000	7.00000000	7	7
0.08000000	8.00000000	8	8
0.09000000	9.00000000	9	9
0.10000000	10.00000000	10	9
0.11000000	11.00000000	11	10
0.12000000	12.00000000	12	11
0.13000000	13.00000000	13	13
0.14000000	14.00000000	14	14
0.15000000	15.00000000	15	15
0.16000000	16.00000000	16	16
0.17000000	17.00000000	17	17
0.18000000	18.00000000	18	18
0.19000000	19.00000000	19	19
0.20000000	20.00000000	20	20

Keep up the good work.

Paul Jaeger

[Thanks for passing along this information, Paul. I would assume that Fortran and Pascal users can use some of the same hints mentioned in the article ("MICROSOFT BASIC AND FLOATING POINT NUMBERS", Northern Bytes Volume 6, Number 6, page 17) to minimize these errors, especially the technique of doing all computations using integers and carrying all variables as integers whenever possible.]

MODEL III TRSDOS 1.3 - THE BEST DOS !

by Jim Whittaker

[Reprinted from SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, Australia.]

A lot of people often talk about such and such a DOS being the best DOS and how they use one DOS exclusively. Well, I am familiar with 3 DOS's and I use each one regularly and in about the same proportion. As far as I am concerned, its horses for courses and some DOS's are great for doing work in some areas and lousy in others and if you don't exploit their benefits, a lot of potential power is going to waste.

NEWDOS/80 I use if I need to do any file manipulating or transferring. It has the best range of options for reading files, manipulating data and then writing either to the same file or a new file in different format.

LDOS I use when I need to handle a special file or print job where I can quickly and easily install a filter or link to another device (real or imaginary). The best area to demonstrate this is when MODEMING where the type ahead buffer and easy file handling allow hassle free operations.

TRSDOS I use with most of my standard Radio Shack programs. Cobol and Superscripsit are two large systems that are split into many smaller files for manageability. TRSDOS, being a "bare bones" DOS, is not very large and can thus accommodate all these programs and many of my own files on the one system disk. This allows "1 DISK" operation which I find is more convenient to use. WIFE/CMD also finds it is less hassle to find and boot a single disk.

I find under the other DOS's that although powerful, you pay for it with lack of user space on the system disk. Often, people transfer these programs to LDOS or NEWDOS, but there is no real advantage as you cannot use type ahead or filters or other enhancements and there is very little room left on the system disk for your own files. The Radio Shack programs use their own drivers and do NOT refer to the DOS routines. The other problem is that although the programs don't use the DOS, the DOS still thinks that it does and consequently everything runs a lot slower trying to interpret interrupts that are not used. Cobol and Superscripsit and the like were written specifically to run under TRSDOS and this is where they perform at their best.

To make life just a little easier, you will find below a whole heap of TRSDOS 1.3 system patches and enhancements that have been collected from here there and everywhere over the years. My compliments to the original authors (unknown) for their perseverance and foresight. You can either enter them individually from DOS Ready or put them in a DO file. I will upload them to the BBS as a DO file called TRSPATCH/BLD and they will be available at the club. You can change any of them by putting the file through Superscripsit's ASCII option, making the changes, deletions or additions, QUIT the document and reuse the ASCII option.

MAJOR PATCHES TO THE TRS-DOS SYSTEMS.

1. Disable Password checking in both PATCH and DEBUG.
```
PATCH *2 ( ADD=4ED4, FIND=20, CHG=18 )
PATCH *5 ( ADD=52EB, FIND=CB, CHG=36 )
PATCH *5 ( ADD=52ED, FIND=BE, CHG=00 )
```

2. Allow DEBUG to inspect below 5600H.
```
PATCH *5 ( ADD=4EDF, FIND=38E6, CHG=0000 )
PATCH *5 ( ADD=4F04, FIND=D0, CHG=C9 )
PATCH *5 ( ADD=506E, FIND=38E3, CHG=0000 )
```

3. Correct BASIC arrow functions (one or both may already be done).
```
PATCH BASIC/CMD ( ADD=58C4, FIND=D5, CHG=00 )
PATCH BASIC/CMD ( ADD=58F8, FIND=F1, CHG=00 )
```

4. Start directory searches from sector 3 not 2 on track 17.
```
PATCH *10 ( ADD=4E47, FIND=02, CHG=03 )
PATCH *10 ( ADD=4E2A, FIND=3ADA4E, CHG=784F00 )
```

5. Only zero the first two bytes in the directory when KILLing a file. This allows you to UNKILL them if need be.
```
PATCH *3 ( ADD=4FB5, FIND=2F, CHG=01 )
```

6. Correct an error in FORMAT an I/O retry counter error.
```
PATCH *7 ( ADD=4E55, FIND=12, CHG=0F )
PATCH *7 ( ADD=4E5B, FIND=0C, CHG=09 )
```

7.(a) Disable boot-up Logo.
```
PATCH *0 ( ADD=4E85, FIND=216C51, CHG=C39D4E )
```
(b) Disable both Date and Time prompts.
```
PATCH *0 ( ADD=4EA9, FIND=CA, CHG=C3 )
```

or selectively eliminate the boot up logo.

8.(a) Eliminate Model III picture.
```
PATCH *0 ( ADD=4E89, FIND=1B, CHG=27 )
```
(b) Eliminate message from "VERSION to SERIAL NUMBER".
```
PATCH *0 ( ADD=4E95, FIND=1B, CHG=27 )
```
(c) Eliminate TANDY copyrite message.
```
PATCH *0 ( ADD=4E9B, FIND=1B, CHG=27 )
```
(d) Disable Time prompt only.
```
PATCH *0 ( ADD=4EFE, FIND=215451, CHG=C32E4F )
```

9. Print long error messages instead of numbers.
```
PATCH *4 ( ADD=4E28, FIND=20, CHG=18 )
```

10.Remove periods from DOS READY prompt.
```
PATCH *1 ( ADD=4E78, FIND=2E, CHG=20 )
```

11.Change TRSDOS READY prompt (max 12 characters). Because of the length of this patch, I have split it into 2 lots of 6 characters. Just enter the new message in HEX format. NB a space = 20 HEX.
```
PATCH *1 (ADD=509C, FIND=545253444F53, CHG=535550455220)
PATCH *1 (ADD=50A2, FIND=205265616479, CHG=204D4943524F)
```

12.Change Track-Step rate. Use the table below to select the different rates of step and use them in the "CHG=" spots.

Current	6ms	12ms	20ms	30ms
0C	08	09	0A	0B
1C	18	19	1A	1B
58	58	59	5A	5B

```
PATCH *0 ( ADD=42EE, FIND=0C, CHG=08 )
PATCH *0 ( ADD=4516, FIND=0C, CHG=08 )
PATCH *0 ( ADD=4544, FIND=1C, CHG=18 )
PATCH *0 ( ADD=4FE1, FIND=0C, CHG=08 )
PATCH *7 ( ADD=580E, FIND=0C, CHG=08 )
PATCH *7 ( ADD=5841, FIND=0C, CHG=08 )
PATCH *7 ( ADD=5923, FIND=1C, CHG=18 )
PATCH *7 ( ADD=5B3C, FIND=58, CHG=58 )
```

13. Change Track Count. Mainly (but not limited to) to use 80 track drives. Select the count (in HEX) and use it in the "CHG=" spots.

Current	40T	42T	80T
28	2A	50
29	2B	51

```
PATCH *0 ( ADD=4926, FIND=28, CHG=2A )
PATCH *0 ( ADD=499B, FIND=28, CHG=2A )
PATCH *0 ( ADD=4B29, FIND=29, CHG=2B )
PATCH *2 ( ADD=4F61, FIND=28, CHG=2A )
PATCH *6 ( ADD=5C06, FIND=28, CHG=2A )
PATCH *6 ( ADD=5D53, FIND=28, CHG=2A )
PATCH *7 ( ADD=5203, FIND=28, CHG=2A )
PATCH *7 ( ADD=52D4, FIND=28, CHG=2A )
PATCH *7 ( ADD=53FF, FIND=28, CHG=2A )
PATCH *7 ( ADD=54D7, FIND=29, CHG=2B )
PATCH *7 ( ADD=5504, FIND=28, CHG=2A )
PATCH *7 ( ADD=554A, FIND=28, CHG=2A )
PATCH *7 ( ADD=5C4F, FIND=28, CHG=2A )
PATCH *7 ( ADD=5CD4, FIND=28, CHG=2A )
```

Now for some welcome additions to TRS-DOS.

14.This addition will halve the loading time of BASIC programmes.

```
PATCH BASIC/CMD (ADD=5BFE, FIND=2AA440, CHG=CD8754)
PATCH BASIC/CMD (ADD=5C07, FIND=FF, CHG=FE)
PATCH BASIC/CMD (ADD=5C0D, FIND=CD535F7723,
                 CHG=CD9B540000)
PATCH BASIC/CMD (ADD=53CC, FIND=8754, CHG=4A1E)
PATCH BASIC/CMD (ADD=5487, FIND=E17EFE26, CHG=2323E5DD)
PATCH BASIC/CMD (ADD=548B, FIND=C24A1ED7, CHG=E1ED5BA4)
PATCH BASIC/CMD (ADD=548F, FIND=C24A1EE5, CHG=40013300)
PATCH BASIC/CMD (ADD=5493, FIND=219B54CD, CHG=0901FF00)
PATCH BASIC/CMD (ADD=5497, FIND=3F56E1C9, CHG=EDB0EBC9)
PATCH BASIC/CMD (ADD=549B, FIND=35013D3C, CHG=DD7503DD)
PATCH BASIC/CMD (ADD=549F, FIND=26751734, CHG=7404DD36)
PATCH BASIC/CMD (ADD=54A3, FIND=263C3675, CHG=0500DDCB)
PATCH BASIC/CMD (ADD=54A7, FIND=3C267516, CHG=01EEDD34)
PATCH BASIC/CMD (ADD=54AB, FIND=1A050C07, CHG=0A2003DD)
PATCH BASIC/CMD (ADD=54AF, FIND=1C121D01, CHG=340B24C3)
PATCH BASIC/CMD (ADD=54B3, FIND=1011, CHG=535F)
```

15. Create a "R"epeat command similar to NEWDOS80.

```
PATCH *1 (ADD=4E36, FIND=2642013F003600,
                 CHG=8041013F000000)
PATCH *1 (ADD=5191, FIND=4D4153544552, CHG=522020202020)
PATCH *9 (ADD=603E, FIND=CD8E5FFE0D2003,
                 CHG=21804111254201)
PATCH *9 (ADD=6045, FIND=AF1829111060CD,
                 CHG=2500EDB0212542)
PATCH *9 (ADD=604C, FIND=5444C2, CHG=C39942)
```

16.Change the CONVERT command to allow Y/N/Q selection.

```
PATCH CONVERT/CMD (ADD=520A, FIND=06, CHG=1C)
PATCH CONVERT/CMD (ADD=5511, FIND=21955B, CHG=CD065A)
PATCH CONVERT/CMD (ADD=5536, FIND=ED53, CHG=135A)
PATCH CONVERT/CMD (ADD=5A06, FIND=4D6F64656C20,
                 CHG=3E86CD155A21)
PATCH CONVERT/CMD (ADD=5A0C, FIND=3120746F204D,
                 CHG=955BC0E1C39E)
PATCH CONVERT/CMD (ADD=5A12, FIND=6F64656C2033,
                 CHG=553E7032F153)
PATCH CONVERT/CMD (ADD=5A18, FIND=20436F6E7665,
                 CHG=CDED53C9436F)
PATCH CONVERT/CMD (ADD=5A1E, FIND=7273696F6E,
                 CHG=6E76657274)
```

MODEL 4 SETDATE UPGRADE
by Jack Decker

In NORTHERN BYTES Volume 6 Number 6 I published the Model 4 version of SETDATE, the ever popular date-setting routine that ends the drudgery of setting the date when you boot up your DOS system disk. Although version 6.0 did not contain any major bugs, I have released an upgrade (version 6.1) that includes a couple of minor improvements.

The first change eliminates the requirement that SETDATE/CMD be saved only under that filename on your disk drive. Now you may rename it to whatever you like. This might be useful in certain circumstances. Sorry, I can't do this in the Model I/III version due to the different ways that the various DOSes load and execute a program.

The other change is for people who want to call SETDATE from within another program or a /JCL file. Note that SETDATE was never intended to be operated from within a /JCL file. Rather, you should execute SETDATE first, then DO your /JCL file. An AUTO command line similar to this can be used for this purpose:

```
AUTO SETDATE DO YOURFILE/JCL
```

This will work (because SETDATE is specifically designed to allow this type of syntax) but I won't guarantee that running SETDATE from within a /JCL file will. Anyway, the specific complaint that I had was that SETDATE did not exit with the HL register pair containing a value of zero, which indicates an error to the calling program. This would not have any effect on those who run SETDATE from DOS READY, but would affect those who would try to run SETDATE from within a /JCL file or another program.

I will also mention that I now have SD (the version of SETDATE that also writes the date out to a file called DATE/TXT, which can then be imbedded into word-processing documents to automatically use today's date) up and running on the Model 4, thanks to a little help from Andy Levinson of Studio City, California (as long as I'm passing out credits, I should also mention that I received some help on the original version of SETDATE from Ray Erickson of Escanaba, Michigan). I'm not going to print the Mod 4 version of SD/CMD in this issue due to space limitations, but it will undoubtedly show up on a TAS Public Domain Library disk (and/or your favorite Bulletin Board System?) in the near future. Maybe when things slow down next summer, I'll try to squeeze it in.

Anyway, here are the changes you need to make to SETDATE6/ASM version 6.0 to produce an "official" version 6.1:

1. DELETE the following lines: 180, 1780, 2880, 4300, 4310.

2. CHANGE the lines shown below (some of the changes are cosmetic, I'll agree, but you do want an "official" version, don't you?):

```
00110 ;Version 6.1 for the Model 4 - creation date 12/14/85.
00190 ;Normally, this program is used by placing the
00530 ;                Sault Ste. Marie, Michigan  49783-1268
01510          DEFM    'SETDATE version 6.1 is a public domain program by Jack Decker.'
01550          DEFM    'owners and users of the TRS-80 Models I, III, 4, 4D and 4P.'
01620          DEFM    'Lansing, Michigan 48906-5319 (U.S.A.) or telephone (517) 482-8270.'
03050          JP      NC,GETKEY       ; than ASCII 1CH
04290 FCB      DEFS    20H             ;File Control Block area
```

3. ADD the following lines:

```
01745 CMD      DEFM    'CMD'           ;Default extension string

01771 START    LD      (EXIT2+1),SP    ;Save stack pointer
01772          PUSH    HL              ;Save input buffer ptr
01773          LD      H,B             ;Transfer filespec start
01774          LD      L,C             ; to HL
01775          LD      DE,FCB          ;Where to put filespec
01776          LD      A,78            ;Move to FCB
01777          RST     28H
01778          JP      NZ,ERREXT       ;Go if error
01779          LD      HL,CMD          ;Point to CMD extension
01780          LD      A,79            ;Insert default extension
01781          RST     28H
01782          POP     HL              ;Restore buffer pointer
01783          LD      A,(HL)          ;Get argument (!) if any
02860 EXIT2    LD      SP,0            ;Restore stack pointer
02861          LD      HL,0            ;No error condition
02862          LD      DE,CMDBFR       ;Point to new command bfr
02863          LD      A,(DE)          ;Get first char of buffer
02864          CP      0DH             ;Is first char a <CR>?
02865          RET     Z               ;Normal exit if cmd end
02866          EX      DE,HL           ;Point HL to new cmd bfr
02867          LD      A,18H           ;Execute next DOS command
```

PROSOFT ON PIRACY
by Chuck Tessler

[Reprinted from the Adelaide Micro-User News, 36 Sturt Street, Adelaide, South AUSTRALIA 5000. Although this article is not directly relevant to our North American readers, I think it has some interesting implications for all of us. When was the last time you saw a really good piece of new TRS-80 software? Maybe there's a reason that folks don't feel inclined to spend hours upon hours writing programs for the TRS-80 market...]

(The following article is the text of the letter I received from Chuck Tessler, co-founder of Prosoft, and author of such programs as 'Allwrite'. The letter was in response to my request that he explain his policy of not selling software to Australians. I think the points he makes are very interesting and, to a large extent, they provide valid reasons for his policy. I would welcome any comments which members may have on this subject for publication in future issues. -Ed Grigonis [Editor, Adelaide Micro-User News])

We stopped accepting orders from many countries, including Australia, early in 1984. There were several reasons for this:

1. Our products and documentation are specifically oriented to "American English" and the printer character sets for the United States. They often do not work satisfactorily for other languages, and that includes English as it is used in Great Britain and Australia. We are a small company and do not have the resources to add proper support for other languages. Rather than taking people's money when they are likely to be unhappy with our products, I decided to reject the orders. We do accept overseas orders when the customers sign an agreement that they understand the language restriction.

2. We offer after-the-sale support. As you probably know, we are virtually unique in this, at least among micro-computer software houses. The basis for this support is telephone calls from customers to my technical support group. The system works well for the U.S. However, most overseas customers write for help, since the telephone calls are very expensive.

This raises a problem: I have to answer most of the written questions myself. Only one other person on the technical staff has the time and knowledge to help me with this. And, I cannot justify the cost of my time (or his) to answer a lot of letters, given the relatively low prices we charge for our software. I did a cost analysis of domestic retail, overseas retail, and dealer orders in 1984, and found that we lost money on overseas retail and all dealer sales. My choices were to 1) shut down; 2) raise the prices significantly; or 3) discontinue the unprofitable sales channels. I chose the latter, and think weekly that I should have chosen the first alternative.

3. Above and beyond the widespread theft of software that occurs world-wide, we found several flagrant cases that affected us in West Germany and Australia. The laws at that time offered us no protection at all. I do know that the laws in Australia have been changed, and that the original Court decision was over-turned. However, do you think the new laws have deterred even one person from stealing my products? Neither do I.

My attorney told me in 1983 that it would cost me at least $10,000 (U.S.) to even begin to take legal action in Australia. He advised me that there was no upper-limit he could promise to me, and that, even if we obtained a favorable judgement, the odds against collecting were almost as high as the additional costs would be for trying to collect.

The point here is that civil laws (as opposed to criminal laws, in which the STATE is the plaintiff) are not enforced automatically. I cannot afford to buy the legal protection to which I am theoretically entitled.

Now, after considering all of the above, and also realizing that my refusal to sell my products overseas would only slow, but not prevent, their theft, I decided that I would not be an active party to my own rape. I cannot prevent people from obtaining my products, whether or not they pay someone for them, but I certainly can refuse to knowingly participate in the distribution chain!

The trade-offs to PROSOFT (and to me) in this position are the following:

1. I lose some sales revenues from the people who tried to place legitimate orders with us. My estimate is that about one percent of our sales used to come from overseas before we stopped accepting them, and that less than half a percent of the potential orders we now receive are from overseas. This means the success or failure of PROSOFT will not be affected by these sales.

2. I do not have to provide support when we receive overseas letters or calls for help. We received several times as many requests for help (and complaints) from our overseas NEWSCRIPT customers (based on the number of sales made), as we did from our domestic NEWSCRIPT customers. I believe we lost money on those NEWSCRIPT sales due to the high cost of support, and the returns. This means we probably are either breaking-even or even showing more of a profit by not accepting overseas orders. I realize you probably will not agree with this claim, and I cannot actually prove it; but I'm trying to tell you what I believe.

3. Whereas the employees of large companies do not usually have a personal relationship with their company, let alone the products they sell, I'm the one who co-founded PROSOFT, wrote most of the software, and all of the manuals. I take great pride in my work, and when my products are stolen, it is not just an impersonal thing: those concepts and all the thousands of hours of work it took to make them real, came from my mind. So, it is my thoughts and ideas that people are stealing; and that is just about as personal as you can get.

4. I can afford to take legal action to protect our property here in the U.S., but not overseas.

Now, may I pose one simple question to the people who have given away or stolen my products: if you paid me for those products, and I refused to support you, how would you feel? Well, that's what you've done to me, and that's how I feel, too.

Sincerely,
Chuck Tessler

* * * * *

COMMENTS ON THE PREVIOUS LETTER
by Ed Grigonis

Firstly, I would like to thank Chuck very much for having taken the time to explain his position.

I think the letter serves to highlight the conflict between the just expectations of software authors and the often unrealistic expectations of the software buyer.

Consider the neophyte computer buyer. He buys the latest computer with all sorts of fancy features and then proceeds to buy the most sophisticated software available. As soon as he gets the program he puts it straight into the computer and then expects to breeze through the whole thing (usually without so much as a cursory glance at the manual). When he finds he is unable to use it he blames the company who sold it to him. Never mind that he probably wasn't anywhere near experienced enough with computers to be able to appreciate what they can and cannot achieve. How often have we seen the comments in magazines about people who ask dumb questions about software simply because they have thrown themselves in at the deep end with no understanding of what they are doing. OK, the software and hardware companies like to stress how easy their products are to use, but I still think that they would generally (and justifiably) expect the consumer to have some idea of what they are doing. You don't buy a plane if you don't know how to fly it, you don't buy a guitar if you don't know how to play it, and you don't buy a sophisticated computer program if you have no idea of the basics required to run it UNLESS you are prepared to put in the effort to learn how to use it.

So if you must buy a fancy program that you don't really understand, take the time to read the manual, follow the examples and start at the basics. If it doesn't work as advertised (and be sure you know enough about it to know that is the case) then you have got a legitimate complaint against the company who sold it to you. But if you are just floundering around, over your head in confusion, learn from your mistake and get something a bit simpler next time. Keep the program you don't understand because with experience will eventually come the knowledge to put it to the use it was intended.

Above all, DON'T try and make it do something it wasn't designed to do and which the company never said it would do. That way lies many wasted hours.

As to PROSOFT, it is probably too late to reverse their current policy, which is a pity as they have some excellent software available. Perhaps some enterprising Australian company could try and sway them back but I see this as an impossible task as I would expect the conditions of any such about-face to be along the following lines:-

1. The Australian company would have to agree to provide all support to purchasers of the software in Australia. Naturally the company would be able to refer legitimate bugs to Prosoft for correction but the existence of such is unlikely.

2. Australian purchasers of the software would need to understand that ALL queries would have to be referred to the Australian distributor. (Although, perhaps they could have the option of referring their queries by mail direct to Prosoft with a 'Query Fee' of, say, $US30. This would avoid the ticky-touchwood silly questions from the hapless and would compensate Prosoft for any effort required to solve the problem. Perhaps if a real bug is uncovered the 'Query Fee' could be refunded)

3. The Australian company would have to be able to provide audited statements to justify the levels of any royalties sent to Prosoft.

As I said, the chances of this happening would have to be zero but I think such conditions would be most reasonable.

* * * * *

NORTHERN BYTES EDITOR'S COMMENTS ON THE FOREGOING

I'm a little surprised at Chuck's comments regarding "American English", because Allwrite has excellent capabilities for printing any character that your printer is capable of producing. For example, if I wish to use characters such as £, ¥, ß, ñ, ¿, or any of several other foreign characters or special symbols that my Tandy DMP-2100P can produce, there are at least two easy ways that I can think of offhand (and maybe more that I haven't thought of) to get ALLWRITE to put them into my text. The process is only a bit more complicated than entering "normal" characters (although you can easily define soft keys so that frequently-used characters can be entered using only a couple of keystrokes), but you won't be able to do it without reading the manual!

This whole situation is a bit unusual because PROSOFT does not sell Allwrite through dealers. If you buy a legitimate copy of Allwrite, you buy it from Prosoft. Further, the person who places the order is considered the "registered owner". Thus, even if someone in another country was able to get a U.S. resident to purchase a copy of Allwrite for him, that copy would be registered in the U.S. resident's name. Because of this "closed" distribution system, it's now nearly impossible for anyone living in one of the countries to which PROSOFT will not sell their products to obtain a legitimate copy of Allwrite - and even if someone does, PROSOFT would not be obligated to provide support because it would be a "second-hand" purchase.

Do I approve of this? I have mixed feelings. Normally, I do not support any sort of restraint of trade (I don't like utility monopolies, either), but in this situation it is certainly understandable that Mr. Tesler may be less than enthusiastic in, as he puts it, being an active party to his own rape. In any case, Chuck certainly has the right to operate his company as he sees fit.

Why did I reprint this? Just because I thought our readers might find it interesting. But also, maybe by printing this some of the people who never buy a legitimate piece of software will begin to realize why good, new software products are getting harder to find. It is a sad but true fact of life that commercial enterprises can't continue to operate if they aren't profitable.

We welcome any comments from readers on this discussion, provided you send them on magnetic media or via electronic mail (see the opening of our Letters Column for details).

EASTER

Back in the October, 1981 issue of the TRS-80 Microcomputer News, John T. Livingstone wrote a program to find the date that Easter falls on in any given year. Although originally written for the pocket computer, it was easy to convert to standard TRS-80 BASIC. The original pocket computer program was checked and found to be accurate for the years 1900 to 2089 (it wasn't checked for years before or after that period). This may prove helpful to those writing calendar and/or scheduling programs. If anyone can extract the basic mathematical algorithm used in this program, please send it in as it may prove more useful to many readers in that format (particular to those readers coding in languages other than BASIC).

```
10 INPUT "YEAR WANTED";Y
20 N=Y-1900
30 Z=N/19
40 V=INT(Z)
50 A=(Z-V)*19: A=A+.05: A=INT(A*10)/10
60 B=((7*A)+1)/19: B=INT(B)
70 T=((11*A)+4-B)/29
80 M=(T-INT(T))*29
90 Q=N/4: Q=INT(Q)
100 D=(N+Q+31-M)/7
110 W=(D-INT(D))*7: W=W+.05: W=INT(W*10)/10
120 E=25-M-W: E=E+.05: E=INT(E*10)/10
130 PRINT "EASTER IS ";: IF E>0 THEN PRINT "APRIL"; E ELSE PRINT
"MARCH"; E+31
140 GOTO 10
```

AN EXPLANATION ABOUT ELECTRICITY
by John Buxton 'Catcher' 1984.

[As an educational service to our readers, we are reprinting this fine tutorial from the MicroBee News of South Australia, by way of the Adelaide Micro-User News, 36 Sturt Street, Adelaide, South AUSTRALIA 5000. For the benefit of editors of newsletters on our publication exchange list, you might want to keep this article in mind for reprinting, perhaps in your April issue...]

I was never taught about electricity at school, nor was it a topic of dinner conversation between my parents. But with reading, and having to change light bulbs or tune a transistor radio, I have picked up a pretty sound working knowledge of electrical matters. It's not comprehensive. I still don't fully understand why you can't boil an egg on an electric guitar or run a Basic program on a washing machine.

Most electricity is manufactured in power stations where it is fed into wires which are then wound around large drums.

Some electricity, however, does not need to go along wires; that used in lightning for example. This kind of electricity is not generated, but just hangs around in the air, loose.

Electricity makes a low humming noise. This noise may be pitched at different levels for use in door bells, telephones, and electric organs.

Electricity has to be earthed, that is to say it has to be connected to the ground before it can function, except in the case of aircraft which have separate arrangements.

Although electricity is said not to leak out of an empty power point, that power is, nevertheless, live if you happen to shove your finger in it when the switch is on. If it is not leaking, what is it doing?

Electricity is made up of two ingredients, negative and positive. One ingredient travels along a wire covered in red (lately changed to brown which doesn't show the dirt so much) plastic, the other along a wire covered in black (lately changed to blue in the interests of racism) plastic. These two wires meet together in a plug, which mixes the ingredients to form what we call electricity.

Electricity may be stored in batteries. Big batteries do not necessarily hold more electricity than small batteries. In big batteries electricity is just shoved in, whereas in small batteries (for transistors) it is packed flat.

A switch controls a small clamp or vice, which grips the wires very hard so that the electricity cannot get through. When the switch is flicked on, the wire is relaxed and the electricity travels to the light bulb where a piece of wire, called the element, is left bare. Here for the first time, we can actually see the electricity in the form of a spark. This spark is enlarged many many times by the curved bulb which is made of magnifying glass.

I have not yet touched on fuse wire. It has always amazed me that an industry which is so enterprising in most respects - the invention of colored electricity for use in traffic lights and the harnessing of negative electricity for refrigeration are two inventions that come to mind - should still be manufacturing fuse wire too thin. By using chicken wire I now have a fuse box which - even when the spin dryer burst into flames because of too much electricity having been fed to it - has for six months been as impregnable as the Bank of Argentina.

In some respects, I know my knowledge is imperfect. I have not yet explored the field of neon signs - how do they make the electricity move about? And the pop-up toaster - how does the electricity know when the toast is ready?

Logic would answer all these questions, but the light on my desk has just gone out.

SAVESUB: THE BASIC VERSION NUMBERING UTILITY

by Mohammad Dadashzadeh and Ted Byrne

It's too deep into the night in a room lonely as a lighthouse. Through a dry film your eyes peer at the flashing screen. Hours of work on two programs, but now the last bug's caught and you press the <ENTER> key as the thought hits. It hits without reason or sense, like a dropped brick. A pool of darkness opens around a terrible truth ---- "IT'S THE WRONG NAME - OMIGAWD! THIS PROGRAM'S ERASING THE OTHER!" You've misnamed the file and that millisecond disaster has zapped the other away. Nothing now but to flop into bed and fitlessly curse until it becomes morning.

Why hadn't you backed it up? Why didn't you remember the right name? Why's the machine out to get you? A paranoid lump gnarls your stomach.

In every programmer's life a misnamed file has snuffed out some crucial work. Do it once and you're suddenly serious about backups and hard copy. Still, as you scurry in hot pursuit of some wild bug, it's seductive to postpone a back-up for just another minute - until ...

To avoid this, some people include a last line which might read: 65529 SAVE"filespec/ext":END. Upon a session's completion, they do a 'GOTO 65529' and that helps a lot. But suppose you're experimenting with a version and you GOTO 65529 - OOOOOPS! The main version is now replaced with the experiment. And how do you keep a lot of similarly named files on different disks straight? Which one is the latest?

That's why SAVESUB exists. It's a BASIC utility to prevent disaster. SAVESUB lets you <ENTER> a command from the last line of your current program and ----- BAM! It saves the present program on the disk drive of your choice under the correct name with an extension that contains the number of the most recent version. If you've been working on Version 19, then this one's saved as V20! Even better, the first line of your program listing will now contain this amended file name. And as a special convenience, SAVESUB will also POKE the current date and time into that first line. And of course, you can include any valid password as part of the file specification.

No more confusion over which file is the most current. Each version is numbered and time stamped. And, if you want to get rid of the earlier versions it's simple. Just <ENTER> the command from the next to last program line and SAVESUB cleans-up your disks. And while you're at it, SAVESUB also lets you backup this latest file - with a separate extension!

Considering what it does, SAVESUB is very short. No long typing exercises - and it's delightfully easy to use. Just MERGE it onto the program you're working on, it'll fit right in at the very end. And then <ENTER> a new first line. Here's the only thing that takes a little care. The first line of your program will have to look like this:

```
1 REM filespec/V00  00/00/00 00:00:00
```

Or it could look like:

```
10 'filespec/V00        00/00/00 00:00:00
```

The point is that the line needs to be a REM or REMARK statement. The number of spaces preceding the filespec (the program's name) aren't critical nor are the number of spaces between the filespec and what will become the latest date. However, the date and time once begun, must occupy 17 contiguous spaces including one space between the date and time. Include anything you like after these 17 characters.

For example, you might enter a first line which looks like this:

```
100 'SAVESUB/V00      00/00/00 00:00:00   Copyright (c) 1984.
```

Notice that the filespec is composed of up to eight characters beginning with a letter and that the extension has three places. And if you wish, you can also enter an optional password.

That's it! Go on back to your program. Ignore SAVESUB until you need it. Change your work any way you want. RENUMBER the program at your whim - SAVESUB doesn't care. Quiet as a sleeping cat, it will just fit itself to its bed awaiting a call. When you're ready, LIST the last line of the BASIC program in memory. You'll see something like:

```
65528 END:RUN65516:REM to save new version.
```

Well do it. Type RUN 65516 and <ENTER>. What's happening? Look at the program listing. To start SAVESUB peeks into memory to find your first line. That's really not too hard since the TRS-80 BASICs store the start address of BASIC programs at memory location 40A4H. Looking around, it hunts up the filename and extension in your first line. Next it builds up the filespec as a variable called PRGM$. Then SAVESUB checks to see if you've got the extension you were supposed to include. If it can't find a "/V" it'll stop in line 65498.

The following line POKEs the current DATE and TIME into the first line of the program at the reserved spots. Lines 65500 - 65502 determine the current version of the program and update it by one.

In line 65517 SAVESUB asks you for the disk drive you prefer to direct the program toward. Note line 65520 contains the number seven in quotation marks. Should you have fewer than eight drives on line, then alter this error check to your configuration. Hence, if you have two drives, then substitute "1" for the "7" and SAVESUB will not permit you to <ENTER> an erroneously high number. The following lines save the program under its new name and ask whether you want to make a backup. Of course you should. If you answer 'Y' or 'y', a second copy will be made under the name filespec/BKU.

Suppose your first line was the same as proposed above. You'll now notice the first line of that program has been changed to look something like this:

```
100 'SAVESUB/V01      11/17/85 11:08:01   Copyright (c) 1984
```

neatly listing the version (V01), the date (11/17/85) and the time (11:08:01).

Want to KILL-off the debris - the older versions of this file? List the next-to-last line. Remember, if you've RENUMBERed your program, your listing will contain a different line number than this example. But it should still take this kind of form:

```
65527 END:RUN65504:REM to kill old versions
```

Do it. From the listing you can see what will happen. SAVESUB again determines the latest version from the filespec written in the first (REMark) line in the program. Then it kills each of the lower-numbered versions of the file which reside on your disks. Finally it requests where you want to direct your new file, SAVEs the current version and again asks you to back it up.

The mainframe world has such convenience. For example the VMS operating system on DEC's VAX series is one of the operating systems that has extended the concept of filespec to include a version number field. In its own way SAVESUB will bring that concept to the TRSDOS world. And, by the way, SAVESUBroutine can be packed if you want it to take up less space.

SAVESUB will work on a TRS-80 MODEL I/III with any major DOS including TRSDOS, NEWDOS, MULTIDOS, LDOS and DOS PLUS. It's written to be unobtrusive and run in a tiny amount of space. While it's true that even paranoids have enemies, your machine's not one of them. Unless you're a reckless sort pickled sour by paranoia, SAVESUB'll relax your stomach and become one of the handiest utilities you've got. So, while it wasn't designed as a sleep aid, it is.

```
65486 '-------------------------------------------------------------
-
65487 'SAVESUB/ASC, By Mohammad Dadashzadeh, MDZ Associates.
65488 '** NOTICE **
65489 'The first line in the program MUST be of the form:
65490 '1 REM  filename/V00      00/00/00 00:00:00            OR
65491 '1 'filename/V00 00/00/00 00:00:00
65492 STOP
65493 M=256*PEEK(&H40A5)+PEEK(&H40A4)+4
65494 IFPEEK(M)<>147THENM=M+2
65495 M=M+1:IFPEEK(M)=32THENGOTO65495ELSED=M
65496 IFPEEK(M)<>32THENPRGM$=PRGM$+CHR$(PEEK(M)):M=M+1:GOTO65496
65497 M=M+1:IFPEEK(M)=32THENGOTO65497
65498 MD=INSTR(PRGM$,"/"):IFMD=0THENSTOP:'missing /Vxx
65499 FORZ=1TO17:POKEM+Z-1,ASC(MID$(TIME$,Z,1)):NEXTZ
65500 LSD=VAL(MID$(PRGM$,MD+3,1)):MSD=VAL(MID$(PRGM$,MD+2,1))
65501 LSD$=MID$("1234567890",LSD+1,1):MID$(PRGM$,MD+3,1)=LSD$:POKED
+MD+2,ASC(LSD$)
65502 IFLSD=9THENMSD$=MID$("1234567890",MSD+1,1):MID$(PRGM$,MD+2,1
)=MSD$:POKED+MD+1,ASC(MSD$)
65503 RETURN
65504 CLEAR100:GOSUB65493:REM code to kill old versions starts
here
65505 ONERRORGOTO65514
65506 FILE$=PRGM$
65507 LSD=VAL(MID$(FILE$,MD+3,1)):MSD=VAL(MID$(FILE$,MD+2,1))
65508 IFLSD=0ANDMSD=0THENONERRORGOTO0:GOTO65517
65509 LSD=VAL(MID$(FILE$,MD+3,1)):MSD=VAL(MID$(FILE$,MD+2,1))
65510 LSD$=MID$("9012345678",LSD+1,1):MID$(FILE$,MD+3,1)=LSD$
65511 IFLSD=0THENMSD$=MID$("9012345678",MSD+1,1):MID$(FILE$,MD+2,1)
```

```
=MSD$
65512 PRINT"Killing ";FILE$:KILLFILE$
65513 GOTO65507
65514 PRINT"+++ NOT FOUND +++"
65515 RESUMENEXT
65516 CLEAR75:GOSUB65493:REM code to save new version starts
here
65517 PRINT"Saving  ";PRGM$;":? <=-- to which drive?";
65518 FORM=ØTO21:PRINTCHR$(24);:NEXTM
65519 FORD=ØTOØ:D$=INKEY$:D=(D$="");NEXTD
65520 IFD$<"Ø"ORD$>"7"THENGOTO65519ELSEPRINTD$;CHR$(3Ø)
65521 PRGM$=PRGM$+";"+D$:SAVE PRGM$
65522 PRINT"Save a backup copy (Y/N) --=> ";
65523 M$=INKEY$:IFM$=""THENGOTO65523ELSEPRINTM$
65524 ONINSTR("YyNn",M$)+1GOTO65522,65525,65525,65527,65527
65525 MID$(PRGM$,MD+1,3)="BKU"
65526 SAVE PRGM$
65527 END:RUN65504:REM to kill old versions
65528 END:RUN65516:REM to save new version
```


THE COMPLETE HANDBOOK OF PERSONAL COMPUTER COMMUNICATIONS
(Subtitle: Everything You Need to Go Online with the World)
A Book Review by Jack Decker

A successful person doesn't have to know all the answers to the problems he encounters, as long as he knows where to get the answers. When you have a problem in getting one computer to talk to another over the telephone lines, this book is the place to find the solution to that problem. In fact, if you don't even know whether you have any need for your computer to be talking to other computers, this book will probably give you the answer to that question as well.

The book I speak of is the new, completely revised and updated edition of Alfred Glossbrenner's definitive work entitled "The Complete Handbook of Personal Computer Communications - Everything You Need to Go Online with the World". Mr. Glossbrenner is a prolific writer of computer-related books and has been a very successful author for one reason - his books are very understandable by the average computer user. There are many books for the computer whizzes, but Glossbrenner brings the mysteries of computer communications down to the level of the masses.

This new edition is about 550 pages long (over 200 pages longer than the original bestseller), and has been almost completely rewritten. This rewrite was necessitated by the speed at which things change in the computer communications world. Indeed, portions of the book were probably already a bit outdated when it was published (at the end of October, 1985) but it is probably as up-to-date as any other book you will find on this fast-changing field.

There isn't much about personal computer communications that this book doesn't cover. Want to know how to buy a modem, or how to use bulletin boards, electronic mail, or online utilities such as Compuserve, Delphi, or The Source? Want to find out about packet networks or Videotex? How about the Telex system, or computerized conferencing? How can you get instant stock quotations, airline schedules, weather reports, or abstracts of magazine articles by computer? It's all in here!

One thing I appreciate about the book is Glossbrenner's cost-cutting Online Tips. When you are online with a service that charges by the minute, it's easy to make mistakes that can cost you lots of money. For example, even if you have a 1200 baud modem, it may be more cost-efficient to log onto some services at 300 baud if the service in question charges a premium price for 1200 baud connections. The book will tell you which services do charge a premium, and which don't. Many systems have "shortcuts", ways to skip menus and get around faster that aren't readily apparent to the inexperienced user. Glossbrenner tells you about these. Some types of information are available from many online service providers, but the same information may cost you more through one service than another. The book will tell you how to get the service you want at the lowest cost. If you go online very often, this book will probably save you many times its $14.95 price.

Unfortunately, the book does not have much information that is specific to the TRS-80. In particular, though it tells how to get free (public domain) communications programs for many types of machines, the TRS-80 is not one of them (I'll rectify that oversight here and tell you that many good, free terminal programs for the TRS-80 are available from several sources, including many computer Bulletin Board System operators and on certain TAS Public Domain Library disks). But that is a minor complaint. Most of the techniques of online communication are not machine specific, so most of the information in this book will apply as much to TRS-80 owners as to other computer users.

The one complaint that I have seen about this book in other publications that we receive is that Glossbrenner occasionally plugs his other books (particularly "How to Get Free Software") in the pages of this book. True enough, but since this book is already about twice the size of many other publishers' books that sell for the same price, I hardly begrudge the author an occasional reference to his other books, particularly when a subject of interest may be more thoroughly covered in another volume.

One insight that I have received from "hanging around" The Alternate Source is that you can sell a fairly complex word processing program or some fairly complicated utility program and you will get relatively few calls from users requiring assistance, but if you sell even an easy-to-use terminal program, you'll get all kinds of calls from users that can't seem to understand the mechanics of computer communications, and that seem to be scared to death of the whole process of logging onto a Bulletin Board System or a service such a Compuserve. I don't know why this seems like such a mysterious process to some people (it's really simple once you've done it a couple of times), but perhaps it's because they don't really understand the process that brings words and data from many miles away into their home computers. These people really need Glossbrenner's book, since it explains the whole process of going online. However, even the experienced online veteran will most likely find a cost-cutting tip, or information on a service that he didn't know was even available, that will be worth the cost of the book.

I'll close with a personal note. I receive a lot of reading material in the mail, and it's very rare for me to spend a lot of time with any one piece of literature. With the possible exception of a couple of chapters that covered material of little interest to me, I read this book all the way through! It was that interesting. This new printer I have won't print stars, but if it would, I'd give this book four of 'em!

* * * * *

Title: THE COMPLETE HANDBOOK OF PERSONAL COMPUTER COMMUNICATIONS--COMPLETELY REVISED AND UPDATED: Everything You Need to Go Online with the World. Author: Alfred Glossbrenner. Price: $14.95 Paperback. ISBN 0-312-15760-6

A REAL TIME CLOCK

by David Kennedy (phone 011+61+47+30-2577)

[Reprinted from SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, Australia. This article is intended for Model I users running NEWDOS/80, but should be easily adaptable to the Model III/4. If anyone adapts this project for use with another Model TRS-80 and/or another DOS, please send the details and we will publish them here.]

The circuit presented here uses the National Semiconductor MM58274, which is a bus oriented microprocessor real time clock, it has a major advantage over previous designs which used the MSM5832 in that it is fast enough to be connected directly to the processor bus, thus avoiding the use of a PIO or similar device to latch the data presented to the clock chip by the micro, as well as simplifying the circuit and software design.

The counters of the MM58274 are arranged as 4-bit words and can be randomly accessed for time reading and setting. The clock may be assembled on a piece of Vero-Board, it may be powered by either 3 dry cells or 4 NiCads. The clock and 4016 CMOS draw approximately 150 microamps from the batteries. The 4016 CMOS pulls the chip select, read and write lines high so as to put the MM58274 in standby mode. Without the 4016 the clock draws 1 milliamp.

The MM58274 can be obtained from Geoff Wood Electronics in Sydney at about $15, with the National Semiconductor Data sheet for this product. The circuit is a modified version of that which appeared as Project 80 in the July '84 issue of 80 Micro and uses the same port decoding. For a more detailed description of the operation of the circuit see the original article by Roger C. Alford.

The first listing should be assembled and then patched into SYS0/SYS beginning at FRS 14, BYTE 06 (ie. immediately after the bytes 00 00 00 01 F7 E6 50), using a utility such as SUPERZAP or SUPER UTILITY. The execution address of SYS0 must also be changed at the end of the sector from 02 02 00 4D to 02 02 E6 50.

```
                ;*****************************************************
                ;*      NEWDOS/80 ZAP FOR REALTIME CLOCK             *
                ;*  INSTALL INTO LAST SECTOR ON SYS0/SYS             *
                ;*          BY DAVID KENNEDY                         *
                ;*****************************************************
0020            CNTRL EQU   20H     ;CONTROL REG
4041            SECLOC EQU  4041H   ;SECONDS STORAGE LOC.
                ;*****************************************************
                ;*     DATE,TIME INPUT TO RAM BUFFER 4041-4046       *
                ;*****************************************************
                      ORG   5DE6H           ;START OF PATCH.
5DE6 A5         START AND   L               ;CLEAR
5DE7 3E14             LD    A,20            ;TAPE
5DE9 D3FF             OUT   (0FFH),A        ;LATCH.
5DEB F3               DI                    ;DISABLE INTERRUPTS.
5DEC DB22             IN    A,(22H)         ;CHECK
5DEE FEFF             CP    255             ;CLOCK IS
5DF0 2825             JR    Z,GO            ;CONNECTED.
5DF2 0122FF           LD    BC,0FF22H       ;LOAD COUNT AND PORT 22.
5DF5 21414D           LD    HL,SECLOC       ;LOAD SECONDS TABLE.
5DF8 CD1B51           CALL  GETTIM          ;GET VALUE.
5DFB 77               LD    (HL),A          ;LOAD SECONDS.
5DFC 23               INC   HL              ;NEXT LOCATION.
5DFD CD1B51           CALL  GETTIM          ;GET VALUE.
5E00 77               LD    (HL),A          ;LOAD MINUTES.
5E01 23               INC   HL              ;NEXT LOCATION.
5E02 CD1B51           CALL  GETTIM          ;GET VALUE.
5E05 77               LD    (HL),A          ;LOAD HOURS.
5E06 23               INC   HL              ;NEXT VALUE.
5E07 23               INC   HL              ;NEXT VALUE.
5E08 CD1B51           CALL  GETTIM          ;GET VALUE.
5E0B 77               LD    (HL),A          ;LOAD DAY.
5E0C 23               INC   HL              ;NEXT VALUE.
5E0D CD1B51           CALL  GETTIM          ;GET VALUE.
5E10 77               LD    (HL),A          ;LOAD MONTH.
5E11 2B               DEC   HL
5E12 2B               DEC   HL
5E13 CD1B51           CALL  GETTIM          ;GET VALUE.
5E16 77               LD    (HL),A          ;LOAD YEAR.
5E17 FB         GO    EI                    ;ENABLE INERRUPTS.
5E18 C3004D           JP    4D00H           ;GO DOS.
                ;*****************************************************
                ;*                READ CLOCK REGISTERS.              *
                ;*****************************************************
5E1B DB20       GETTIM IN   A,(CNTRL)       ;CHECK FOR
5E1D CB5F             BIT   3,A             ;CLOCK IS
5E1F 20FA             JR    NZ,GETTIM       ;READY.
5E21 ED78             IN    A,(C)           ;READ VALUE IN.
5E23 ED78             IN    A,(C)           ;MAKE SURE.
5E25 A0               AND   B               ;ZERO HIGH BITS.
5E26 57               LD    D,A             ;SAVE VALUE.
5E27 0C               INC   C               ;NEXT PORT.
5E28 ED78             IN    A,(C)           ;READ IN VALUE.
5E2A ED78             IN    A,(C)           ;MAKE SURE.
5E2C A0               AND   B               ;ZERO HIGH BITS.
5E2D 07               RLCA                  ;VALUE TIMES 2.
5E2E 0C               INC   C               ;NEXT PORT.
5E2F 5F               LD    E,A             ;SAVE VALUE.
5E30 07               RLCA                  ;VALUE TIMES 2.
5E31 07               RLCA                  ;VALUE TIMES 4.
5E32 83               ADD   A,E             ;ADD VALUES.
5E33 82               ADD   A,D             ;ADD AGAIN.
5E34 C9               RET
5DE6                  END   START
00000 TOTAL ERRORS

CONTRL 0020   GETTIM 5E1B   GO     5E17   SECLOC 4041   START  5DE6
```

The following program allows the reading and setting of the clock from DOS (a BASIC program to allow time setting and reading follows).

```
                ;*****************************************************
                ;*                REAL TIME CLOCK                    *
                ;*         SETTING AND READING ROUTINE               *
                ;*                BY DAVID KENNEDY                   *
                ;*****************************************************
                ;
3022            CRSTOP EQU  3022H   ;CURSOR POSITION +2
4020            CURPOS EQU  4020H   ;CURSOR POSITION
01C9            CLEAR  EQU  01C9H   ;CLEAR SCREEN
3C00            SCREEN EQU  3C00H
401D            DODCB  EQU  401DH   ;VIDEO D.C.B
0020            CONTRL EQU  20H     ;CLOCK CONTROL REDG.
6000                   ORG  6000H
                ;*****************************************************
                ;*              WRITE SCREEN MESSAGES                *
                ;*****************************************************
6000 F3         START  DI                   ;DISABLE INTERRUPTS.
6001 31FF60            LD    SP,START      ;LOAD STACK POINTER.
6004 CDC901            CALL  CLEAR         ;CLEAR SCREEN.
6007 21583C            LD    HL,SCREEN+88  ;SCREEN LOCATION
600A 222040            LD    (CURPOS),HL   ;LOAD CURSOR POS.
600D 21F762            LD    HL,MESG1      ;GET MESSAGE
6010 CDC361            CALL  VIDEO         ;PUT ONTO SCREEN
6013 21973C            LD    HL,SCREEN+151 ;N
6016 222040            LD    (CURPOS),HL   ;E
6019 219F62            LD    HL,MESG2      ;X
601C CDC361            CALL  VIDEO         ;T
601F 21D83C            LD    HL,SCREEN+216
6022 E5                PUSH  HL            ;SAVE LOCATION
6023 222040            LD    (CURPOS),HL   ;LOAD CURSOR.
6026 218062            LD    HL,MESG3      ;LOAD MESSAGE.
6029 CDC361            CALL  VIDEO         ;PUT ON SCREEN.
602C CD2000     LOOP   CALL  20H           ;CALL KEYBOARD
602F FE52              CP    'R'           ;CHECK FOR "R"
6031 CACC61            JP    Z,READ        ;GO READ
6034 FE57              CP    'W'           ;CHECK FOR "W"
6036 20F4              JR    NZ,LOOP       ;LOOP IF NOT WRITE
6038 21CB3C            LD    HL,SCREEN+203 ;SCREEN LOCATION.
603B E5                PUSH  HL            ;SAVE LOCATION.
603C 222040            LD    (CURPOS),HL   ;LOAD CURSOR.
603F 21C862            LD    HL,MESG4      ;LOAD MESSAGE.
6042 CDC361            CALL  VIDEO         ;LOAD SCREEN.
6045 212230            LD    HL,CRSTOP     ;SCREEN POSITION.
6048 222040            LD    (CURPOS),HL   ;LOAD CURSOR.
604B 3EB0              LD    A,176         ;CURSOR CHARACTER.
604D 322240            LD    (DODCB+5),A   ;LOAD D.C.B.
6050 ED56              IM    1             ;MODE I INTERRUPTS.
6052 FB                EI                  ;INABLE INTERRUPTS.
6053 3E3E              LD    A,'>'         ;LOAD ">".
6055 CD3300     LOOP1  CALL  33H           ;DISPLAY IT.
```

```
6058 CD2B00   00600 LOOP2  CALL   28H          ;CALL KEYBOARD.
605B FE08     00610        CP     08H          ;BACKSPACE.
605D 2813     00620        JR     Z,BSPACE     ;GO BACKSPACE.
605F FE18     00630        CP     18H          ;ERASE
6061 281C     00640        JR     Z,ERASE      ;GO ERASE.
6063 FE0D     00650        CP     0DH          ;ENTER.
6065 CA8960   00660        JP     Z,ENTER      ;GO ENTER.
6068          00670 SAVE   EQU    $-2          ;ENTER1 JUMP.
6068 FE2F     00680        CP     '/'          ;COMPARE FOR "/".
606A 38EC     00690        JR     C,LOOP2      ;LOOP "/"OR LESS.
606C FE3B     00700        CP     ';'          ;COMPARE FOR ";".
606E 30E8     00710        JR     NC,LOOP2     ;LOOP ";"OR GREATER.
6070 18E3     00720        JR     LOOP1        ;GO WITH NUMBER.
              00730 ;*************************************************
              00740 ;*          BACKSPACE CHARACTER.                 *
              00750 ;*************************************************
6072 2A2840   00760 BSPACE LD     HL,(CURPOS)  ;CURSOR LOCATION.
6075 2B       00770        DEC    HL
6076 3E3E     00780        LD     A,'>'        ;LOAD ">".
6078 BE       00790        CP     (HL)         ;COMPARE SAME.
6079 2800     00800        JR     Z,LOOP2      ;IF SO LOOP.
607B 3E08     00810        LD     A,08H        ;LOAD BACKSPACE.
607D 18D6     00820        JR     LOOP1        ;LOOP BACK.
              00830 ;*************************************************
              00840 ;*          ERASE LINE ROUTINE.                  *
              00850 ;*************************************************
607F 212330   00860 ERASE  LD     HL,CRSTOP+1  ;LOAD HL 3023H.
6082 222840   00870        LD     (CURPOS),HL  ;LOAD CURSOR LOC.
6085 3E1F     00880        LD     A,1FH        ;ERASE FROM CURSOR ON.
6087 18CC     00890        JR     LOOP1        ;GO DO IT.
              00900 ;*************************************************
              00910 ;*          ENTER DAY MONTH YEAR.                *
              00920 ;*************************************************
6089 212530   00930 ENTER  LD     HL,CRSTOP+3  ;LOAD HL 3025H
608C 3E2F     00940        LD     A,'/'        ;LOAD "/".
608E BE       00950        CP     (HL)         ;COMPARE SAME.
608F 20EE     00960        JR     NZ,ERASE     ;ERASE IF NOT.
6091 212830   00970        LD     HL,CRSTOP+6  ;LOAD HL 3028H.
6094 BE       00980        CP     (HL)         ;COMPARE "/".
6095 20E8     00990        JR     NZ,ERASE     ;ERASE IF NOT.
6097 210364   01000        LD     HL,TABLE     ;LOAD STORAGE.
609A 010F00   01010        LD     BC,15        ;COUNT.
609D 3600     01020 LOOP3  LD     (HL),0       ;ZERO STORAGE.
609F 23       01030        INC    HL           ;NEXT LOCATION.
60A0 0B       01040        DEC    BC           ;DEC. COUNT.
60A1 79       01050        LD     A,C          ;SAVE COUNT.
60A2 B1       01060        OR     C            ;COMPARE FOR "0".
60A3 20F8     01070        JR     NZ,LOOP3     ;RETURN IF NOT.
60A5 3602     01080        LD     (HL),2       ;LOAD CONTROL REG.
60A7 3E0F     01090        LD     A,0FH        ;TURN OFF CURSOR.
60A9 CD3300   01100        CALL   33H          ;PRINT IT.
60AC 212930   01110        LD     HL,CRSTOP+7  ;LOAD HL 3029H.
60AF 010564   01120        LD     BC,TABLE+2   ;LOAD TENS YEARS.
60B2 7E       01130        LD     A,(HL)       ;GET CHARACTER.
60B3 02       01140        LD     (BC),A       ;LOAD TABLE.
60B4 23       01150        INC    HL
60B5 03       01160        INC    BC
60B6 7E       01170        LD     A,(HL)       ;GET CHARACTER.
60B7 02       01180        LD     (BC),A       ;LOAD TABLE.
60B8 212630   01190        LD     HL,CRSTOP+4  ;LOAD HL 3026H.
60BB 03       01200        INC    BC
60BC 7E       01210        LD     A,(HL)       ;GET CHARACTER.
60BD 02       01220        LD     (BC),A       ;LOAD TABLE.
60BE 23       01230        INC    HL
60BF 03       01240        INC    BC
60C0 7E       01250        LD     A,(HL)       ;GET CHARACTER.
60C1 02       01260        LD     (BC),A       ;LOAD TABLE.
60C2 03       01270        INC    BC
60C3 212330   01280        LD     HL,CRSTOP+1  ;LOAD HL 3023H
60C6 7E       01290        LD     A,(HL)       ;GET CHARACTER.
60C7 02       01300        LD     (BC),A       ;LOAD TABLE.
60C8 23       01310        INC    HL
60C9 03       01320        INC    BC
60CA 7E       01330        LD     A,(HL)       ;GET CHARACTER.
60CB 02       01340        LD     (BC),A       ;LOAD TABLE.
60CC E1       01350        POP    HL           ;SCREEN LOCATION.
60CD C5       01360        PUSH   BC           ;SAVE TABLE LOC.
60CE 222840   01370        LD     (CURPOS),HL  ;LOAD CURSOR.
60D1 214F63   01380        LD     HL,MESG7     ;GET MESSAGE.
60D4 CDC361   01390        CALL   VIDEO        ;PRINT IT.
60D7 21EB60   01400        LD     HL,ENTER1    ;LOAD TIME.
60DA 226660   01410        LD     (SAVE),HL    ;SAVE NEW JUMP.
60DD 212330   01420        LD     HL,CRSTOP+1  ;LOAD HL 3023H.
60E0 222840   01430        LD     (CURPOS),HL  ;LOAD CURSOR.
60E3 3E0E     01440        LD     A,0EH        ;TURN ON
60E5 CD3300   01450        CALL   33H          ;CURSOR.
60E8 C37F60   01460        JP     ERASE
              01470 ;*************************************************
              01480 ;*          TIME ROUTINE.                        *
              01490 ;*************************************************
60EB 212530   01500 ENTER1 LD     HL,CRSTOP+3  ;LOAD HL 3025H.
60EE 3E3A     01510        LD     A,':'        ;LOAD ":".
60F0 BE       01520        CP     (HL)         ;COMPARE FOR ":".
60F1 C27F60   01530        JP     NZ,ERASE     ;GO IF NOT
60F4 212330   01540        LD     HL,CRSTOP+1  ;LOAD HL 3023H.
60F7 C1       01550        POP    BC           ;POP TABLE COUNT.
60F8 03       01560        INC    BC
60F9 7E       01570        LD     A,(HL)       ;GET CHARACTER.
60FA 02       01580        LD     (BC),A       ;LOAD TABLE.
60FB 23       01590        INC    HL
60FC 03       01600        INC    BC
60FD 7E       01610        LD     A,(HL)       ;GET CHARACTER.
60FE 02       01620        LD     (BC),A       ;LOAD TABLE.
60FF 23       01630        INC    HL
6100 23       01640        INC    HL
6101 03       01650        INC    BC
6102 7E       01660        LD     A,(HL)       ;GET CHARACTER.
6103 02       01670        LD     (BC),A       ;LOAD TABLE.
6104 23       01680        INC    HL
6105 03       01690        INC    BC
6106 7E       01700        LD     A,(HL)       ;GET CHARACTER.
6107 02       01710        LD     (BC),A       ;LOAD TABLE.
6108 3E0F     01720        LD     A,0FH        ;TURN OF
610A CD3300   01730        CALL   33H          ;CURSOR.
610D E1       01740        POP    HL           ;SCREEN LOCATION.
610E 222840   01750        LD     (CURPOS),HL  ;LOAD CURSOR.
6111 3E10     01760        LD     A,10H        ;POSITION CURSOR
6113 CD3300   01770        CALL   33H          ;END OF LINE.
6116 3E1F     01780        LD     A,1FH        ;ERASE FROM CURSOR
6118 CD3300   01790        CALL   33H          ;TO END OF FRAME.
611B 21D43C   01800        LD     HL,SCREEN+214 ;SCREEN LOCATION.
611E 222840   01810        LD     (CURPOS),HL  ;LOAD CURSOR POS.
6121 217563   01820        LD     HL,MESG8     ;GET MESSAGE.
6124 CDC361   01830        CALL   VIDEO        ;PRINT MESSAGE.
6127 3E0E     01840        LD     A,0EH        ;TURN ON CURSOR.
6129 CD3300   01850        CALL   33H          ;DO IT.
612C 210364   01860        LD     HL,TABLE     ;LOAD TABLE.
612F E5       01870        PUSH   HL           ;SAVE LOCATION.
6130 CD2B00   01880 LOOP4  CALL   28H          ;CALL KEYBOARD.
6133 FE41     01890        CP     'A'
6135 2806     01900        JR     Z,LOOP5      ;IF "A" GO.
6137 FE50     01910        CP     'P'
6139 20F5     01920        JR     NZ,LOOP4     ;IF NOT "P" GO.
613B CBCE     01930        SET    1,(HL)       ;SET AM PM BIT.
613D CD3300   01940 LOOP5  CALL   33H          ;LOAD SCREEN AM PM.
6140 3E0F     01950        LD     A,0FH        ;TURN OF CURSOR.
6142 CD3300   01960        CALL   33H          ;DO IT.
6145 21D430   01970        LD     HL,SCREEN+268 ;SCREEN LOCATION.
6148 222840   01980        LD     (CURPOS),HL  ;LOAD CURSOR
614B 218A63   01990        LD     HL,MESG9     ;GET MESSAGE.
614E CDC361   02000        CALL   VIDEO        ;PRINT IT.
6151 3E0E     02010        LD     A,0EH        ;TURN ON CURSOR.
6153 CD3300   02020        CALL   33H          ;DO IT
6156 E1       02030        POP    HL           ;TABLE LOCATION.
6157 CD2B00   02040 LOOP6  CALL   28H          ;CALL KEYBOARD.
615A FE30     02050        CP     '0'          ;COMPARE FOR "0".
615C 38F9     02060        JR     C,LOOP6      ;LOOP IF BELOW "0".
615E FE34     02070        CP     '4'          ;COMPARE FOR "4".
6160 30F5     02080        JR     NC,LOOP6     ;LOOP IF ABOVE "3".
6162 CD3300   02090        CALL   33H          ;PRINT CHARACTER.
6165 17       02100        RLA                 ;MULTIPLE BY 2.
6166 17       02110        RLA                 ;MULTIPLE BY 4.
6167 86       02120        ADD    A,(HL)       ;ADD AM PM VALUE.
6168 77       02130        LD     (HL),A       ;RESTORE VALUE.
6169 3E0F     02140        LD     A,0FH        ;TURN OF CURSOR.
616B CD3300   02150        CALL   33H          ;DO IT.
              02160 ;*************************************************
              02170 ;*          INPUT DAY ROUTINE.                   *
              02180 ;*************************************************
616E 214830   02190        LD     HL,SCREEN+328 ;SCREEN LOCATION.
```

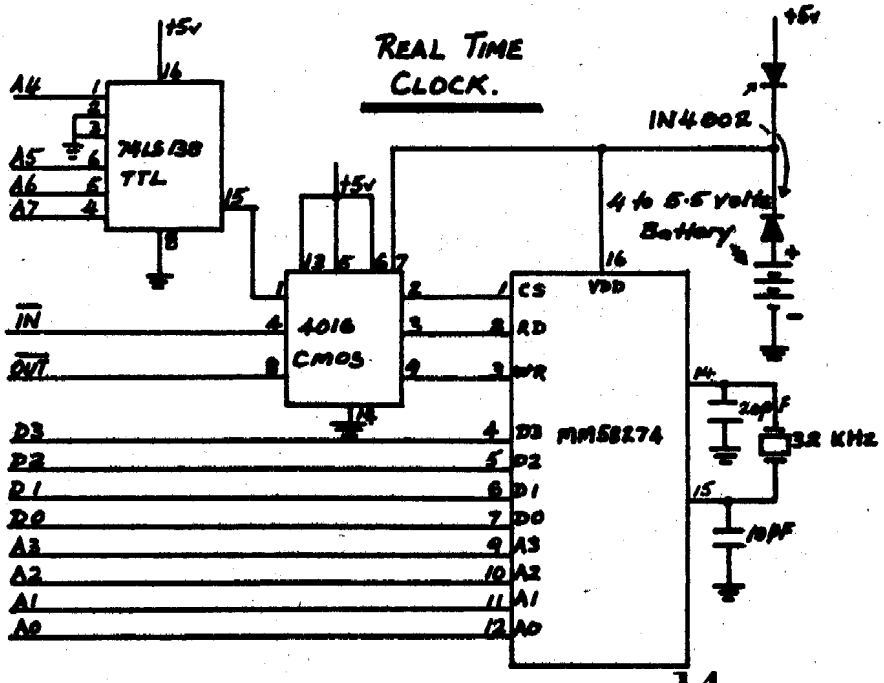
```
6171 222D40  0220          LD      (CURPOS),HL     ;LOAD CURSOR.
6174 21B163  0221          LD      HL,MESG10       ;GET MESSAGE.
6177 CDC361  0222          CALL    VIDEO           ;LOAD SCREEN.
617A 3E0E    0223          LD      A,0EH           ;TURN ON CURSOR.
617C CD3300  0224          CALL    33H             ;DO IT.
617F CD2800  0225  LOOP7   CALL    28H             ;CALL KEYBOARD.
6182 FE31    0226          CP      '1'             ;COMPARE FOR "1".
6184 38F9    0227          JR      C,LOOP7         ;LOOP IF BELOW "1".
6186 FE38    0228          CP      '8'             ;COMPARE FOR "8".
6188 30F5    0229          JR      NC,LOOP7        ;LOOP IF ABOVE "7".
618A 326464  0230          LD      (TABLE+1),A     ;LOAD TABLE WITH DAY.
618D CD3300  0231          CALL    33H             ;LOAD SCREEN.
6190 3E0F    0232          LD      A,0FH           ;TURN OFF CURSOR.
6192 CD3300  0233          CALL    33H             ;DO IT
             0234  ;*************************************************
             0235  ;*      ENTER TO START CLOCK ROUTINE.            *
             0236  ;*************************************************
6195 218430  0237          LD      HL,SCREEN+388   ;SCREEN LOCATINE.
6198 222D40  0238          LD      (CURPOS),HL     ;LOAD CURSOR.
619B 210463  0239          LD      HL,MESG11       ;LOAD MESSAGE.
619E CDC361  0240          CALL    VIDEO           ;PUT ON SCREEN.
61A1 AF      0241          XOR     A               ;CLEAR A.
61A2 D320    0242          OUT     (20H),A         ;CLEAR CONTROL REG.
61A4 D32F    0243          OUT     (2FH),A         ;CLEAR SETTING REG.
61A6 3E05    0244          LD      A,5             ;BITS 4 AND 0.
61A8 D320    0245          OUT     (20H),A         ;LOAD CONTROL REG.
61AA 211164  0246          LD      HL,TABLE+14     ;LOAD HL DAYS.
61AD 010F21  0247          LD      BC,0F21H        ;COUNT PORT C.
61B0 7E      0248  LOOP8   LD      A,(HL)          ;GET VALUE.
61B1 ED79    0249          OUT     (C),A           ;LOAD CLOCK.
61B3 2B      0250          DEC     HL              ;DEC TABLE.
61B4 0C      0251          INC     C               ;INC. CLOCK PORT.
61B5 10F9    0252          DJNZ    LOOP8           ;AGAIN 16 TIMES.
61B7 CD2800  0253  LOOP9   CALL    28H             ;CALL KEYBOARD.
61BA FE0D    0254          CP      0DH             ;COMPARE FOR ENTER.
61BC 20F9    0255          JR      NZ,LOOP9        ;LOOP IF NOT.
61BE AF      0256          XOR     A               ;ZERO A.
61BF D320    0257          OUT     (20H),A         ;START CLOCK.
61C1 18B9    0258          JR      READ            ;GO READ ROUTINE.
             0259  ;*************************************************
             0260  ;*          VIDEO WRITE ROUTINE.                 *
             0261  ;*************************************************
61C3 7E      0262  VIDEO   LD      A,(HL)          ;GET LETTER FROM MESS.
61C4 B7      0263          OR      A               ;TEST FOR END OF MESS.
61C5 C8      0264          RET     Z               ;GO IF END OF MESSAGE.
61C6 CD3300  0265          CALL    33H             ;PRINT CHARACTER.
61C9 23      0266          INC     HL              ;NEXT CHARACTER.
61CA 18F7    0267          JR      VIDEO           ;LOOP TILL ALL DONE.
             0268  ;*************************************************
             0269  ;*          CLOCK READ ROUTINE                   *
             0270  ;*          IN 32 CHR0 MODE                      *
             0271  ;*************************************************
61CC CDC901  0272  READ    CALL    CLEAR           ;CLEAR SCREEN.
61CF 3E17    0273          LD      A,17H           ;SELECT 32 CHR$
61D1 CD3300  0274          CALL    33H             ;MODE AND PRINT.
61D4 214830  0275          LD      HL,SCREEN+328   ;LOAD CURSOR
61D7 222D40  0276          LD      (CURPOS),HL     ;POSTION.
61DA 21E962  0277          LD      HL,MESG5        ;PRINT
61DD CDC361  0278          CALL    VIDEO           ;MESSAGE.
             0279  ;*************************************************
             0280  ;*          FIND DAY ROUTINE                     *
             0281  ;*************************************************
61E0 21543C  0282          LD      HL,SCREEN+84    ;LOAD CURSOR
61E3 222D40  0283          LD      (CURPOS),HL     ;POSTION.
61E6 0600    0284          LD      B,0
61E8 21FB62  0285          LD      HL,MESG6        ;LOAD DAYS.
61EB DB2E    0286          IN      A,(2EH)         ;GET DAY VALUE.
61ED E60F    0287          AND     0FH             ;CLEAR TOP BITS.
61EF 3D      0288          DEC     A               ;MAKE ZERO.
61F0 17      0289          RLA                     ;ROTATE
61F1 17      0290          RLA                     ;TWICE.
61F2 4F      0291          LD      C,A             ;LOAD C WITH VALUE.
61F3 17      0292          RLA                     ;ROTATE TWICE
61F4 17      0293          RLA                     ;MORE.
61F5 91      0294          SUB     C               ;SUBTRACT 0CH.
61F6 4F      0295          LD      C,A             ;LOAD VALUE.
61F7 09      0296          ADD     HL,BC           ;FIND DAY.
61F8 CDC361  0297          CALL    VIDEO           ;PRINT IT.
             0298  ;*************************************************
             0299  ;*          AM PM ROUNTINE                       *
             0300  ;*************************************************
61FB 21EA3C  0301          LD      HL,SCREEN+234   ;SCREEN LOCATION.
61FE 222D40  0302          LD      (CURPOS),HL     ;LOAD CURSOR.
6201 21FC63  0303          LD      HL,MESG12       ;GET MESSAGE.
6204 0600    0304          LD      B,0
6206 DB2F    0305          IN      A,(2FH)         ;CLOCK SET REGISTERS.
6208 E602    0306          AND     2H              ;AM OR PM SET.
620A 17      0307          RLA                     ;MOVE RIGHT.
620B 4F      0308          LD      C,A             ;SAVE IN C.
620C 09      0309          ADD     HL,BC           ;FIND AM OR PM.
620D CDC361  0310          CALL    VIDEO           ;PRINT VALUE.
             0311  ;*************************************************
             0312  ;*       LOAD DATE AND TIME ONTO SCREEN          *
             0313  ;*************************************************
6210 21E03C  0314  LOOP10 LD      HL,SCREEN+224   ;SCREEN POSITION.
6213 0E21    0315          LD      C,21H           ;LOAD PORT 21H.
6215 CD7B62  0316          CALL    GETTIM          ;DISPLAY IT.
6218 363A    0317          LD      (HL),':'        ;PRINT COLEN.
621A 2B      0318          DEC     HL
621B 2B      0319          DEC     HL
621C CD7B62  0320          CALL    GETTIM          ;DISPLAY NEXT.
621F CD7B62  0321          CALL    GETTIM          ;DISPLAY NEXT.
6222 363A    0322          LD      (HL),':'
6224 2B      0323          DEC     HL
6225 2B      0324          DEC     HL
6226 CD7B62  0325          CALL    GETTIM          ;DISPLAY NEXT.
6229 CD7B62  0326          CALL    GETTIM          ;DISPLAY NEXT.
622C 363A    0327          LD      (HL),':'
622E 2B      0328          DEC     HL
622F 2B      0329          DEC     HL
6230 CD7B62  0330          CALL    GETTIM          ;DISPLAY NEXT.
6233 CD7B62  0331          CALL    GETTIM          ;DISPLAY NEXT.
6236 2E90    0332          LD      L,090H          ;SCREEN LOCATION.
6238 CD7B62  0333          CALL    GETTIM          ;DISPLAY NEXT.
623B CD7B62  0334          CALL    GETTIM          ;DISPLAY NEXT.
623E 2E92    0335          LD      L,092H          ;SCREEN LOCATION.
6240 362F    0336          LD      (HL),'/'        ;LOAD SLASH.
6242 2E96    0337          LD      L,096H          ;SCREEN LOCATION.
6244 CD7B62  0338          CALL    GETTIM          ;DISPLAY NEXT.
6247 CD7B62  0339          CALL    GETTIM          ;DISPLAY NEXT.
624A 2E98    0340          LD      L,098H          ;SCREEN LOCATION.
624C 362F    0341          LD      (HL),'/'        ;LOAD SLASH.
624E 2E9C    0342          LD      L,09CH          ;SCREEN LOCATION.
6250 CD7B62  0343          CALL    GETTIM          ;DISPLAY NEXT.
6253 CD7B62  0344          CALL    GETTIM          ;DISPLAY NEXT.
6256 CD2800  0345          CALL    28H             ;CALL KEYBOARD.
6259 FE52    0346          CP      'R'             ;COMPARE RETRY.
625B 280A    0347          JR      Z,RETRY         ;GO RETRY.
625D FE45    0348          CP      'E'             ;COMPARE EXIT.
625F 20AF    0349          JR      NZ,LOOP10       ;GO IF NOT
6261 CDC901  0350          CALL    CLEAR           ;CLEAR SCREEN.
6264 C32D40  0351          JP      402DH           ;GO DOS.
6267 218960  0352  RETRY   LD      HL,ENTER        ;RESAVE ENTER.
626A 226660  0353          LD      (SAVE),HL       ;SAVE IT.
626D C36600  0354          JP      START           ;GO TO BEGINING.
             0355  ;*************************************************
             0356  ;*          READ CLOCK ROUTINE.                  *
             0357  ;*************************************************
6270 DB20    0358  GETTIM IN      A,(CONTRL)      ;CONTROL REGISTER.
6272 CB5F    0359          BIT     3,A             ;IS CLOCK READY.
6274 28FA    0360          JR      NZ,GETTIM       ;RETURN IF SO.
6276 ED78    0361          IN      A,(C)           ;CLOCK REGISTER.
6278 E63F    0362          AND     3FH             ;CONVERT TO NUMBERS.
627A 0C      0363          INC     C               ;NEXT PORT.
627B 77      0364          LD      (HL),A          ;LOAD SCREEN.
627C 2B      0365          DEC     HL              ;NEXT SCREEN LOCATION.
627D 2B      0366          DEC     HL              ;AGAIN.
627E C9      0367          RET
             0368  ;*************************************************
             0369  ;*                MESSAGES.                      *
             0370  ;*************************************************
627F 52      0371  MESG1   DEFM    'Real Time Clock setting program'
     65 61 6C 20 54 69 6D 65 20 43 6C 6F 63 6B 20 73
     65 74 74 69 6E 67 20 70 72 6F 67 72 61 6D
629E 00      0372          DEFB    0
629F 62      0373  MESG2   DEFM    'by David Kennedy'
     79 20 44 61 76 69 64 20 4B 65 6E 6E 65 64 79
62AF 00      0374          DEFB    0
62B0 45      0375  MESG3   DEFM    'Enter;(R)ead or (W)rite'
     6E 74 65 72 3B 28 52 29 65 61 64 20 6F 72 20 28
```

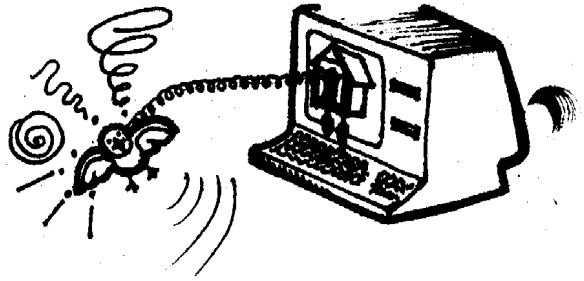
```
    57 29 72 69 74 65
62C7 00         03760         DEFB    0
62C8 49         03770 MESG4   DEFM    'Input Day/Month/Year,eg 00/00/00'
    6E 70 75 74 20 44 61 79 2F 4D 6F 6E 74 68 2F 59
    65 61 72 2C 65 67 20 30 30 2F 30 30 2F 30 30
62E8 00         03780         DEFB    0
62E9 28         03790 MESG5   DEFM    '(R)ETRY OR (E)XIT'
    52 29 45 54 52 59 20 4F 52 20 28 45 29 58 49 54
62FA 00         03800         DEFB    0
62FB 53         03810 MESG6   DEFM    'SUNDAY    '
    55 4E 44 41 59 20 20 20 20 20
6306 00         03820         DEFB    0
6307 4D         03830         DEFM    'MONDAY    '
    4F 4E 44 41 59 20 20 20 20 20
6312 00         03840         DEFB    0
6313 54         03850         DEFM    'TUESDAY   '
    55 45 53 44 41 59 20 20 20 20
631E 00         03860         DEFB    0
631F 57         03870         DEFM    'WEDNESDAY '
    45 44 4E 45 53 44 41 59 20 20
632A 00         03880         DEFB    0
632B 54         03890         DEFM    'THURSDAY  '
    48 55 52 53 44 41 59 20 20 20
6336 00         03900         DEFB    0
6337 46         03910         DEFM    'FRIDAY    '
    52 49 44 41 59 20 20 20 20 20
6342 00         03920         DEFB    0
6343 53         03930         DEFM    'SATURDAY  '
    41 54 55 52 44 41 59 20 20 20
634E 00         03940         DEFB    0
634F 20         03950 MESG7   DEFM    ' Input Hours:Minutes,eg 00:00
    49 6E 70 75 74 20 48 6F 75 72 73 3A 4D 69 6E 75
    74 65 73 2C 65 67 20 30 30 3A 30 30 20 20 20 20
    20 20 20 20
6374 00         03960         DEFB    0
6375 50         03970 MESG8   DEFM    'Press (A)M or (P)M )'
    72 65 73 73 20 28 41 29 4D 20 6F 72 20 28 50 29
    4D 20 3E
6389 00         03980         DEFB    0
638A 4E         03990 MESG9   DEFM    'Number of Years since last Leap Year )'
    75 6D 62 65 72 20 6F 66 20 59 65 61 72 73 20 73
    69 6E 63 65 20 6C 61 73 74 20 4C 65 61 70 20 59
    65 61 72 20 3E
63B0 00         04000         DEFB    0
63B1 49         04010 MESG10  DEFM    'Input: Sunday to Saturday 1 to 7 )'
    6E 70 75 74 3A 20 53 75 6E 64 61 79 20 74 6F 20
    53 61 74 75 72 64 61 79 20 31 20 74 6F 20 37 20
    3E
6303 00         04020         DEFB    0

6304 45         04030 MESG11  DEFM    'Enter with seconds at 00 to start clock'
    6E 74 65 72 20 77 69 74 68 20 73 65 63 6F 6E 64
    73 20 61 74 20 30 30 20 74 6F 20 73 74 61 72 74
    20 63 6C 6F 63 6B
63FB 00         04040         DEFB    0
63FC 41         04050 MESG12  DEFM    'AM '
    4D 20
63FF 00         04060         DEFB    0
6400 50         04070         DEFM    'PM'
    4D
6402 00         04080         DEFB    0
6403            04090 TABLE   EQU     $
6000            04100         END     START
00000 TOTAL ERRORS

BSPACE  6072    CLEAR   01C9    CONTRL  0020    CRSTOP  3022    CURPOS  4020
DODCB   4010    ENTER   6089    ENTER1  60EB    ERASE   607F    GETTIM  6270
LOOP    602C    LOOP1   6055    LOOP10  6210    LOOP2   6050    LOOP3   6070
LOOP4   6130    LOOP5   613D    LOOP6   6157    LOOP7   617F    LOOP8   6180
LOOP9   6187    MESG1   627F    MESG10  63B1    MESG11  63D4    MESG12  63FC
MESG2   629F    MESG3   62B0    MESG4   62CB    MESG5   62E9    MESG6   62FB
MESG7   634F    MESG8   6375    MESG9   638A    READ    61CC    RETRY   6267
SAVE    6066    SCREEN  3C00    START   6000    TABLE   6403    VIDEO   61C3
```

```
10 REM* BASIC PROGRAM TO INTIALIZE CLOCK
20 CLS:CLEAR100:
30 DIMD$(7)
40 INPUT"DO YOU WANT -12- OR -24- HOUR FORMAT";B$
50 IF B$="24"THENB1=B1+1:GOTO100
60 CLS:INPUT"DO YOU WANT -AM- OR -PM- (A/P)";P$
70 IFP$="P"THENB1=B1+2
80 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY
90 DATA SATURDAY
100 FORI=0TO6:READD$(I):NEXTI:
105 CLS:PRINT"ENTER Hours & Minutes - H10,H1,M10,M1";
110 INPUT H1,H,M1,M
115 CLS:PRINT"ENTER Year, Month & Date - Y10,Y1,M10,M1,D10,D1";
120 INPUT Y1,Y,MB,ML,D1,D
130 CLS:INPUT"ENTER DAY CODE (SUN TO SAT =0 -6)";W
140 CLS:INPUT"ENTER YEAR - LEAP YEAR VALUE";LY
150 IFLY>3THENGOTO140
160 E=LY*4+B1
170 OUT32,5
180 OUT47,B1
190 OUT34,0:OUT35,0
200 OUT36,M:OUT37,M1
210 OUT38,H:OUT39,H1
220 OUT40,D:OUT41,D1
230 OUT42,ML:OUT43,MB
240 OUT44,Y:OUT45,Y1
250 OUT46,W
260 OUT47,E
270 CLS:INPUT"ENTER TO START CLOCK";
280 OUT32,1
290 CLS
300 IFINP(32)=0THEN310
310 PRINT@1,D$(INP(46)AND15)
320 REM* READ CLOCK'S TIME
330 IFINP(32)=0THEN340
340 PRINT@50,INP(33)AND15
350 PRINT@44,INP(35)AND15:PRINT@46,INP(34)AND15;":";
360 PRINT@38,INP(37)AND15:PRINT@40,INP(36)AND15;":"
370 PRINT@32,INP(39)AND15:PRINT@34,INP(38)AND15;":";
380 PRINT@24,INP(45)AND15:PRINT@26,INP(44)AND15
390 PRINT@18,INP(43)AND15:PRINT@20,INP(42)AND15;"/"
400 PRINT@12,INP(41)AND15:PRINT@14,INP(40)AND15;"/"
410 IF(INP(47)AND3)=2THENPRINT@55,"P.M"
420 IF(INP(47)AND3)=1THENPRINT@53,"24 HR.MODE"
430 IF(INP(47)AND3)=0THENPRINT@55,"A.M"
440 GOTO330
```





MODEL I LEVEL II LOWERCASE ON POWERUP

by Jack Decker
‡(with help from Bob Seaborn)

I recently received a letter from Bob Seaborn, requesting information on modifying the Model I ROM to provide for correct handling of lowercase characters on powerup without using a separate lowercase driver routine (usually part of the Disk Operating System). This would be useful in situations where a user does not have disk drives (are there any such systems left?) or where a program bypasses the DOS and directly calls the ROM routines (not good practice, but it is done occasionally).

Bob had recalled an article that appeared in the November/December 1981 issue of 80-U.S. Journal, which used two added integrated circuits to modify ONE memory location in the video driver routine in the ROM. The change was to 0473H, where a 38H (JR C instruction) is changed to an 18H (JR instruction). So, in effect we are adding two integrated circuits to change one bit in memory (bit 5 of location 0473H, from 1 to 0).

Well, that change WILL work, but there is a much better way. Rather than hacking up your computer to change one bit, why not simply change the ROM? I wrote up a first draft of this article and sent it up to Bob, who actually performed the modification I'm about to describe on his Model I. He was kind enough to send back several comments, suggestions, and additions to my original article, and many of Bob's comments have been incorporated into this article. If you're having difficulty with this mod, you might want to contact Bob Seaborn at Post Office Box 1741, Saskatoon, Saskatchewan, Canada S7K 3S1 (telephone: (306) 343-1305, MCI Mail ID: 268-7906). As usual, the information in this article is provided for experimental purposes and is intended for use by experienced hardware hackers only, and is NOT guaranteed to be error-free. We assume NO liability whatsoever for any damage that your computer and/or data may sustain if you actually attempt to install or use this modification. That said, here's how to replace your Radio Shack ROM with an EPROM that will provide much better lowercase support than the one-byte patch mentioned above.

If you have the three IC ROM set (using the satellite board that is mounted to the upper side of the keyboard PCB with double-stick tape), the change is relatively easy. You simply need to swap the "A" ROM with a 2532 EPROM that has been properly programmed (the 2532 EPROM is pin-compatible with the 4K ROM IC's). Note that not all 2532 EPROMs are the same, but I do know that a Texas Instruments 2532 (not a 25L32!!) will work correctly. In the three-chip ROM set, the TI 2532 can (when properly programmed) substitute for any of the three ROM chips (this may be a handy bit of information if you ever need to repair a "dead" Model I). If the TI 2532 EPROM or an exact equivalent is not available, an Intel (or equivalent) 2732 EPROM can be substituted, but you will have to do a bit of rewiring. When using the 2732, pin 21 (A11 on 2732 and Vpp on 2532) must be reversed with pin 18 (CE on 2732 and A11 on 2532). This is because of the difference between the TI pinout and the INTEL pinout. Naturally, it's preferable to use the 2532 if at all possible, since no rewiring is necessary when that chip is used.

If you have the two IC ROM set (no satellite ROM board), you can use a 2764 EPROM in place of the ROM A (the 8K one with a part number ending in '64). A 2532 EPROM (or 2732 rewired as described in the previous paragraph) works in place of the ROM B (which ends in a '32); as already noted, the 2532 is pin compatible with the ROM. However, it is just a little more difficult to use the 2764 EPROM in place of the 8K ROM A, because the 2764 is a 28 pin chip while the 8K ROM is only a 24 pin chip. This difference in IC packaging creates less of a problem than you might at first think. Bob studied the pinout diagrams of the two IC's and came up with an easy way to make a 28 pin to 24 pin adapter plug. You plug the 2764 EPROM into the adapter and then plug the adapter into the socket formerly occupied by the 8K Radio Shack ROM A. Here are Bob's instructions for constructing the adapter (with a few added comments in parenthesis for the technically minded):

1) Get two IC sockets, a 24 pin one and a 28 pin one.

2) Jumper together pins 1, 28, 27, and 26 on the 28 pin socket (this will put +5 volts on these pins, obtained from pin 26 of the 28 pin socket which will be plugged into pin 24 of the 24 pin socket).

3) Jumper together pins 14 and 20 on the 28 pin socket (pin 14 will be connected to ground).

4) Bend pins 20 and 23 on the 28 pin socket at right angles to the socket.

5) Plug the 28 pin socket into the 24 pin socket so that pins 1, 2, 27, and 28 of the 28 pin socket are not inserted in the 24 pin socket (pin 3 of the 28 pin socket should plug into pin 1 of the 24 pin socket, etc.).

6) Jumper pin 2 on the 28 pin socket to pin 20 on the 24 pin socket.

7) Jumper pin 23 on the 28 pin socket to pin 18 on the 24 pin socket (A11).

Whenever you purchase an EPROM to replace a TRS-80 ROM chip, be sure to get the highest speed (lowest access time in nanoseconds) unit that you reasonably can (without spending huge amounts of money), especially if you have installed a high-speed modification in your Model I. I once tried using a 2532 EPROM in place of the "A" ROM, and found that although it worked just great at normal CPU speed, it refused to work at more than 150% of normal CPU speed (but, strangely enough, this failure only occurred after the computer had warmed up for a few minutes). Bob suggests that changing the values of resistors R61 and R68 from 4.7K to 1K might have helped, but I have no way to verify that now.

Anyway, getting back to the ROM lowercase mod, the first thing you will need to do is to perform a standard lowercase modification to your Model I, which is to say that you will need to add an eighth video RAM IC and perhaps a new character generator chip. This information has been covered in great detail elsewhere, so I won't go into it here (see my letter to the editor in the August, 1983 issue of BASIC COMPUTING, or the last few paragraphs of this article).

If you have lowercase available under the DOS or by loading a special driver routine into memory, then the above mods have already been installed. This brings us back to the subject of this article - modifying the ROM to make lowercase permanently available.

You will first need to find someone with an EPROM programmer that's willing to help - try your local computer user group; a local college or university with an electronics lab; or a firm that specializes in electronic design or computer repair (or see the last few paragraphs of this article). Take in your "A" ROM chip and your blank EPROM. Whoever burns the EPROM for you should first erase it under ultraviolet light, then copy your "A" ROM to the EPROM. However, before the copy is made, you will first want to make the necessary change(s). You can make the above-mentioned change to memory location 0473H, or you can make a more involved series of changes as documented in the assembly-language source code listing below.

The source code below shows the patches to the "A" ROM which provide the following features: Lowercase is enabled at power-up, SHIFT-ZERO toggles a "CAPS lock" feature as on the Model III, the cursor character can be defined by POKEing the desired character to memory location 4023H (also as in the Model III), the cassette tape leader is shortened to permit faster reading of data from tape, and all improvements found in what used to be called the "NEW" Model I ROM are included (such as built-in keyboard debounce, improved cassette read timing, etc.). The "RADIO SHACK LEVEL II BASIC" (or "R/S L2 BASIC") message is deleted to make room for these patches.

A special note about the cassette read timing patches at 0248H and 024FH. Although they are the values used in the "NEW" Model I ROMs, some folks have experienced difficulty in loading tapes after installing them. Therefore, I suggest that you do NOT use these two patches if 1) You have little or no difficulty loading tapes using the old values, or 2) You are using an expansion interface and/or disk system, or 3) You have the Radio Shack hardware fix (the "XRX-III" modification - a small PC board that is usually also mounted with double-stick tape onto the keyboard PCB), unless you disconnect the XRX-III mod (recommended, see details in the next paragraph). If any of these conditions apply to you, I suggest using the old timing values (the byte at 0249H should be 41H, and the byte at 0250H should be 76H).

If you would like to remove the XRX-III modification, Bob informs me that it can be done as follows. There are only six wires to remove and one cut trace to restore. First, you remove the red wire from Z25 pin 14, the black wire from Z4 pin 7, the yellow wire from Z43 pin 9, and the blue wire from Z24 pin 12. At this point, the only remaining connections between the XRX-III mod and the main circuit board will be the green wire on Z4 pin 10 and the violet wire on Z24 pin 9. Between these two

connections there is a trace that was cut during the XRX-III installation. Restore the trace by scraping the solder from each side of the cut and carefully soldering a short piece of hookup wire across the cut. Then remove both the remaining wires. This completes the removal of the XRX-III cassette modification. If you can't seem to locate the modification in your computer, keep in mind that it was installed only in some of the "old ROM" Model I's (that display 'MEMORY SIZE'), and was never installed in the "new ROM" units that show 'MEM SIZE' when first reset.

There are a couple of caveats to be observed when using the patches shown below. For one thing, if you have a program that is so rude as to make calls directly into the guts of the keyboard driver routine (very few programs do this, and if a program is Model I/III compatible it almost certainly will not do this), it will probably not work correctly with this mod installed. Calls made to the normal starting point of the keyboard driver are OK. Then, too, the use of the 4023H memory location for cursor character storage may conflict with other Model I programs that also expect that memory location to be unused (again, this problem will probably never occur when a Model I/III compatible program is used). I have never had a problem with a conflict for the use of 4023H, but apparently others either have had such a problem, or were worried about the possibility of a conflict. If you don't want to take a chance, delete lines 1020 and 1030, and remove the instructions in lines 910 and 920, replacing them with two NOP's and a LD (HL),5FH instruction (or use a different default cursor character if you prefer).

Now a couple of paragraphs for NEWDOS/80 version 2.0 users. Bob has used this mod with NEWDOS/80 and reports that it is best to set the NEWDOS/80 SYSTEM options BF=N and BG=N. Otherwise the NEWDOS lowercase driver and the ROM lowercase driver compete and make it nearly impossible to to shift case using SHIFT-ZERO.

Also, here are a couple of zaps that Bob has provided for NEWDOS/80 version 2.0 users that have installed this modification. The first loads the cursor character selected by the SYSTEM option BI into the user programmable cursor character storage location (4023H) when the disk is rebooted:

Change SYS0/SYS,11,40 from B7 28 03 to 32 23 40

The following zap permits the LC,(Y/N) command to work with the EPROM lowercase driver installed.

Change SYS3/SYS,00,A1
from 32 B4 45 AF C9 CD EF 4D 28 F5 3E C9 18
to 32 19 40 AF C9 CD EF 4D 20 F5 3E 01 18

When the above zap is used, the BASIC caps lock control is changed to POKE 16409,0 to set mixed (upper/lower) case, and POKE 16409,1 to set uppercase only (the same as under Model III BASIC). This location replaces the 17644 (45B4H) location that Apparat documented (reluctantly) in Model I ZAP 082.

Now, then - for those of you who don't like to hack around and want to make the whole process as cut and dried as possible, I suggest you order the following kits from Don McKenzie: 1) If you have never had the hardware lowercase modification installed, order a JACKGEN 3 WITH OPTIONAL PRINTED CIRCUIT BOARD (total cost $18.90). Then, if you have the three-chip ROM set and do not have, or are willing to disconnect, the XRX-III modification, you may also order a DONPATCH EPROM (the Model I "A" ROM with the extra routines installed, MINUS the 4023H cursor character storage) for $19.90. If you feel you simply must have the user-definable cursor character, contact Don first and ask him if he'll burn you a DONPATCH with the code shown below included - I suspect he will do it, but don't come crying later if you discover a software compatibility problem. All items come with installation instructions that assume at least some hardware experience (they are not designed for the rank beginner. If you've never done any modifications of electronic circuits, you will need some help from an experienced technician or hardware hacker!).

The prices shown above are in Australian dollars and YOU MUST ADD POSTAGE if you live outside Australia. Postage to North America is $4.45, to Europe it's $4.70, and to New Zealand and Papua New Guinea it's $2.80, again all in Australian dollars. These amounts actually cover Air Mail postage, foreign currency exchange and stamp duty. Add up the cost of your order, add the appropriate postage rate, and you will have the total in Australian dollars. Then call your bank and ask them to convert that amount to your local currency and send a check in that amount, payable to Don McKenzie. Don ships most orders within 48 hours of receipt and the normal turnaround time on an order from the U.S.A. is three weeks, counting the time it takes for the post office to do its thing. This assumes that you send your order to Don via Air Mail, of course. If you want to write to Don for information, be sure to enclose some money for postage - NOT foreign stamps as the Australian post office won't accept them! Please keep in mind that Don is a hobbyist doing this in his spare time, not a big company that can afford to send out free catalogs.

That said, here's Don's address: Don McKenzie, 29 Ellesmere Crescent, Tullamarine, Victoria 3043, AUSTRALIA. You can also telephone Don at (03) 338 6286 (from North America dial 011+61+3+338 6286, on your nickel of course, and please remember the time difference so you don't wake Don at 3 A.M.). When ordering, be sure to specify exactly what type of computer you have (TRS-80, PMC-80, SYSTEM-80). Don also has many other interesting hardware products including a SPEEDON bare printed circuit board which, when stuffed with about $5 worth of commonly-available parts, will speed up a Model I CPU to as much as 5.32 MHz, with or without added "wait states" (if your system isn't capable of running that fast, you can select lower speeds that are still faster than the normal 1.77 MHz speed). SPEEDON slows down automatically for disk operations and you can command it to run at normal or fast speed by outputting a value to port 254 (or using the LDOS "SYSTEM (FAST)" or "SYSTEM (SLOW)" commands). The price of the bare board with full instructions is $17.90 - add an additional $2.50 and Don will also throw in the resistors, capacitors, and LED used in the project (but not the I.C.'s. Remember, these prices are Australian dollars, and you must add postage as shown above).

Now, without further ado, here are the patches to the 4K "A" ROM of the Model I. Note that if you are using the 8K (2764) EPROM, you may wish to also add the "new ROM" changes between 1226H and 1265H, which correct an invalid test for a double precision number. These are not listed below, but are given in Appendix III of TRS-80 ROM Routines Documented.

```
0059            0100        ORG     0059H
0059 00         0110        DEFB    00H         ;SHIFT DOWN-ARROW NOW CTL
                0120
0085            0130        ORG     0085H
0085 21FF00     0140        LD      HL,00FFH    ;POINT HL TO "MEM SIZE"
                0150
00FC            0160        ORG     00FCH
00FC C3191A     0170        JP      1A19H       ;SKIP "R/S L2 BASIC" MSG
00FF 4D         0180        DEFM    'MEM SIZE'  ;NEW LOCATION FOR MSG
     45 4D 20 53 49 5A 45
0107 00         0190        DEFB    00H         ;MESSAGE TERMINATOR
0108 DD7404     0200        LD      (IX+04H),H  ;SAVE CURSOR LOCATION MSB
010B 79         0210        LD      A,C         ;GET ORIGINAL CHARACTER
010C C9         0220        RET                 ;EXIT VIDEO DRIVER
010D 2807       0230        JR      Z,0116H     ;IF "@" WAS PRESSED
010F DDCB0446   0240        BIT     0,(IX+04H)  ;CHECK SHIFT-LOCK FLAG
0113 C0         0250        RET     NZ          ;IF "ALL CAPS" MODE
0114 EE20       0260        XOR     20H         ;MAKE LWR CASE (20H-3FH)
0116 CB00       0270        RLC     B           ;SHIFT KEY FROM BIT 7
0118 D0         0280        RET     NC          ;IF SHIFT KEY NOT PRESSED
0119 EE20       0290        XOR     20H         ;REVERSE CASE
011B C9         0300        RET                 ;TO 042BH
011C C5         0310        PUSH    BC          ;KEYBOARD
011D 010005     0320        LD      BC,0500H    ;  DEBOUNCE
0120 CD6A00     0330        CALL    006AH       ;    ROUTINE
0123 C1         0340        POP     BC          ;      FROM
0124 0A         0350        LD      A,(BC)      ;        "NEW"
0125 A3         0360        AND     E           ;          ROM
0126 C0         0370        RET     Z
0127 7A         0380        LD      A,D
0128 07         0390        RLCA
0129 07         0400        RLCA
012A C3FE03     0410        JP      03FEH
                0420
0248            0430        ORG     0248H       ;SEE TEXT BEFORE USING
0248 0660       0440        LD      B,60H       ;NEW CASSETTE READ TIMING
                0450
024F            0460        ORG     024FH       ;SEE TEXT BEFORE USING
024F 0685       0470        LD      B,85H       ;NEW CASSETTE READ TIMING
                0480
0287            0490        ORG     0287H
0287 0653       0500        LD      B,53H       ;MOD III LENGTH CASS LEAD
                0510
02E2            0520        ORG     02E2H
02E2 20ED       0530        JR      NZ,02D1H    ;"SYSTEM" COMMAND ROUTINE
```

```
02E4 23         00540           INC     HL              ;  CHANGE FROM "NEW" ROM
                00550
03FB            00560           ORG     03FBH
03FB C31C01     00570           JP      011CH           ;TO NEW KEYBOARD DEBOUNCE
                00580
0410            00590           ORG     0410H
0410 FE20       00600           CP      20H             ;ALPHABETIC OR CNTL CHAR?
0412 3015       00610           JR      NC,0429H        ;GO IF NOT ALPHA/CONTROL
0414 CB00       00620           RRC     B               ;SHIFT INTO CARRY & BIT 7
0416 3009       00630           JR      NC,0421H        ;IF NOT SHFT-ALPHA OR CTL
0418 57         00640           LD      D,A             ;SAVE CHAR IN D REGISTER
0419 3A4038     00650           LD      A,(3840H)       ;GET DOWN-ARROW KEY ROW
041C E610       00660           AND     10H             ;MASK OUT OTHER KEYS
041E 282C       00670           JR      NZ,044CH        ;IF CONTROL CHARACTER
0420 7A         00680           LD      A,D             ;RESTORE ORIG ALPHA CHAR
0421 B7         00690           OR      A               ;SET Z FLAG IF "@" KEY
0422 CD9D01     00700           CALL    019DH           ;GET CASE OF CHARACTER
0425 C640       00710           ADD     A,40H           ;OFFSET (CHAR = 40H-7FH)
0427 1822       00720           JR      044BH           ;GO TO END ROUTINE
0429 D630       00730           SUB     30H             ;0-9, PUNCTUATION OFFSET
                00740
0437            00750           ORG     0437H
0437 DC7304     00760           CALL    C,0473H         ;IF SHIFT KEY PRESSED
043A C34B04     00770           JP      044BH           ;GO TO END ROUTINE
                00780
0471            00790           ORG     0471H
0471 188A       00800           JR      04FDH           ;SKIP BAD PART VIDEO DRVR
0473 EE10       00810           XOR     10H             ;COMPLEMENT BIT 4
0475 FE20       00820           CP      20H             ;WAS SHIFT-ZERO PRESSED?
0477 C8         00830           RET     NZ              ;IF NOT SHIFT ZERO
0478 DD3404     00840           INC     (IX+04H)        ;COMPLEMENT BIT 0 OF FLAG
047B AF         00850           XOR     A               ;RETURN WITH NO CHARACTER
047C C9         00860           RET                     ;TO 043AH
                00870
0488            00880           ORG     048BH
0488 2807       00890           JR      Z,0494H         ;CHANGE FORWARD JUMP
048A DD7205     00900           LD      (IX+05H),D      ;SAVE CHR COVERED BY CRSR
048D DD5606     00910           LD      D,(IX+06H)      ;GET CURSOR CHARACTER
0493 72         00920           LD      (HL),D          ;DISPLAY IT
0494 DD7503     00930           LD      (IX+03H),L      ;SAVE LSB CURSOR LOCATION
0497 C39801     00940           JP      0198H           ;TO VIDEO DRIVER PATCH
                00950
0683            00960           ORG     0683H
0683 28F1       00970           JR      NZ,0676H        ;NEW ROM POWER-UP FIX
                00980
06EB            00990           ORG     06EBH
06EB 01         01000           DEFB    01H             ;DEFAULT SHIFT-LOCK ON
                01010
06F5            01020           ORG     06F5H
06F5 5F         01030           DEFB    5FH             ;DEFAULT UNDERSCORE CRSR
                01040
0EF2            01050           ORG     0EF2H
0EF2 D7         01060           RST     10H             ;FIX ROM BUG (WAS INC HL)
                01070
0000            01080           END
00000 TOTAL ERRORS
```

These patches make the period an acceptable input from the keyboard, the slash still remains a valid input and some other characters will also be acceptable, they only affect the character separating MM DD YY and therefore the date routine still rejects invalid dates. The patches may be typed in from the keyboard, exactly as shown above, or put into a BUILD or FIX file for patching more than one copy of the DOS. The normal precautions apply, namely try them out on a backup copy of the DOS before applying them to a good disk and test them out.

MORE PATCHES TO TRSDOS

by Reg Birks

[Reprinted from the Adelaide Micro-User News, 36 Sturt Street, Adelaide, South AUSTRALIA 5000]

When answering the Date? prompt on power up, I find it more convenient to use the Numeric Keypad, unfortunately there is no slash (/) character on the keypad. The answer is quite simple however, patch the DOS to accept the period (.) character which is on the keypad.

Set out below are the patches I have made to TRSDOS 1.3, 6.0 & 6.1

```
TRSDOS 1.3
PATCH *0 (ADD=4EC3,FIND=2F,CHG=30)
PATCH *0 (ADD=4FB8,FIND=28,CHG=38)

TRSDOS 6.0.x
PATCH SYS0/SYS.DBO (D0E,A4=30:F0E,A4=2F)
PATCH SYS0/SYS.DBO (D10,7B=30 03 18 DD:F10,7B=28 DF 18 04)

TRSDOS 6.1.x
PATCH SYS0/SYS.DBO (D0E,B3=30:F0E,B3=2F)
PATCH SYS0/SYS.DBO (D10,AF=30 06 18 DD:F10,AF=28 DF 18 04)
```

FIX YOUR BLOOMING SCREEN!!

by John Phillipp
Information provided by Mike Santana

[Reprinted from The Interface newsletter of the San Gabriel Valley TRS-80 Users Group.]

Well, friends, SAGATUG - that is to say, Mike Santana - bailed me out again.

Those of you who have been following the various stories of my Model 4s and 4Ps may remember a problem I had with the screen when I first got my 4P. The picture would get larger and smaller, then larger again, while the cooling fan ran faster, and slower and faster. Finally, the computer would give up entirely and reboot itself, committing murder on whatever I was working on at the time. I sent the computer back to Displayed Video, from whence it came, and they fixed it... or at least I thought they had. Anyway, the computer worked fine for about a year.

The problem, which is called screen "blooming", came back about six weeks ago with a vengeance. The 4P couldn't be used. You may have seen me wandering around the last SAGATUG meeting, looking for someone who knew how to fix it. George Fisher said Mike had the fix, and Mike - bless his heart - not only sent me the directions to fix the power supply, but he sent me the resistors I needed as well!!

I can't enclose resistors with each issue of the INTERFACE, but I think the fix should be made available to everyone who has an old 4P with the problem of a blooming screen.

THESE MODIFICATIONS ARE TO ELIMINATE SCREEN BLOOMING ON THE MODEL 4P

Note: the following resistor modifications pertain ONLY to the Tandy power supply. Do not install on the Aztec unit.

Parts list:
- 2 - 1000 (1K) ohm ½ watt resistors
- 2 - 470 ohm ½ watt resistors
- 1 - 68 ohm ½ watt resistors

Two of the resistors (1KΩ and 68Ω) are part of a revised version of Tech Tip 4P:4 (Rev: August 10, 1984). To add this modification, replace R-7 with the 1KΩ (brown, black, red) resistor, and replace R-35 with the 68Ω (blue, grey, black) resistor.

The remaining resistors are used in a second modification. To install it, remove the trim pot R-15 (R-15 is located next to the metal shield/heat sink on the circuit board. It looks like a tight squeeze, but if the solder is removed from its three connections on the solder side of the board with solder wick, R-15 can be easily lifted up and off the board).

Replace R-15 with the two 470Ω (yellow, violet, brown) resistors, joining them where the center tap on the trim pot was.

Check the 5 volt (V1) for a tolerance of 4.95 to 5.25 volts. To trim the power supply you may need to add the remaining 1KΩ (brown, black, red) resistor. To raise the output, locate R-14 and install the 1KΩ resistor across the 510Ω (green, brown, black) resistor closest to R-14. To lower the output, locate R-16 and install the 1KΩ resistor across the 510Ω (green, brown, black) resistor closest to R-16.

I put the modification in my 4P over two weeks ago, and the screen has been steady as a rock. Now, if I could just get Mike to write a short article explaining exactly what was wrong with the 4P power supply, and why changing a few resistors fixed it...

[NORTHERN BYTES editor's note: Although I can't explain the reasoning behind the Tech Tip modification, it sounds as though the major part of the problem was caused by trim pot R-15, which apparently was of such low quality that its resistance jumped all over the place. Cleaning and/or replacing R-15 might also have helped (at least for a while), but the fix proposed above sounds much more reliable.]

TELECOMMUNICATIONS NEWS

Before I begin this issue's column, here's a note for MCI Mail users. In the past we've had two MCI Mail accounts, one for The Alternate Source and one for Northern Bytes. We don't really need both accounts, and the duplication creates extra expense for us, so from now on please use MCI Mail ID number 109-7407 to contact either TAS or Northern Bytes. Please note that I will get your correspondence just as fast on this account as I did on the other. Of course, this also means that our MCI Telex number has changed, the new number is 6501097407 (answerback 6501097407 MCI).

A few issues back I mentioned that some phone companies have implemented a feature where you can temporarily disable Call Waiting, if you have that service. This is essential if you are making a modem call, since an incoming "Call Waiting" call momentarily interrupts the signal on your phone line, which will at least garble the data being transmitted on the line, and will likely cause one or both of the modems (yours or the one at the computer you're talking to) to hang up.

Well, it turns out that most of the Bell Operating Companies will receive this feature automatically as part of the software upgrade to convert to "equal access" (the service that lets you select a carrier other than AT&T to handle your calls when you dial "1" plus the area code and number). So, even if this feature hasn't been officially announced in your area, it may still be available. If it is, you can use it as follows:

1) If you have a touch-tone line, dial "*70". If you don't have touch-tone service, or if your auto-dial modem can't produce the "*" tone from the touch-tone keypad, dial "1170" instead.

2) Then simply dial the number you want to call. Call Waiting will be suspended for the duration of the call. It will be re-enabled when you hang up, even if the call doesn't go through or you get a busy signal.

Note that this doesn't help a bit on incoming calls, so you probably would not want to have Call Waiting service if you frequently receive modem calls (you certainly wouldn't want it on a BBS line, for example). But the average computer user that occasionally uses the family phone line to call up Compuserve or the local BBS is no longer forced to do without Call Waiting.

Last issue I mentioned the new Fastlink™ modem that operates at 10,000 bits per second over ordinary, dial-up telephone lines (™Fastlink is a trademark of DCA/Telebit Data Systems, and DCA is a trademark of Digital Communications Associates, Inc.) This modem is revolutionary in that it will even work over cruddy lines that would give regular modems a lot of problems. You've probably been told for years that 4800 bps (or 2400 bps or 1200 bps) was the upper limit for reliable communications over a standard telephone line, and to run at that blinding speed you'd need a very "clean" telephone line - no static, no glitches, no noise of any kind to disrupt communications. So how does the Fastlink manage to operate at speeds more than twice as fast as the former supposed upper limit?

To answer that question, you have to realize that conventional modems transmit bits serially - one after the other. A binary zero is called a "space" and a binary zero is called a "mark". Thus, to transmit a full eight bits of one byte, we need to send eight signals (some combination of eight "marks" and "spaces"), one right after the other (in actual practice, there will usually be two or three extra bits sent for each byte; these are called "start" and "stop" bits).

A standard modem uses one audio frequency to represent a "space", and another to represent a "mark". When two modems are connected to each other, one modem will use one pair of frequencies, and the other will use a different pair (if this were not the case, each time a modem transmitted a bit, it would scramble the incoming signal from the other computer). We normally refer to one pair of frequencies as the "originate" signal, and the other as the "answer" signal. Here in the United States and Canada, modems transmitting on the originate frequencies use an audio frequency of 1070 Hz to represent a "space", and 1270 Hz to represent a "mark". If the modem is transmitting on the answer frequencies, it uses 2025 Hz and 2225 Hz as the "space" and "mark" frequencies, respectively. These frequencies are known as the Bell standard frequencies (named after the old Bell System, which developed the first modems - after all, at the time Ma Bell didn't let anyone else connect anything to her lines anyway, which is just the sort of behavior that got her in trouble with the Justice Department). Other countries use the CCITT standard (which uses different frequencies - under the CCITT standard, 1180 Hz represents a "space" and 980 Hz a "mark" in originate mode. In the answer mode, 1850 Hz and 1650 Hz represent "space" and "mark" respectively), which is why you can't take your U.S. modem overseas and use it to connect to a bulletin board or packet network (quite apart from local regulations which would probably also forbid you to do so).

The point is that out of the entire audio spectrum, only four frequencies are used - two going in one direction, and two travelling in the reverse direction. But a standard telephone line is usually capable of carrying audio signals with frequencies as high as 3000 Hz (usually higher yet on local circuits). It's quite obvious that many more frequencies could be used simultaneously.

Up until now, modem communication has been treated like a dancing bear. When one considers a dancing bear, one does not critique how well the bear dances, one simply marvels that the bear is able to dance at all. Similarly, computer users have been enchanted by the fact that computer-to-computer communication over telephone lines is even possible, and until recently, no one has really questioned whether there could be any improvement in performance by using a different method of transmitting the data. That's not to say that we haven't tried to push the conventional system to the extreme limit of transmission speed, but since we've limited ourselves to using only four frequencies out of a much larger audio spectrum, we've ignored the most obvious method of increasing telecommunications speed.

Permit me to use an analogy. Suppose that you had an AM radio, but that there was only one radio station transmitting Morse Code, and furthermore, it transmitted the dots on one frequency on the radio dial, and the dashes on another frequency. You might think that a very inefficient use of the AM radio spectrum! Now suppose that suddenly there was a sharp increase in the number of messages to be transmitted, so many that even though the station was operated 24 hours a day, it was impossible to get all the messages through. Then suppose that the the only solution offered was to train the operators of the station to transmit faster, but when the operators transmitted faster, the operators at the receiving end had difficulty keeping up (especially when there was a lot of interference and static on the airwaves) and as a result, received messages were sometimes garbled. That situation almost parallels the difficulties we've had in increasing modem transmission speed.

Now, suppose that instead we open up a second transmitter using a different pair of frequencies on the AM radio band. We could instantly double our message transmission capacity. Now think of how many stations could be put on the AM broadcast band, and you begin to see how transmission capacity can dramatically increase when additional frequencies are used.

Well, the audio spectrum can be similarly divided up. Suppose that instead of using one frequency pair, we used eight. We could transmit an entire byte at once! If we use more than eight frequencies, multiple bytes can be transmitted simultaneously. If we use the full audio spectrum available on the phone line, we can transmit many bytes at once! The Fastlink modem uses a system similar in concept to this, although I don't have specific information about how their system is implemented.

Of course, raising transmission speed in this manner is not quite as simple as it sounds when real telephone lines are involved. Phone lines tend to vary widely in frequency response; it's entirely possible that on any given connection a portion of the audio spectrum may be unusable (either due to interference on a given frequency, or due to a loss of signal at that frequency due to uneven amplification, loading coils, filters, and who knows what else that may be used on a given connection). The Fastlink adjusts for this. It has the ability to monitor 512 different frequency intervals, and to "pick and choose" the frequencies that are capable of being used reliably. Furthermore, it can adapt to changing line quality, decreasing or increasing speed until it finds the highest dependable operating speed. For example, if noise is present on a line, it will lower its speed in 50 bps increments until it can establish reliable communications.

There are other technical considerations relating to the telephone system that must be taken into account on any system of this type. For example, you can't transmit a continuous signal at or near 2600 Hz, because that will immediately break the connection on many long distance telephone circuits (not on the newer digital links, but there's a large amount of non-digital

AT&T has already led the way in this with their "Reach Out America" plan (which considers time only, not distance, when charging for calls made during the night/weekend rate period. During that period, customers pay $9.45 for the first hour of interstate calling per month, or 15¾¢ per minute. That rate drops to $8.25 per hour, or 13¾¢ per minute, after the first hour. However, those who are signed up for this plan must pay a minimum of $9.45 per month, even if a full hour of interstate long distance service during the night/weekend rate period is not used in a given month). But even among the competing long distance services, I think that you'll see pricing become even less distance-sensitive as time goes by, especially as satellite communications are used for more and more of the nation's phone traffic. After all, once you've shot a call several thousand miles into space and back, a difference of a few hundred miles on the ground begins to look mighty puny.

But the use of satellites for phone calls brings their own set of problems. The biggest is that it takes a finite amount of time to shoot a signal up to a satellite and have it come down again. If we assume that a call has to travel 22,000 miles to reach the communications satellite, that's a 44,000 mile round trip. Microwaves travel at about 186,000 miles per second, so simple division tells us that it takes about a quarter of a second for the signal to make the round trip. An international call may make two or three satellite hops before it reaches its destination, which makes the delay very noticeable. If you've ever talked on an international call that went via satellite, you know what I'm talking about. There's just enough of a delay to make conversation difficult - both parties start speaking at the same time, then after a fraction of a second each hears the other begin to speak, whereupon both parties pause to let the other finish (which each hears a fraction of a second later), and so on. And if that isn't bad enough, sometimes your <u>own</u> voice will come back at you after about a ½ second delay. For some reason, people hearing their own voice come back after a short delay tend to stop speaking, perhaps in an effort to let their voice catch up! This effect can make a reasonably intelligent person turn into a gibbering idiot (or at least it can make him SOUND like one).

For modem users, that delay can add expense to modem calls. Consider how the XMODEM file transfer protocol works: It sends a block of data plus checksum information, then waits for the receiving computer to send back an acknowledgment that the block of data has been sent correctly before it transmits the next (or, alternately, that the block of data was received incorrectly, in which case it needs to be retransmitted). Now, if there's a delay in receiving back the acknowledgment due to satellite lag (or delays in transmission through a packet network, which can be worse yet), there will be a delay before the next block of data can be sent. This delay is cumulative from block to block, and could make long distance data calls (especially international calls) very expensive.

One solution is to use a larger block size, say 1K instead of 256 bytes, and there are some versions of MEX (a terminal program that, unfortunately, does not run on the TRS-80 unless you are using CP/M) that support this larger packet size (for more information, try calling the MEX BBS at (414) 563-9932, 300 or 1200 baud). The problem with that is that if you do get an error in transmission, you have to retransmit a 1K block instead of a 256 byte block. Thus, smaller blocks are better for "dirty" phone lines, while larger blocks are better when the transmission is relatively error-free. At present, I'm not aware of a TRS-80 program that supports anything other than the standard 256 byte block size, however.

I'm going to close this month with another of my thoughts on how to bring telecommunications to everyone at a reasonable price. As I've mentioned before, folks in the larger population areas have many inexpensive ways to access computer communications services. For example, Lansing, Michigan has nodes for Telenet, Tymnet, Uninet, Autonet, and Compuserve's packet network (and probably others I don't know about). Here in Sault Ste. Marie we have zilcho - in fact, to my knowledge the nearest pack network node is about 150 miles away!

We have a similar problem with airline service. None of the big airlines wants to come all the way up here. But, we have a feeder airline that flies from here to Detroit (stopping at other small towns along the way), and there you can connect with just about all the major air carriers.

switching equipment that is still in use). Conversely, the phone system is designed to "listen" for normal modem tones, and if heard, to switch out any echo cancellation circuitry that is normally in effect on a call (echo cancellation plays havoc with normal modem communications).

As you might realize, there is no national or international standard for a modem similar to the Fastlink. The Fastlink <u>is</u> compatible with Bell-standard modems (212A and 103 type), but only at their normal speeds. If you want to use the 10,000 bps speed, both modems must be a Fastlink. It's possible that other manufacturers may come out with something similar to the Fastlink, which may or may not be compatible. On the other hand, it may be that the Fastlink will define the standard for this type of modem, simply by virtue of being the first on the market with this type of system (the Hayes "AT" command set is a de facto standard only because Hayes had the first "smart" modem on the market, and everyone else wanted to be compatible).

If you think you'd like a Fastlink, there are a couple of things you should know before you reach for your wallet. One is that it is intended for use with an IBM-PC or compatible computer. That's not to say that it can't be made to work with a TRS-80, but so far I've never heard of an actual case where the two have been mated. What might give you greater reason to pause is the current price - around $2,000. Obviously, this price will come down eventually, especially when other manufacturers start playing "catch-up" and producing their own versions of the super-fast modems (which may turn out to be even "Faster and Better", to steal a line from Lewis Rosenfelder). But if you really need lightning-fast communications capability right now, you can call 1-800-241-4762 for more information on the Fastlink.

This new technology does have certain implications. For one thing, if you're running up huge phone bills on computer-to-computer communications, it might be possible to amortize the cost of a Fastlink over a relatively short time, depending on your application. For another, you can expect that 300 and probably even 1200 baud modems will be dinosaurs in five or ten years (if that long), and that 2400 baud modems will never really become a mass market item. Smart modem manufacturers ("smart" as applied to the manufacturer, not the modem) will either start developing their own versions of a high-speed modem or will get a license from DCA to produce a Fastlink-compatible modem. The Fastlink will be a boon for alternate long distance carriers due to its relative immunity to problems caused by low-grade circuits. Finally, if I had the money to invest in the stock market, I think I'd buy a few shares of Digital Communications Associates stock. They ought to be in for some good times in the near future (if you become a millionaire on this tip, contributions to the Home for Retired Newsletter Editors will be gratefully accepted).

Turning to other matters, have you noticed that the cost of long distance calling is becoming less distance-sensitive? Take the night rates (when else do computer hobbyists make long distance calls?) of one major carrier as an example. The rates climb sharply for the first 56 miles away from your location (to call up to ten miles away costs about 6¢ per minute, to call 23-55 miles away about 10¢, and to call from 56 to 124 miles away costs 13.6¢. But after that, the rates rise rather slowly as you go farther away. You can call a location from 926 to 1910 miles away for 16.1¢, and from there to 3000 miles away for 18¢ per minute. The maximum you can pay to anywhere in the nation (up to 5750 miles away) is 19.6¢ per minute. These prices do not include volume discounts. Note that a 10-mile distant call costs about 6¢ per minute while a 56-mile distant call costs 13.6¢ per minute, which is an increase of over 7½¢ per minute in just 46 miles. But after that, there is only a 6¢ per minute increase to the most distant point you can call in the U.S. (for most of us that's probably Hawaii, which is now tariffed under the domestic rate structure). What this means is that if you have to call long distance to access your communications utility, you may not really be saving any money by accessing the nearest node, particularly if you have any problems with that node. Most people still think that more distant calls cost them much more money, but as we see from this comparison, that becomes much less true once you get farther than easy driving distance from your home (if you think I'm crazy to refer to 56 miles as "easy driving distance", you've obviously never lived in the wide-open spaces of the Great White North).

That makes me wonder if maybe the packet networks couldn't do something similar. Suppose we had a "regional" or "feeder" packet network that covered, for example, towns of any significant size in Michigan's Upper Peninsula (alternately, it could be a single "front end" computer serving a single town). This could then connect into all of the major packet networks at the nearest major city (probably Green Bay, Wisconsin in this case). Perhaps there would be only one private data line out of the Sault, which could carry maybe 6, 12, or 24 data "conversations" at once. You would log onto this computer, then tell it which packet network you wanted to be connected to. It could then make a connection to Green Bay and either dial up the desired network there (or even a bulletin board system!), or directly connect to the packet network's lines. The secret would be that anyone accessing the regional packet could then in turn access any of the major packet networks. The only hassle would be in the billing. Would each user have to have his own account, or would the usage be billed to the called packet network (which would in turn bill it to the host computer, which would in turn bill it to the customer)?

By the way, lest you think this sounds farfetched, consider that Bell Datapac performs almost exactly this type of service for Canadians calling U.S. packet networks. Once you get onto Datapac, you can connect with Tymnet, Telenet, Compuserve or other U.S. networks. Datapac bills the U.S. network, which in turn bills the called host computer. So it can be done, and I don't understand why nobody's doing it. It might not be economical for one of the major packet networks to put nodes in every small town, just as it might be uneconomical for a major airline to run planes to every small town. Just as "feeder" lines have their place in the transportation industry, I believe they also have their place in the communications industry. It's also worth noting that many of the small, regional long distance phone companies are currently on better financial ground than the large carriers that try to cover the entire country, and that's in spite of the fact most of the analysts predicted that these small resellers would go belly up as soon as equal access took effect (starting last year). Which only goes to prove that you can't always believe what "they" say.

Anyway, if anyone has the knowledge and financial backing to put an experimental "feeder" packet switch into the Sault, I'm available for consultation. I don't charge an arm and a leg either.

That's all for this month. I'd be interested to know if you folks enjoy these columns on telecommunications, or do you just think I'm wasting space that could be put to better use? Drop me a line and let me know.

P BUFF KIT REVIEW
by Eeon MacAulay

[Reprinted from the Adelaide Micro-User News, 36 Sturt Street, Adelaide, South AUSTRALIA 5000]

In the April issue of the news John Ross advised members of Don McKenzie's latest project [also mentioned in NORTHERN BYTES Volume 6, Number 4] - an 8 to 64 K printer buffer unit kit. At that time I was half way through constructing such a unit and I thought members would be interested in my experiences. I saw this project as being a great relief of frustration in not having to "standby" while my printer laboriously churned out 20 - 25K of program...

The short form kit as supplied by Don consists of a printed circuit board (one side only) together with an EPROM containing the program for the processor. The basic kit (available from John Ross [or directly from Don McKenzie by mail] at a cost of $35.00 [Australian (around $22 U.S.), plus $5.00 for postage to North America], with extras bringing a total cost up to $150.00 [Australian - much less in the U.S.], depending on where you obtain the Z80A, 4164 RAM and the 8255 PIO. Commercially available printer buffers with 16 K can cost $450 and with hefty extra $ for each additional 8 K. Therefore this project is very attractive cost wise. Don has done the good job of providing a unit to suit most needs and when complete it will work with any Centronics configured computer and printer. If you are short of cash you can start off with as little as 8 K buffer which means you only need to buy one 4164 chip, and with prices varying from $3 - $10.00 the remaining seven represents a very big saving. The kit is relatively easy to construct provided you have a good soldering iron. Perhaps the two most time consuming tasks are installing the 30 wire straps and the Input/Output cable connections. I am a novice at electronics, if I can do it then there is no reason why others cannot put this kit together quite easily also.

A few little wrinkles which do not appear in the instructions, which I might say are very comprehensive - 12 pages in all. Firstly, on the subject of power supply I found it better to use the Dick Smith Electronics catalogue M-2155 transformer [page 117 of 1985-86 U.S.A. catalog] together with a Dick Smith case catalogue H-2744 [page 42 of U.S.A. catalog]. This enables all the components including power supply to be included in a compact form without having to find a place for a plug pack on your power board which might already be crowded. To accommodate this some simple wiring is required to provide a power switch at the rear of the case.

If you are using a Mitsubishi 8255 then it may be necessary to connect a 150 PF ceramic capacitor between pins 6 and 7 to ensure that it operates properly. This can be quite easily carried out on the solder side of the board without much trouble, providing you keep the capacitor leads very short. If you are using one of the faster printers (120 CPS+) you may have to wire in an initialisation switch (momentary) by attaching wire from pin 31 on the output side of the printer cable (presently unused in the buffer circuit) through a 10K resistor to a momentary switch which is then connected to ground. On the subject of printer cables I found it better to fix the female centronics plug to the case (I used a solder type here), rather than letting it "float". The output cable must float and should be long enough to suit positioning of the buffer in relation to the printer.

On page 5 of the instructions Don states, "on power up, LED should be off". If you find that on power up that the LED remains on (and everything else checks out), then immediately suspect a track short in the tracks associated with Pin 2 of chip E13, that is whilst only E12 and E13 have been installed. Finally, if all tests are complete, and you come across an intermittent fault in that the test message only appears once on initial power up and not on subsequent power ups then the problem is resistor R9. Don advises that this resistor was included on the advice of Z80 buffs and in effect can be deleted if the intermittent test problem arises. When I cut mine out the unit worked perfectly with every test sequence. Coming down to operation it is better that your printer power and printer buffer power supply come from the same source, i.e. through one switch. My solution was to install a double adapter to my switch power board so that when I switch the printer off I also switch the printer buffer off. If you switch off all units except the printer buffer you will note that the data LED lights up brightly (this effectively activates the LED circuit). Although Don doesn't mention this as a feature I found it quite useful in that it was unnecessary to then install a power LED to signify power on.

How does it work? In my opinion it is a great addition especially where long print runs are required, and especially if you are using a slow printer, e.g. 80 CPS or less. My test was a simple 22 line BASIC program (the menu program from the Creator Series) and while this was running on my Epson (80CPS) I noted that within a second I had command back over my CPU was able to do two DOS enquiries, and a number of other keyboard functions before the printing completed. With particularly long programs (and with a 64K buffer) the benefits really add up. Just a small point, if you do not power down the printer buffer everything that is in the buffer is still there, even if your CPU has been powered down, and if your printer is still connected you will find that you can still dump the buffer memory to the printer without any influence from the CPU. All in all I can recommend this project as a useful addition to a dining room table full of computer gear.

[Addresses you may want in connection with the above article: Don McKenzie, 29 Ellesmere Crescent, Tullamarine, Victoria 3043, AUSTRALIA sells the PBUFF kit for $35.00 Australian, and can also supply the required 3.58 MHz crystal for $2.90 Australian, plus $5 postage to North America (consult your bank for the current exchange rate - delivery is usually within 2-3 weeks from the time you mail your order to Don, provided you send your letter Airmail, of course). Dick Smith Electronics' U.S.A. mail order operation can be reached through P.O. Box 8021, Redwood City, California 94063 or by toll-free telephone at (800) 332-5373 (outside the U.S.A., use the regular phone number (415) 368-1066).]

TRANSFERRING A BASIC PROGRAM FROM MOD I TO MS-DOS

by Don Gruenther

[This article is reprinted from the NCTCUG Newsletter (Fairfax, Virginia). Your editor finds himself in agreement with the closing paragraph of this article!]

Recently I had occasion to convert some mail list programs, which I had written in BASIC for the Model I, to a Model 2000. There were over 3200 records in the data file and new members were being added daily. The Model I had a hard drive, but the major limitation was that whatever sort field was desired was maintained in an array in memory. There isn't that much memory in the Model I. In fact each array element had been pared to 8 bytes, one for a bulk mail flag, five for ZIP code, and two for an integer record number. Alternatively the first six bytes could be used for name, date of birth, membership number, or whatever. The entire sort field for all 3200 records could not fit in memory at one time, so all processing had to be done in two batches. For example, one mailing might include all ZIP codes less than 40000, with the others being consigned to another mailing. In another case, the roster had to be printed in two separate passes - one for those names less than "L" and a second for the remaining.

The computer owner bought a Model 2000, also with a hard drive. This should be no different from any other MS-DOS machine in so far as my project was concerned. My job was to transfer the data files and the programs that manipulated the data, all in BASIC, from the Model I to the Model 2000. Since I don't own an MS-DOS machine and probably never will, I had very little interest in learning any more about the Model 2000 than what was essential to do the program conversions. Since others may be tempted to undertake this same type of task, I will relate my trials and tribulations. The BASIC on the MS-DOS machine was version 01.03.00, 1982.

BASIC on the Model I does not require spaces between commands or variables; BASIC on the Model 2000 does. The main reason for this is that the Model I limits variable names to two letters. More may be used but only the first two actually define the variable. The Model 2000 permits a large number. Since key words embedded within a variable name is permitted, or else you could never keep track of legal and illegal variable names, the BASIC interpreter must insist that all variable names be delineated with a space or punctuation character. The first task then is to edit the Model I BASIC program to ensure that the appropriate spaces are present. I did this manually, though there are programs to do that automatically.

All BASIC commands are normally stored on a disk as single byte tokens. For example, the command GOTO is stored as a single byte. These bytes are not the same between different model machines that use BASIC, and particularly between an 8-bit and a 16-bit machine. Many of the commands in the BASIC on the Model 2000 actually require two tokens. Therefore the program on the Model I had to be saved to disk using the "A" option to ensure that it is saved in ASCII. Then it could be transferred to the Model 2000 either via a modem, null modem, or a program that reads disks of different formats. I had Roger Fujii and Rich Deglin perform that task for me.

At this point, the fun began; or rather I should say the frustration. While most BASIC commands are standard, there are several that are not. The Model I has a PRINT @ statement for controlling the cursor. The screen is divided into 1024 individual locations, starting at the upper left and ending at the lower right. The Model 2000 handles this function with a LOCATE command, with the parameters being the row and the column desired. I had to find all the PRINT @ commands on the Model I and translate them for the other computer.

I had a lot more trouble controlling the cursor on the Model 2000 than I had had on the Model I, but it was all due to my own ignorance. Specifically, a <LINEINPUT IN$;> (emphasis on the semi-colon) in the Model I would suppress a line feed on the screen after the return key was hit. The Model 2000 wasn't nearly as polite; I had to put a LOCATE command before a <LINE INPUT IN$> and after it, if I wanted to preserve the video just below that prompt. (Note also that #&%led required space between LINE and INPUT; but not in DEFINT). I make no judgement as to whether one syntax is better or worse than the other, but when you treat one as second nature, the other can be very frustrating.

I'm sure that you are all well aware that a sort in BASIC, even a very clever one, takes forever when you start sorting 1000 or more. My solution to that problem on the Model I was very simple; I wrote a short machine language routine to to the sorting. I understand the Z80 somewhat; the 68000 is a big mystery. Those who do know something about it have suggested I leave it alone. Fortunately MS-DOS has its own sort routine, but it is not in BASIC. It took many hours to go from that knowledge to making any use of it. A random file in BASIC is just one continuous stream of bytes. The BASIC interpreter is able to delineate individual records by its knowledge of the number of bytes in each record. Further these bytes need not be pure ASCII. The MS-DOS sort turns out to be line oriented; in other words there has to be a carriage return and line feed after each individual record. It also will make good sense out of data records only if they are pure ASCII. Where I had kept all my sort fields in an array in memory, now I had to write them to a file on disk before the sort. Further I had to make sure the record number associated with a particular sort field was in ASCII and that the record ended with a CR LF. None of these things are significant problems when you know about them, but the only way I learned was by getting burned. I also learned to be very wary about the length of the file; MS-DOS and my BASIC program never did agree on that limit. I solved the problem by appending several dummy records while in BASIC, ones that I could recognize as being beyond my last valid data record. By filling those records with "zzzzzzz" or an appropriate graphic symbol, I always got them to sort to the bottom of the pile. You also have to be sure there is a 1AH at the very end of the file as an MS-DOS EOF mark. Whenever I jumped back into BASIC to make use of the sorted records, I always made an EOF test in BASIC. It didn't matter that there might be more than one phony record. In fact, due to a bug in the SORT MS-DOS command, without a few extras I could never sort beginning at a column other than the first [MS-DOS allows you to specify the beginning column]. One other observation; the MS-DOS sort is case independent. I wanted it that way, but if you don't, that's another complication. Perhaps all this information is in the manual, but I doubt it. As I mentioned before, my main concern was to get the programs converted, not to learn to use the Model 2000.

In the Model I, after you have opened a random file with a certain record length, the LOF command tells you how many records you have. Not so with the Model 2000. LOF tells you how many bytes are in a file, including those unused ones at the end of the last block of disk space allocated to the file. This required that the program divide the LOF by the record length to get an approximate answer. Then each record from some convenient one near the last had to be checked in turn to see if it was the dummy record at the end of the data file.

I have always been enamored with the flexibility a file buffer offers for string manipulation. You can field a buffer with 30 bytes allocated to an entire name, the first 14 allocated to a first name, the next 16 allocated to a last name, the 3d & 4th allocated to anything else you wish, and so on. It's very handy to stuff things into one slot with one variable name and take out selected pieces using another variable name, even when no actual data file is involved. I always open one more buffer than I have data files, just as a place for me to play in. Having opened it, I usually close it right away, but the fielding remains valid until a CLEAR or NEW statement. This is not true of the Model 2000. If a file tied to a buffer is closed, the buffer fielding is lost. Any dummy buffer must remain open! You also have to worry about closing any file during program execution because if you reopen it, fielding of the buffer is not automatic.

Editing a BASIC program on the Model 2000 was a nightmare. I don't fault the computer; it's just that I edit on the Model I as second nature. I can spot a single "x" hiding on the line, enter <Sx>, perform <Cy>, and everything is done. The Model 2000 enters every key you press, except for the arrow keys, an <END> key and some <CTL> sequences. I had to be very careful to avoid typing over line numbers and other things by giving Model I commands. The delete key and the insert key are very useful, but if you digress the tiniest bit, say by backing up a character, you just exited either of those modes. It takes a while to get used to that editing. One nice feature is that you can change a line number and change the order of code without retyping it. A bad feature is the scrolling. I haven't figured out exactly why it is so jumpy, but it isn't nearly as pretty as on the Model I. One cure is to give a CLS before every LIST command; then at least the first 25 lines are displayed smoothly. If I owned a Model

2000, I would figure out how to make the function key that issues the command, LIST, perform a CLS:LIST instead.

Eventually in every program, generation of a report is required. In my case for a roster of members, I needed to print in 132 column mode. Would you believe the Model 2000 BASIC doesn't want to do that -- until you place in the program somewhere, WIDTH LPRINT 132?

The final set of programs is less than ideal. Some language other than BASIC would be a better alternative. Even with all its memory, the Model 2000 only permits use of 64K for the user's BASIC program. This does not include the BASIC interpreter. In this case, since the 500K main data file and the smaller sort key file are on disk and not in memory, we can live with the memory limitation. The nuisance of having to exit BASIC in order to sort is a big disadvantage. In this application, one BASIC program examines each record and selects those meeting input criteria, which are ANDed and ORed in any conceivable fashion. Those fields selected are written, with their appropriate record numbers in the main data file, to a key file. After selection, the operator has to exit BASIC in order to sort the key file in MS-DOS. He than has to enter any one of several different BASIC programs depending on whether he wants to print a roster, telephone list, membership cards, labels, nasty notices, etc. A language other than BASIC would probably have been more appropriate.

For those of you thinking of porting over your Model I BASIC programs to an MS-DOS machine, I hope I have given you some idea of what you're up against. It's not impossible, but it is tedious. It helps to learn something about the particular syntax of the new BASIC commands, but the manual often isn't very illuminating in this regard. Certainly I saw no explanation in the manual of the limitations of the SORT command in MS-DOS.

At this point, I'm not at all certain that progress is really a virtue. Why not just keep your older machine?

MORE MODEL 4 PATCHES

The following two patches will allow the M-ZAL assembler to run under TRSDOS 6.2:

```
PATCH MASM/CMD (X'6401'=18 78)
PATCH MASM/CMD (X'6542'=00 26 06 06 3E 0F EF 18 06 0E 3F 06 02 18 82)
```

The stack used by TRSDOS 6.2 can become corrupted under certain conditions. This has come to light using the PRO-NTO program along with some applications such as ALLWRITE and MULTIPLAN. The following patch will correct this problem:

```
PATCH BOOT/SYS.LSIDOS (D06,2C=58:F06,2C=60)
```

When using PRO-NTO or other systems which allow a file to be accessed by multiple programs at the same time, the following statement should be entered and SYSGENed. This sets the "network" flag ON in TRSDOS 6.2. The "network" flag establishes a lock on files which have already been opened by another program. This prevents them from being written to while they are already open.

```
MEMORY (A="N",B=1)
```

[This original source of this information is unclear, although it may have come directly from Logical Systems, Inc. We got it from the Milwaukee Area TRS-80 User's Group newsletter].

ZAP TO MODEL I VERSION OF ALAN JOHNSTONE'S NEWDOS/80 MODS

If you have purchased TAS Public Domain Library disk #ND-1 (Alan Johnstone's NEWDOS/80 mods), you may have discovered that Alan's Model I printer driver sends most characters to the printer unchanged (in other words, it doesn't convert a form feed character to a series of linefeeds, etc. It just passes the character straight through). However, for some unfathomable reason, if the character is an ASCII zero (Ø), it is not passed through unchanged, but is handed over to the TRS-80 ROM (which also will not LPRINT a CHR$(Ø)!). There are times, however, that you need to be able to send a zero byte to some printers (particularly if you are doing printer graphics), so here is a zap to SYSØ/SYS (Model I version, as modified with Alan Johnstone's mods) that will allow zero bytes to be passed straight through to the printer. This zap is recommended:

SYSØ/SYS,23,3Ø change B7 CA D1 Ø5 18 25 to B7 ØØ ØØ ØØ 18 25

This zap was extracted from an article in Christchurch-80, the official magazine of the Christchurch 80 Users Group in New Zealand.

CONFIGURATION IN LDOS
by Rod Stevenson

[Reprinted from the Adelaide Micro-User News, 36 Sturt Street, Adelaide, South AUSTRALIA 5000]

There appears to be some confusion possible for a newcomer to a disk system by the explanation in the manual of configuration. I don't wish to duplicate the instructions in the manual for installing, removing, or viewing a configuration. Reading will explain these mere mechanical points I believe.

It seems to me the trouble lies with not recognising that the configuration is really a section of high memory reserved and occupied by certain drivers, filters, or whatever the user has built into his customised configuration. The CONFIG file on the disk is nothing more than this section of high memory (with a few other things we don't need to delve into here) saved to disk in such a way that it will load itself into high memory on boot-up. So there is nothing at all special about the file saved on disk, it is just one of the ways of loading whatever is required into high memory. Another way is to type each one in from the keyboard as it is required. A common way is to have a JCL file to save having to type it in from the keyboard each time. Below is my own JCL which I call SETUP/JCL and change whenever I want to custom-configure a disk:

```
SYSTEM(BLINK,LARGE)
FILTER *PR USING PR/FLT(C=72,I=6,L=62)
FILTER *PR USING SLASHO/FLT
SET *KI TO KI/DVR(T,J,D=10,R=1)
VERIFY
PDUBL/CMD
FILTER *KI TO KSM/FLT USING KEYMULT/KSM
FILTER *KI USING MINIDOS/FLT
```

I change it using Scripsit, save it as an ASCII file, then DO = SETUP while the system has no configuration at all.

There is another source of possible problems if you do a fresh configuration on top of an existing one. Most of the time you'll get away with it, but there are some things that can't be re-done or done twice; the manual does advise to configure only if there is not one present.

When the system is configured as you require, insert the disk you want that configuration saved to and save it by SYSTEM (SYSGEN). It will write a new CONFIG file over any that already exists on that disk.

It should be apparent that it matters not if you configure having booted up a Model I on a single density disk, then write the configuration to a double density one. Probably this is the easiest way to do it. Or you can boot up with the configuration required, then write that configuration to another disk; or simply copy the CONFIG file.

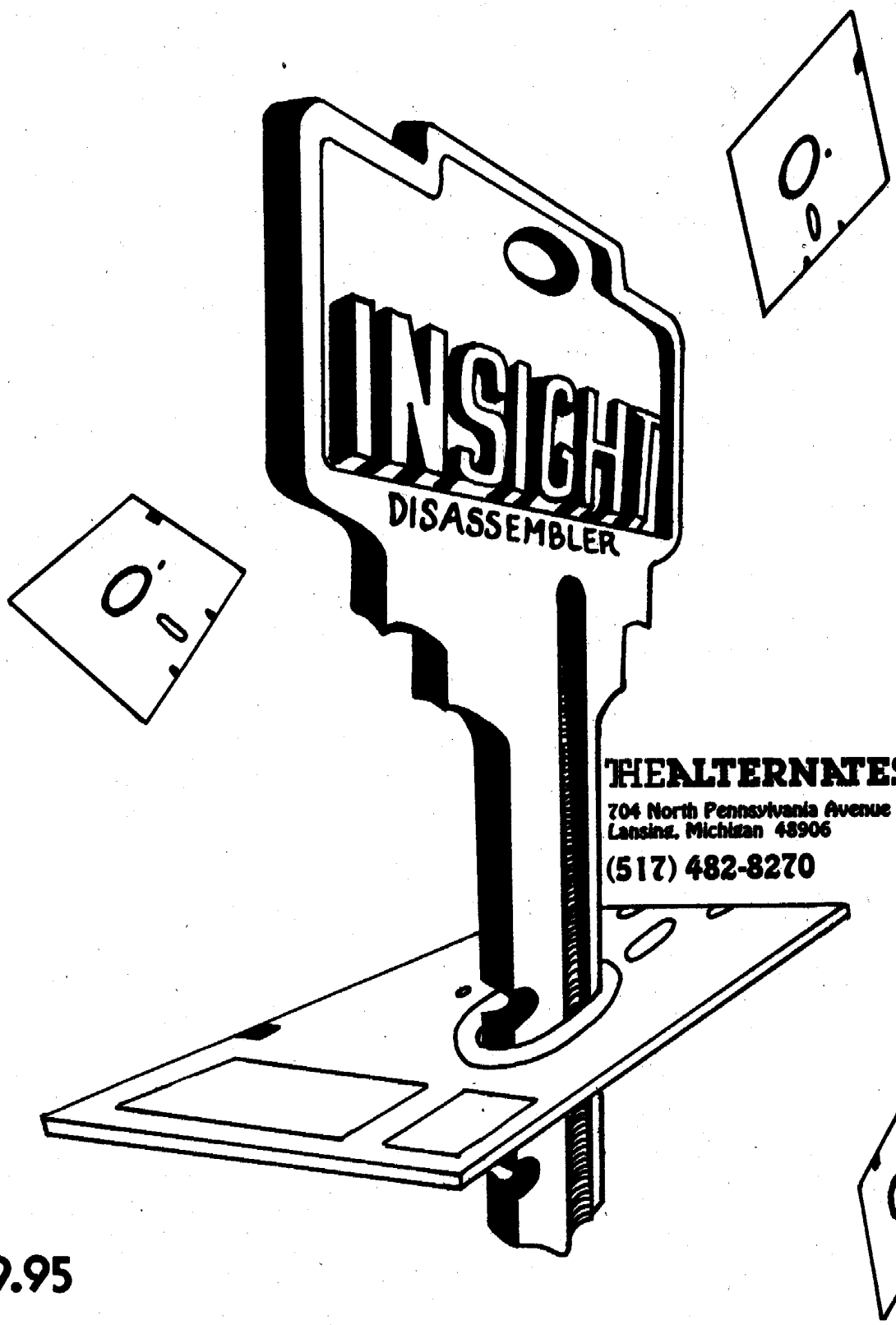
I hope that by pointing out that the configuration is actually in memory I've started some thoughts along the right track.

ADAPT MODEL III INFOCOM GAMES TO RUN ON THE MODEL I
by Bob Seaborn / MCI ID: 268-7906

Here's a tidbit of information to help all those with Model I's that would like to be able to play the latest Infocom games that are available only for the Model III, or those who dislike custom disks. This works under Newdos/80 version 2.0 and requires a double density adapter, only because of the requirement for more disk storage area. It could possibly work if the files are spread over two disks, but I'm not sure. The same goes for other DOS's too.

Simply copy over from the Model III disk the two files on it (FILESPEC/CMD and FILESPEC/DAT). Then, using SUPERZAP search through the FILESPEC/CMD file for the hex byte sequence "21 25 42" and change it to the new sequence "21 18 42".

What happens is, when you execute the program by typing from DOS "FILESPEC", the program loads and executes; then after it has initialized it looks back to the DOS command buffer (at 4225 Hex in the Model III and 4218 Hex in the Model I) for the name of the program, adds the "/DAT" to it, and then opens the proper data file. Infocom explains that the only reason a custom disk was used for the Model I was because there isn't enough disk space available on a single density 35 track system. Some copy protection was used for both, but it is easily defeated, on the Model I by using TRAKCESS or SUPER UTILITY and on the Model III by using a DOS other than TRSDOS.



THE ALTERNATE SOURCE
704 North Pennsylvania Avenue
Lansing, Michigan 48906
(517) 482-8270

$19.95

TRSDOS 6

NORTHERN BYTES

Subscription Information

Northern Bytes is edited by Jack Decker and published on an irregular basis by The Alternate Source Information Outlet. Back issues are available starting with Volume 5, Number 1. Issues prior to that are not available. Some of the most valuable articles from earlier issues may be reprinted in future issues of Northern Bytes. Currently there are eight back issues available for Volume 5, as well as all issues from Volume 6. All back issues are $2 each.

It is very easy to be placed on the Northern Bytes REGULAR list. Simply place your address, Visa or MasterCard number and expiration date on file with us. We will start with the issue you request. We do not bill you for ANY ISSUE until that issue has been mailed. This way, we can continue to offer you top quality information with absolutely no risk to you. There's no question of what to do about unfulfilled issues if we decide to quit publishing. Unless otherwise requested, we presume your subscription will extend through the month of your expiration date.

Don't have a charge card, huh? We understand the myriad of reasons for not having them and we feel that a "To-Be-Invoiced" policy could help increase the demand for Northern Bytes. If you'll do it for us, we'll do it for you. Would you like to be placed on a regular list TO BE BILLED for each issue? You could then send a check for the issues as they are mailed. If you didn't send a check, we would presume that your interest has died and discontinue your subscription. The only requirement for getting onto the list is to pay for the first issue up front; the next will be mailed automatically, if you request. You are insured that you will receive top of the line TRS-80 information as it is released. Ask to be placed on the NO RISK "To-Be-Invoiced Northern Bytes List".

Call or write, but SIGN UP TODAY!

The Alternate Source Information Outlet
704 North Pennsylvania Avenue
Lansing, Michigan 48906-5319

Telephone (517) 482-8270, or use our toll-free lines for ORDERS ONLY:
(800) 632-7818 ext. 700 in Michigan, (800) 253-3200 ext. 700 elsewhere in U.S.A.
EMAIL: CompuServe: 72167,161 / Delphi: TASIO / MCI Mail: 109-7407 / Telex: 6501097407 MCI

NORTHERN BYTES

c/o Jack Decker
1804 West 18th Street # 155
Sault Ste. Marie, Michigan 49783-1268
U.S.A.

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

POSTMASTER: Address Correction Requested.
Send address changes, USPS Form 3579, and undeliverable copies to: The Alternate Source Information Outlet, 704 North Pennsylvania Avenue, Lansing, Michigan 48906-5319

To:

ELECTRONIC MAIL ADDRESSES:

Compuserve EasyPlex:	72167,161
Delphi Mail:	TASIO
MCI Mail:	109-7407
Telex:	6501097407
Answerback:	6501097407 MCI