NORTHERN BYTES

Volume 6 Number 5

Welcome to another issue of NORTHERN BYTES. Summer has arrived in Sault Ste. Marie, and so has the state bird of Michigan's upper peninsula. I'm talking about the mosquito, of course. State bird?! Well, you should see how big they get up here. My wife whacked one against the wall the other day, and insists that she heard it hit the floor. One night (after staying up at the keyboard all night), I thought I saw a swarm of them carrying away a small squirrel.

Of course, the mosquitoes we have are nothing compared to the blowflies of Australia - or so Clifford S. Richards of Sydney would have me believe. He wrote me a letter recently and included the following bit of information:

"We Aussies are known all over the world for our bush 'Blowies', sometimes they grow so big they are mistaken for crows. The result is 'The Great Aussie Salute' which I will now describe for you underprivileged Americans.

"This manuever commences with a movement similar to the first part of the American Military salute. But, before the fingertips reach the forehead, the hand passes quickly back and forth several times in front of the face, before returning to the side.

"Only the uninitiated, however, mistake blowflies for crows. It is easy to pick them. Because of the heat, crows fly backwards, so as to keep their backsides cool."

Well, Clifford, perhaps someday we should stage a match between a bush blowfly and an upper peninsula mosquito. Our mosquitoes may be a bit smaller, but they're mean!

What has all this got to do with computers? Not a thing! But, it's summer, and since summer is so short up here I refuse to ruin it by thinking about computers all the time!

Oh, well, on to more serious stuff. The "international bulletin board via a packet network" idea we mentioned in Volume 6, Number 3 has been put on indefinite hold. You may recall that I asked folks to drop us a line if they'd be interested in such a system. I got one letter of support for this idea. Not only that, but it seemed that no matter which way we went, the charges would have been too high to be competitive. Sorry about that, we were hoping...

I would like to ask folks once again to please bear with me, as sometimes I get mail that for one reason or another I don't get around to answering (I'm only one person, after all, and can only get so much accomplished in a day). Then it goes into one of my famous "piles" on my desk (some archeologist will have a field day going through these someday, if the pressure doesn't turn the bottom layers into coal first). If you wrote me or sent me something and haven't received a proper reply, please drop me a postcard and jog my memory. Of course, if you live in the U.S.A. and did not include a self-addressed stamped envelope, that is probably the reason you didn't get a reply. As I've mentioned before, I prefer phone calls to letters, but please understand that I do not just sit by the phone waiting for calls. If you get my wife and ask her to have me call you back, please be sure to specify that I may call you collect, otherwise your call will not be returned (in fact, she won't even bother to pass on the message because she knows that I won't return the call on my nickle)! If you're willing to gamble on me being home, the number is (906) 632-3248 (there, I just saved you the 50 cents you would have paid AT&T to get my number, so you can afford to call me back if I'm not home).

Would you believe I don't have any other comments to make this issue? Well, it's true! Hope you enjoy the summer (or whatever's left of it when you receive this issue) - I know that we folks up in these parts thoroughly enjoy the entire week!

LETTERS DEPARTMENT

Reminder: Persons sending letters intended for publication should send them on magnetic media or via 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!

Date: Tue Apr 16, 1985 7:26 pm EST
From: Orange Controls / MCI ID: 267-2560

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

Well Sir,

On your advice I joined the ranks of MCI users. So far I like the service. The only thing wrong is that my area does not have a local dial up, I have to use the WATS service. Have you made any headway in rounding up other SYSOPs for a network?

Say, if you know of anyone that has a decent inventory control program for the TRS-80 Model I or LNW systems please put them in contact with me. I need a good manufacturing inventory control program. Tandy's is a total disaster.

Well goodbye for now...

Lou Gomez
Voice (305) 631-0579 - BBS (305) 631-2061

[Well, Lou, as you have found out, MCI Mail is not such a bargain as it used to be when I first recommended it. Use of the WATS line now costs 15 cents per minute and MCI is even dropping the free 20 minutes of WATS line access per month for advanced service users. However, MCI Mail is now available through TYMNET at a five cents per minute surcharge, but that doesn't help us folks who live in the boondocks. I have noticed that the number of messages we receive via MCI Mail has taken a dramatic drop since the new charges were imposed at the first of the year. We may even drop the service at the next annual renewal. I personally find it a bit offensive that many of us were lured into signing up for MCI Mail at giveaway prices, and now that they figure they have us hooked, they've started charging for things that used to be free, like (800) number access.

Unfortunately, we haven't had much response to the idea of a network of SYSOPs (but then I wasn't trying to organize such a project anyway - I just suggested it, hoping someone else would pick up on the idea).

Just for your information, The Alternate Source now has a Compuserve account, and the ID number is 70147,3557. However, this account is checked very infrequently at present, so time-critical orders or other material should not be sent this way for the time being. It would be possible to send an article to NORTHERN BYTES using this account, but I don't recommend it because of the lengthy time delay that may pass before I receive it.

Perhaps one of our readers will be able to help you obtain the manufacturing inventory control program that you seek. Let us know if you find a good one!]

Date: Sun May 26, 1985 8:47 am EDT **PRIORITY **RECEIPT
From: Anthony J. Domigan / MCI ID: 254-5121

TO: *Jack Decker / MCI ID: 102-7413
Subject: Newdos Zap

Dear Jack,

Just lately I have been using more and more programs which reserve HIMEM. In order to use the CLEAR command to zero memory it is necessary to specify the START and END parameters. Failure to do so causes HIMEM to be reset to FFFFH and memory 5200-FFFFH being cleared. Forgetting to specify an argument has brought me grief with my 4P and often results in me having to reload the ROM image from disk. It seems to me that a more sensible upper limit default (for users in my situation) should be the HIMEM address.

In the following ZAP I have defaulted the nil-argument form of the command i.e. CLEAR<CR> to clear 5200-HIMEM. The command CLEAR *<CR> will invoke the original default and thereby clear all user routes/enqueues etc. The original START, END, and MEM arguments can still be used.

```
ZAP     SYS14/SYS,3,82
CHANGE FE 0D 28
TO      C3 87 51
ZAP     SYS14/SYS,04,9B
CHANGE 00 00 00 00 00 00 00 00 00 00 00 00 00 00
       00 00 00 00 00 00 00 00 00 00
TO     FE 2A CA 8B 50 FE 0D C2 76 50 2A 11 44 22
       AE 50 E5 D1 21 FF FF C3 A6 50
END
```

Tony Domigan, P.O. Box 150, Thomastown, Victoria, 3074, Australia
MCI-ID: 254-5121

[Thanks for the information, Tony!]

Attention!

Anyone who has a L.N.W. doubler with a Radio Shack interface—or anyone who is hard-core hardware hacker: HELP!!

I purchased a Holmes doubler, a L.N.W. interface, and a MPI-92S drive. When I try to load MULTIDOS (80 track, double density, version 1.7d) with the doubler installed, I get the banner and the system hangs. Without the doubler, I get the banner and then a "boot error." I have been told that the Holmes doubler will not work with the L.N.W. interface. Does anyone out there know how to modify either of these to make them compatible? Or, barring that, will anyone with a LNDoubler 5/8 and a Radio Shack interface swap with me? My doubler is in mint condition...it WILL work with a Radio Shack interface.

If you can help, please write or call: Steve Winokur, 435 Norristown Road, Horsham, Pennsylvania 19044. My phone number is (215) 675-7708 (after 3:00 P.M. E.D.T.). Any assistance will be greatly appreciated and I will provide anyone who wants to help with a photocopy of the L.N.W. schematics if you send a self addressed stamped envelope with sufficient postage. Unfortunately, I do not have a schematic of the Radio Shack interface, which I also need. As I said before, HELP!!!

Steve Winokur

[Ah, yes, the thrills of trying to force incompatible equipment into a shotgun wedding. I strongly suspect this is some minor hardware problem that can be easily resolved by someone with hardware expertise, so if anyone has the answer, how about passing it along to Steve and to the rest of our readers?

And while we're at it, this brings up an interesting question. When using NEWDOS/80, the TI flag of PDRIVE must be set to C if a Percom doubler type interface is in use, or to E if an LNW type interface is in use. I personally use a Holmes Engineering Expansion Mainframe with one of their doublers installed, and I found that I could make it work using either the C or E flag, although I normally use the C flag and haven't done any extensive testing using E. Anyway, there must be some obvious differences (from the DOS viewpoint) between the various makes of double-density adapters, so what are they?? Perhaps some hardware hacker and/or NEWDOS/80 expert could enlighten us on this subject!]

Dear Jack,

As a followup to the article "UPGRADING THE TRS-80 MODEL I TO 64K OF DRAMS" by Errol Rosser in the Volume 5, Number 8 issue of NORTHERN BYTES, I thought you might be interested to know of the BIGMEM upgrade kit.

This kit has been sold now for three years and hundreds of TRS-80 Model I users have converted their keyboards to BIGMEM capability. Note that this kit retains access to the 32K RAM in the expansion interface and optionally fills the 2K space above ROM as well as enables 0.75K RAM in the unused portion of keyboard space. It also provides full CP/M 2.2 hardware compatibility which we support with software as well.

There are a few kits left which are available for $129 including the $20 software package Disk01.

Dick Kinsey, MICROHATCH, P.O. Box 501, DeWitt, New York 13214

[Readers who would prefer to contact Mr. Kinsey by telephone may call (315) 446-8031 after 6 P.M. Eastern time. Yet another 64K memory expansion that installs inside the Model I keyboard is sold for $59.95 by International Carbide & Engineering, Inc. (see their ad elsewhere in this issue of NORTHERN BYTES, or phone them at (800) 424-3311 or (804) 568-3311 for more information).]

Mr. Jack Decker,

..... you may be able to give some advice or shed some light on a problem I am having. I recently upgraded my CPU to a Z80A (4 MHz) and changed the clock circuitry to provide the CPU with a 3.55 MHz clock. The problem is, when I invoke the speed-up, the screen fills up with the commercial at (@) and 6's, all in perfect order and sequence. The funny part is when I first hit <RESET>, the MEM SIZE prompt comes up, I can enter anything I desire into the keyboard and it is displayed to the video, but the moment I hit <ENTER> that is when I get the pattern described above and the keyboard locks up.

I believe the problem may be in the memory itself, being I am using 4116-4 (250 ns) chips and they may not be fast enough, but I can't be sure. I am hoping you (or someone else) can verify my claim and suggest the proper RAM chips to use, or make a wild guess as to what my problem could possibly be.

I am hoping you can help or suggest someone I can contact to help me with my problem. Thanks a bunch!!

Carlos H. Matos

[The TRS-80 Model I will usually run fine at 150% of normal speed (2.66 MHz). However, when using a 200% speedup, problems sometimes arise.

One problem is with the memory select circuits. The solution to this problem comes from Dennis Kitsz's book, "The Custom TRS-80 & Other Mysteries" (if you'd like a copy of this book, check with The Alternate Source to see if any copies are still available). Dennis says you must "Locate Z69, and cut the trace running from pin 5 to pin 12. Connect pin 12 to pin 13. This speeds up the memory-select process (from MREQ and RD) just a tad, but enough to cope with the 200 percent modification."

Also, if you are using a newer Radio Shack Expansion Interface (one that does not use the "buffered" cable), you may need to make the following modifications <u>in the Expansion Interface</u> (these are also from Dennis' book):

1. Find Z37 and Z38
2. Cut trace leading from Z38, pin 9. Call the end furthest from Z38 'Trace A'.
3. Cut trace leading from Z38, pin 8. Call the end furthest from Z38 'Trace B'.
4. Cut trace leading from Z37, pin 4. Call the end furthest from Z37 'Trace C'.
5. Attach Trace A to Z38, pin 11.
6. Attach Trace B to Z37, pin 4.
7. Attach Trace C to Z38, pin 9.

Final notes - your memory must be 200 ns to handle the above, so if it still doesn't work, try new (faster) memory chips. Of course, if you have made the change from 4116 to 4164 chips in the keyboard (as described in NORTHERN BYTES Volume 5, Number 8, also note the correction in THE EXTERMINATOR column in Volume 6, Number 3), you don't have to worry about any of the Expansion Interface mods.

If all else fails, I have heard of cases where the problems have been solved by replacing the buffer IC's in the keyboard unit with non-LS series (that is, replace the 74LS367's with 74367 TTL IC's). It may also help to replace the 74LS157 multiplexers for video and dynamic RAM if they are weak (not likely). But you should not normally need to resort to such drastic measures for a 200% speedup. I hope this helps.]

Dear Mr. Decker:

..... I have one question that I hope you or one of your readers can answer. Is there any simple way to send a character or control code to the printer from TRSDOS READY in TRSDOS 6.2 on my Model 4P? Since this can be done in LDOS (using MINIDOS), and since TRSDOS is supposed to be LDOS, I wonder if the necessary code might be there, undocumented? Or is there some other way. This would be useful, for example, to put the printer into compressed mode before printing a disk directory to put on the jacket. Now I have to go to BASIC, LPRINT the code, and exit to DOS - very awkward!

Don Singer, 3726 Skyline Drive, Scottsbluff, Nebraska 69361

[Readers, any suggestions? If you write to Don with a solution, please send a copy to us here at NORTHERN BYTES so we can share it with everyone!]

Dear Readers:

Since June 7, 1981, I've been running the 'Chicago Greene Machine' BBS at (312) 622-4442. Since that date I've made many improvements and modifications to the system. The software is still being offered as 'public domain' to anyone who asks.

Many of our callers have asked me, the SYSOP, if I would ever offer a 'chat mode' similar to The Source or Compuserve. I said I would if I could get enough subscribers and another Model I or III. I now have another Model I and would like to get the two machines talking to one another. The only problem is that I can't use the RS-232 because that will be used by each machine to each caller.

My question is this: Can two Model I's or a Model I and a Model III communicate with each other thru the cassette port? The only other two stipulations are that the software has to be in BASIC, and run under NEWDOS/80 version 2.0. Can this be done, with only BASIC, and any other minor hardware modifications ????

George Matyaszek, 1718 North Long Avenue, Chicago, Illinois 60639

[George, I assume you mean that you want a subroutine or two that will operate in a pure BASIC environment, without needing a machine language driver in high memory or some such thing. In other words, you probably realize that you have to use a little bit of machine language to access the cassette read and write routines in ROM, but these routines could be totally contained within a BASIC 'packed string' and therefore would not require any use of high memory.

To partially answer your question, yes, it is possible for a Model I to communicate with another Model I or a Model III through the cassette port. You would need a pair of small audio amplifiers (one small stereo amplifier could be used), and you would connect the output of each computer to one amplifier input. You would then connect the output of each amplifier to the audio input on the opposite computer. The volume on each amp would have to be adjusted for best results (high enough for each computer to read the data coming from the other, but not so high that noise is erroneously seen as data). The amps selected for this application should not have Automatic Gain Control, since this will tend to boost the level of noise to the same level as the data bits.

The maximum data transmission speed would be limited to 500 baud, of course, but what you may not realize is that there is no minimum transmission speed. Each byte transmitted through the cassette port contains its own timing bit, so even if bytes are transmitted individually, with relatively large gaps between each one, there should be no problem so long as there is no noise on the audio line.

TRS-80 ROM Routines Documented lists the various cassette input/output routines on pages 15 through 18. The routines most useful in this application would be the one to read a byte from the tape into the A register (at 0235H), and the one to write the byte stored in the A register to the tape (at 0264H). Having gone this far, I am going to leave it to our readers to suggest some actual routines, since I don't have two TRS-80's sitting side-by-side to experiment in this way. I do see one problem, in that when a computer is reading (or waiting for) a cassette pulse, it can't be doing anything else, which just may make it impossible to use the cassette ports for your particular application. But I've learned never to say "impossible", especially when faced with a software problem.

I just know that some wise guy hardware hacker is smugly reading this and saying, "why doesn't he forget the cassette ports and just whip up a small hardware interface using TTL logic, that would permit sending characters from one machine to another by simply outputting them to a port?" Well, I'm sure that this would work (probably much more reliably) and that George is open to such suggestions, but like me, he doesn't have the hardware expertise to build such a device. If you'd like to build such a device, I'm sure that George would appreciate it. Keep in mind that it has to permit two-way communication, and that it preferably should hold an incoming byte until the receiving computer acknowledges receipt, to prevent lost characters (the cassette port won't do this, which is why I say that it may be unsuitable for this application).

If you want to communicate directly with George, his voice phone number is (312) 622-5969 and his BBS phone number is (312) 622-4442. George has kindly consented to allow his rewrite of the "Greene Machine" BBS software to be placed on a TAS Public Domain Library diskette, it can be found on disk # 008.]

[The following is a portion of a letter that was sent to Charley Butler at The Alternate Source, that includes some patches for Super Scripsit:]

Dear Mr. Butler,

... These are some patches that I was asked to work out by a member of the user group. I include them as they could be of some interest. Please check the Model III patches though, I'm not sure if they are correct.

The standard version of Super Scripsit uses an underline character code on the screen as "shorthand" for every two spaces together. This has the disadvantage that the screen layout then does not match the printer layout. This is especially true of the Model 4 version (with its 80 column screen).

To overcome this, I have located the offending code and changed it. After amendment, as described below, spaces are not compressed and the screen layout will match the printer output for all except proportional spaced printing. The two changes are to the Scripsit/ASCII conversion and the keyboard text input routines. In each case a conditional jump is replaced by an unconditional jump, removing the test for double space.

Zaps are as follows:

Model I/III:

```
SCR35/CTL 30H change byte from 20 to 18
Context: from 2EH onwards FE20 200B 2BBE

SCR64/CTL 38H change byte from 20 to 18
Context: from 36H onwards FE20 2007 36F7
```

Model 4:

```
SCR35/CTL   FRS 20 28H change from 20 to 18
Context: from 26H onwards FE20 200B 2BBE
PATCH SCR35/CTL (ADD=99C9,FIND=20,CHG=18)

SCRIPSIT/CTL FRS 20 30H change from 20 to 18
Context: from 2EH onwards FE20 2007 36F7
PATCH SCRIPSIT/CTL (ADD=43D4,FIND=20,CHG=18)
```

Yours Sincerely,

P. Knaggs, 12 Seymour Road, Chippenham, Wiltshire, ENGLAND,
SN15 3NH

CALL FOR INFORMATION

It seems a bit amazing, but there does not seem to be much effort underway to disassemble and decode Model 4 BASIC. Surely some of you Model 4 hackers have compiled information on the entry points for various BASIC routines and/or reserved RAM location (pointers, tables, data storage locations, etc. used by BASIC). If you have such information, please send it (preferably on disk in a text file) to NORTHERN BYTES. We'll try to pull together a preliminary list of pertinent addresses to which others can add.

ROM/ASM
by Tony Domigan

In using my Model 4P with programs that modify the standard MICROSOFT ROM, I have at rare times come to grief when exiting these programs and attempting to reboot. When I try to reboot the screen fills with @'s and the machine locks up.

The short-term solution to this was to reinstate the old ROM prior to rebooting NEWDOS80.

ROM/ASM will produce a short /CMD file to either save the ROM to an upper memory bank or to retrieve the ROM from upper memory prior to rebooting.

To execute the program the following conventions apply.

ROM<cr> . . .Will present you with menu option to Save or Load ROM.
ROM S<cr> . .Will Save ROM to upper memory.
ROM R<cr> . .Will Retrieve ROM from upper bank and reboot NEWDOS/80.

As the ROM will be saved to an upper memory bank, your 4P should have 128K of memory installed.

```
             00010 ;               ROM/ASM  - Version 1.0
             00020 ;                  NEWDOS80/V2  (III)
             00030 ;                  3rd March 1985
             00040 ;          For NORTHERN BYTES & the PUBLIC DOMAIN
             00050 ;                    by Tony Domigan
             00060 ;  PO Box 150, Thomastown, Victoria, 3074, Australia
             00070 ;MCI-ID:2545121. SOURCE-ID:BCT039. TAB-ID:DOMIPOBOMINN
             00080 ;---------------------------------------------------
             00090 ;
5200         00100        ORG     5200H
5200 E5      00110 START  PUSH    HL               ;Save buffer pointer
5201 CDC901  00120        CALL    01C9H            ;CLS
5204 E1      00130        POP     HL               ;Restore cmd pointer
5205 7E      00140        LD      A,(HL)           ;Command parsed
5206 FE0D    00150        CP      0DH              ;CR=NO
5208 2808    00160        JR      Z,WHAT           ;Skip if CR
520A E6DF    00170        AND     0DFH             ;Mask out L/C
520C FE53    00180        CP      'S'              ;Saveron?
520E CA4652  00190        JP      Z,SVROM          ;Yes, doit
5211 FE52    00200        CP      'R'              ;Restore ROM?
5213 2816    00210        JR      Z,GETROM         ;Yes, doit
5215 217A52  00220 WHAT   LD      HL,PROMPT        ;Menu
5218 CD6744  00230        CALL    4467H            ;Disp menu
521B CD4900  00240 ILOOP  CALL    49H              ;Wait for key
521E E6DF    00250        AND     0DFH             ;Mask out L/C
5220 FE52    00260        CP      'R'              ;Restore ROM
5222 2807    00270        JR      Z,GETROM         ;Yes, doit
5224 FE53    00280        CP      'S'              ;Save ROM?
5226 CA4652  00290        JP      Z,SVROM          ;Yes, doit
5229 18F0    00300        JR      ILOOP            ;Try again
             00310 ;
522B 21F552  00320 GETROM LD      HL,BOOMSG        ;Romboot msg
522E CD6744  00330        CALL    4467H            ;Display msg
5231 CD7052  00340        CALL    DELAY            ;Wait a while
5234 3E31    00350        LD      A,31H            ;Mv upper bank & open ROM
5236 326652  00360        LD      (SWITCH+1),A     ;Post code in swap
5239 210080  00370        LD      HL,8000H         ;Start of moved RAM
523C 110000  00380        LD      DE,0000H         ;Dest of saved ROM
523F 01FF37  00390        LD      BC,37FFH         ;ROM length
5242 CD6452  00400        CALL    SWAP             ;Swap banks & mv data
5245 C7      00410        RST     00H              ;RE-BOOT
             00411 ;
5246 210000  00420 SVROM  LD      HL,0000H
5249 110080  00430        LD      DE,8000H         ;Start bank 3 (high)
524C 01FF37  00440        LD      BC,37FFH         ;Bytes to Transfer
524F CD6452  00450        CALL    SWAP             ;Transfer ROM to Bank3
5252 212F53  00460        LD      HL,MSG1          ;Pt to info message
5255 CD6744  00470        CALL    4467H            ;Print message
5258 CD6000  00480        CALL    60H              ;delay
525B CD6000  00490        CALL    60H              ;delay
525E CD6000  00500        CALL    60H              ;delay
5261 C32D40  00510        JP      402DH
             00520 ;
5264 F3      00530 SWAP   DI                       ;must do this
5265 3E30    00540 SWITCH LD      A,30H            ;upper bank switched high
5267 D384    00550        OUT     (84H),A          ;switch bank to 8000-FFFF
5269 EDB0    00560        LDIR                     ;Move ROM to 8000+
526B AF      00570        XOR     A                ;A=0
526C D384    00580        OUT     (84H),A          ;Reinstate banks
526E FB      00590        EI
526F C9      00600        RET
5270 010000  00610 DELAY  LD      BC,0000H
5273 110300  00620        LD      DE,0003H         ;3 passes
5276 CD954C  00630        CALL    4C95H            ;Multiple delay
5279 C9      00640        RET
527A 0A      00650 PROMPT DEFB    0AH
527B 10      00660        DEFB    10H
527C 20      00670        DEFM    ' Upper Bank Memory Transfer Utility  - by
Tony Domigan '
     55 70 70 65 72 20 42 61 6E 6B 20 4D 65 6D 6F 72
     79 20 54 72 61 6E 73 66 65 72 20 55 74 69 6C 69
     74 79 20 20 2D 20 62 79 20 54 6F 6E 79 20 44 6F
     6D 69 67 61 6E 20
52B3 11      00680        DEFB    11H
52B4 0A0A    00690        DEFW    0A0AH
52B6 10      00700        DEFB    10H
52B7 53      00710        DEFM    'Save ROM to upper 64K "S" '
     61 76 65 20 52 4F 4D 20 74 6F 20 75 70 70 65 72
     20 36 34 4B 20 22 53 22 20
52D1 11      00720        DEFB    11H
52D2 0A0A    00730        DEFW    0A0AH
52D4 10      00740        DEFB    10H
52D5 52      00750        DEFM    'Return the saved ROM  "R" '
     65 74 75 72 6E 20 74 68 65 20 73 61 76 65 64 20
     52 4F 4D 20 20 22 52 22 20
52EF 11      00760        DEFB    11H
52F0 0A0A    00770        DEFW    0A0AH
52F2 1107    00780        DEFW    0711H
52F4 0D      00790        DEFB    0DH
52F5 0A      00800 BOOMSG DEFB    0AH
52F6 10      00810        DEFB    10H
52F7 52      00820        DEFM    'Restoring Saved ROM to allow the booting of
NEWDOS80 '
     65 73 74 6F 72 69 6E 67 20 53 61 76 65 64 20 52
     4F 4D 20 74 6F 20 61 6C 6C 6F 77 20 74 68 65 20
     62 6F 6F 74 69 6E 67 20 6F 66 20 4E 45 57 44 4F
     53 38 30 20
532C 1107    00830        DEFW    0711H
532E 0D      00840        DEFB    0DH
532F 52      00850 MSG1   DEFM    'ROM SAVED - USE R option to REBOOT '
     4F 4D 20 53 41 56 45 44 20 2D 20 55 53 45 20 52
     20 6F 70 74 69 6F 6E 20 74 6F 20 52 45 42 4F 4F
     54 20
5352 0D      00860        DEFB    0DH
5200         00870        END     START
00000 TOTAL ERRORS

BOOMSG  52F5   DELAY   5270   GETROM  522B   ILOOP   521B   MSG1    532F
PROMPT  527A   START   5200   SVROM   5246   SWAP    5264   SWITCH  5265
WHAT    5215
```

EXTRA INSTRUCTIONS FOR EDAS 3.5
by Warick Sands

[This article is reprinted from the TRS-80 SYSTEM 80 Computer Group newsletter (MacGregor, Queensland, Australia).]

In using EDAS 3.5 to load a source file which had been saved from a different Editor Assembler, you may find that although the drive fires up and the program loads, you get the message:

BAD PARAMETERS

If you examined memory with DEBUG, you would find that the file is there, all nicely loaded, but EDAS just doesn't want to know about it.

To overcome this problem, put a minus sign before the filename when you enter the command. This and other commands that are not documented, but which can be used with EDAS 3.5 are:

a) L-filename will load an EDTASM file successfully into EDAS.
b) L-#filename will load an EDAS file that has NO line numbers.
c) W-#filename will write a file without line numbers.

The use of option c) would conserve disk space which may be important in a very long source listing. In loading back a file without line numbers, option b), immediately after it is loaded execute the command:

N100,10

so as to implant line numbers to the listing.

STANDARD SORT ROUTINES

by Ron Zajac

When Charles Butler asked me to document some of these sort routines, I asked "Chuck, who cares about BASIC sorts; everyone got their very own machine code sort with their DOS!" This observation is not without some relevance, but Chuck reminded me that good sort routines are still helpful to people coding applications for compilation, for instance (good point!). So, for those of you interested in 1) the esthetics of good sort techniques, and 2) programs that are fun and interesting to run and watch, here are demo programs for two efficient sort routines called "Heapsort" and "Shell sort".

HEAPSORT

This algorithm is attributed to J. W. J. Williams (Communications of the ACM, vol. 7, June 1964, pp. 347-348). The heapsort treats the one dimensional array as a binary tree with each parent node N having two children; the (2N)th and the (2N+1)th nodes. For example, the a(2) array element has for its children a(4) and a(5) (2N=4 and 2N+1=5; see Figure h1).

```
Unsorted          a(1): 11
                     /   \
                    /     \
                   /       \
                  /         \
         a(2): 4              a(3): 14
              / \                  / \
             /   \                /   \
    a(4): 8  a(5): 18  a(6): 15   a(7): 6   (etc. . .)
         / \       / \       / \       / \
a(8): 20    9    5    3    16   12   17    2
```

Figure h1

The heap process simply puts the biggest (for an ascending sort) value at the top of the heap; at a(1) (see Figure h2).

```
Heaped              20
                   /  \
                  /    \
                 /      \
                /        \
             18            17
            / \           / \
           /   \         /   \
          9     11     16     14
         / \    / \    / \    / \
        8   4  5   3  15  12  6   2
```

Figure h2

Once this has been done, we perform these steps:

step 1: We exchange this largest value with the value at the end of the list.

step 2: We shorten the effective list length by 1, thereby excluding this largest value from subsequent heap processings. If the effective list length is already ONE, we are obviously DONE.

step 3: If we are not finished, however, we again perform a heaping process to bring the largest value in the smaller heap (the next-to-largest value in the total list) to the top. When this is done, we can continue to step 1 and repeat this process.

DONE: Figure h3 shows the finished sort!

```
Sorted                 2
                      / \
                     /   \
                    /     \
                   /       \
                 3           4
                / \         / \
               /   \       /   \
              5     6     8     9
             / \   / \   / \   / \
           11  12 14  15 16 17 18  20
```

Figure h3

```
1000 ' HEAPSORT demonstration program by Ron A. Zajac
1010 '
1020 CLEAR 1000: DEFINT A-V: DEFSTR W-Z: DIM A(15)
1030 '
1040 GOSUB 1400 ' Load array, initialize PRINT USING strings
1050 '
1060 ZZ="Unsorted": GOSUB 1290 ' Display unsorted heap
1070 FOR I=INT(N/2) TO 1 STEP -1: GOSUB 1180: NEXT I
1080 ZZ="Heaped": GOSUB 1290 ' Displ. heap after one pass
1090 '
1100 ZZ="Sorted": I=1: C=N-1
1110 '
1120 FOR M=C TO 1 STEP -1
1130 B=A(1): A(1)=A(M+1): A(M+1)=B
1140 N=M: GOSUB 1180: GOSUB 1290 ' Display after each pass
1150 NEXT M
1160 END
1170 '
1180 K=I
1190 J=2*K
1200 IF J>N THEN RETURN
1210 IF J<N THEN IF A(J+1)>A(J) THEN J=J+1
```

```
1220 IF A(K)<=A(J) THEN B=A(K): A(K)=A(J): A(J)=B: K=J ELSE
RETURN
1230 GOTO 1190
1240 '
1250 ' ****************************************************
1260 '
1270 ' This subroutine prints the tree in a graphic form.
1280 '
1290 PRINT CHR$(28)
1300 IF M<2 THEN PRINT USING W0;ZZ,A(1) ELSE PRINT USING
WT;M,A(1)
1310 PRINT X: PRINT USING W5;A(2),A(3)
1320 PRINT Y: PRINT USING W8;A(4),A(5),A(6),A(7): PRINT Z
1330 PRINT USING WA;A(8),A(9),A(10),A(11),A(12),A(13),A(14),A(15)
1340 PRINT: PRINT: LINE INPUT "ENTER to continue ==>";X0
1350 RETURN
1360 '
1370 ' This subroutine initializes the PRINT USING strings
1380 ' and loads the array of integer values to be sorted.
1390 '
1400 W0="%    %              ##"
1410 WT="Sort Proc, M=##:        ##"
1420 W1="                    /   \"
1430 W2="                   /     \"
1440 W3="                  /       \"
1450 W4="                 /         \"
1460 W5="               ##          ##"
1470 W6="              / \          / \"
1480 W7="             /   \        /   \"
1490 W8="            ##     ##     ##     ##"
1500 W9="           / \    / \    / \    / \"
1510 WA="          ##   ## ##   ## ##   ## ##    ##"
1520 '
1530 X=W1+CHR$(13)+W2+CHR$(13)+W3+CHR$(13)+W4
1540 Y=W6+CHR$(13)+W7: Z=W9
1550 M=0: READ N: FOR A=1 TO N: READ A(A): NEXT A
1560 CLS: RETURN
1570 '
1580 DATA 15
1590 DATA 11,4,14,8,18,15,6,20,9,5,3,16,12,17,2
```

SHELL SORT

The Shell sort is attributed to someone named Shell (sorry, no reference). This sort is an elaboration on a simple bubble sort. The first variation involves the interval over which array elements are compared. In a standard bubble sort, item N is compared with item N+1 over the entire array, and these passes are repeated until no exchanges are made. In the Shell sort, an initial interval is calculated from the number of items in the list divided by 2. And items to be compared in the array are offset by this interval. Subsequent passes divide this interval by 2, as shown in Figure s1.

```
interval = N = 15 (number of items in list)

Pass one, interval =   INT(15/2) =    7
 |-------------------------|
11  4 14  8 18 15  6 20  9  5  3  1 12 17  2

Pass two, interval =   INT(7/2) =     3
 |----------|
 2  4  5  3  1 12  6 11  9 14  8 18 15 17 20

Pass three, interval = INT(3/2) =     1
 |---|
 2  1  5  3  4  9  6  8 12 14 11 18 15 17 20

All Done
                                          |---|
 1  2  3  4  5  6  8  9 11 12 14 15 17 18 20
```

Figure s1

There is one other interesting attribute of this method of "bubbling" through the array. When a comparison shows that a swap of two elements is called for, the values are swapped and a new backwards scan is initialized. The main element pointer (S in the program) is decremented by the interval and these elements are compared for swapping. This continues until no swap is found, or the beginning of the array has been reached. Note that when this phase of the sort is finished, the regular forward movement of comparisons is resumed where it last left off. The best way to see this process is to run the program shown below and study the code. Note that you must press ENTER to initiate the next comparison.

```
1000 ' SHELSORT demonstration program by Ron A. Zajac
1010 '
1020 CLEAR 1000: DEFINT A-V: DEFSTR W-Z: DIM A(15)
1030 GOSUB 1220 ' Load array, initialize PRINT USING strings
1040 '
1050 IF M=1 THEN END ELSE M=M/2: T=1
1060 S=T
1070 GOSUB 1140: IF A(S)>A(S+M) THEN B=A(S): A(S)=A(S+M):
A(S+M)=B: IF  THEN S=S-M: GOTO 1070
1080 T=T+1: IF T+M<=N THEN 1060 ELSE 1050
1090 '
1100 ' ****************************************************
1110 '
1120 ' This subroutine prints the array in a graphic form.
1130 '
1140 FOR A=1 TO N: IF A<S THEN PRINT W1;: ELSE IF A=S THEN
PRINT W2;: F A<S+M THEN PRINT W3;: ELSE IF A=S+M THEN
PRINT W4;
1150 NEXT A: PRINT
1160 FOR A=1 TO N: PRINT USING W0;A(A);: NEXT A: PRINT
1170 LINEINPUT "ENTER ==>";X: RETURN
1180 '
1190 ' This subroutine initializes the PRINT USING strings
1200 ' and loads the array of integer values to be sorted.
1210 '
1220 W0=" ## ": W1="    ": W2="  |-": W3="-----": W4="--| "
1230 '
1240 READ N: FOR A=1 TO N: READ A(A): NEXT A: M=N: RETURN
1250 '
1260 DATA 15
1270 DATA 11,4,14,8,18,15,6,20,9,5,3,1,12,17,2
```

EXTRA RAM FOR THE MODEL I

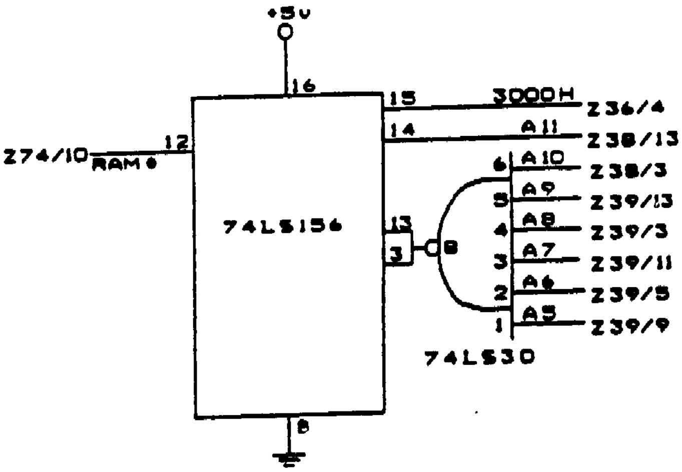
by Dave Kennedy

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

Have you fitted a bank of 64K RAMs to your Model I as per Errol Rosser's instructions that appeared in the May 1984 (Volume 4 Issue 9) issue of the SYDTRUG News (or Volume 5, Number 8 issue of NORTHERN BYTES)? The decoder circuit presented here will enable the RAM addressed in the unused portion of the Model I memory map (3000H through 37FFH, less those addresses used for the Printer and Disk I/O).

I mounted it on a small piece of Vero board and used a piece of double-sided tape to attach it to the main circuit board. It could just as readily be used to enable an EPROM or a 6116 CMOS static RAM in an unmodified keyboard.

The output of the 74LS156 at pin 12 is used to enable the Data bus buffers via the RAM* signal whenever the specified block of memory is addressed.



ADDRESS DECODER

INTERNATIONAL NEWDOS/80 USERS' GROUP

by Greg Small

Welcome to the third installment of this column devoted to the understanding, use, and modification of NEWDOS/80.

This issue's column will be devoted to a small collection of ZAPs. Most I developed, some are from other sources but bear repeating. But first, a few words about the present state of this endeavour.

Since the last issue I have made contact with a number of you. I have had calls or letters from California, Connecticut, Iowa, Massachusetts and Saskatchewan. I have also had a call and a logon to my BBS from Australia.

The general gist of all the contacts that have been made is that all of us, in our own way, are doing something that we need done to Newdos/80. Much duplication is occurring!

I have asked each person I spoke with to send me a disk detailing the work they are doing or have already finished. When I have sufficient material it will be published here and/or copies will be sent to those who contributed. I do not need amazing work, nor do I need a massive amount from any one person.

What is needed is unique things that you have not seen published before. The addresses for all this sort of thing are listed at the end of this column.

Now, on to the ZAPs.

****

This ZAP is for NEWDOS/80 Version 2 for the Mod I. This patch changes DOS to display the current time setting under the NEWDOS/80 READY prompt when the prompt appears. It does not update the time in this location.

```
SYS1/SYS,00,F6  Change CC 67 44 21 to CC C8 51 21

SYS1/SYS,04,DB       Change 03 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 00
                         to 03 CD67 44D9
2157 433E 0D32 5F43 E5CD 6D44 E1CD 6744
D9C9 00
```

I have never done a similar patch for the III but it would be easily accomplished. Send it to me or Jack and we will see it gets published.

****

The next ZAPs are for NEWDOS/80 Version 2 for the Model I.

When the Model I TRS-80 is equipped with a Percom type double density adaptor and the last disk access was to a double density diskette the system will not re-boot properly on a HALT or JP 0000H instruction. Since this is the way APPARAT has chosen to reset the computer under two conditions the following ZAPs were constructed. They will allow proper reset to occur under double density from the BOOT command or from the FATAL DOS ERROR. KEY "R" TO RESET error message.

The ZAPs use a patch area that may be pre-empted by APPARAT without warning or notice.

```
SYS9/SYS,00,CF  Change  02 CD 60 00 76 to 02 C3 D7 51 76

SYS9/SYS,04,EA  Change  00 00 00 00 00 00 00 00 00 00 02
                    to  00 CD 60 00 21 EC 37 36 FE 76 02
```

This ZAP is obviously not needed for the Model III.

****

The following ZAPs are for SUPERZAP/CMD and DIRCHECK/CMD for either Mod I or III. They allow exit from one program and automatic entry to the other if either shift key is pressed.

Optional ZAP to SUPERZAP to exit to DIRCHECK if a shift key is pressed as the T in EXIT is entered.

```
SUPERZAP/CMD,04,D4     Change  C3 2D 40 47  to  C3 E2 6F 47

SUPERZAP/CMD,28,F7
    Change    00 00 00 00 00 00 00 00 00
        to    00 3A 80 38 FE 00 CA 2D 40

SUPERZAP/CMD,29,00
    Change    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
              00 00 00 00
        to    21 F0 6F C3 05 44 44 49 52 43 48 45 43 4B 2F 43
              4D 44 0D 00
```

Optional ZAP to DIRCHECK to exit to SUPERZAP if a shift key is pressed as the N is entered at the main menu.

```
DIRCHECK/CMD,07,10     Change CA 2D 40 FE to CA F7 60 FE

DIRCHECK/CMD,14,E0
    Change    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
              00 00 00 00 00 00 00 00 00 00 00 00 02
        to    00 3A 80 38 FE 00 CA 2D 40 21 05 61 C3 05 44 53
              55 50 45 52 5A 41 50 2F 43 4D 44 0D 02
```

****

This next ZAP came from CompuServe in 1982 or 1983. I have used the Model I version since then and it works well.

This is an example of the duplication that is going on. I have had two people tell me that they have developed similar patches, one of whom was forced to add a /SYS module to get it to work.

This zap to NEWDOS/80 version 2 for the Model III allows changing PDRIVE parameters without writing to disk.

Syntax: Same as using parameter "A" except use "B"
Example: PDRIVE,0,1=7,B

This would set PDRIVE for drive 1 to equal 7 until a reset, or another PDRIVE command with either "A" or "B" is called. It will not change the PDRIVE on the disk, only in memory.

```
SYS16/SYS  02,E8  Change 7E FE 41 12 20 to 7E C3 CB 51 20

SYS16/SYS  04,DF
   Change                                                00
           00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
           00 00 00 00 00 00 00 00 00 00 00 00 02
      to                                                 FE
           41 12 20 03 C3 E0 4F 3D FE 41 20 F8 C5 E5 01 00
           07 21 D3 4D 71 23 10 FC E1 C1 18 E3 02
```

This area is presently unused. If future APPARAT zaps use this area then of course it will wipe this zap out.

You can now put a file protect tab on drive 0 and leave it on.

Here is the same zap for Model I users:

```
SYS16/SYS  02,F7      change  7E FE 41 12 20 06 23
                          to  7E C3 C0 51 00 00 23

SYS16/SYS  04,D3
Change         00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00
   to          00 FE 41 12 20 03 C3 F1 4F FE 42 C2 F7
      4F 3E 41 12 77 C5 E5 01 00 07 21 E2 4D 71 23 10
      FC E1 C1 C3 F1 4F 00
```

****

The following ZAPs are for NEWDOS/80 Version 2 for the Model I and III.

In SUPERZAP/CMD, DEBUG, and the JKL screenprint module non-displayable or non-printable characters are changed to periods (2EH). In the first two items, a space (20H) is defined as non-displayable or non-printable.

As well, in the JKL module, graphics are changed to periods (2EH), if the SYSTEM parameter AX indicates that the printer is not capable of printing graphics. Some printers, (i.e. Radio Shack LP4 - Centronics 737) will print very limited graphics for certain ASCII codes.

The following ZAPs allow you to define the character to be displayed or printed instead of the period, if a replacement is to take place. Also given, is the byte to ZAP, to experiment with, to change the lowest displayable or printable character.

All ZAPs are in the normal APPARAT format excepting that the Mod III addresses are given in brackets beside the Mod I addresses.

This is an optional ZAP to SUPERZAP/CMD to change the character displayed when a non-displayable character is encountered.

```
SUPERZAP/CMD,12,DF (12,E5)  change 3E 2E FE to 3E xx FE
                    replacing xx with the ASCII code (in hex)
                    of the character to be displayed.
```

This is an optional ZAP to SUPERZAP/CMD to change the character printed when a non-printable character is encountered.

```
SUPERZAP/CMD,16,DC (16,E5)  change 3E 2E FE to 3E xx FE
                    replacing xx with the ASCII code (in hex)
                    of the character to be printed.
```

This is an optional ZAP to SUPERZAP/CMD to allow the space (20H) to be displayed as such and not changed to a period (2EH).

```
SUPERZAP/CMD,12,D7 (12,E0)  change D6 21 B9 to D6 20 B9
```

This is an optional ZAP to SUPERZAP/CMD to allow the space (20H) to be printed as such and not changed to a period (2EH).

```
SUPERZAP/CMD,16,DE (16,E7)  change FE 20 30 to FE 1F 30
```

This is an optional ZAP to SYS5/SYS (DEBUG) to change the character displayed when a non-displayable character is encountered.

```
SYS5/SYS,03,75 (03,75)  change 3E 2E E3 to 3E xx E3
                replacing xx with the ASCII code (in hex)
                of the character to be displayed.
```

This is an optional ZAP to SYS5/SYS (DEBUG) to allow the space (20H) to be displayed as such and not changed to a period (2EH).

```
SYS5/SYS,03,6D (03,6D)  change D6 21 B9 to D6 20 B9
```

This is an optional ZAP to SYS3/SYS (JKL module) to change the character printed when a non-printable character is encountered.

```
SYS3/SYS,04,C1 (04,9D)  change 3E 2E CD to 3E xx CD
                replacing xx with the ASCII code (in hex)
                of the character to be printed.
```

****

Next, we have some ZAPs to make us feel like MS-DOSers.

Optional ZAP to NEWDOS/80 Version 2.0 for the Mod I and III to allow the interchanging of the period and slash in filespec definition. Thus a valid filespec would be:

filespec.ext/password:dn

This will be of use to anyone working full time with other systems which use this convention.

This ZAP works! But it has not been extensively tested and feedback on any problems would be appreciated.

One known inconsistency is in the COPY function where the parameter "/ext" is used to define a group of files to be copied. For the time being this must still be stated as "/ext" not ".ext". I would appreciate any feedback from readers who root out the location for this part of the ZAP.

Any programs that do not use the NEWDOS/80 system calls for I/O or filespec handling will most likely need modification to work with this ZAP.

The addresses in the ZAP are identical for Model I or III unless noted.

```
SYS1/SYS,02,5E,(Mod I)      change   3E 2F 12   to   3E 2E 12
SYS1/SYS,02,75,(Mod III)    change   3E 2F 12   to   3E 2E 12
SYS2/SYS,01,48              change   FE 2F 06   to   FE 2E 06
SYS2/SYS,01,4F              change   FE 2E CD   to   FE 2F CD
SYS3/SYS,03,7B,(Mod I)      change   FE 2F 20   to   FE 2E 20
SYS3/SYS,03,58,(Mod III)    change   FE 2F 20   to   FE 2E 20
SYS3/SYS,04,17,(Mod I)      change   3E 2F CD   to   3E 2E CD
SYS3/SYS,03,F0,(Mod III)    change   3E 2F CD   to   3E 2E CD
SYS6/SYS,11,64              change   3E 2F C4   to   3E 2E C4
SYS8/SYS,00,B6              change   FE 2F C2   to   FE 2E C2
SYS8/SYS,01,62              change   3E 2F C4   to   3E 2E C4
```

****

Following patch for NEWDOS/80 Version 2 for the Mod I will do the following on a RESET:

1) Check to see if ALPHA PRODUCTS NEWCLOCK-80 is present and functional.

2) If it is, patch SYS0 temporarily with the two ZAPs outlined in ALPHA PRODUCTS NEWCLOCK-80 instruction sheet.

3) If a printer is on-line and ready a series of bytes will be sent to the printer for initialization provided a shift key was not pressed.

4) If LEFT ARROW is held during reset and the clock is not present the old time and date will be used regardless of the setting of SYSTEM option AY and AZ.

```
SYS0/SYS 11,BF change CD 67 44 3A to CD E6 50 3A

SYS0/SYS 14,06
change            5000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
to                5009 08DD E5FD E50E B0ED
78FE FF28 4B21 0951 113D 4F01 1800 EDB0
11CD 4401 1E00 EDB0 1836 F3CD CD44 2E44
01BC 01CD D544 042E 46CD D544 04CD 0344
FBC9 2143 4001 B503 1603 ED78 0DA2 160F
0777 0707 8677 ED78 A200 8677 2B10 EDC9
3A80 38FE 0020 24DD 21EB 37FD 218A 51DD
7E00 FE3F 2015 06xx FD7E 00F5 DD7E 00FE
3F20 F9F1 D077 00FD 2310 ED3A 4038 CB6F
280F 3EA5 32AB 433A F942 CB87 CB8F 32F9
42FD E1DD E108 D9CD 6744 C9yy
```

In the above patch you must replace "xx" with the hex number of bytes to be sent to the printer and start inserting the bytes to be sent at "yy".

That is about it for this issue. There have been a number of things keeping me from doing more along these lines recently. Probably the most exciting right now is that I have received my US Robotics Courier 2400 modem and my BBS is now capable of running at blinding speed.

Next issue I will be explaining some of the less understood and thus, less used, features of some DOS commands.

If you have any specific requests for this column or wish to contact me full details were in the Volume 6, Number 2 issue of Northern Bytes. The three major changes to those details are that my MCI Mail account has been activated, a section has been created on TBBS - The Business Board System for the Users' Group that is fully accessible by first time callers without any delay, and that the board now supports 1200 and 2400 bps callers as well as 300 bps.

Here is a summary of all the previous details:

Mail: Greg Small, Box 607, Stouffville, Ontario, Canada L0H 1L0
Phone: (416) 640-4400 - 7PM-9PM/weeknights/10AM-6PM/weekends
Dataline: (416) 640-3434 - 24 hours a day (2400/1200/300 - 8N1)
MCI Mail: Username: GSMALL/EAGLE MCI Mail ID #: 251-7579
CompuServe EasyPlex: PPN 72335,560

GOOD NEWS DEPARTMENT

by Bill Baker

[Reprinted from the K.C. South Computer Club newsletter]

The April 22 issue of the Telephone Times, the monthly publication of Southwestern Bell, has some good news for computer users with MODEMS.

A new service recently approved by the Missouri Public Service Commission offers an enhancement to Call Waiting. The service: Cancel Call Waiting. The price: Free.

Hard to believe, isn't it, something free from Bell? Well, it's true and what it means to computer users is that you will not have to be plagued by interruptions on your modem by an incoming call waiting tone because you will be able to cancel this feature on a temporary basis while using the modem. When finished the feature can be reinstated from your phone, and remember it will be FREE.

Southwestern Bell believes that this new feature will make Call Waiting more saleable, not only to owners of computers, but to many others that have seemed to resent being interrupted during a conversation with the beep tone.

You will receive a direct mailing from SW Bell advising you when this feature is available for your line.

[NORTHERN BYTES editor's note: Folks in Missouri are lucky to have this option. However, now that the software to do this is available, is is only a matter of convincing other regional telephone companies to get it and implement it on their lines. Perhaps some of the computer user groups that get NORTHERN BYTES should consider formally contacting your area phone companies, letting them know about this feature and suggesting that you'd like to see it implemented in your area!]

REAL TIME CLOCK/CALENDER

by Stefan Keller (phone 011-61-2-533-1612)
and Chris Tinney (phone 011-61-2-759-5052)

[This article is reprinted from SYDTRUG NEWS, P.O. Box 297, Padstow, New South Wales 2211, AUSTRALIA. The authors of this article may be reached through the club address, or by telephoning the numbers shown above (which include the international access codes for direct dialing from the U.S.A.). If you do telephone, remember the time difference - Sydney is 16 hours ahead of New York and 19 hours ahead of California time. Also note that in the U.S. you can dial toll-free (800) 874-4000 for information on overseas phone rates and the best time to call to get the lowest rate.]

Hello again! Those of you that know which end of a soldering iron to hold might like to try your hand at this Saturday afternoon project. You've all heard of TIME$ (We hope), and you all know how useless it is. (Come on.. how many of you actually type the time and date every time you boot? (LDOS users excepted)). A real time clock/calender with battery back up is exactly what the doctor ordered for you lazy typers. This clock is even smart enough to work out that February has 29 days in a leap year.

If you've glanced at the circuit diagram, you will have seen that the circuit is very simple (considering that it is technically quite complex). The clock consists of a MSM5832 clock chip, (the same as the one used in the ETI 'Little Big Board' computer) a 8255 PIO, and 2 small LSTTL decoder chips. The PIO is only half used, so you can run something else from it as well (We are looking for a speech synthesizer to fit the application). In total, 4 units have been successfully constructed, 3 using 8255 PIOs, and one using a Z-80 PIO (Others are under construction). To use a Z-80 PIO, you need certain control signals not available on the Tandy 40 pin edge connector. If you own a SCRAP-80, you could (if you desire) use a Z-80 PIO.

Well, let's get straight into a description of the clock chip. It is a CMOS chip and hence a little on the slow side for interfacing directly to a microprocessor running at the speed of our Z-80s, this is why the PIO is needed. It gains us the time that the Z-80 just doesn't want to give. The clock has 13 internal registers and one internal frequency generation function. The registers are for things like SECONDS, MINUTES, HOURS, DAY OF WEEK, DAY IN MONTH, MONTH and YEAR.

The hours function has 2 different modes. <1>.. 12 hour mode (with an AM/PM indicator bit). <2>.. 24 hour mode (the same as TIME$). The mode desired is selected when the time is first set. It can be changed however, it is just a matter of resetting a bit in the hours*10 register.

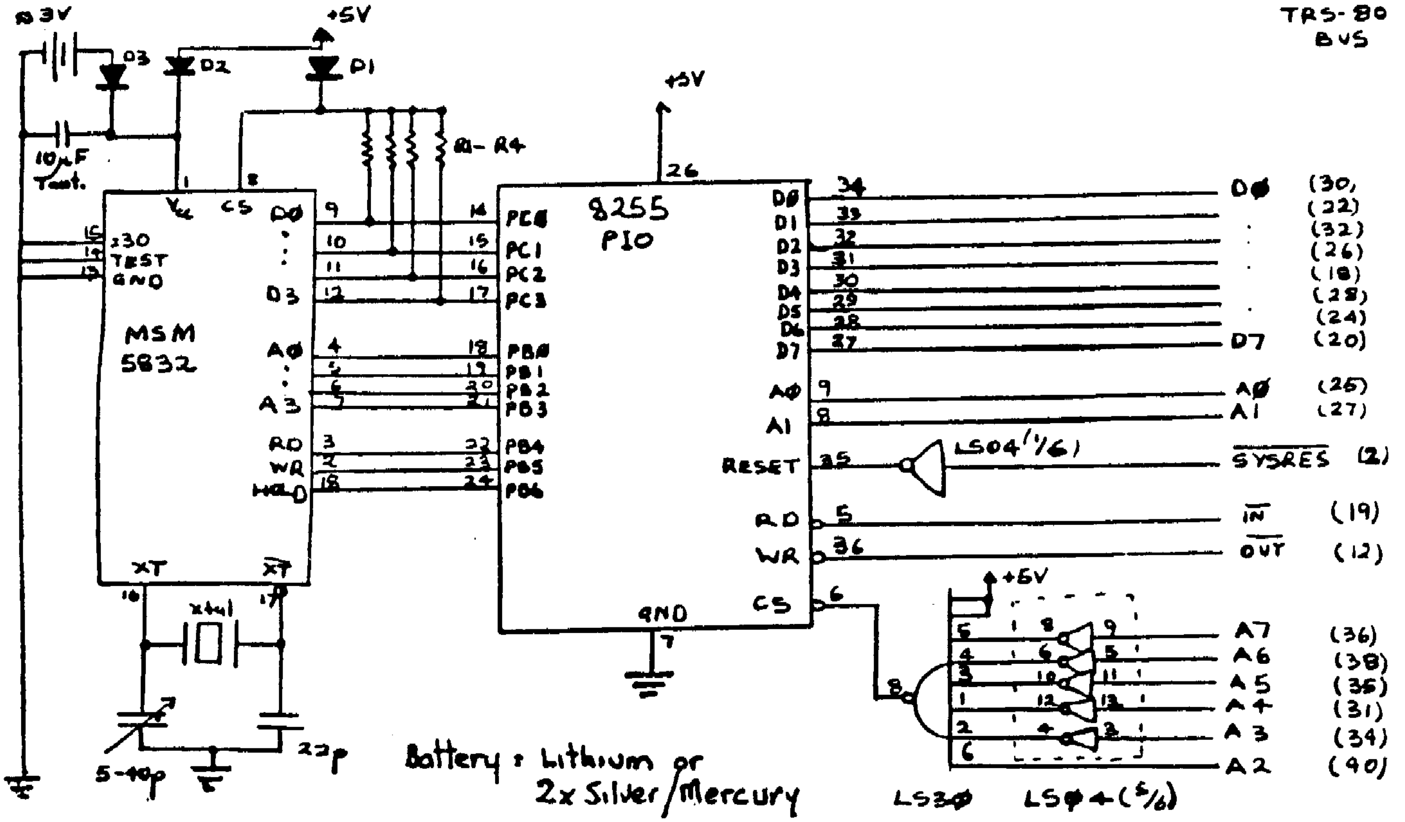
The leap year function is not as intelligent as it appears. The 2 sections (i.e. two 4 bit ports) as an internal function of the PIO. We have used this function, so you are free to use the other 4 bits totally independently for something else. Port C is bi-directional. When you set the clock, it is written to, otherwise the port is a read only port. The addresses, Read, Write and Hold are sent to the clock through port B of the PIO. This port is always a write port. By the way, the Hold function does exactly as its name says... it stops the clock from ticking over.

The Hold function is necessary when you set the time, and optional when you read it. If using a machine language reader, the Hold function is totally unnecessary when reading the time, but still necessary when setting it. A BASIC driver programme needs Hold all the time that the clock is accessed. Don't misunderstand the Hold function, it doesn't stop the clock, just stops it ticking over, so long as the Hold stays high for less than 1 second (easily done if using machine language) then the time will not be slowed. (We haven't said it yet, but the chip is totally POSITIVE logic.. if that means anything to you).

Now a bit on the software that you'll need to run this thing. Because the chip is not directly connected to the Z-80, you have to muck about a little with sending commands. If for instance, you wish to read from the units section of the seconds counter (register 0), you have to send 16 (0 for the register, 16 for the READ line (And optionally another 64 for the HOLD line)) to port 5. (Port B of the PIO) Then you can read the value through port 6 (Port C of the PIO).

Even though you have to go to all this extra trouble to do anything with the clock, all the hard work has been done by us. We can give anyone that wants it, a copy of a program to set the time up, and another that you can AUTO when you boot that says things like "Good Morning! It's 10:34:48 Sunday, 23rd of November 1984". (Well that's what it says at 10:34:48 on the 23rd of November - at other times it will probably be different). This program also resets the value of TIME$ every 6 seconds that the computer is turned on, so all you lazy people don't have to type in a time and date when you boot.

[TEXT CONTINUES NEXT PAGE --->]



Xtal : 32.768 kHz

D1, D2, D3 : 1N914

R1 - R4 : 10K

It's written for NEWDOS, and will probably not work under LDOS - any other DOS's are in the Lap of the Gods. If you want some technical data on the clock or PIO chips we can give you some of that too.

After all that verbose crap on what the thing does, here is a little note on what you will have to do to make it. There are only 4 chips, so you can build the lot onto a piece of veroboard, or wire wrap board (as we have done). The size of the lot should be no more than 5cm * 5cm. This is allowing room for the small battery (either 2 mercury or silver cells, or one lithium cell) needed for power down times, and for the voice synthesizer chips that you may wish to add at a later date.

You will have to run around 20 lines back to the expansion bus of the computer (8 data, 8 address, IN, OUT, GROUND, +5, and SYSRES). We have both mounted ours in our (home built) expansion interfaces. We suppose that you could do the same thing with yours once you have built it and verified that is working. However, you could do as one of our mates at uni has done (not in SYDTRUG)... hang it out the back of his SCRAP-80 on 2 dozen pieces of old wires.

A suggested parts layout is shown [Not in my copy of the SYDTRUG News, it wasn't! -Jack]. This layout uses one lithium battery. Alternatively use the two silver or mercury cells. For those who want to know where to get parts, we know for sure that Sheridan sells the MSM5832 and the crystal [a handy thing to know if you plan on visiting Australia, I guess. -Jack]. Other stores probably sell them also.

This is the second lot of articles we have contributed to the newsletter so far, and as you may have realized we are very into hardware mods and assembly language programming. The few mods we have presented so far are only some of ones we have constructed. Others which we have made internal to our CPU's are a DeGlitch (get rid of that video snow), a speedup for Japanese CPU's, a Dottizer to turn those adjacent video dots on the screen (which are lines now) into real dots, and a custom Character Generator.

Anyone interested in these mods or having any good ones of their own, can see us at the user group meetings (just yell out our names) or give us a call at the above numbers.

If anyone has bothered to build the synthesizer we described several months ago, from the SN76489, then you may be interested to know that I have written a music editor for it. It operates similarly to the FRED program, using EDAS as an editor, and has all sorts of nifty features like CALLS, JUMPS, conditional statements, repeats, etc. Any SYDTRUG member who wants a copy merely has to ask! So all of you who want to play around with computer music, and can't afford a pre-built synthesizer - GET HACKING!

WARNING ON ADDING 64K MEMORY IC'S TO THE MODEL 4/4P

Not all 64K memory IC's use the same method of memory refreshing. For example, in the Model 4/4P a seven bit refresh (on lines A0-A6) is used. Some IC's use an eight bit refresh, and the two types are not interchangeable. If the data sheet for a given 64K memory IC says that it uses "8 refresh lines" or "256 refresh cycles", it will not work in a Model 4/4P - you need IC's with "7 refresh lines" or "128 refresh cycles" (please note that this information is unverified - I sure hope I didn't get it backwards!). If you've tried to add expansion memory to your 4/4P and it didn't work, this may be the reason why.

SELF-BOOTING NEWDOS/80 system disk for the MODEL 4P.....AGAIN.

by Art Rasmussen

After reading the several articles on self-booting NEWDOS disks, I decided to try my luck. Although Mr. Domigan's method in NORTHERN BYTES Volume 6, Number 2 does work, I didn't like the idea of moving my directory to lump 29 from its normal position on lump 17. It would cause PDRIVE incompatibility problems with all the other Model III NEWDOS disks I have and I really didn't relish the thought of converting all of them, so I modified his process.

I am using a standard 40 track double density NEWDOS/80 version 2 with a normal 10 sector (2 gran) directory which begins on lump 17.

PDRIVE setting both drives 0 and 1:

```
TI=A,TD=E,TC=40,SPT=18,TSR=0,GPL=2,DDSL=17,DDGA=2
```

STEP 1.

Make a backup of your NEWDOS disk by typing:

```
COPY,0,1,,CBF,/SYS,FMT<CR>
```

STEP 2.

Insert the newly created system disk in drive 0 and create a directory entry for the MODELA/III file by typing:

```
CREATE MODELA/III:0<CR>
```

From here on I used the utilities in SUPER UTILITY. Make sure you use the N3DR specifier for the configuration table.

STEP 3.

If you did not create a system disk as outlined in step one you will need to make sure that lumps (relative tracks) 30 through 36 are not assigned to any file. The ALLOCATION MAP routine of SUPER UTILITY will tell you this.

The reason for this is that beginning on lump 30, relative sector 6 we will create a dummy directory for the MODEL 4P loader. This sector corresponds exactly to real track 17, sector 0 which is where relative byte 2 on the boot sector says the directory is supposed to be. Notice we will NOT move the real directory from lump 17. We will also copy the MODELA/III file to the sectors immediately following this dummy directory.

STEP 4.

Using SUPER UTILITY, zero out lump 30 relative sectors 6,7,8. Modify lump 30, relative sector 6, relative byte 0, from 00 to 17. This sector is the fake HIT (Hash Index Table) sector and 17 is the hash code for MODELA/III.

Next go to lump 30 relative sector 8 and beginning at relative byte 0 enter the following:

```
1000 7C00 004D 4F44 454C 4120 2049 4949
9642 9642 3900 1129 FFFF FFFF FFFF FFFF
```

This is the fake FPDE for the MODEL 4P loader. Make sure that these modifications begin exactly on the lumps, sectors, and bytes specified. Also note we do not need to read-protect these sectors as if they were part of a real directory, the 4P loader doesn't care.

STEP 5.

Next I used the FILE LOCATIONS routine of SUPER UTILITY to find out where the 57 sectors of the MODELA/III ROM image were located on the MODEL 4 disk. (On mine it began on track 36, sector 12. If you are copying from the MODELA/III file disk, it begins on track 0 sector 4)

STEP 6.

Using the COPY SECTORS routine of SUPER UTILITY copy the 57 sectors from the MODEL 4 disk to the NEWDOS disk BEGINNING at lump 31 relative sector 2 (this corresponds to real track 17 sector 6).

STEP 7.

Modify the real MODELA/III FPDE (lump 17) from:

```
1000 0000 004D 4F44 454C 4120 2049 4949
9642 9642 0000 FFFF FFFF FFFF FFFF FFFF
```

TO

```
1000 0000 004D 4F44 454C 4120 2049 4949
9642 9642 3F00 1E2C FFFF FFFF FFFF FFFF
```

You may wish to change the first byte from 10 to 18 to make the MODELA/III file invisible.

STEP 8.

Use the REPAIR GAT SECTOR routine of SUPER UTILITY to correct the GAT table. This adjusts the GAT table for the new MODELA/III ROM image file and the transferred directory. NEWDOS will treat this as one large file.

You may now reboot the disk by pressing the reset key. You do not need to hold any keys down, even if you have just turned the 4P on, the ROM image will load and NEWDOS will boot.....I hope.

Art Rasmussen
612 West Hillcrest
Keene, Texas 76059
Phone (817) 641-8922

The TRUTH about RELATIVE Tracks

by John T. Phillipp

[This article is reprinted from The INTERFACE newsletter of the San Gabriel Valley TRS-80 Users Group.]

There have been a few questions regarding the articles reprinted from Northern Bytes in the past few issues of the INTERFACE explaining how to make a NEWDOS/80 system disk which will self boot on the Model 4P - that is, a disk which will load the ROM image file (MODELA/III) and then load the NEWDOS/80 system.

Much of the confusion centers on the concept of "true tracks", "physical tracks", and "relative tracks", so I will try to explain these concepts here. In order to make the explanation as clear as possible, it is necessary to go back to basic basics, so I hope the more advanced members of the club won't get bored.

A floppy disk is a circular piece of magnetic media. The head of the disk drive writes data onto the disk in concentric circles called "tracks". There are usually 40 tracks on a disk (although there may be as many as 80). The tracks are numbered 0 to 39. The ones closest to the center of the disk are the higher numbered ones, and the outermost track is track 0.

Note that these tracks are concentric circles, not a continuous spiral like the groove on a phonograph record. Once a track has been written, the head of the drive steps in or out a small distance and writes another one.

Each track is broken up into 256 byte pieces of information, and each of these pieces is called a sector. When floppy disks used single density recording, there were 10 sectors on each track. For obscure reasons, 5 sectors (1/2 track) was called a "granule", usually shortened to "gran". The granule was the smallest amount of space that the Disk Operating System (DOS) could allocate to a file when it was written to the disk. Every file took up at least 1 gran of disk space (5 sectors) even if it was less than one sector long. The rest of the space was just wasted.

As long as everyone used single density, it was generally agreed that there would be 10 sectors per track, and 5 sectors per gran (which gave 2 grans per track). But when double density recording became popular the problems started.

"Double density" is not quite twice single density. That is, there are not 20 sectors per track as you would expect, but only 18. If 5 sectors equal one gran, then that means each track would have 3 3/5 grans. TRSDOS, LDOS, and MULTIDOS solved the problem of an incomplete number of grans per track by assigning 6 sectors to each gran, with 3 grans per track. Each track was numbered by its physical location on the disk - the outermost was 0, next one in was 1, next one in was 2 and so on.

NEWDOS/80 handled things differently. The NEWDOS/80 system continued to consider a "relative" track as being 10 sectors, and a gran as 5 sectors just as they had under single density. Each physical track on the double density disk had 18 sectors, but the system considered the first 10 sectors to be "relative" track 0, the next 10 sectors to be "relative" track 1, the next 10 to be "relative" track 2, and so on.

As you can see, the first 10 sectors ("relative" track 0) will be on physical (or "true") track 0. The next 10 sectors ("relative" track 1) will be made up of the last 8 sectors of physical track 0, and the first 2 sectors of physical track 1. The next 10 sectors ("relative" track 2) will be sectors 3 to 12 of physical track 1. The next 10 sectors ("relative" track 3) will be sectors 13 to 18 of physical track 1 (6 sectors), and the first 2 sectors of physical track 2. Get the picture? A "relative" track might be partly on one "true" (physical) track and partly on another.

A NEWDOS/80 disk with 40 physical tracks contains 40 * 18 = 720 sectors (18 sectors on each physical track). Since there are 10 sectors in each "relative" track, that SAME disk has 720 / 10 = 72 "relative" tracks.

Now, most of the time the user doesn't know or care about "relative" or "true" tracks. We hire a DOS to worry about that. We SAVE a BASIC program, or write a data file, and the DOS figures out where there is free space on the disk, and writes the file or program there. The DOS makes a note in the directory of where it put the file. Later, when we LOAD the program, or read the file, the DOS looks in the directory, notes where on the disk the file is, moves the head of the disk drive to that location, and reads the file. The USER still doesn't know where on the disk the file was, but the DOS does. TRSDOS writes in the directory which "true" (physical) track contains the file, NEWDOS/80 writes which "relative" track contains the file. That's all.

For example, if a file starts on "true" track 30, TRSDOS writes a 30 (actually 1E hex, which is the same as 30 decimal) in the directory. If the file is in the SAME physical location (track 30) on a NEWDOS/80 disk, the NEWDOS/80 directory entry will say the file starts on "relative" track 54 (36 hex). Why "relative" track 54? Because the file starts at sector 540 on the disk (30 physical tracks * 18 sectors per track), and each "relative " track consists of 10 sectors. Sector 540 is the beginning of "relative" track 540 / 10 = 54.

A file which begins at "relative" track 54 starts at exactly the same place on the disk as one that begins at "true" track 30. But for NEWDOS/80 to find the file, the directory entry must say 54, for TRSDOS to find the file, the entry must say 30.

Normally, it makes no difference to us, the users. All NEWDOS/80 directory entries are in "relative" tracks, but only NEWDOS is going to read that directory, and it expects to find its entries in "relative" tracks. TRSDOS can't read the NEWDOS/80 directory because it expects the file locations to be given in "true" tracks (besides, TRSDOS can't even FIND the directory on a NEWDOS/80 disk because the directory location itself is given in "relative" tracks, and TRSDOS thinks that number is "true" tracks and it gets lost => DIRECTORY READ ERROR).

But, in order to make a self-booting NEWDOS/80 disk for the 4P (that's what all this is about, remember?) we have to put the ROM image file (MODELA/III) on the NEWDOS/80 disk in such a way that TRSDOS can find it, because the boot loader in the Model 4P hardware uses "true" tracks, just like TRSDOS.

How can we do that?

First, we must copy the MODELA/III file onto the NEWDOS/80 disk so that it starts at the beginning of a "true" track, even though NEWDOS/80 uses "relative" tracks. Using the CREATE command of NEWDOS, we create a file called MODELA/III. This creates a directory entry for the file, even though there is no file on the disk. Then, using SUPERZAP we go to the directory entry, and change the start of the file to "relative" track 54, which we know is the start of "true" track 30 (if "we" don't know that, "we" had better go back and re-read the first part of this article). Now we copy the MODELA/III file from a TRSDOS disk. NEWDOS looks at the directory of its disk, sees that a file called MODELA/III exists there, and that it starts at "relative" track 54 ('cuz we zapped that in, remember?), and copies the MODELA/III file from the TRSDOS disk, thinking it's over-writing the file already on the NEWDOS disk. Result: MODELA/III ends up on the NEWDOS disk starting at "relative" track 54, which is the same as "true" track 30.

The problem is that the 4P won't be able to find the MODELA/III file. The loader will read the directory entry and look for the file on "true" track 54 - it sees the 54 in the directory and thinks all the entries are in "true" tracks. So we use SUPERZAP again. This time we change the 54 to 30. Now the 4P loader sees the 30 in the directory, goes to "true" track 30, finds the MODELA/III file there, and loads it. Mission accomplished.

There are still a few loose ends, though. For one thing, NEWDOS/80 now cannot find the MODELA/III file. NEWDOS will read the directory entry, and look for the file on "relative" track 30, which is "true" track 16 and a fraction (30/1.8). It obviously won't find the file there, since we know we put it on "true" track 30!! But NEWDOS never NEEDS to find the file. Once the 4P loader loaded MODELA/III into RAM, the 4P became a Model III. The ROM image file won't be used again.

If you use DIRCHECK, the NEWDOS utility for checking the disk directory for errors, the directory appears to be a big mess!

```
NEWDOS4P    01/17/85
1E,0   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          42   SYS16/SYS
          67   MODELA/III

1E,1   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          25   SYS11/SYS
          67   MODELA/III

1F,0   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          41   SYS15/SYS
          67   MODELA/III

1F,1   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          40   SYS14/SYS
          67   MODELA/III

20,0   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          21   SYS7/SYS
          67   MODELA/III
```

```
20,1   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          45   SYS19/SYS
          67   MODELA/III

21,0   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          44   SYS18/SYS
          67   MODELA/III

21,1   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          27   SYS13/SYS
          67   MODELA/III

22,0   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          24   SYS10/SYS
          67   MODELA/III

22,1   xxxxx GRANULE ALLOCATED, BUT ASSIGNED TO MULTIPLE FILES
          46   SYS20/SYS
          67   MODELA/III

36,0   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

36,1   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

37,0   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

37,1   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

38,0   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

38,1   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

39,0   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

39,1   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

3A,0   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE

3A,1   xxxxx GRANULE ALLOCATED BUT NOT ASSIGNED TO ANY FILE
```

But it's not as bad as it looks. Since NEWDOS thinks the MODELA/III file starts at "relative" track 30 (since we put "30" into the directory so the 4P loader could find the file on "true" track 30) which is 1E hex, DIRCHECK warns us that other files start at "relative" track 30 also. We needn't worry about that, because the MODELA/III file isn't really there. DIRCHECK also informs us that the granules starting at "relative" track 54 (36 hex) are allocated, but not assigned to a file. NEWDOS doesn't know that the MODELA/III file actually starts at "relative" track 54, because the directory entry tells it that the file starts at "relative" track 30. WE know the file starts at "relative" track 54, though, since we went to a fair amount of trouble to put it there. We marked the granules where the file is actually located as "used" (allocated) so that NEWDOS won't use those granules when it writes another file to the disk, and accidentally overwrite the MODELA/III file.

I hope this clarifies things a little.

NEWDOS/80 ENHANCEMENTS WANTED

by Jack Decker

With all the ZAPs that have been developed for NEWDOS/80, there are still a few categories of patches that I'd like to see for NEWDOS/80 users:

1) Zaps to enable date stamping of files, in a manner similar to DOSPLUS, LDOS or MULTIDOS.

2) Zaps to use the password fields of a directory as "library" identifiers. NEWDOS/80 supports passwords, but hardly anyone uses them. But, since a SYSTEM flag can be set to tell the DOS to ignore passwords, it seems to me that the password area would be a good place to store information that "links" files together.

For example: Suppose you buy a program called CHEKSPEL (I just made up that name), a spelling checker program that has, let's say, 15 support programs on the disk (main program, dictionary files, configuration programs, etc.). Now, suppose you want to be able to copy those 15 programs to one of your 80 track double-sided disks (that already has, let's say, 150 programs on it). Later you want to know exactly which of the 15 programs on that disk go with CHEKSPEL, and perhaps you want to copy those programs only to another disk. This would be easily accomplished if one parameter were added to DIR and four to COPY:

To DIR - An argument format that searches for a given user password (perhaps something similar to DIR *CHEKSPEL which would display only files with a user password of CHEKSPEL, and which of course could be used with other arguments to further narrow the search).

To COPY - First, a new parameter DUPD (Destination User PassworD). For example, COPY 1 2,,CBF, ... ,DUPD=CHEKSPEL would copy all files specified by the COPY command, and add the "user password" of CHEKSPEL to each file copied to the destination disk. Second, new parameter NDF (No Destination File) would tell NEWDOS/80 <u>not</u> to copy a file if it already exists on the destination disk, and ideally, it should also print an advisory message to the operator warning that the file has not been backed up. For example, COPY 1 2,,CBF, ... ,NDF would copy all files specified by the COPY command <u>unless</u> a file with the same filename and extension exists on both the source and destination disks. In that case, a message similar to:

filename/ext FILE ALREADY EXISTS

would be printed (notice that we have used NEWDOS/80 error message 53 decimal/35 hexadecimal to save space) and the COPY routine would skip that file and continue on with the next file to be copied, if any.

Third, RUPD (Remove User PassworD) would be similar to DUPD except that it would write a "null password" to the destination disk, in effect deleting any user passwords encountered during the copy. For example, COPY 1 2,,CBF, ... ,RUPD would Remove User Passwords from the filenames as they are copied to the destination disk. Finally, a new parameter SUPD (Source User PassworD) would only copy files that have the specified user password on the source disk. For example, COPY 1 2,,CBF, ... ,SUPD=CHEKSPEL would only copy files that have the user password CHEKSPEL on the source disk. Note that none of these arguments would modify the user password on the source disk - only the user password on the destination disk would be affected.

In other words, you could copy all the CHEKSPEL files from your master disk to your 80-track disk, using the DUPD (Destination User PassworD) parameter to add the same password to all files, and the NDF parameter to make sure you don't accidentally copy over another program already on the disk. Then, later on, if you wanted to re-create the original master disk, you could use the SUPD and RUPD parameters together (SUPD to select only the files with the proper password on the source disk, and RUPD to "strip off" the passwords when copying to the new disk). This could be an easy way to keep track of "libraries" of related files on a single disk.

3) One more thing for you NEWDOS/80 hackers to consider. NEWDOS/80 has a nasty habit of directly "PEEKing" the keyboard (rather than going through the keyboard driver) for keys such as the up-arrow, left-arrow, etc. See the notes on ZAP 061 (Model I version) for a description of the problem. Since "device independence" is one of the features that LDOS/TRSDOS 6 users constantly mention as being a superior feature of their DOS ("device independence" really means that input and output can be "routed" from one device to another or from a device to a disk file), I feel that NEWDOS/80 could be improved by a) Patching the ROUTE command to allow routing to a disk file, as well as to other output devices, and b) As much as possible, getting rid of code that bypasses the keyboard (or other) drivers and making NEWDOS/80 go through the proper drivers for everything. Modification (b) would also eliminate the problem of keystrokes being "stored" by the type-ahead feature of certain programs (Allwrite!, Alan Johnstone's NEWDOS/80 modifications, etc.) while using the left-arrow and ENTER keys to scroll through a LISTed file, and it would also eliminate most of the problems encountered when trying to run NEWDOS/80 from a remote site, through a phone line and MODEM.

Sorry, none of these modifications to NEWDOS/80 have been created yet (to my knowledge), and I simply don't have the time nor the knowledge of the inner workings of NEWDOS/80 to create such patches. However, I know that there are many NORTHERN BYTES readers that like to hack around with NEWDOS/80, so if you manage to figure out how to do any of the above mods, please, by all means let us know about it!

FINANCIAL PLANNER

by Bob Koehler

While visiting some young friends during a trip to Florida, we were discussing life insurance, IRA's, and annuities, and I realized that they had no idea how much retirement income could be derived from a modest annual investment, over a period of 20 or 30 years. I threw together a couple of short programs, to show them, and they were so useful that I decided to combine them into one, dress it up a bit, for convenience, and send it to Northern Bytes.

I think the instructions within the program are complete enough that I don't have to add any more explanation here.

```
1 CLS:PRINTTAB(15)"F I N A N C I A L  P L A N N E R":PRINT
2 PRINT"By: R. B. Koehler, R.D. #4, Box 174, Hopewell Junction, NY
 12533"
3 PRINT"Do you want instructions -
 (Y/N)":INPUTR$:IFR$="N"THEN11
4 CLS:PRINT"This program may be used in three ways:
1. To determine the final value of an investment of a fixed annual am
ount, at a constant interest rate, for a specified num-ber of years."
5 PRINT"2. To calculate the annual income from an initial investmen
t, at a constant interest rate, that will reduce the investment to zer
o in a specified number of years. (An annuity.)"
6 PRINT"3. A combination of the two, that will show the annual inco
me  that can be derived from the proceeds from an annual investment
 program."
7 PRINT:PRINT"At the end of each calculation using options 1 or 2, p
ressing the spacebar will start a new cycle. To return to the menu,
  press 'M'. It is not necessary to re-enter a value that is un-
 changed from the previous calculation. ";
8 PRINT"Just press <ENTER>.":PRINTTAB(19)"(PRESS ANY KEY TO
CONTINUE)";:GOSUB26
9 CLS:PRINT"Option 3 returns to the menu at the end of each set of c
al-
 culations. However, if the interest rate and/or number of yearsis th
e same for each part of a set, they need not be entered thesecond tim
e, as above."
10 PRINTTAB(20)"PRESS ANY KEY TO BEGIN":GOSUB26
11 A$="$$######,###.##":DEFDBLP,A
12 CLS:PRINT"ENTER:
<1> FOR INVESTMENT VALUE
<2> FOR ANNUITY INCOME
<3> FOR ANNUITY INCOME FROM ANNUAL INVESTMENT
<4> TO END":INPUTK:ONKGOTO20,13,19,29
13 CLS:PRINT"ANNUAL INCOME FROM AN ANNUITY":PRINT
14 PRINT"ENTER STARTING AMOUNT":INPUTP:PRINTCHR$(27);P:G
OSUB27
15 I=P*R/100*(1+R/100)[N/((1+R/100)[N-1)
16 PRINT"ANNUAL INCOME IS:";USINGA$;I:AI=I*N
17 PRINT"TOTAL VALUE IS:";USINGA$;AI;
18 GOSUB26:IFK=3THEN12ELSEIFQ$="M"THEN12ELSEPRINT:PRINT
:GOTO14
19 CLS:GOTO21
20 CLS:PRINT"TOTAL VALUE OF AN INVESTMENT OF A FIXED ANN
UAL AMOUNT":PRINT
21 PRINT"ENTER ANNUAL AMOUNT TO BE INVESTED:":INPUTD:PR
INTCHR$(27);D:GOSUB27
22 P=0:FORI=1TON:P=(P+D)*(1+R/100):NEXT
23 PRINT"VALUE AT MATURITY:";USINGA$;P:PRINT:PRINT
24 IFK=3PRINT"FOR ANNUITY:":GOSUB27:GOTO15
25 GOSUB26:IFQ$="M"THEN12ELSE21
26 Q$=INKEY$:IFQ$=""THEN26ELSERETURN
27 PRINT"ENTER EFFECTIVE ANNUAL INTEREST RATE (%)":INPUT
R:PRINTCHR$(27);R
28 PRINT"ENTER NUMBER OF YEARS":INPUTN:PRINTCHR$(27);N:RE
TURN
29 END
```

PATCHES FOR TRSDOS 6.2

These patches were obtained from various sources, but it can be assumed that many of them originated with Logical Systems, Inc. The information for PATCH #3 below was extracted from MICRO NOTES, the newsletter of the Dearborn TRS-80 Users Group. All patches are for use with TRSDOS version 6.2 only, and remember - NEVER patch your master disk, always do your patching on a backup copy! One more thing - none of these patches were developed by us here at NORTHERN BYTES, and we haven't tested them, so apply them at your own risk!!!

PATCH #1: Two optional patches to TRSDOS 06.02.00 to force logical drives two and three to be present upon boot. Note that these patches are applicable to TRSDOS 06.02.00 level AN ONLY!! Do not try to use them for any other release of the DOS.

To enable drive two:

```
PATCH BOOT/SYS.LSIDOS (D02,84=C3:F02,84=C9)
```

To enable drive three:

```
PATCH BOOT/SYS.LSIDOS (D02,8E=C3:F02,8E=C9)
```

It is not necessary to apply these patches to enable drives two and three, since the following three commands may be used to achieve the same effect:

```
SYSTEM (DRIVE=2,ENABLE)<ENTER>
SYSTEM (DRIVE=3,ENABLE)<ENTER>
SYSGEN
```

Note that SYSGEN files also store the state of other system parameters, such as the FORMS filter, etc.

PATCH #2: The following patch may be used to force the FORMAT utility to prompt for the number of sides when formatting. Note that this patch is not to FORMAT!

```
PATCH SYS0/SYS.LSIDOS (D00,81=11:F00,81=31)
```

If you don't want to patch the DOS, the same effect may be accomplished by using the command:

```
MEMORY (A="L",B=X'11')
```

and then doing a SYSGEN. NOTE: Read Patch #3 below before doing this one, you may prefer it and the two patches are mutually exclusive (that is, you may do either Patch #2 or Patch #3, but not both, since they change the same byte!)

PATCH #3: This is a variation of patch #2. Not only can FORMAT be made to prompt for the number of sides, it can also be made to prompt for the boot stepping rate. You can even use 8" drives if you use the FLOPPY/DCT driver. This is accomplished by altering LFLAG$, a low memory flag that is defined as follows:

Bit 7 - Reserved for Interrupt Mode 2 hardware.
Bit 6 - Reserved for Interrupt Mode 2 hardware.
Bit 5 - If set, FORMAT will not prompt for the number of sides.
Bit 4 - If set, FLOPPY/DCT will inhibit the 8" query.
Bit 3 - Reserved.
Bit 2 - Reserved.
Bit 1 - Reserved.
Bit 0 - If set, FORMAT will not prompt for step rate.

So, if you want FORMAT to prompt for the number of sides and the boot stepping rate, or if you want to be able to use 8" drives when using FLOPPY/DCT, you can use:

```
PATCH SYS0/SYS.LSIDOS (D00,81=00:F00,81=31)
```

Once again, if you don't want to patch the DOS, the same effect may be accomplished by using the command:

```
MEMORY (A="L",B=X'00')
```

and then doing a SYSGEN.

In case you're wondering about the strange syntax of the above MEMORY command, 'A="L"' means that LFLAG$ should be changed, regardless of where it is in memory.

PATCH #4: This is a repeat of an earlier patch, just in case you missed it the first time around. It was first revealed by Hardin Brothers in *80-Micro*. To activate the KILL command without affecting the REMOVE command (that is, either KILL or REMOVE will work interchangeably), use this patch:

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

PATCH #5: The following are optional patches for FORMAT on TRSDOS 06.02.00 level AN that will allow the format of a disk with garbage data on it left over from the certification process, or a "software bulk-erase". Please note that these patches merely "band-aid" the problem, and the disk should really be properly bulk erased with a bulk tape eraser (available at Radio Shack or HiFi/Stereo or Video stores).

It is believed that the problem is related to the disk driver, specifically that the read status loop at X'0E70' to X'0E75' may not terminate on a "CRC error during sector header read" error passed back from the FDC. It is not possible to patch the driver, and a true solution must wait for the next re-assembly of the system.

```
. FORMATA/FIX - 04/23/85 by --jjkd--
.
. Optional patch to FORMAT on TRSDOS 06.02.00
. This patch is for FORMAT from TRSDOS 06.02.00 Level AN ONLY!!!
```

```
.
. This patch will force FORMAT to always skip the "is this disk formatted"
. test when formatting a disk to prevent a system hang when farkled data is
. present on the disk. You should really bulk erase the diskette anyway.
.
. Use build to enter the patch lines and apply using the command
.
. PATCH FORMAT/CMD.UTILITY FORMATA/FIX
.
X'3492'=00 00 00 C3
.
. End of patch

. FORMATB/FIX - 04/23/85 by --jjkd--
.
. Optional patch to FORMAT on TRSDOS 06.02.00
. This patch is for FORMAT from TRSDOS 06.02.00 Level AN ONLY!!!
.
. This patch will force FORMAT to skip the "is this disk formatted"
. test when formatting a disk if the ABS parameter is specified.
. This is to prevent a system hang when farkled data is
. present on the disk. You should really bulk erase the diskette anyway.
.
. Use build to enter the patch lines and apply using the command
.
. PATCH FORMAT/CMD.UTILITY FORMATB/FIX
.
X'3492'=CD 16 3A
X'3A16'=3A 27 35 B7 C0 CD 27 2A C9
.
. End of patch
```

PATCH #6: These patches may prove useful when running some of the new 3.5 inch drives.

```
. Possible patches to 6.2.0 Level AN for
. additional CKDRV time for bizarre index
. pulses from new 3.5 inch drives.
.
. The single byte changes may be increased up to FF
. The double byte changes are in standard LSB MSB
. format, and may go up to FF FF
.
. The first and middle lines for BACKUP, FORMAT and
. SYS12 are probably the crucial ones. They may have
. to be increased a whole bunch.
.
. Note that this will slow down the performance of
. the system considerably whenever a @CKDRV is done.
.
PATCH BACKUP/CMD.UTILITY (D15,F6=12:F15,F6=09)
PATCH BACKUP/CMD.UTILITY (D15,FC=40 00:F15,FC=20 00)
PATCH BACKUP/CMD.UTILITY (D16,10=40 00:F16,10=20 00)
.
PATCH FORMAT/CMD.UTILITY (D0D,D4=18:F0D,D4=0F)
.
PATCH SYS2/SYS.LSIDOS (D00,E2=12:F00,E2=09)
PATCH SYS2/SYS.LSIDOS (D00,EC=40 00:F00,EC=20 00)
PATCH SYS2/SYS.LSIDOS (D01,02=40 00:F01,02=20 00)
.
PATCH SYS12/SYS.LSIDOS (D03,7A=12:F03,7A=09)
PATCH SYS12/SYS.LSIDOS (D03,84=40 00:F03,84=20 00)
PATCH SYS12/SYS.LSIDOS (D03,9A=40 00:F03,9A=20 00)
.
. END OF PATCHES
```

WHICH LONG DISTANCE CARRIER ARE YOU CONNECTED TO?

If you are in an area that has "Dial 1" access to all long distance carriers (one in which you were asked to select a long distance carrier that would handle your calls when you dial "1" plus an area code and phone number), here's how you can tell which carrier you're hooked up to: Dial 1-700-555-4141 (yes, that's a 700 area code, it's a new code that does not have a fixed function and can be utilized by the various long distance carriers as they see fit. AT&T uses it for customer-controlled teleconferences). When you dial this number, you should hear a recording telling you which carrier you are connected to.

This can be important for two reasons. In the first place, the Federal Communications Commission has said that local telephone companies may no longer route all "default" traffic to AT&T. Thus, if you live in an area where "Dial 1" access is available and you do not specifically choose a long distance carrier, you may one day find yourself assigned to a carrier at random. If the quality of your MODEM transmissions suddenly drops, you might want to dial the 700 number to make sure you're still with the carrier that you think you're with!

Also, it turns out that AT&T has submitted some false order for customers during the selection process (source: Communications Week). Thus, in a few areas, customers who thought they were signing up with MCI or Sprint or some other carrier actually remained with AT&T due to the fact that a false order was submitted by AT&T for their line (the most severe case of this happend in Bell operating areas served by Nynex Corporation, where AT&T may have submitted as many as 61,000 false orders!). Naturally, AT&T blamed the foulup on a faulty computer software program (this correlates with Decker's Law of the Assignment of Fault! When all else fails, blame the computer!).

Then, of course, there are areas like Sault Ste. Marie, where I got a form from Michigan Bell, asking me to select a long distance carrier. To sign up with the one company I wish to use, I was instructed to place an X next to the company name on the response card. So, I referred to the card, eagerly anticipating the ability to select a long distance carrier other than AT&T - but the card reminded me of a Russian ballot, since there was only one long distance carrier listed on the card (you get one guess as to who it was)! No, I didn't check the box and send it back - I figure that if I don't get a choice, I'll just let them assign me to one of the "participating carriers" at random. Let's see, if I tell my computer to PRINT RND(1), maybe it will predict which carrier (out of the one that is participating) that I will be assigned to...

SUPER UTILITY+ 3.2 AND THE MODEL 4
Information supplied by Herbert L. Smith

Herbert L. Smith of the Tandy Hobart Users' Group (located in Hobart, Tasmania, Australia) passed along some correspondence he had recently had with the people at Powersoft. Apparently he had sent his Super Utility Plus version 3.2 disk in for upgrading, and received back revision number 33. He then discovered he had problem, which he described as follows (quoting from Herbert'. letter to Powersoft):

"I have found that the program 'hangs-up' during special backup; it makes no difference whether the source disk is in drive 0 or 1 of my TRS-80 Model 4 computer (in slow speed or in the Model III mode). After answering 'N' to the query do you wish to 'view the formatted buffers?', the correct drive is selected, the formatting message for track 00 appears on the screen, but the drive times out. No tracks on the destination disk are formatted, but the graphics characters are still 'alive'. The program responds to 'BREAK' or 'SHIFT-BREAK' to return to the BACKUP and MAIN menus, respectively.

"If 'Y' is answered to the query of viewing the formatted buffers, the program does not correctly write to disk when the spacebar is pressed. The destination drive is selected, followed by the source drive which then is on continuously without apparently doing anything (until I give up and press 'BREAK', that is). No formatting message appears on the screen during this time. Thus I am unable to perform a special backup where it is needed..... My backup copy of SU+ 3.2 (revision 16) is able to satisfactorily special backup those disks that revision 33 is hanging up on."

Powersoft's reply:

"These patches were tested as early as January, and this is the first we heard of this. We were able to make it happen, but not in high speed mode. The change is at memory location 54B4H. Change the 1C to 16. We found the problem at any number higher than 17. This was the change to allow SDEN to work on newer Mod 4's, so we must test this value with others. Thank you - let us know how it works.

"After you verify that it works, you might try setting SPEED=Y."

If you're having the same problem, here's how to make the patch: Go to ZAP utilities and select the DISPLAY DISK SECTORS option. Then use the specification 0TS,0,8 to access the patch sector. Type M for modify, then use the arrow keys to get to byte 19H which should presently contain a value of 1CH (just as a check, the surrounding bytes are: 03 B4 54 1C 01 05 81). Change this value to 16, press ENTER, save it and reboot. Hopefully, this should fix the problem!

GHOST MENU

by Phil Holden

What is a Ghost Menu, you may ask? It's a short machine language program-selection menu that loads and executes from protected memory. And the protected memory is the safest of them all...it's video memory!

The program is especially handy when you're working with assembler or compiled programs. It lets you flip back and forth easily between monitors, disassemblers, text editors, and various application programs without fear of overwriting some of your code. The menu program does not corrupt user memory at all.

The menu program is simple...nothing fancy. But it's fast, idiot-proof, and easy to adapt to your own filenames. Moreover, it will run on either a Model I or III, and with any DOS [however, on the Model I, a lowercase modification must be installed; otherwise only seven bits are available in video memory and you need a full eight bits for executable machine code! -editor].

Type in the code of the assembly language listing, assemble it under the filename of GOSTMENU/CMD (or whatever you like), execute it, and almost instantly you see your screen change - and you immediately notice three things about the display.

First of all, you notice the video screen is flickering rather wildly. It's not something you'd want to stare at all day, but for our purposes it's quite legible. Then you see a menu to select one of ten programs with keys 0 thru 9. Finally, you notice some garbage at the bottom of the screen. That "garbage" is the menu program. That's where it's executing from.

The flicker is an interesting phenomenon. Normally the Z80 and video driver chain do not compete for the same memory, but here the program is executing from the same memory that is used to refresh the CRT. This causes bus arbitration delays between the video chain and the Program Counter. Hence, a flickering screen.

Here's a few helpful hints about using GOSTMENU/ASM.

(1) You may need to use a different syntax on some of the assembly language opcodes, depending on your assembler. I used ZEN/CMD. Watch particularly for the colon after label and multiple DEFBs on the same line. See your editor/assembler manual for syntax.

(2) When plugging-in the filenames, you can use anything the syntax of your DOS permits, with the following constraints: Each filename string must end with a 0DH byte. Leading blanks (spaces) are a no-no, trailing blanks are okay.

And DO NOT add so much space to the filenames that the program won't fit within the video memory. After any modification, you should first do a "dummy" assembly to ensure that the last byte of the program will be absolutely no higher than 3FFFH. Otherwise you'll overwrite a critical DOS area with guaranteed catastrophic results.

(3) Once you have a CMD file of the menu on disk, you don't have to re-assemble it to make minor changes or additions to filenames. Just use Superzap (or equivalent) to change them directly in the CMD disk file. I've structured the layout of the nine user filenames to permit 7 with full XXXXXXXX/YYY capability, and 2 for even longer names (if your DOS permits it). As you can see from the PROG8 name, entries can be all blanks. You'll also notice that key "0" is dedicated to DOS entry via the 402DH vector.

(4) If you need more than nine program entries, you can chain menus together by having PROG9 call up MENU2, etc.

Now, about the listing. The code in the program is relatively simple, and the comments should help you wade thru it, if you care to. But the overall structure needs an explanation.

My intent was to leave as much room for filenames as possible. To this end, I put some of the initialization code up in the 3D00H area that subsequently gets overwritten by the menu display. Hence, the two ORGs. The MAIN code at ORG 3F00H does not get destroyed by the video print routine. It's the "garbage" you see on the screen.

Having "integrated software" on MS-DOS machines may be great, but the next best thing is having only to hit reset (if you've AUTOed the menu) and one key to flip back and forth between programs.

Hope you find it useful. I have.

```
                    ;GOSTMENU/LST:1   12/04/84
                    ;Executes from video memory.
                            ORG  3D00H
3D00 F3             INIT:   DI                ;All INIT code gets
3D01 DD21483F               LD   IX,TABLE     ; overwritten.
3D05 215A3F                 LD   HL,PROG1
3D08 DD7500                 LD   (IX+0),L     ; Create TABLE of
3D0B DD7401                 LD   (IX+1),H     ; filename vectors.
3D0E 21673F                 LD   HL,PROG2
3D11 DD7502                 LD   (IX+2),L
3D14 DD7403                 LD   (IX+3),H
3D17 21743F                 LD   HL,PROG3
3D1A DD7504                 LD   (IX+4),L
3D1D DD7405                 LD   (IX+5),H
3D20 21813F                 LD   HL,PROG4
3D23 DD7506                 LD   (IX+6),L
3D26 DD7407                 LD   (IX+7),H
3D29 218E3F                 LD   HL,PROG5
3D2C DD7508                 LD   (IX+8),L
3D2F DD7409                 LD   (IX+9),H
3D32 219B3F                 LD   HL,PROG6
3D35 DD750A                 LD   (IX+10),L
3D38 DD740B                 LD   (IX+11),H
3D3B 21A83F                 LD   HL,PROG7
3D3E DD750C                 LD   (IX+12),L
3D41 DD740D                 LD   (IX+13),H
3D44 21B53F                 LD   HL,PROG8
3D47 DD750E                 LD   (IX+14),L
3D4A DD740F                 LD   (IX+15),H
3D4D 21D53F                 LD   HL,PROG9
3D50 DD7510                 LD   (IX+16),L
3D53 DD7411                 LD   (IX+17),H
                    ; Start displaying
3D56 21653D                 LD   HL,MENU
3D59 CD2C3F                 CALL LOOP1        ; In display
3D5C 21713D                 LD   HL,DOS
3D5F CD1F3F                 CALL DISP
3D62 C3003F                 JP   MAIN         ; Go to protected mem here
3D65 1C1ECD2A MENU:         DEFM 1CH,1EH,205,'*Menu*',0DH,1DH,00H
3D69 4D656E75
3D6D 2A0D1D00
3D71 444F5300 DOS:          DEFM 'DOS',0DH,1DH,00H
3D75 1D00
                    ;
                            ORG  3F00H        ; This is in
3F00 215A3F         MAIN:   LD   HL,PROG1     ;"protected"
3F03 CD213F                 CALL NUMB         ;  memory.
3F06 CD2B00         GETKEY: CALL 2BH          ; ROM "Inkey$".
3F09 0609                   LD   B,9          ; For 9 keys.
3F0B 0E31                   LD   C,'1'        ; First key.
3F0D 21483F                 LD   HL,TABLE     ; Filename vectors.
3F10 FE30                   CP   '0'          ; Is it "0" key?
3F12 CA2D40                 JP   Z,402DH      ; If so, exit to DOS.
3F15 B9             LOOP2:  CP   C            ; First key?
3F16 2823                   JR   Z,DOIT       ; If so, do it.
3F18 23                     INC  HL           ; Point to...
3F19 23                     INC  HL           ; next vector
3F1A 0C                     INC  C            ; Test next key.
3F1B 10F8                   DJNZ LOOP2        ; Check 9 keys.
3F1D 18E7                   JR   GETKEY       ; Keep on till
                                              ; valid key
                    ;
3F1F 0600           DISP:   LD   B,0          ;Display and
3F21 78             NUMB:   LD   A,B          ; number
3F22 C630                   ADD  A,30H        ; filenames
3F24 CD3300                 CALL 33H
3F27 3E2D                   LD   A,'-'
3F29 CD3300                 CALL 33H
3F2C 7E             LOOP1:  LD   A,(HL)
3F2D CD3300                 CALL 33H
3F30 FE00                   CP   00H          ; Print till
3F32 C8                     RET  Z            ; null byte
3F33 23                     INC  HL
3F34 FE0D                   CP   0DH
3F36 20F4                   JR   NZ,LOOP1
3F38 04                     INC  B
3F39 18E6                   JR   NUMB
                    ;
3F3B 5E             DOIT:   LD   E,(HL)       ; Decode vector
3F3C 23                     INC  HL           ; table
3F3D 56                     LD   D,(HL)
3F3E EB                     EX   DE,HL
3F3F 7E                     LD   A,(HL)       ; What is 1st char?
3F40 FE20                   CP   ' '          ; Leading blanks invalid
3F42 28C2                   JR   Z,GETKEY
3F44 FB                     EI
3F45 C30544                 JP   4405H        ; DOS Cmd execution
                    ;
```

```
86                TABLE:    DEFS 18         ; 9 addresses
87                ;
88 3F5A 44495220 PROG1:     DEFM 'DIR :1        ',0DH
88 3F5E 3A312020
88 3F62 20202020
88 3F66 0D
89 3F67 45445820 PROG2:     DEFM 'EDX           ',0DH
89 3F6B 20202020
89 3F6F 20202020
89 3F73 0D
90 3F74 53544552 PROG3:     DEFM 'STERM12       ',0DH
90 3F78 4D313220
90 3F7C 20202020
90 3F80 0D
91 3F81 53544552 PROG4:     DEFM 'STERM3        ',0DH
91 3F85 4D332020
91 3F89 20202020
91 3F8D 0D
92 3F8E 4D4F4445 PROG5:     DEFM 'MODEM T.F     ',0DH
92 3F92 4D20542E
92 3F96 46202020
92 3F9A 0D
93 3F9B 56494454 PROG6:     DEFM 'VIDTEX12 T    ',0DH
93 3F9F 45583132
93 3FA3 20542020
93 3FA7 0D
94 3FA8 56494454 PROG7:     DEFM 'VIDTEX3 T     ',0DH
94 3FAC 45583320
94 3FB0 54202020
94 3FB4 0D
95 3FB5 20202020 PROG8:     DEFM '                                  ',0DH
95 3FB9 20202020
95 3FBD 20202020
95 3FC1 20202020
95 3FC5 20202020
95 3FC9 20202020
95 3FCD 20202020
95 3FD1 2020200D
96 3FD5 42415349 PROG9:     DEFM 'BASIC,3,65000,RUN"DIAL/BAS:0   ',0DH
96 3FD9 432C332C
96 3FDD 36353030
96 3FE1 302C5255
96 3FE5 4E224449
96 3FE9 414C2F42
96 3FED 41533A30
96 3FF1 2020200D
97 3FF5 1DEB1B4B QUIT:      DEFM 1DH,235,1BH,'Key?',00H
97 3FF9 65793F00
98                ;
99                          EXEC 3D00H
100                         END
Exec Addr 3D00
```

KARAKTERSET (CHARACTER SET) IN EPROM

by Ruud Broers, Joop Jansen and Timo Lampe

Translated from Dutch to English by Paul Fransen

[This program is reprinted from REMARKS, the publication of the TRS-80 Gebruikers Vereniging (TRS-80 Users Group) in The Netherlands. IMPORTANT NOTE: This article has been reprinted for the benefit of the hardware hackers among us. We do NOT guarantee that it is completely accurate, especially since it has been translated from one language to another, and the accuracy of the actual translation was beyond our control (I'm sure that Paul did an excellent job, but when you're dealing with technical subjects, errors are bound to creep in somewhere). So, don't attempt this modification unless you have some hardware experience and can repair any problems that might arise. WE ASSUME NO LIABILITY IN THE EVENT THAT YOU TRY THIS MODIFICATION AND DAMAGE YOUR COMPUTER!!!!! This mod should be considered an experimenter's circuit, and not as a guaranteed fully working circuit.

Also, please note that you will have to program your own EPROM, and you will have to determine how to that from the rather limited instructions given. That should not be terribly difficult, just time consuming. I believe that the authors will sell both a printed dump of the EPROM contents and a pre-programmed EPROM, however, I do not know what the cost would be to ship these items to North America. Should you wish to contact the authors, you could write to them c/o TRS-80 Gebruikers Vereniging, Postbus 551, 2070 AN Santpoort-Noord, HOLLAND. If you are able to read Dutch and would like a copy of the original article, please send a self-addressed stamped envelope and 50 cents to cover photocopying to NORTHERN BYTES.]

The TRS-80 Model I can only display the ASCII codes 32 - 191 (on the screen). The Model III can also display the codes 192 - 255. The Model I doesn't have those characters and you will get the graphic characters twice on the screen. The following program puts all ASCII codes on your screen:

```
10 CLS
20 FOR X = 0 TO 255
30 POKE 15360 + X , X
40 NEXT X
50 PRINT @ 320,"ASCII CHARACTERS 0 - 255"
60 END
```

The codes 0 - 31 will be displayed as uppercase characters on the Japanese Model I and on the American Model I without a lowercase modification. An American Model I with lowercase can have uppercase characters or control symbols (Bell, FF, CR etc.) at these locations, depending on which lowercase modification is used.

The following modification has been installed successfully in an American and in a Japanese Model I. The American Model I must have a lowercase modification, otherwise you only have 7 bits (so the maximum is 128 characters). The Japanese Model I already has lowercase.

After the modification you will be able to produce 256 characters on your screen. The Tandy character generator will be disabled and its function will be transferred to a (programmable) EPROM, in which the information and build up of the characters will be stored. Because the standard Model I (Japanese and American) doesn't get the information for the block graphics from the character generator, the modification will cover this too.

To generate characters the EPROM must not only know which character you want to see, but also on which scan line (the lines from the video display) the build up takes place. The video display is built up from 312.5 lines, from which 192 can be used for displaying information. There are 16 text lines, so there are 192/16 = 12 scan lines per text line. The character needs 8 data lines (D0-D7) and the scan line needs 4 lines (L1, L2, L4 and L8). So there are 12 lines as address lines for the EPROM. That means you need an EPROM of 2↑12 = 4K. We used an EPROM of the type 2732.

The information in the EPROM:

The first byte of the EPROM is a 0 (build up of the first scan line).

Every character has a width of 6 screen points, so there is information on the first 6 bits.

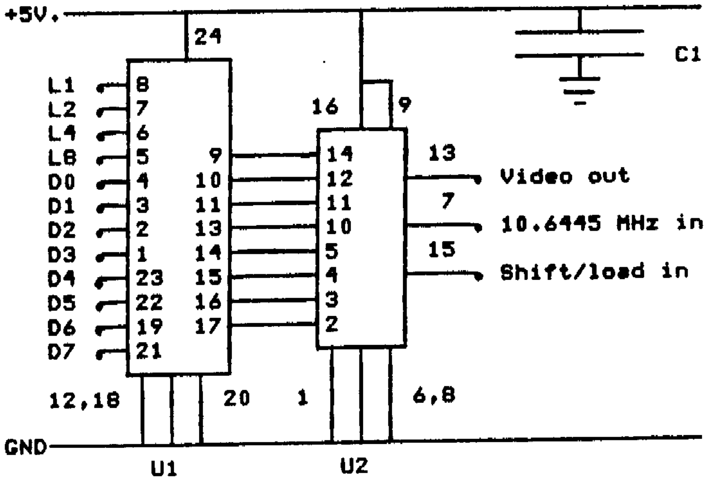
Bit 0 is the leftmost screen point. A 1 means the point is on, a 0 means the point is off.

There are 16 bytes reserved for a character (only 12 used).

The character 191 (graphic - all pixels on) means that there are 12 bytes of 63 (3FH) in the EPROM.

The information from the EPROM will be put parallel in a shift register (74LS166) and put to the video part of the TRS-80 serial. We only need two IC's to realise this.

The schematic:



U1 = 2732 EPROM
U2 = 74LS166 (a shift register)
C1 = an uncoupler capacitor of 0.1 microFarad.

In some TRS-80 you use one of the two 74LS166 from the computer, because they are in a socket.

The modification:

Disconnect the following:

AMERICAN	JAPANESE
Z30 pin 2 -- Z11 pin 13	Z54 pin 11 -- Z57 pin 13
Z30 pin 3 -- Z10 pin 13	Z54 pin 12 -- Z58 pin 13
Z26 pin 4 -- Z27 pin 3	Z55 pin 13 -- Z56 pin 2

LEGEND:
Z8 : 74LS153
Z10: 74LS166
Z11: 74LS166
Z12: 74LS93
Z26: 74LS175
Z27: 74LS20
Z30: 74LS02

Z39: 74LS153
Z50: 74LS04
Z54: 74LS02
Z55: 74LS20
Z56: 74LS175
Z57: 74LS166
Z58: 74LS166

Connect on the TRS-80 printed circuit board:

AMERICAN: Z26 pin 4 with Z26 pin 14 (+5V)
JAPANESE: Z55 pin 13 with Z55 pin 14 (+5V)

The connection of the extensions with the TRS-80 board can be done the easiest with a ribbon cable of at least 17 conductors, so the new character generator can be placed in a socket of the Model I.

The ribbon cable should connect the following:

from EXTENSION	to AMERICAN	to JAPANESE
U1 pin 24, U2 pins 9 & 16	Z8 pin 16	Z37 pin 18
U1 pin 8 (L1)	Z12 pin 12	Z37 pin 11
U1 pin 7 (L2)	Z12 pin 9	Z37 pin 10
U1 pin 6 (L4)	Z8 pin 14	Z37 pin 8
U1 pin 5 (L8)	Z8 pin 2	Z39 pin 2
U1 pin 4 (D0)	Z8 pin 6	Z37 pin 7
U1 pin 3 (D1)	Z8 pin 10	Z37 pin 6
U1 pin 2 (D2)	Z8 pin 5	Z37 pin 5
U1 pin 1 (D3)	Z8 pin 11	Z37 pin 4
U1 pin 23 (D4)	Z8 pin 4	Z37 pin 3
U1 pin 22 (D5)	Z8 pin 12	Z37 pin 2
U1 pin 19 (D6)	Z27 pin 15	Z56 pin 15
U1 pin 21 (D7)	Z27 pin 3	Z56 pin 3
U1 pin 12,18,20 (ground)	Z8 pin 8	Z37 pin 9
U2 pin 1,6,8 (ground)	Z8 pin 8	Z37 pin 9
U2 pin 13 (video out)	Z30 pins 2 & 3	Z54 pins 11 & 12
U2 pin 7 (10.6445 MHz)	Z11 pin 7 or Z9 pin 8	Z50 pin 12
U2 pin 15 (shift/load)	Z26 pin 6	Z55 pin 8

When you boot up your system without an active lowercase driver, you will see a lot of strange characters on your screen, which tells you: MEMORY SIZE? or MEM SIZE?. The ROM sends in place of characters the codes 1-31 to the screen, which are strange characters now. Loading a lowercase driver will help you to get rid of these characters. To see if the modification works, you can type:

```
10 FOR X = 0 TO 255 : POKE 15360 + X , X : NEXT
RUN
```

The screen should show the following:
Line 1: Special graphics, special characters and the numbers
Line 2: The alphabet in uppercase and in lowercase
Line 3: The Tandy graphics
Line 4: Special characters, numbers and uppercase alphabet reversed.

If you do not have those characters then switch off your system. If all is well except the first line (there are uppercase characters in place), then you will probably have a lowercase modification installed, that manipulates bit 6 in such a way that without a lowercase driver installed, you will have the normal uppercase characters. To solve this problem you can do one of two things:
- connect the exit of the extra 2102 (pin 12) directly with Z27 pin 13.
- the same but with a switch in between.

If this is not the case, then check the connections and disconnections you have made. If you only miss some points of the Tandy graphics it could be a too slow 74LS166.

You can use the modification in BASIC. Calculate the position of the cursor by typing:

```
P = PEEK(16416)+256*PEEK(16417)
```

POKE the character you like at that place and move the cursor. That's a hard way to get the characters on the screen, so here are two video drivers. They are meant as a supplement of an existing driver.

```
100 REM.THIS.LINE.IS.35.CHARACTERS.LONG...
110 REM*********************************************
120 REM*.......SPECIAL VIDEO DRIVER 1 (DISK)..........*
130 REM*..................BY..........................*
140 REM*..............RUUD BROERS......................*
150 REM*********************************************
160 REM* DON'T CHANGE LINE 100. IT SHOULD BE 35.......*
170 REM* CHARACTERS LONG !!!!!!!!!!!!!!!!!!!!!!!........*
180 REM*********************************************
250 BP = 16548 : REM 40A4H IS BASIC POINTER
260 LS = PEEK(BP) : MS = PEEK(BP+1)
270 BS = LS + 256 * MS : REM BS = START OF PROGRAM
280 IF BS > 32767 THEN BS = BS - 65536
290 L2 = PEEK(BS) : H2 = PEEK(BS+1)
300 POKE BP , L2 : POKE BP + 1 , H2 : REM L2 AND H2 = LINE 2
310 FOR X = 0 TO 34 : READ D : POKE BS+X, D : NEXT
320 POKE BS+X, PEEK(16414) : POKE BS+X+1, PEEK(16415)
330 POKE 16414, BS AND 255 : POKE16415, BS/256
340 DATA 33, 61, 64, 203, 126, 40, 18, 126
350 DATA 230, 95, 119, 221, 110, 3, 221, 102
360 DATA 4, 221, 126, 5, 119, 121, 195, 125
370 DATA 4, 121, 183, 32, 5, 126, 246, 128
380 DATA 110, 201, 195
400 CLS
410 PRINT"ALL POSSIBLE CHARACTERS:":PRINT
420 FOR X = 0 TO 255 : PRINT CHR$(0); CHR$(X); : NEXT
430 PRINT
440 PRINT"WITH CHR$(0); CHR$(X); CHARACTER 'X' WILL BE SEND
TO THE VIDEO"
```

The second driver is a stand alone driver with a simple lowercase driver (specially for cassette users).

```
100 REM.THIS.LINE.CONTAINS.68,CHARACTERS.IN.TOTAL.4.6.8.0.2
.4.6.8.0.2.4.6.8
110 REM*********************************************
120 REM*.......SPECIAL VIDEO DRIVER 2 (CASSETTE)....*
130 REM*.......................BY....................*
140 REM*...................RUUD BROERS..............*
150 REM*********************************************
160 REM* THE FIRST LINE (100) MUST CONTAIN 68.......*
170 REM* CHARACTERS.................................*
180 REM*********************************************
250 BP = 16548 : REM 40A4H IS BASIC POINTER
260 LS = PEEK(BP) : MS = PEEK(BP+1)
270 BS = LS + 256 * MS : REM BS = START OF PROGRAM
280 IF BS > 32767 THEN BS = BS - 65536
290 L2 = PEEK(BS) : H2 = PEEK(BS+1)
300 POKE BP , L2 : POKE BP + 1 , H2 : REM L2 AND H2 = LINE 2
310 FOR X = 0 TO 67 : READ D : POKE BS+X, D : NEXT
320 POKE 16414 , BS AND 255 : POKE 16415 , BS/256
330 DATA 245, 209
340 DATA 33, 61, 64, 203, 126, 40, 18, 126
350 DATA 230, 95, 119, 221, 110, 3, 221, 102
360 DATA 4, 221, 126, 5, 119, 121, 195, 125
370 DATA 4, 121, 183, 32, 5, 126, 246, 128
380 DATA 119, 201, 221, 110, 3, 221, 102, 4
390 DATA 213, 241, 218, 154, 4, 221, 126, 5
395 DATA 183, 40, 1, 119, 121, 254, 32, 218
396 DATA 6, 5, 254, 128, 210, 166, 4, 195
397 DATA 125, 4
400 CLS
410 PRINT"ALL POSSIBLE CHARACTERS:":PRINT
420 FOR X = 0 TO 255 : PRINT CHR$(0); CHR$(X); : NEXT
430 PRINT
440 PRINT"WITH CHR$(0); CHR$(X); CHARACTER 'X' WILL BE SEND
TO THE VIDEO"250
```

The way both the drivers protect and activate themselves is identical. Both programs put the driver on the first line and then move the BASIC start-of-program pointer. The pointer then points at the second line. BASIC can't access the first line anymore.

You can't use this method with DOS or EDTASM, because the driver will be overwritten. So, here are the machine language codes, so you can put them where you wish. [Editor's note: These sample programs are ORGed at 3000H, but obviously, you can't actually put them there unless you have added some memory at that location, since a stock Model I has no memory there at all!]

```
                00100 ;=============================;
                00110 ;.SPECIAL VIDEO DRIVER 1 (DISK).;
                00120 ;.......25 SEPTEMBER 1983.......;
                00130 ;..........R.C.H.BROERS.........;
                00140 ;=============================;
4030            00150 CONBYT  EQU     403DH           ;VIDEO CONTROL BYTE
3000            00160         ORG     3000H
3000 213D40     00170 START   LD      HL,CONBYT       ;GET VIDEO CONTROL BYTE
3003 CB7E       00180         BIT     7,(HL)          ;TEST BIT SPECIAL
3005 2812       00190         JR      Z,NOSP          ;NO, JP TO NO-SPECIAL
3007 7E         00200 SPEC    LD      A,(HL)          ;SPECIAL PROCESSING
3008 E65F       00210         AND     5FH             ;RESET BIT 7
300A 77         00220         LD      (HL),A          ;PUT IT BACK
300B DD6E03     00230         LD      L,(IX+3)        ;READY FOR JUMP TO ROM
300E DD6604     00240         LD      H,(IX+4)
3011 DD7E05     00250         LD      A,(IX+5)
3014 77         00260         LD      (HL),A
3015 79         00270         LD      A,C
3016 C37D04     00280         JP      047DH           ;TO ROM
                00290 ;-----------------------------------------;
3019 79         00300 NOSP    LD      A,C             ;NO SPECIAL
301A B7         00310         OR      A               ;TEST FOR CODE (0)
301B 2005       00320         JR      NZ,NOST         ;NO, JP TO NO-CODE
301D 7E         00330         LD      A,(HL)
301E F680       00340         OR      80H             ;PUT BIT 7 BACK
3020 77         00350         LD      (HL),A
3021 C9         00360         RET
3022 C30000     00370 NOST    JP      0000            ;GO NORMALLY
3023            00380 PATCH   EQU     $-2             ;DRIVER JUMP
                00390 ;-----------------------------------------;
3025 2A1E40     00400 SETUP   LD      HL,(401EH)      ;VIDEO DRIVER ADRESS
3028 222330     00410         LD      (PATCH),HL      ;CONNECT WITH ROUTINE
302B 210030     00420         LD      HL,START
302E 221E40     00430         LD      (401EH),HL
3031 C32D40     00440         JP      402DH           ;BACK TO DOS
                00450 ;       JP      06CCH           ;BACK TO LEVEL 2
3025            00460         END     SETUP
00000 TOTAL ERRORS

CONRYT  403D    NOSP    3019    NOST    3022    PATCH   3023    SETUP   3025
SPEC    3007    START   3000
```

The driver with a lowercase driver. By pressing SHIFT you will get lowercase characters.

```
                00100 ;=============================;
                00110 ;..SPECIAL VIDEO DRIVER 2 (CASS)..;
                00120 ;.........WITH LOWER CASE.........;
                00130 ;........25 SEPTEMBER 1983........;
                00140 ;...........R.C.H.BROERS..........;
                00150 ;=============================;
4030            00160 CONBYT  EQU     403DH           ;VIDEO CONTROL BYTE
3000            00170         ORG     3000H
3000 F5         00180 START   PUSH    AF
3001 D1         00190         POP     DE
3002 213D40     00200         LD      HL,CONBYT       ;GET VIDEO CONTROL BYTE
3005 CB7E       00210         BIT     7,(HL)          ;TEST BIT SPECIAL
3007 2812       00220         JR      Z,NOSP          ;NO, JP TO NO-SPECIAL
3009 7E         00230 SPEC    LD      A,(HL)          ;SPECIAL PROCESSING
300A E65F       00240         AND     5FH             ;RESET BIT 7
300C 77         00250         LD      (HL),A          ;PUT IT BACK
300D DD6E03     00260         LD      L,(IX+3)        ;READY FOR JUMP TO ROM
3010 DD6604     00270         LD      H,(IX+4)
3013 DD7E05     00280         LD      A,(IX+5)
3016 77         00290         LD      (HL),A
3017 79         00300         LD      A,C
3018 C37D04     00310         JP      047DH           ;TO ROM
                00320 ;-----------------------------------------;
301B 79         00330 NOSP    LD      A,C             ;NO SPECIAL
301C B7         00340         OR      A               ;TEST FOR CODE (0)
301D 2005       00350         JR      NZ,NOST         ;NO, JP TO NO-CODE
301F 7E         00360         LD      A,(HL)
3020 F680       00370         OR      80H             ;PUT BIT 7 BACK
3022 77         00380         LD      (HL),A
3023 C9         00390         RET
3024 DD6E03     00400 NOST    LD      L,(IX+3)        ;LOWERCASE DRIVER
3027 DD6604     00410         LD      H,(IX+4)
302A D5         00420         PUSH    DE
302B F1         00430         POP     AF
302C DA9A04     00440         JP      C,049AH
302F DD7E05     00450         LD      A,(IX+5)        ;TEST CURSOR ON
3032 B7         00460         OR      A
3033 2801       00470         JR      Z,SKIP0
3035 77         00480         LD      (HL),A  ;RESTORE CHARACTER UNDER CURSOR
3036 79         00490 SKIP0   LD      A,C
3037 FE20       00500         CP      20H             ;TEST FUNCTION CODE
3039 DA0605     00510         JP      C,0506H
303C FE80       00520         CP      80H             ;DELETE IF...
303E D2A604     00530         JP      NC,04A6H        ;SPACE COMPRESS
3041 C37D04     00540         JP      047DH
                00550 ;
3044 210030     00560 SETUP   LD      HL,START
3047 221E40     00570         LD      (401EH),HL
                00580 ;       JP      402DH           ;BACK TO DOS
304A C3CC06     00590         JP      06CCH           ;BACK TO LEVEL 2
3044            00600         END     SETUP
00000 TOTAL ERRORS

CONBYT  403D    NOSP    301B    NOST    3024    SETUP   3044    SKIP0   3036
SPEC    3009    START   3000
```

If one the drivers is active you will see normal readable characters on your screen. You will notice the special characters if you print something preceded with the control code (CHR$(0)). For every special character you wish on the screen you have to repeat the control code:

```
PRINT CHR$(0);CHR$(5);CHR$(0);CHR$(6)
```

The normal use of CHR$(0) can be compared with CHR$(27) ("ESC") on some printers.

And now a simple program to reverse normal text:

```
100 REM NORMAL TO INVERSE
130 CLEAR 400
140 REM INPUT QUESTION
150 INPUT "ENTER A TEXT STRING"; N$
160 REM A SPACE BEFORE AND AFTER
170 N$ = " " + N$ +" "
180 REM CONVERT IT
190 GOSUB 500
200 REM DISPLAY IT
210 PRINT N$ : PRINT NI$
220 REM AGAIN
230 GOTO 150
500 REM GET LENGTH AND INIT NI$
510 L = LEN(N$) : NI$ = ""
520 REM THROUGH THE WHOLE STRING
530 FOR X = 1 TO L
540  REM WHAT'S THE ASCII CODE ?
550  C = ASC( MID$( N$, X, 1))
560  REM IS IT LOWERCASE ?
570  IF C > 95 AND C < 128 THEN C = C AND 223
580  REM LOWERCASE TO UPPERCASE
590  NI$ = NI$ + CHR$(0) + CHR$(C+160)
600  REM NORMAL TO INVERSE
610 NEXT X
620 REM READY
630 RETURN
```

With this modification the space compression codes won't work anymore [this is a function of the video driver, not the hardware, so they should work when the normal DOS video driver is in use -editor]. The underline option of ZORLOF (CLEAR U) will produce inverse characters. The JKL option will, on some printers, have strange effects.

MAAK EENS EEN DOOLHOF (MAKE A MAZE)

by Richard A. Keijzer

Translated from Dutch to English by Paul Fransen

[This program is reprinted from REMARKS, the publication of the TRS-80 Gebruikers Vereniging (TRS-80 Users Group) in The Netherlands. PLEASE NOTE: In the BASIC program listings, there are a few places where the number of spaces to be inserted is critical. At those points we have used the symbol "‸" to represent a space. So, wherever you see a "‸", type a space instead.]

Now and then you will find in computer magazines a program that will make mazes for you. Here is another one. You have to input the width and length of the maze.

```
10 REM ******************************
20 REM * A maze generator in Basic *
30 REM * by: Richard A. Keijzer    *
40 REM * 16 jan 1984              *
50 REM ******************************
100 CLS: K=1
200 DEFINTA-Z:DIMB(4),C(4),D(4)
210 B(1)=-1:B(2)=0:B(3)=1:B(4)=0
220 C(1)=0:C(2)=1:C(3)=0:C(4)=-1
230 D(1)=2:D(2)=4:D(3)=8:D(4)=16
240 INPUT"Dimensions (width/length)";Y,X
250 CLS:DIMA(X+1,Y+1)
260 FORN=0TOX+1:A(N,0)=42:A(N,Y+1)=42:NEXT
270 FORN=0TOY+1:A(0,N)=52:A(X+1,N)=52:NEXT
280 A(X+1,Y+1)=63
290 I=1:J=RND(Y):A(I,J)=2
300 C=RND(4):T=C
310 P=I:Q=J
320 I=I+B(C):J=J+C(C)
330 IFA(I,J)=0THEN370
340 I=P:J=Q
350 C=C+1:IFC>4THENC=C-4
360 IFC=TTHEN430ELSE320
370 A(P,Q)=A(P,Q)ORD(C)
380 SET(Q*2,P):SET(Q*2+1,P)
390 C=C+2:C=C+4*(C>4)
400 A(I,J)=A(I,J)ORD(C)
410 SET(J*2,I):SET(J*2+1,I)
420 GOTO300
430 REM **** stuck ****
440 U=0
450 FORN=KTOX:FORM=1TOY:IFA(N,M)<>0THEN530
460 FORZ=1TO4
470 I=N:J=M:P=I:Q=J
480 I=I+B(Z):J=J+C(Z)
490 IFA(I,J)>0ANDA(I,J)<40THENC=Z:Z=14ELSEI=P:J=Q
500 NEXT
510 IFZ<10THEN530
520 U=1:K=N:N=X:M=Y
530 NEXTM,N
540 IFU=0THEN600
550 A(P,Q)=A(P,Q)ORD(C)
560 C=C+2:C=C+4*(C>4)
570 A(I,J)=A(I,J)ORD(C)
580 I=P:J=Q
590 GOTO310
600 REM **** open bottom ****
610 J=RND(Y):A(X+1,J)=A(X+1,J)OR2
620 REM **** print out ****
630 FORN=1TOX+1:FORT=1TO2:FORM=1TOY+1
640 IFT=2THEN670
650 IFA(N,M)AND2THENLPRINT"+‸‸";ELSELPRINT"+--";
660 GOTO680
670 IFA(N,M)AND16THENLPRINT"‸‸‸";ELSELPRINT"L‸‸";
680 NEXTM:LPRINT:NEXTT,N
```

You can make some nice mazes with this program. But for some people it is too slow. So we will use a machine language routine. It's called as a USR routine from BASIC.

```
B000         00100        ORG    0B000H
C000         00110 HOFBEG EQU    0C000H          ;START OF MAZE
B000 CD7F0A  00120        CALL   0A7FH           ;WIDTH & LENGTH IN HL
B003 E5      00130        PUSH   HL
B004 210000  00140        LD     HL,0000H        ;ZERO TO
B007 22FFB0  00150        LD     (RNDLOK),HL     ;RANDOM POINTER
B00A D1      00160        POP    DE              ;HOR/VER IN DE
B00B 2100C0  00170        LD     HL,HOFBEG       ;START CODE
B00E 7A      00180        LD     A,D             ;HOR. DIMENSION
B00F F5      00190        PUSH   AF              ;SAVE IT
B010 C602    00200        ADD    A,2             ;PLUS 211
B012 47      00210        LD     B,A             ;COUNTER
B013 3640    00220 VUL    LD     (HL),40H        ;TOPSIDE
B015 23      00230        INC    HL              ;NEXT BYTE
B016 10FB    00240        DJNZ   VUL             ;UNTIL READY
B018 23      00250        INC    HL              ;1ST LOCATION
B019 E5      00260        PUSH   HL              ;TO STACK
B01A DDE1    00270        POP    IX              ;IX=1ST LOCATION
B01C 2B      00280        DEC    HL              ;CORRECT IT
B01D F1      00290        POP    AF              ;HOR.DIM (ORIG)
B01E 43      00300        LD     B,E             ;VER.DIM IN B
B01F C5      00310 INVUL  PUSH   BC              ;SAVE THEM
B020 F5      00320        PUSH   AF              ;TO STACK
B021 C1      00330        POP    BC              ;HOR.DIM IN B
             00340 ;TOP OF STACK IS NOW VER.DIM
B022 3640    00350        LD     (HL),40H        ;1ST BYTE (LEFT ONE)
B024 23      00360        INC    HL
B025 3600    00370 VUL2   LD     (HL),00H        ;FILL WITH ZERO
B027 23      00380        INC    HL              ;NEXT BYTE
B028 10FB    00390        DJNZ   VUL2
B02A 3640    00400        LD     (HL),40H        ;RIGHT ONE
B02C 23      00410        INC    HL              ;NEXT BYTE
B02D C1      00420        POP    BC              ;VER.COUNTER IN B
B02E 10EF    00430        DJNZ   INVUL           ;REPEAT IT
             00440 ;NOW THE BOTTOM HALF
B030 C602    00450        ADD    A,2             ;PLUS 2
B032 47      00460        LD     B,A             ;MAKE IT THE COUNTER
B033 3640    00470 VUL3   LD     (HL),40H
B035 23      00480        INC    HL              ;PLUS 1
B036 10FB    00490        DJNZ   VUL3
B038 3600    00500        LD     (HL),00H        ;FILL WITH ZERO
B03A 23      00510        INC    HL
B03B 36FF    00520        LD     (HL),0FFH       ;END MARKER
             00530 ;THAT'S THE WHOLE MAZE
B03D ED5F    00540        LD     A,R             ;RANDOM VALUE
B03F 92      00550 AFTREK SUB    D               ;MINUS HOR.VALUE
B040 30FD    00560        JR     NC,AFTREK       ;POSITIVE, THEN AGAIN
B042 82      00570        ADD    A,D             ;ADD IT AGAIN
             00580 ;THAT IS THE HOR.RANDOM VALUE
B043 DDE5    00590        PUSH   IX              ;1ST LOCATION MAZE
B045 E1      00600        POP    HL              ;PUT IT IN HL
B046 4F      00610        LD     C,A             ;BC=00'AA'
B047 09      00620        ADD    HL,BC           ;FIRST BOX
B048 3601    00630        LD     (HL),01H        ;OPEN UPSIDE
B04A 7A      00640        LD     A,D             ;HOR.VALUE
B04B 3C      00650        INC    A               ;PLUS 1
B04C 3C      00660        INC    A               ;PLUS 1
B04D 4F      00670        LD     C,A             ;C=HOR.VALUE + 2
B04E C5      00680        PUSH   BC              ;SAVE IT
B04F D1      00690        POP    DE              ;IN DE
             00700 ;DE NOW CONTAINS THE VER.MOVEMENT THROUGH THE MAZE
B050 ED5F    00710 RANDOM LD     A,R             ;RANDOM VALUE
B052 E5      00720        PUSH   HL              ;SAVE...
B053 D5      00730        PUSH   DE              ;REGISTERS
B054 2AFFB0  00740        LD     HL,(RNDLOK)     ;GET RANDOM POINTER
B057 1600    00750        LD     D,0
B059 5F      00760        LD     E,A
B05A 19      00770        ADD    HL,DE           ;ADD RND VALUE
B05B 7E      00780        LD     A,(HL)          ;GET THE BYTE
B05C 22FFB0  00790        LD     (RNDLOK),HL     ;NEW POINTER
B05F D1      00800        POP    DE              ;GET THE
B060 E1      00810        POP    HL              ;REGISTERS
B061 E603    00820        AND    03H             ;MASK OUT
B063 E5      00830        PUSH   HL              ;OLD HL TO STACK
             00840 ;BY MASKING OUT WITH 'AND 03' WE NOW HAVE A VALUE
             00850 ;BETWEEN 0 AND 3 (THE 4 DIRECTIONS)
B064 0605    00860        LD     B,5             ;COUNTER
B066 1802    00870        JR     K0123
B068 78      00880 LOOP   LD     A,B             ;COUNTER IS DIRECTION
B069 3D      00890        DEC    A               ;COMPENSATE
B06A B7      00900 K0123  OR     A               ;IS IT ZERO?
B06B 4E      00910        LD     C,(HL)          ;OLD VALUE IN C
B06C 2817    00920        JR     Z,NUL
B06E FE01    00930        CP     01H             ;IS IT A ONE?
B070 2823    00940        JR     Z,EEN
B072 FE02    00950        CP     02H             ;IS IT A TWO?
B074 282E    00960        JR     Z,TWEE
             00970 ;IF IT IS NOT A 0,1 OR 2 THEN IT IS A 3
```

```
B076 CBDE       00980 DRIE    SET     3,(HL)          ;OPEN LEFT
B078 2B         00990         DEC     HL              ;ONE PLACE TO THE LEFT
B079 AF         01000         XOR     A               ;ZEROING A
B07A BE         01010         CP      (HL)            ;NEW ONE ALSO A ZERO?
B07B 2802       01020         JR      Z,DRIE2         ;YES, THEN JUMP
B07D 1834       01030         JR      FOUT
B07F FDE1       01040 DRIE2   POP     IY              ;NO NEED FOR OLD HL
B081 CBCE       01050         SET     1,(HL)          ;OPEN RIGHT
B083 18CB       01060         JR      RANDOM          ;NEXT BOX
B085 CBC6       01070 NUL     SET     0,(HL)          ;OPEN UP
B087 AF         01080         XOR     A               ;ZEROING A
B088 ED52       01090         SBC     HL,DE           ;ONE BOX UP
B08A BE         01100         CP      (HL)            ;IS IT ZERO?
B08B 2802       01110         JR      Z,NUL2          ;YES, THEN JUMP
B08D 1824       01120         JR      FOUT            ;TO END OF LOOP
B08F FDE1       01130 NUL2    POP     IY              ;NO NEED OF OLD HL
B091 CBD6       01140         SET     2,(HL)          ;OPEN BOTTOM
B093 18BB       01150         JR      RANDOM          ;NEXT BOX
B095 CBCE       01160 EEN     SET     1,(HL)          ;OPEN RIGHT
B097 23         01170         INC     HL              ;ONE BOX TO THE RIGHT
B098 AF         01180         XOR     A
B099 BE         01190         CP      (HL)
B09A 2802       01200         JR      Z,EEN2
B09C 1815       01210         JR      FOUT
B09E FDE1       01220 EEN2    POP     IY              ;NO NEED OF OLD HL
B0A0 CBDE       01230         SET     3,(HL)          ;OPEN LEFT
B0A2 18AC       01240         JR      RANDOM          ;NEXT BOX
B0A4 CBD6       01250 TWEE    SET     2,(HL)          ;OPEN BOTTOM
B0A6 19         01260         ADD     HL,DE           ;ONE BOX DOWN
B0A7 AF         01270         XOR     A
B0A8 BE         01280         CP      (HL)
B0A9 2802       01290         JR      Z,TWEE2
B0AB 1806       01300         JR      FOUT
B0AD FDE1       01310 TWEE2   POP     IY              ;NO NEED OF OLD HL
B0AF CBC6       01320         SET     0,(HL)          ;OPEN UP
B0B1 189D       01330         JR      RANDOM
                01340 ;
                01350 ;IF WE COME HERE THEN IT WAS NOT POSSIBLE TO FILL
                01360 ;A BOX. SO WE HAVE TO GO THROUGH THE LOOP AGAIN
                01370 ;
B0B3 E1         01380 FOUT    POP     HL              ;OLD BOX BA
B0B4 71         01390         LD      (HL),C          ;OLD VALUE BACK
B0B5 E5         01400         PUSH    HL              ;HL TO STACK AGAIN
B0B6 10B0       01410         DJNZ    LOOP
B0B8 E1         01420         POP     HL              ;IX TO MUCH PUSHED
                01430 ;IF WE ARE HERE THEN WE CAN'T GO ANY WAY
                01440 ;THEN WE HAVE TO DO THE FOLLOWING
B0B9 DDE5       01450 VASTGL  PUSH    IX              ;1ST LOCATION
B0BB E1         01460         POP     HL              ;TO HL
B0BC AF         01470 ANUL    XOR     A               ;ZEROING A
B0BD 0E00       01480         LD      C,0             ;BC IS ZERO NOW
B0BF EDB1       01490         CPIR                    ;FIND A ZERO
B0C1 CB7E       01500         BIT     7,(HL)          ;IS IT FF?
B0C3 2020       01510         JR      NZ,BASIC        ;YES, THEN TO BASIC
B0C5 CDF9B0     01520         CALL    TEST            ;LOOK FOR 0 OR 64
                01530 ;IF YES THEN WE MAY NOT USE THIS BOX
B0C8 380B       01540         JR      C,RECHTS
B0CA 2B         01550         DEC     HL              ;1 BOX BACK
B0CB 2B         01560         DEC     HL              ;AND AGAIN ONE BOX
B0CC CDF9B0     01570         CALL    TEST            ;LOOK AGAIN
B0CF 380C       01580         JR      C,LINKS         ;TO THE LEFT
B0D1 23         01590         INC     HL              ;NEXT BOX
B0D2 23         01600         INC     HL              ;AND NEXT AGAIN
B0D3 18E7       01610         JR      ANUL            ;EXAMINE
B0D5 CBDE       01620 RECHTS  SET     3,(HL)          ;OPEN LEFT
B0D7 2B         01630         DEC     HL              ;TO THE LEFT
B0D8 CBCE       01640         SET     1,(HL)          ;OPEN RIGHT
B0DA C350B0     01650         JP      RANDOM          ;GO AGAIN
B0DD CBCE       01660 LINKS   SET     1,(HL)          ;OPEN RIGHT
B0DF 23         01670         INC     HL              ;TO THE RIGHT
B0E0 CBDE       01680         SET     3,(HL)          ;OPEN LEFT
B0E2 C350B0     01690         JP      RANDOM          ;GO AGAIN
B0E5 AF         01700 BASIC   XOR     A               ;ZEROING A
B0E6 ED52       01710         SBC     HL,DE           ;1 ROW UP
                01720 ;HL POINTS TO 1ST BOX OF THE BOTTOM ROW
B0E8 7B         01730         LD      A,E             ;HOR.+2 IN A
B0E9 3D         01740         DEC     A               ;HOR.+1 IN A
B0EA 3D         01750         DEC     A               ;HOR. IN A
B0EB 57         01760         LD      D,A             ;HOR.VALUE TO D
B0EC ED5F       01770         LD      A,R             ;RANDOM VALUE
B0EE 92         01780 AFTR    SUB     D               ;HOR.VALUE MINUS
B0EF 30FD       01790         JR      NC,AFTR         ;TILL IT IS POSITIVE
B0F1 82         01800         ADD     A,D             ;ADD AGAIN
B0F2 0600       01810         LD      B,00H
B0F4 4F         01820         LD      C,A             ;BC='00AA
B0F5 09         01830         ADD     HL,BC
B0F6 CBC6       01840         SET     0,(HL)          ;OPEN UP
B0F8 C9         01850         RET
B0F9 7E         01860 TEST    LD      A,(HL)          ;GET VALUE
B0FA B7         01870         OR      A               ;SET THE FLAGS
B0FB C8         01880         RET     Z               ;RETURN IF ZERO
B0FC FE40       01890         CP      64              ;<64?
B0FE C9         01900         RET                     ;THEN CARRY IS 1
B0FF 0000       01910 RNDLOK  DEFW    0000H           ;RANDOM POINTER
402D            01920         END     402DH           ;"DOS READY"
00000 TOTAL ERRORS

AFTR    B0EE  AFTREK  B03F  ANUL    B0BC  BASIC   B0E5  DRIE    B076
DRIE2   B07F  EEN     B095  EEN2    B09E  FOUT    B0B3  HOFBEG  C000
INVUL   B01F  K0123   B06A  LINKS   B0DD  LOOP    B068  NUL     B085
NUL2    B08F  RANDOM  B050  RECHTS  B0D5  RNDLOK  B0FF  TEST    B0F9
TWEE    B0A4  TWEE2   B0AD  VASTGL  B0B9  VUL     B013  VUL2    B025
VUL3    B033
```

If we have assembled the program and saved it, we have to enter a Basic program like the following:

```
100 W=0
110 INPUT"Hor. and Vert.";H,V
120 S=256*H+V
130 DEFUSR1=&HB000
140 K=USR1(S)
150 A=-16384+H+3
160 FORN=0TOH-1
170 IFK=255THENSTOP
180 K=PEEK(A+N)
190 IFKAND1THENLPRINT"+^^";ELSELPRINT"+--";
200 NEXT:LPRINT"+"
210 IFW=VTHENSTOP
220 FORN=0TOH-1
230 K=PEEK(A+N)
240 IFKAND8THENLPRINT"^^^";ELSELPRINT"I^^";
250 NEXT:LPRINT"I"
260 W=W+1
270 A=A+H+2:GOTO160
```

To use the above programs, from DOS READY set HIMEM or TOPMEM to AFFFH (you may also do this be entering a MEMORY SIZE of 45055 when you go into BASIC, but it's safer to do it from DOS if possible). Then LOAD MAZE/CIM (or whatever name you have given the machine language object code). Finally, go into BASIC and RUN"MAZE/BAS". These programs are known to run under NEWDOS/80, but should run under other Disk Operating Systems as well.

OBERON READER PATCH FOR MODEM-80

If you are thinking of buying one of the new Oberon text readers (a device which is supposed to be able to read text from a printed page directly into the computer), you may want to wait a bit until the next model comes out. We tried one at The Alternate Source and unfortunately, we found that the error rate in reading text was much too high, and the reader was only capable of reading a very limited number of type styles (four, to be exact) with any success at all. Unless you have a few hundred dollars burning a hole in your pocket, the advice from here is, "wait!"

But, if you buy one anyway, you'll find that the supplied software only works with IBM compatibles. No problem, say the folks at Oberon, any standard terminal package can be used. Well, not quite. The Oberon reader actually uses the DTR (Data Terminal Ready) line from the RS-232 interface (pin 20 of the DB-25 connector). I won't bore you with all the gory details of what we went through before we figured this out (suffice it to say that I'm a bit balder now), but what we found was that during UART initialization, the value output to port EAH must have bit 0 reset (i.e. Bit 0 = 0). If you are using MODEM-80, use DEBUG to modify the byte at 9121H from 6DH to 6CH to accomplish this. If you don't do this, no amount of twiddling with the reader or MODEM-80 will permit you to read text!

MODEL I/III TO MODEL 4 SUPERSCRIPSIT PRINTER DRIVERS

by Michael R. Freifeld

[Excerpted from the TBUG-80 newsletter.]

With the development of SuperScripsit for the Model 4 there exists one minor problem to Model III customers of SuperScripsit. No, it is not getting the update to the Model 4 version. Radio Shack adequately handled that with a very fair update fee of twenty-five dollars. Rather, it is in converting printer drivers to work with the Model 4 program. The Model 4 version of SuperScripsit DOES contain all the printer drivers for the printers which they are currently selling. But what about printer drivers for those which they no longer sell? Moreover, what about all those printer drivers for printers which Radio Shack has never sold???

Fortunately, certain key memory locations have been maintained in the transition from Model III to Model 4 SuperScripsit. Of importance to us is the fact that the "protocol" for the printer drivers between the two are very much the same. Three things must be done in order to convert a Model III printer driver to work on the Model 4. This article will explain, in general, how to go about doing this. Also, I have generated two patch files (listed at the end of this article) which will convert Powersoft's POWERDRIVER-E drivers (Epson printer drivers) so they will operate on the Model 4.

Before I go on, I must say that you will need a disassembler (preferably one which disassembles to disk such as Apparat's or Misosys's [or the disassembler in TASMON -editor]), the technical portion of your SuperScripsit manual (turned to the section on printer drivers), and a little bit of experience in assembler if you wish to convert your "non-standard" Model III printer driver. Make sure that you have a hardcopied disassembly of your particular driver before going on.

The first thing you will need to do is insert the following routine which will store the location of the printer (PR) device control block in memory. This is the @GTDCB SuperVisory Call (SVC). For further information consult the Model 4 Technical Manual (or other such publication). In order to accomplish this you will need to find some (or make some) free space in your driver routine and change the JumP located at BB37H to the start of this routine. Note the location you are changing this from. This is where PRINIT begins. When through, it will then jump to the standard PRINIT routine (you know where this is, right???).

```
BB37            JP      START           ;go START then PRINIT
BB76            NOP                     ;this is where RS chose to
BB77            NOP                     ;store the PR DCB

        START   LD      DE,5250H        ;load DE with PR
                LD      A,52H           ;store SVC #
                RST     28H             ;call @GTDCB
                LD      (0BB76H),HL     ;store PR location
                CALL    RDYTST          ;check for printer ready
                JP      Z,PRINIT        ;okay, initialize printer
                JP      0BAB5H          ;huh?, goto ERROR message
```

The next thing that needs to be done is to change the RDYTST (check printer for ready status test) so that it will use the appropriate SVC which in this case is #5, @CTL. The following are a few of examples of typical RDYTST routines:

```
1)      RDYTST  LD      A,(37E8H)       ;load status
                AND     0F0H            ;strip off lower nybble
                CP      30H             ;compare for ready status
                RET

2)      RDYTST  PUSH    BC              ;save registers
                LD      B,A             ;save character
                IN      A,(0F8H)        ;get printer status
                NOP                     ;(used for Model I only)
                AND     0F0H            ;strip off lower nybble
                CP      30H             ;compare for ready status
                LD      A,B             ;restore character
                POP     BC              ;restore registers
                RET                     ;Z flag set if ready

3)      RDYTST  PUSH    BC              ;save registers
                LD      B,A             ;save character
                LD      A,(293)         ;test for....
                CP      'I'             ;Model III
                JR      Z,MDLIII        ;JP if Model III
                LD      A,(37E8H)       ;get printer status (Mod 1)
                JR      STATUS          ;perform status check
        MDLIII  IN      A,(0F8H)        ;get printer status (Mod 3)
        STATUS  AND     0F0H            ;strip off lower nybble
                CP      30H             ;compare for ready status
                LD      A,B             ;restore character
                POP     BC              ;restore registers
                RET                     ;Z flag set if ready
```

To locate the RDYTST routine in your driver look at memory location BB52H of your disassembly. There will be a jump ????H here. This is where your RDYTST routine begins. Change it to the following routine:

```
        RDYTST  PUSH    BC              ;save registers
                PUSH    DE              ;      "
                PUSH    HL              ;      "
                LD      B,A             ;store character
                PUSH    BC              ;save character
                LD      DE,(0BB76H)     ;point to PR DCB
                LD      C,00H           ;tell SVC to return status
                LD      A,05H           ;store SVC #
                RST     28H             ;call @CTL
                POP     BC              ;load character
                LD      A,B             ;restore character
                PUSH    HL              ;restore registers
                PUSH    DE              ;      "
                PUSH    BC              ;      "
                RET                     ;Z flag set if ready
```

The final change which needs to be made to your driver is to the PRTCHR routine. Like the RDYTST routine, this call makes use of port (F8H) or specific memory location (37E8H) i/o. This needs to be changed to the appropriate SVC (@PRT). The following are a few examples of the CRITICAL AREAS in the PRTCHR routines:

```
1)              LD      A,(0054H)       ;check for Model 1 or 3
                DEC     A               ;set Z flag?
                JR      NZ,MOD3PR       ;NZ if Model 3
                POP     AF              ;restore character
                LD      (37E8H),A       ;output character
                RET                     ;done
        MOD3PR  POP     AF              ;restore character
                                        ;(previously pushed)
                OUT     (0F8H),A        ;output character

2)              LD      A,E             ;restore character
                                        ;(previously stored in E)
                OUT     (0F8H),A        ;output character
```

These sections (which may appear in your driver more than once) need to be changed to the following:

```
                ???     ???             ;restore character
                PUSH    BC              ;store registers
                PUSH    DE              ;      "
                LD      C,A             ;place character in C
                LD      A,06H           ;store SVC #
                RST     28H             ;call @PRT
                POP     DE              ;restore registers
                POP     BC              ;      "
```

To locate this section of code trace through the PRTCHR routine. This call is JumPed to in your driver at location BB3DH.

After making these changes and upon reassembly you should have a working printer driver for your particular printer for use with Model 4 SuperScripsit. If your driver doesn't work correctly study this article carefully and make sure that you have replaced ALL routines which use specifically located memory or port i/o to the appropriate SVC routines. Good Luck!

The following patches are possible because POWERSOFT left enough room inside their drivers for me to be able to do so. As a result, all of POWERDRIVER-E's (unless they have had revisions which I am unaware of) can be patched with these files. Also, the GPLUS and GPLUS2 patches are the same since the two drivers are virtually identical. Notice also that the CTLF patch is not much different than the CTL patch (this is because the two patches do the same thing except in different places).

One final note, I would enjoy hearing from anyone out there who has benefited from this article. This took me quite a bit of extra time to work-up and I would like to know that it was worth it.

```
.CTL/FIX - M.R. Freifeld for Powersoft customers - 7 February 1984
.
.PATCH GPLUS/CTL using CTL... or
.PATCH GPLUS2/CTL using CTL... are VALID.
.This patch will convert Powersoft's POWERDRIVER-E for use on the
.Model 4 with SuperScripsit.
.
X'BB38'=7E
X'BB7E'=11 50 52 3E 52 EF 22 76 BB CD B5 BB CA DF BB C3 B5 BA
X'BBA6'=00 00 00 00 00 F1 C5 D5 4F 3E 06 EF D1 C1
X'BBB5'=C5 D5 E5 47 C5 C3 C6 BB
X'BBC6'=ED 5B 76 BB 0E 00 3E 05 EF C1 78 E1 D1 C1 C9
.
.EOP - End of patch

.CTLF/FIX - M.R. Freifeld for Powersoft customers - 7 February 1984
.
.PATCH GRAF/CTL using CTLF
.This patch will convert Powersoft's POWERDRIVER-E for use on the
.Model 4 with SuperScripsit.
.
X'BB38'=7E
X'BB7E'=11 50 52 3E 52 EF 22 76 BB CD C7 BB CA F1 BB C3 B5 BA
X'BBB8'=00 00 00 00 00 F1 C5 D5 4F 3E 06 EF D1 C1
X'BBC7'=C5 D5 E5 47 C5 C3 D8 BB
X'BBD8'=ED 5B 76 BB 0E 00 3E 05 EF C1 78 E1 D1 C1 C9
.
.EOP - End of patch
```

ALLWRITE HINTS

by Jack Decker

Here's an Allwrite command you may have missed. Try pressing the BREAK key, then when you see the CMD=> prompt, type a question mark ("?") followed by ENTER. You'll see that the last editor command you entered is redisplayed, and you can edit and/or re-execute it. If you just want to repeat the previous command without re-displaying it first, use an equals sign ("=") instead. This little goodie is hidden away on page 123 of the Allwrite manual.

Pressing the CLEAR key plus the 1, 2, or 3 keys respectively generates a ;pp, ;pa, or ;ce code respectively. However, there's actually a bit more to it than that. Pressing one of those soft keys actually generates the following sequence:

<CLEAR> <N> <;> <p> <p> <CLEAR> <n> <SPACE> <BACKSPACE>

Translation: <CLEAR>+<N> is the "new blank line" editor control key. <;>+<p>+<p> is the actual ";pp" code (this could of course be ";pa" or ";ce" instead, depending on which soft key is depressed). Another <CLEAR>+<n> forces another new blank line, and the <SPACE> followed by a <BACKSPACE> gets rid of the carriage return character on the new line.

Now this is all well and good, but I have found that sometimes I wish to use one of these soft keys to insert one of these control codes, and the code winds up one line below where I really want it. If your sense of logic is also confused by the way these keys operate, try redefining them as follows:

<SHIFT-BACKSPACE> <CLEAR> <N> <;> <p> <p> <ENTER>

(use ;pa or ;ce for keys 2 and 3 respectively). Remember to terminate each line with a <CLEAR>+<Q>. I have found that this tends to make these three particular soft keys work in what seems (to me, anyway) to be a more logical manner.

While I was at it, I redefined soft key "'" (SHIFT-7) as <BREAK> <q> <u> <i> <t> <ENTER>. This is the former definition of soft key ")" (SHIFT-9). So why did I move it? Well, because I redefined soft keys "(" (SHIFT-8) and ")" (SHIFT-9) as follows: Soft key "(" (SHIFT-8) was changed to <CLEAR> <;> <0> <9> <1>. Soft key ")" (SHIFT-9) was changed to <CLEAR> <;> <0> <9> <3>. Now if you press CLEAR plus SHIFT plus 8 or 9, you get a square bracket that corresponds to the parenthesis mark on that key. Makes square brackets easy to enter (and easy to remember which key to press!).

Finally, I defined soft key Z as <CLEAR> <;> <1> <2> <4>. This gives me a vertical "stick" character which I like to use as a tab delimiter. In any event, once you have finished re-defining your soft keys, and are sure you are satisfied with them, press BREAK and type:

KEY S AL/DEF

to save your new soft key definitions. Now enjoy your new, easier to use soft keys!

CHRISTIAN BULLETIN BOARD SYSTEMS

Downloaded from the Good News B.B.S., Plymouth, Michigan

Computers For Christ HQ	San Jose, California	(408) 997-2790
Computers For Christ #2	San Mateo, California	(415) 341-2962
Computers For Christ #6	Columbia, Maryland	(301) 997-7163
Computers For Christ #7	El Dorado, Arkansas	(501) 864-0699
Computers For Christ #8	Washington, D.C.	(301) 596-0123
Prayer In Home B.B.S	California	(714) 974-4294
Kingdom B.B.S.	Santa Barbara, California	(805) 687-2754
Commodore Christian	California	(707) 964-7114
Christian BBS.	Toronto, Ontario	(416) 634-6085
Fishnet	Fort Dodge, Iowa	(515) 576-0591
Good News B.B.S.	Plymouth, Michigan	(313) 459-8375
Good News	Arlington, Texas	(817) 483-1264

If you know of any other Christian Bulletin Boards, please drop a message with the name, city, state, and phone number to the Good News B.B.S. at (313) 459-8735, and they will add it to the list.

PRINTOUT FROM EDITOR/ASSEMBLER-PLUS

by Phil Ereaut

[This article is reprinted from the TRS-80 SYSTEM 80 Computer Group newsletter (MacGregor, Queensland, Australia).]

I generally use roll paper for a printout from a lengthy source listing, and generally prefer to have a continuous print of the code without it being interrupted by the title and page number for each page. By Murphy's Law, the title and page number will invariably occur in the middle of a section that I am working on.

To prevent the message and page number from being printed, use SUPERZAP and make the following changes to EDTASMPL/CMD [or whatever filename your copy of Editor/Assembler-Plus goes by -editor]:

```
FRS 6,9A Change 5E 10 FA 36    to 5E 00 00 36
FRS 6,E6 Change 35 CC 85 5E E5 to 35 00 00 00 E5
```

NOTE: This zap applies to Editor/Assembler-Plus by Microsoft and not EDTASM, the Editor Assembler by Tandy [or Apparat or whomever -editor].

MULTIDOS MODEL 4 BASIC ENTRY POINTS

by Vern Hester (author of MULTIDOS)

ROM BASIC (MODEL I/III) equivalent entry points.

ROM BASIC	MODEL 4
01C9H	529AH
032AH	5266H
0358H	52EBH
035BH	52F4H
0384H	525AH
038BH	5278H
1650H	1698H
1924H	1882H
1929H	187BH
1930H	1897H
19A2H	32EEH
1A19H	35FAH
1AF8H	1680H
1AFCH	1684H
1B10H	1646H
1B2CH	1662H
1B2FH	1665H
1B4DH	35F1H
1B59H	35F4H
1B5DH	35F7H
1BC0H	196EH
1E3DH	5289H
1E4FH	52ADH
1E5AH	52B7H
20F9H	52DFH
20FEH	52E4H
25A1H	32F1H
260DH	35FDH
2B75H	32F8H

The above MODEL 4 addresses are only valid while in BASIC. They are not valid during a CMD"uuuuu" and become invalid after a CMD"S". Although the RAM may not be disturbed around the entry point, other calls/jumps can be altered.

Alternate Support for Your TRS-80 Model I, III or 4

The Alternate BASIC

The Alternate BASIC is a structured BASIC translator. What this means is that you can use your word processor to write high-level BASIC code using commmands like DO/WHILE and DO/UNTIL and SELECT CASE. You can give each BASIC subroutine it's own name and COPY this subroutine into your code as needed. You can CALL that subroutine by it's name, without regard to line numbers. As a matter of fact, after using The Alternate BASIC for a short while, line numbers will become much less important to your programming projects. This means that you will be able to insert as much code as desired, where ever it is needed. Not only will debugging and repairs be much easier, but once a routine is debugged, you will get maximum use from it. Commands included with The Alternate BASIC are very close to ANSI standards for BASIC.

There's more. With The Alternate BASIC, you will be able to use a process called "conditional translation" to create a single master source file that can be translated for TRSDOS, CP/M, MSDOS and any other deriative of BASIC that you should encounter; you define your own Conditionals and can call them what you please. The Alternate BASIC fully respects your coding and "translates" only the high level structured constructs.

The complete Alternate BASIC package, including a hard copy user's manual, the translator and startup subroutine library on diskette, is only $59.95. For a more extensive look into The Alternate BASIC and it's competitors, see the June, 1985 issue of 80-Micro.

The TAS PD Library

For the past two years, The Alternate Source Information Outlet has been collecting, buying, trading and otherwise soliciting programs to include in The TAS Public Domain Library. Many of the public domain programs found in other libraries originated at TAS. We have been especially selective about what programs to include in the library, seeking only the best. We don't have hundreds of diskettes, like many other public domain libraries, even though we could. Our current offering includes only eight single density, 40 track, "FLIPPY" diskettes, jam-packed with top quality software for your TRS-80 Model I, III and 4. Each disk is only $10. Send a SASE with 39 cents in postage for a complete index of all public domain diskettes available in The TAS Public Domain Library.

UNIKEY

UNIKEY is a machine language program for use on the Radio Shack Model 4 using TRSDOS 6.x. It is used in conjunction with BASIC to permit single key entry of 85 key words and phrases. In addition, UNIKEY offers three "programmable" key combinations, each of which can be assigned by the user

UNIKEY is designed to interact with the BASIC that is included with TRSDOS 6.x. UNIKEY is installed using a small filter program and is always available, upon demand, and out of sight when you don't need it. A /JCL file is included on the master disk for easy installation of UNIKEY. When installing UNIKEY, the user has the option of including a 1K HELP screen that quickly displays the key combinations to produce any of the UNIKEY substitutions.

Conversion Utilities

Keep your TRS-80 well supplied with software -- developed on other machines!

Model 4 only: Gee Whiz Convert is a series of programs and utilities that will convert BASIC code written in IBM's version of MicroSoft BASIC called Gee-Whiz BASIC to run on a TRS-80 Model 4. There are hundreds of diskettes in the IBM public domain library that can be converted to run with TRSDOS 6.x; this program makes that a simple process. Several machine language routines are included to assist with the conversion. There is no royalty if these routines are used in commercial packages. This package should be in every software house! Gee-Whiz Convert with it's extensive programmer's manual, detailing how to get around many peculiarities between the two versions of BASIC, and the supplemental, royalty-free routines are $99.95, complete.

Model I/III only: BAS43 is a utility that will assist you with bringing native Model 4 programs to your Model I/III environment. Support is provided for screen formatting, variable translation, code differences and more. BAS34 allows Model I and III owners to perform their own conversions with minimal effort. This is one program that every software house should have. BAS34 is only $49.95 with complete documentation.

BAS34 is a utility that will convert TRS-80 Model I and III code to run with the TRS-80 Model 4 BASIC. In addition to properly setting all spaces and fixing various syntax problems, an optional flag can be used for converting the PRINT @ statements to LOCATE statements (Model 4 can use LOCATE and PRINT @ statements), if you plan to take the code on to MSDOS. All lines that BAS34 does not convert, but contain suspect code, are flagged for easy operator intervention, if necessary. Versions of BAS34 are available that will translate code under all I/III disk operating systems EXCEPT TRSDOS, MSDOS (standard), MSDOS (2000), TRSDOS 6.x and CP/M. Special provisions are made for full utilization of each operating system. First version is $49.95. All additional versions are $10 each. Additional versions do not include additional manuals; these are available for $5 each.

TINYPAY 3.0

TINYPAY is a payroll system for the small businessperson. TINYPAY has been upgraded to support a mixture of payroll periods; if you have certain employees getting paid weekly and others monthly, for example, we've got you covered. TINYPAY also supports both state and local residency and non-residency.

TINYPAY will output a detailed hard-copy report for the payroll period and maintain totals for your specific accounting period. Hard-copy reports can be generated whenever you choose. You may also list a record of any one employee's payroll activity between two dates, as required should an employee become unemployed. Employee 941 withholdings can be previewed at any time. TINYPAY will add new employees, edit all employee data, and process the payroll period. TINYPAY is currently available in three configurations: TRS-80 I/III, TRS-80 4 and MSDOS. TINYPAY Version 3.0 is $29.95 including a new expanded manual.

The Alternate Source Information Outlet
704 North Pennsylvania Avenue
Lansing, Michigan 48906 U.S.A.
Orders and Technical Assistance: (517) 482-8270

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, MI 48906
(517) 482-8270

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

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

POSTMASTER: If undeliverable return to:
The Alternate Source, 704 North Pennsylvania Avenue, Lansing, Michigan 48906

To: