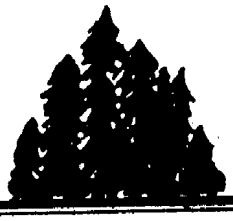
NORTHERN BYTES



Volume 5 Number 5

ATTENTION!! PLEASE READ THIS!!! NORTHERN BYTES is mailed free of charge to any and all computer clubs and user groups THAT EXCHANGE NEWSLETTERS WITH US. Even though we have increased our mailings to such clubs and user groups, we have received correspondingly fewer and fewer exchange newsletters. By failing to send us an exchange newsletter, your club is hurting itself in two ways:

1) You will NOT receive any more issues of NORTHERN BYTES after this one (at least not on a regular basis), and

2) When we get an inquiry from someone in your club's geographical area, as sometimes happens, we will not suggest that they contact your group because we won't have any way of knowing whether or not your club is still in operation. There's a third shortcoming as well - in the future I MAY decide to publish a list of active TRS-80 user groups, and if we haven't heard from you, you won't be on it.

We also exchange with some non-club-related PUBLICATIONS (or at least, have attempted to exchange with them), but have had no response from a few of these. If you fall into this category, this is your last issue as well, unless we hear from you!

A POSTCARD CAN KEEP YOUR GROUP ON THE ACTIVE LIST, if your club or user group does NOT publish a newsletter or has temporarily suspended publication. We may ask you to send in a card once or twice a year, but we'll let you know when. Just drop us a postcard giving the name and address of your TRS-80 club or user group and the approximate number of members in your group. If we've never heard of you before, we reserve the right to ask for some verification of your club or user group status (a copy of a meeting announcement, a newspaper ad or article promoting your club, etc.)

PLEASE CHECK OUR ADDRESS ON YOUR RECORDS - Part of the problem may be that our local post office sometimes "loses" mail that is not addresses correctly. In particular, PLEASE check that you do not have "LOT # 55" as part of our address. This is incorrect and if a substitute mail carrier is working the day you newsletter arrives, we may never see it. The correct lot number is 155. It's amazing how many people seem to miss that first "1" even though it's in plain sight. The ONLY correct address for NORTHERN BYTES is as follows:

NORTHERN BYTES
c/o Jack Decker, editor (this line is optional)
1804 West 18th Street Lot # 155
Sault Ste. Marie, Michigan 49783

Please don't assume that just because you know (or think) that you're sending us your newsletter, that you have our address correct in your records. MANY of the exchange newsletters we receive have an incorrect address!! Some come addressed to "Microcomputer Users International", an organization that no longer exists, and some come addressed to The Alternate Source (sometimes at the Lansing address, no less), which is NOT usually recognized by the post office up here. Sault Ste. Marie is about 250 miles away from The Alternate Source!

THIS IS FOR EVERYONE, whether you're connected with a user group or not. If you need to contact me in a hurry, and especially if you have text intended for the newsletter, please send it via MCI Mail if possible. I check MCI Mail regularly both from Sault Ste. Marie, and from Lansing when I am down there (a distinct advantage of electronic mail - it can follow me where'er I go!). However, whenever you send ANYTHING via MCI Mail, PLEASE include your regular return address. This is doubly important when you are ordering merchandise from The Alternate Source, since it is pretty hard to mail books or diskettes over a MODEM.

Some confusion seems to exist over what should be sent where, when dealing with NORTHERN BYTES or THE ALTERNATE SOURCE. Here are some guidelines:

1) If you're sending something via MCI Mail, don't sweat it. You can even combine TAS orders and newsletter submissions in the same message. This applies ONLY to MCI Mail users.

2) Exchange newsletters, submissions or other editorial material for Northern Bytes and/or the TAS Public Domain Software Library (which we assume you are contributing to the public domain), subscription problems with Northern Bytes (former Opinion-80 subscribers only!), and ALL requests for current or back issues of NORTHERN BYTES to be mailed OUTSIDE of the U.S.A., Canada, or Mexico should be sent to NORTHERN BYTES at the return address on this newsletter (the Sault Ste. Marie address). Also, if you have been instructed by TAS to communicate with me directly in regard to a particular project, follow those instructions.

3) All orders for merchandise (including back issues of Northern Bytes if you live in the U.S.A., Canada, or Mexico), requests for general information about the TAS Public Domain Software Library, or program submissions for The Alternate Source (programs for which you expect to receive a payment and/or royalties MUST be sent to The Alternate Source!) should be sent to TAS at 704 North Pennsylvania Avenue, Lansing, Michigan 48906.

Speaking of SUBMISSIONS, we need them if we are to continue publishing Northern Bytes. Unless you specifically state otherwise, we assume that all submissions to NORTHERN BYTES are eventually intended to be placed in the public domain, so they may show up on a TAS Public Domain Library disk at some point. Keep in mind that contributors to NORTHERN BYTES receive the next six issues FREE (assuming we publish six more issues, which we don't guarantee), and at this point that's the ONLY way to get a "subscription" to this newsletter! And, you also have our heartfelt "Thanks!" for anything you contribute.

Starting next issue, we'll be adding expanded coverage of the FORTH language to NORTHERN BYTES. We've asked Paul Snively to edit a FORTH-related column, and he has said he'd be happy to do it, so if you have questions, comments, tips, techniques, etc. about the FORTH language, send them to Paul c/o The Alternate Source (for the time being, I'll give you another address later on), or directly to Paul via MCI Mail at 176-6817.

Do any of you folks program on the SANYO personal computer, or do you know of anyone that does? Do you have any interest in the SANYO? If so, please send a postcard or letter to The Alternate Source, indicating your interest and/or level of expertise (if any). It's quite possible that TAS may offer expanded support for the Sanyo line in the future, if there is any interest. We've discovered that several TRS-80 owners (including some whose names you'd probably recognize) have recently acquired the Sanyo, so naturally we're interested. As for coverage in NORTHERN BYTES, well, I'm not sure I want to try to support two brands here, but anything's possible. I'd just LOVE to get an article from someone reviewing the Sanyo, and perhaps pointing out the similarities and differences between it and the TRS-80. Who knows, we might even do a totally separate publication just for Sanyo owners one of these days.

Do we owe you anything? Occasionally we may promise someone something, and then forget about it, or we may have kept a disk that you sent us (with a contribution to NORTHERN BYTES on it) when you expected us to return it. If so, don't be shy about reminding us. A postcard will do, and it's a lot better that thinking bad thoughts about us forever more. We're only human, too. By the way, when you send us a contribution on disk and you want the disk back, please SAY SO! You think we'd naturally assume that? Well, not really, because a lot of people save "junk" diskettes (for example, a disk that won't format properly on the last five tracks) and use them to send articles to newsletters and the like. In many cases, they hope they never see that disk again! People tell us that we can keep the disk so often that we tend to not notice when someone doesn't say anything about it. We automatically return disks submitted to the TAS Public Domain Software Library (as stated in the information sheet), but there may be a delay of a couple months or more, so please be patient.

THE EXTERMINATOR – Another attack of the killer BUGS to vanquish from previous issues of NORTHERN BYTES!

Back in Volume 5, Number 3 we told you how to move the Model 4 ROM into RAM. We also printed a zap for NEWDOS/80 to keep it from clobbering the ROM area of memory after it has been transferred to RAM. Trouble is, with NEWDOS/80 one zap is not enough. So far we've found four that need to be made, and we're still counting. The four zaps we have so far are as follows:

SYS0/SYS,02,ABH:	change:	26 01 CD	to:	26 38 CD
SYS6/SYS,13,DCH:	change:	00 00 00	to:	00 38 00
SYS6/SYS,30,EEH:	change:	00 00 CD	to:	00 38 CD
SYS19/SYS,04,29H:	change:	11 00 CD	to:	11 38 CD

The above zaps move the location of a "bit bucket" used during disk verify operations from 0000H or 0100H to 3800H (the memory-mapped keyboard area, which is still considered a "read-only" section of memory). The second through fourth zaps (to SYS6/SYS and SYS19/SYS) fix similar problems that occur only when the FORMAT and BASIC "SAVE" commands are executed. Thanks to GREG SMALL from Stouffville, Ontario, Canada; LYMAN EPP of Omaha, Nebraska; and VERN HESTER of Cosmopolitan Electronics Corporation (author of MULTIDOS and ZEUS) for contributing these zaps.

Users of VIDEO4 (a program sold by The Alternate Source, that lets you utilize the Model 4 24x80 video display while operating in the "Model III mode") should be sure to apply ALL of the above zaps to any NEWDOS/80 system disk that will be used with VIDEO4. None of these zaps adversely affects the normal operation of NEWDOS/80.

Turning to the last issue (Volume 5, Number 4) of NORTHERN BYTES, we almost had a bug in "The Exterminator" column! If you read this column last time, you may recall Paul Snively's comments about not closing SETDATE/CMD, and his method of getting around it. Problem was, about an hour before we started printing that page we discovered that SETDATE wouldn't work at all under Model III TRSDOS 1.3 with Paul's patches installed! We quickly inserted a notice to that effect on the master copy of that page, but didn't have time to redo the page. So, for the record, the "official" version of SETDATE (as found on the latest version of TAS Public Domain Library disk # 001) does NOT include Paul Snively's patches (it doesn't need them, anyway), but DOES include the OR A instruction inserted at line 2045 of the original version of the SETDATE source code (which is required for proper operation). Lest you think too harshly of Paul, he owns a Model I and his suggested patches DO work with every DOS except TRSDOS 1.3!

In the same issue (Volume 5, Number 4, page 15), Paul Snively also provided a subroutine to determine the double density adapter in use on a Model I (Radio Shack or non-Radio Shack). We neglected to mention that the code segment was originally authored by Vern Hester of MULTIDOS fame, and also neglected to state that the fourth instruction from the end [the second occurrence of LD (IY+2),E] resets a Radio Shack type doubler to single density (but does not affect a non-Shack doubler). This instruction should be removed if it is desired to leave the doubler in double density mode.

BINARY CLOCK by Jack Decker – Here's a program that you can use to fascinate kids, or maybe as a quick answer to the age-old question, "But what can you DO with a computer?" If you really need some justification to type it in, try to convince yourself that it's educational because it teaches the binary counting system.

Basically, this is a machine-language version of a binary clock program that was once used as a giveaway by The Alternate Source. The original program was in BASIC and had only one problem – it ran too slow to keep up with the march of time! It worked fine if you had a CPU speed-up or if you compiled it, but otherwise ... well, now you know why it was a giveaway!

This version runs plenty fast (in fact, a bit of delay has been added for "effect"), but it's still not the world's most practical program! Still, it's short and makes a good demonstration program – it has visual impact, if nothing else!

One final note – each time division (hours, minutes, and seconds) are counted in increments from 0-59 or 0-23, represented by eight-bar byte. Each byte is visually subdivided into two four-bar halves, but should be considered as one byte, and NOT as two Binary Coded Decimal digits! It wouldn't be too difficult to convert the program to display the bars in a BCD pattern, but I'll leave that as an exercise for the reader. Here's the program listing:

```
7000                ORG     7000H
7000 FB      START  EI              ;Enable Interrupts
7001 CDC901         CALL    01C9H   ;Clear Screen
7004 21A970  QUERY  LD      HL,MSG  ;Point to question
7007 CD6744         CALL    4467H   ;Display it on video
700A CD4900         CALL    0049H   ;Wait for user response
700D E6DF           AND     0DFH    ;Change lower to uprcase
700F FE59           CP      'Y'     ;"Y" or "y" response?
7011 280C           JR      Z,YES   ;Go if so
7013 FE4E           CP      'N'     ;"N" or "n" response?
7015 2807           JR      Z,NO    ;Go is so
7017 FE01           CP      1       ;<BREAK> key pressed?
7019 20E9           JR      NZ,QUERY ;Get another key if not
701B C32D40  EXIT   JP      402DH   ;Exit to DOS READY
701E 37      NO     SCF             ;Set carry flag = NO
701F F5      YES    PUSH    AF      ;Save carry flag status
7020 CDC901         CALL    01C9H   ;Clear Screen
7023 CD8970         CALL    BORDER  ;Put top & bottom borders
7026 CD2B00  RESTRT CALL    002BH   ;Check for key depressed
7029 3D             DEC     A       ;If it was <BREAK>, A=0
702A 28EF           JR      Z,EXIT  ;Exit if <BREAK>
702C 214140         LD      HL,4041H ;"Seconds" byte in Mod I
702F 3A5400         LD      A,(54H) ;Get Mod I/III test byte
7032 3D             DEC     A       ;A=0 if Model I
7033 2803           JR      Z,MOD1  ;Go if Model I
7035 211742         LD      HL,4217H ;"Seconds" byte in Mod 3
7038 4E      MOD1   LD      C,(HL)  ;Get initial seconds val
7039 7E      LOOPA  LD      A,(HL)  ;Get new seconds value
703A B9             CP      C       ;Has value changed?
703B 28FC           JR      Z,LOOPA ;Wait for change if not
703D F1             POP     AF      ;Get Carry flag status
703E F5             PUSH    AF      ;Also re-save it
703F 380C           JR      C,NODSP ;Go if no date/time disp
7041 E5             PUSH    HL      ;Save "seconds" pointer
7042 21D73F         LD      HL,3FD7H ;Point to video location
7045 CD7044         CALL    4470H   ;Display date
7048 23             INC     HL      ;Skip a space
7049 CD6D44         CALL    446DH   ;Display time
704C E1             POP     HL      ;Restore "seconds" pntr
704D EB      NODSP  EX      DE,HL   ;"Seconds" byte ptr in DE
704E 21B03C         LD      HL,3CB0H ;Top of last time bar
7051 0603           LD      B,3     ;# of time storage bytes
7053 C5      LOOPB  PUSH    BC      ;Save storage byte count
7054 0602           LD      B,2     ;# nybbles per time byte
7056 1A             LD      A,(DE)  ;Get current time byte
7057 C5      LOOPC  PUSH    BC      ;Save nybble count
7058 0604           LD      B,4     ;# time bars per nybble
705A 0EBF    LOOPD  LD      C,0BFH  ;C = Graphics block char
705C 1F             RRA             ;Get bit from time byte
705D 3802           JR      C,SET   ;Go if bit was set (ON)
705F 0E20           LD      C,20H   ;C = Space char (bar OFF)
7061 E5      SET    PUSH    HL      ;Save video loc pointer
7062 D5             PUSH    DE      ;Save time byte pointer
7063 C5             PUSH    BC      ;Save time bar counter
7064 060C           LD      B,0CH   ;# vertical graphic blcks
7066 114000         LD      DE,40H  ;Length one video line
7069 71      LOOPE  LD      (HL),C  ;Put character on video
706A 19             ADD     HL,DE   ;Move down one line
706B 10FC           DJNZ    LOOPE   ;Do til vertical bar done
706D F5             PUSH    AF      ;Save time byte
706E 010004         LD      BC,0400H ;Time delay counter
7071 CD6000         CALL    0060H   ;Call ROM delay routine
7074 F1             POP     AF      ;Restore time byte
7075 C1             POP     BC      ;Restore time bar counter
7076 D1             POP     DE      ;Restore time byte pntr
7077 E1             POP     HL      ;Restore video loc pntr
7078 2B             DEC     HL      ;Backup two spaces to top
7079 2B             DEC     HL      ;  of next vertical bar
707A 10DE           DJNZ    LOOPD   ;If more bits this nybble
707C 2B             DEC     HL      ;Backup two extra spaces
707D 2B             DEC     HL      ;  between nybbles
707E C1             POP     BC      ;Restore nybble count
707F 10D6           DJNZ    LOOPC   ;If only 1st nybble done
7081 2B             DEC     HL      ;Backup two extra spaces
7082 2B             DEC     HL      ;  for hours-mins-secnds
7083 13             INC     DE      ;Bump time byte pointer
7084 C1             POP     BC      ;Restore time byte count
7085 10CC           DJNZ    LOOPB   ;Do remaining byte if any
7087 189D           JR      RESTRT  ;Else wait for nxt second
7089 21013C  BORDER LD      HL,3C01H ;Point to top border loc
708C CD927C         CALL    TEORD   ;Put top border on video
```

```
708F 21C13F    00910          LD      HL,3FC1H        ;Pnt to lower border loc
7092 3E03      00920 TBORD    LD      A,3             ;# of bytes to display
7094 0E02      00930 BLOOPA   LD      C,2             ;# of nybbles per byte
7096 0604      00940 BLOOPB   LD      B,4             ;# of bits (bars)/nybble
7098 36BF      00950 BLOOPC   LD      (HL),0BFH       ;Put graphic blk on video
709A 23        00960          INC     HL              ;Bump video location pntr
709B 23        00970          INC     HL              ;Skip space between blcks
709C 10FA      00980          DJNZ    BLOOPC          ;Do all 4 bits of nybble
709E 23        00990          INC     HL              ;Skip two extra spaces
709F 23        01000          INC     HL              ;  between nybbles
70A0 0D        01010          DEC     C               ;Decrement nybble count
70A1 20F3      01020          JR      NZ,BLOOPB       ;Go if only 1 nybble done
70A3 23        01030          INC     HL              ;Skip two more extra spcs
70A4 23        01040          INC     HL              ;  between hrs-mins-secs
70A5 3D        01050          DEC     A               ;Decrement byte count
70A6 20EC      01060          JR      NZ,BLOOPA       ;Go if more to display
70A8 C9        01070          RET                     ;(Routine calls itself!)
70A9 0A0A      01080 MSG      DEFW    0A0AH           ;Two line feeds
70AB 44        01090          DEFM    'Display time and date at bottom of video
display (Y/N)?'
     69 73 70 6C 61 79 20 74 69 6D 65 20 61 6E 64 20
     64 61 74 65 20 61 74 20 62 6F 74 74 6F 6D 20 6F
     66 20 76 69 64 65 6F 20 64 69 73 70 6C 61 79 20
     28 59 2F 4E 29 3F
70E2 0D        01100          DEFB    0DH             ;<CR> message terminator
7000           01110          END     START
00000 TOTAL ERRORS

BLOOPA  7094    BLOOPB  7096    BLOOPC  7098    BORDER  7089    EXIT    701B
LOOPA   7039    LOOPB   7053    LOOPC   7057    LOOPD   705A    LOOPE   7069
MOD1    7038    MSG     70A9    NO      701E    NODSP   704D    QUERY   7004
RESTRT  7026    SET     7061    START   7000    TBORD   7092    YES     701F
```

FORTH-WISE by Paul Snively [Editor's note: Paul is going to be doing a regular FORTH column for NORTHERN BYTES, so if you have anything FORTH-related to contribute - anything at all - or if you've got questions about FORTH that you'd like answered, please contact Paul directly at MCI Mail I.D. 176-6817, or in care of The Alternate Source.]

I've come up with a few things Forth-wise that might be of interest to TASForth users. Anyway, here goes:

Some implementations of Forth allow recursion. There are basically two ways to do this. One is to allow the use of the word currently being defined within the word itself. This technique is not frequently used because it conflicts with the standards set up for Forth's functioning. The other is to include a word called MYSELF within the Forth vocabulary which, when included in a definition, compiles the code field address of the word being defined into the word. Well, after about 10 minutes of tinkering and referring to the TASForth manual, I came up with MYSELF for TASForth. This word, which makes the TASForth system considerably more powerful than it was (in terms of recursive capabilities) has been placed in the public domain by Paul F. Snively.

```
: MYSELF  SMUDGE LATEST PFA CFA , SMUDGE ; IMMEDIATE
```

That's all there is to it! How is it used? Well, let's give an example. Suppose we wanted to calculate N!, where N! is defined as:

N! = 1 if and only if N = 0
N * (N-1)! if and only if N > 0

As you can see, N! is a recursive function; that is, it's defined in terms of itself. Without MYSELF, TASForth couldn't handle the classical N! definition. With MYSELF, however, the definition is easy. It's:

```
: FAC  DUP 0= IF DROP 1 ELSE DUP 1- MYSELF * THEN ;
```

I won't bother to describe the logic; anyone who's interested enough in Forth to want to do recursion will be able to see what's happening in FAC easily enough.

There's another Forth-related Bulletin Board that I forgot to mention in my article last issue; it's at (415) 538-3580, and they have OODLES and OODLES of Forth goodies. What kind of Forth goodies? Well, here's one (placed in the public domain by Marc Perkel.) The following is the documentation provided by Mr. Perkel, cleaned up and corrected by Paul Snively.

"ARGUMENTS-RESULTS: Local reentrant variables for Forth Overview: One thing Forth lacks that other languages have is local reentrant variables. By local I mean that they serve as temporary storage for stack items in situations in which the stack would be too jumbled if used with conventional methods, i.e., SWAP, ROT, OVER, PICK, ROLL, etc. (See examples.) What I mean by reentrant is that all local variable names act only in their own definition. That means that local variable S6 in one definition will contain a different value than S6 in another definition. This is the factor that separates local and global variables.

"The technique used is shown in high level code but when written in native code runs almost as fast as equivalent code using conventional means. In fact, on the 68000, it will run faster since the 68000 has included an addressing mode that uses my technique exactly (LINK and UNLINK.)

"The technique used allots stack space to be used as temporary storage of transient parameters. The stack pointer is moved down in memory and a pointer is set pointing to the allotted local variable space. Before the pointer is set, however, the pointer's previous contents are saved on the return stack.

"Arguments-results does not make any changes necessary in the present standard; it merely adds to it. Its main disadvantage is that it is so easy to use that it will be used too often. It is intended for use only when the stack would get too nasty otherwise. It is also a stack hog."

Source code in high level Forth:

```
0 VARIABLE <:ARG:>  0 VARIABLE <:TO:>
: +ARG <BUILDS , DOES> @ <:ARG:> @ SWAP - <:TO:> @ EXECUTE
<: FIND @ :> LITERAL <:TO:> ! ;

 0 +ARG S1    2 +ARG S2    4 +ARG S3    6 +ARG S4
 8 +ARG S5   10 +ARG S6   12 +ARG S7   14 +ARG S8
16 +ARG S9

: TO  <: FIND  ! :> LITERAL <:TO:> ! ;
: +TO <: FIND +! :> LITERAL <:TO:> ! ;

: ARGUMENTS ( n - ) R> <:ARG:> @ >R >R SP@ OVER 2 * + <:ARG:>
! 9 SWAP - 0 DO 0 LOOP <: FIND @ :> LITERAL <:TO:> ! ;

: RESULTS ( n - ) 2 * <:ARG:> @ SWAP - SP@ - 2 / 0 DO DROP
LOOP R> R> <:ARG:> ! >R ;

(ARGUMENT EXAMPLE --- Box comes in with height, length &
width and leaves volume, surface area & length of edges):

:  BOX        3  ARGUMENTS
   (  VOLM )  S1  S2  S3  *  *
   (  SURF )  S1  S2  *  S2  S3  *  S1  S3  *  +  +  2  *
   (  EDGE )  S1  S2  S3  +  +  4  *
              TO S3  TO S2  TO S1
              3  RESULTS  ;

CONVENTIONAL CODING OF BOX, 2 EXAMPLES:

:  BOX    >R  2DUP  R@  *  *  R>  SWAP  >R
          >R  2DUP  *  SWAP  ROT
          DUP  R@   *  SWAP  ROT
          DUP  R@   *  SWAP  ROT
          R>  +  +  4  *  >R
          +  +  2  *  R>  R>  ROT  SWAP  ;

: 3DUP  3 PICK 3 PICK 3 PICK ;
: VOLUME * * ;
: S-AREA >R 2DUP * SWAP ROT R@ * SWAP R> * + + 2 * ;
: EDGES + + 4 * ;
: BOX  3DUP EDGES >R 3DUP S-AREA >R VOLUME R> R> ;
```

Well, there it is. Neat, huh? These words really make coding complicated words much simpler, even though there's a pretty hefty price to pay in terms of speed. Someone could conceivably code his word using ARGUMENTS-RESULTS and then recode them after debugging using the conventional stack words.

By the way, there is a bug in early versions of the Forth assembler. On screen 57 (I think) the word RST, should be defined as:

```
: RST,  8 * C7 + ;
```

if the RST, word is to work as it is described in the manual.

All of the above information is in the public domain, and may be freely reprinted or placed on your local Bulletin Board System.

NEWDOS/80 SCREEN DUMP TO DISK FILE PROGRAM - This program was written by Ron Zajac and allows you to dump the current video display to a disk file, which could be useful in any number of applications. Assemble the program using the filename SD456/CMD for the object code. Then, when you will need to use this feature, type SD456 from NEWDOS/80 READY. After that, you may depress the "456" keys simultaneously to dump a "snapshot" of the video display to a disk file. A prompt will appear (temporarily) asking you for the name of the file you wish to create. Once you give a filename, the contents of the video display (BEFORE the prompt appeared) is saved to disk (graphics and "special" characters are saved with their normal ASCII values). The display is then restored to its original appearance (the filename prompt is removed and the original video display is restored).

This program ties into the interrupt chain, so it will not work properly if interrupts are disabled. It will work on the Models I and III, and the Model 4 in the Model III mode only.

Note that the source code listing below was originally created using ALE (the Full Screen Editor and Z80 Assembler program sold by TAS), although the output was modified slightly for publication. Any Editor-Assembler program could be used to assemble the source code, however, some assemblers do not permit multiple bytes in a DEFB statement. In that case you will have to assign each byte value in a DEFB statement to its own DEFB statement. In other words, a line of the form:

```
DEFB 23,47,59
```

would have to be changed to:

```
DEFB 23
DEFB 47
DEFB 59
```

You would have to make this change if you were using the original Radio Shack Editor-Assembler or one of its variants (including the Apparat version supplied with NEWDOS/80).

If you don't feel like typing this program into your TRS-80, both source and object code versions of the program will be available on a TAS Public Domain Library diskette. The source code listing follows:

```
00100                   ;       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00110                   ;       x                              x
00120                   ;       x  Screen Dump to Disk File    x
00130                   ;       x           under              x
00140                   ;       x         NEWDOS/80            x
00150                   ;       x                              x
00160                   ;       x      by Ron A. Zajac         x
00170                   ;       x                              x
00180                   ;       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
00190
00200 035B              INKEY   EQU     035BH           ;Loads ACC w/ current keystroke
00210 3C00              SCREEN  EQU     3C00H           ;CRT Memory
00220 4049              HIMEM1  EQU     4049H           ;Mod I     HIMEM
00230 4411              HIMEM3  EQU     4411H           ;Mod III/4 HIMEM
00240 4410              ENQUE1  EQU     4410H           ;Mod I     Enqueue Interrupt
00250 447B              ENQUE3  EQU     447BH           ;Mod III/4 Enqueue Interrupt
00260
00270 5200                      ORG     5200H
00280
00290 5200              LDSTRT  EQU     $
00300 5200 3A5400               LD      A,(54H)                 ;Check if Mod I or III/4
00310 5203 3D                   DEC     A
00320 5204 280F                 JR      Z,HM1                   ;Go if Model I
00330 5206 211144               LD      HL,HIMEM3               ;Model III MEM SIZE
00340 5209 221652               LD      (HM1+1),HL              ;Patch into program
00350 520C 222052               LD      (HM2+1),HL
00360 520F 217B44               LD      HL,ENQUE3               ;Mod III Enqueue Routine
00370 5212 225552               LD      (ENQ+1),HL              ;Patch into program
00380
00390 5215 2A4940       HM1     LD      HL,(HIMEM1)             ;Get MEM SIZE
00400 5218 01C402               LD      BC,ENDING-DMPSCN        ;Get program length
00410 521B B7                   OR      A                       ;Clear carry flag
00420 521C E5                   PUSH    HL
00430 521D ED42                 SBC     HL,BC                   ;HL <-- new MEM SIZE
00440 521F 224940       HM2     LD      (HIMEM1),HL             ;Posit new MEM SIZE
00450 5222 119356               LD      DE,ENDING-1             ;DE <-- Rev. Source
00460 5225 E1                   POP     HL                      ;HL <-- Destin.
00470 5226 EB                   EX      DE,HL                   ;HL=Source / DE=Destin.
00480 5227 ED88                 LDDR
00490 5229 13                   INC     DE                      ;DE <-- Start Program
00500
00510 522A 211500               LD      HL,PT1-DMPSCN+1
00520 522D CD5D52               CALL    REPLCE
00530 5230 218000               LD      HL,PT2-DMPSCN+1
00540 5233 CD5D52               CALL    REPLCE
00550 5236 218A00               LD      HL,PT3-DMPSCN+1
00560 5239 CD5D52               CALL    REPLCE
00570 523C 219700               LD      HL,PT4-DMPSCN+1
00580 523F CD5D52               CALL    REPLCE
00590 5242 219F00               LD      HL,ERROR-DMPSCN+1
00600 5245 CD5D52               CALL    REPLCE
00610 5248 21B600               LD      HL,PT6-DMPSCN+1
00620 524B CD5D52               CALL    REPLCE
00630 524E 21D500               LD      HL,NOFILE-DMPSCN+1
00640 5251 CD5D52               CALL    REPLCE
00650
00660 5254 CD1044       ENQ     CALL    ENQUE1          ;Enqueue "456" scan/dump
00670 5257 216A52               LD      HL,ENTMSS
00680 525A C36744               JP      4467H           ;Send opening mess. to display,
00690                   ;                                Exit to DOS
00700
00710 525D 19           REPLCE  ADD     HL,DE           ;Get address of label
00720 525E 4E                   LD      C,(HL)
00730 525F 23                   INC     HL
00740 5260 46                   LD      B,(HL)          ;BC <-- displacement
00750 5261 D5                   PUSH    DE              ;Save start address
00760 5262 EB                   EX      DE,HL
00770 5263 09                   ADD     HL,BC           ;Add Start to Displacement
00780 5264 EB                   EX      DE,HL           ;Get address into HL
00790 5265 72                   LD      (HL),D
00800 5266 2B                   DEC     HL
00810 5267 73                   LD      (HL),E          ;(HL) <-- new LD/JP vector
00820 5268 D1                   POP     DE              ;Restore Start address
00830
00840 5269 C9                   RET
00850
00860 526A              ENTMSS  EQU     $
00870
00880                   ;       This is the "SD456" banner message graphics --
00890
00900 526A 0A                   DEFB    10
00910 526B 20202020             DEFB    32,32,32,32,32,32,32,32,32,32,32,32,144,164
                        20202020202020209BA4
00920 5279 89902020             DEFB    137,144,32,32,32,152,151,131,137,32,154,151
                        2098978389209A97
00930 5285 83A99820             DEFB    131,169,144,32,32,160,174,32,32,160,175,131
                        20A0AE2020A0AF83
00940 5291 83812020             DEFB    131,129,32,32,160,158,129,32,32,32,152,161
                        A09E812020209BA1
00950 529D 8490                 DEFB    132,144
00960 529F 0A                   DEFB    10
00970 52A0 20202020             DEFB    32,32,32,32,32,32,32,32,32,32,132,153,162
                        20202020202089992A2
00980 52AD 8499A284             DEFB    132,153,162,132,32,32,32,137,178,137,144,32,149
                        202089B289902095
00990 52B9 95209595             DEFB    149,32,149,149,32,152,171,170,32,32,130,141
                        2098ABAA2020828D
01000 52C5 AC902020             DEFB    172,144,32,32,184,183,176,32,32,32,166,136
                        B8B7B0202020A688
01010 52D1 91A68891             DEFB    145,166,136,145,132
                        84
01020 52D6 0A                   DEFB    10
01030 52D7 20202020             DEFB    32,32,32,32,32,32,32,32,32,32,32,129,134
                        2020202020202088186
01040 52E5 98812020             DEFB    152,129,32,32,136,176,176,157,129,32,189,177
                        88B0B09D81208DB1
01050 52F1 B0872082             DEFB    176,135,32,130,131,171,150,131,32,164,176,186
                        83AB968320A4B0BA
01060 52FD 86208AB5             DEFB    134,32,138,181,176,186,133,32,32,32,137,146
                        B0BA8520202089 92
01070 5309 8481                 DEFB    132,129
01080 530B 0A0A                 DEFB    10,10
01090 530D 20202020             DEFM    '                       Screen Dump Utility is INSTALLED!'
                        2020202020202020202020053637265
                        656E2044756D702055574696C69747920
                        697320494E5354414C4C454421
01100 533E 0A                   DEFB    10
01110 533F 20202020             DEFM    '                       Press the "456" keys simultane....'
                        2020202020202020202020572657373
                        2074686520234353622206865797320
                        736960756C74616E656F75736C79
01120 5371 0A                   DEFB    10
01130 5372 20202020             DEFM    '                             and follow the prompts.'
```

```
                20202020202020202020202020202020
                20616E6420666F6C6C6F772074686520
                70726F6D7074732E
01140 539E 0A            DEFB   10
01150 539F 20202020      DEFM   '                    (created 1984 by Ron A. Zajac)'
                20202020202020202020202020286372
                6561746564203139383420627920526F
                6E20412E205A616A616329
01160 53CE 0A0D          DEFB   10,13
01170
01180
01190 53D0         OFST   EQU    $
01200 53D0         DMPSCN EQU    $
01210
01220 53D0 0000          DEFW   0          ;Used by Enqueue ("forward pointer")
01230 53D2 01            DEFB   1          ;25ms between invocations
01240 53D3 01            DEFB   1          ;Count-down byte
01250
01260 53D4 F5            PUSH   AF
01270 53D5 3A1038        LD     A,(3810H)       ;"01234567" Keyboard Matrix
01280 53D8 EE70          XOR    70H             ;Test "456"
01290 53DA 2802          JR     Z,DUMPIT        ;If Not Zero, Forget it
01300 53DC F1            POP    AF
01310 53DD C9            RET
01320
01330 53DE C5      DUMPIT PUSH  BC
01340 53DF D5            PUSH   DE
01350 53E0 E5            PUSH   HL              ;Save Program Status
01360
01370 53E1 21C03F        LD     HL,3FC0H        ;Last Line of CRT
01380 53E4 110402  PT1   LD     DE,LSTBUF-OFST  ;Last Line Buffer
01390 53E7 014000        LD     BC,40H          ;Length of CRT Line
01400 53EA E5            PUSH   HL
01410 53EB C5            PUSH   BC
01420 53EC EDB0          LDIR                   ;Hold on to it
01430 53EE C1            POP    BC
01440 53EF E1            POP    HL
01450 53F0 EB            EX     DE,HL           ;Set up for prompt line
01460 53F1 EDB0          LDIR
01470
01480 53F3 016400        LD     BC,100
01490 53F6 CD6000        CALL   0060H           ;Delay for Keybounce
01500
01510 53F9 21E93F        LD     HL,TYPEIN-TPPRMT+3FC0H  ;CRT char. insert loc.
01520 53FC 0630          LD     B,30H           ;Cursor Blink Count
01530 53FE 0E16          LD     C,ERRMES-TYPEIN-1     ;C <-- MAX char. entry
01540 5400 1E00          LD     E,0             ;E <-- character entry count
01550
01560 5402 E5            PUSH   HL              ;Hold CRT position
01570
01580 5403 CD5B03  KEYLOP CALL  INKEY           ;Get key
01590 5406 B7            OR     A
01600 5407 282C          JR     Z,CONT
01610 5409 FE08          CP     8               ;Left arrow
01620 540B 281D          JR     Z,BACKSP
01630 540D FE0D          CP     13              ;ENTER key
01640 540F 2836          JR     Z,DEXIT
01650 5411 F5            PUSH   AF
01660 5412 7B            LD     A,E
01670 5413 B9            CP     C
01680 5414 281E          JR     Z,CONT-1        ;Pop AF before continuing
01690 5416 F1            POP    AF
01700 5417 FE2E          CP     '.'
01710 5419 381A          JR     C,CONT
01720 541B FE7B          CP     'z'+1
01730 541D 3016          JR     NC,CONT
01740 541F FE61          CP     'a'
01750 5421 3802          JR     C,PROC
01760 5423 E65F          AND    5FH             ;Mask bit 5 to UPPER CASE
01770
01780 5425 77      PROC  LD     (HL),A
01790 5426 23            INC    HL
01800 5427 1C            INC    E
01810 5428 1808          JR     CONT
01820
01830 542A 7B      BACKSP LD    A,E
01840 542B B7            OR     A
01850 542C 2807          JR     Z,CONT
01860 542E 1D            DEC    E
01870
01880 542F 3620          LD     (HL),' '
01890 5431 2B            DEC    HL
01900 5432 1801          JR     CONT
01910
01920 5434 F1            POP    AF
01930
01940 5435 10CC    CONT  DJNZ   KEYLOP
01950
01960 5437 7E            LD     A,(HL)
01970 5438 FEB6          CP     0B6H
01980 543A 2804          JR     Z,NUNDER
01990 543C 3EB6          LD     A,0B6H          ;Cursor character
02000 543E 1802          JR     INSTAL
02010 5440 3E89    NUNDER LD    A,089H          ;Flip pixels
02020 5442 77      INSTAL LD    (HL),A
02030 5443 0630          LD     B,30H
02040 5445 18BC          JR     KEYLOP
02050
02060 5447         DEXIT  EQU   $
02070 5447 E1            POP    HL              ;Get back CRT position
02080 5448 7B            LD     A,E
02090 5449 B7            OR     A               ;Test for no characters
02100 544A 2858          JR     Z,NOFILE        ;No filespec — EXIT w/o Disk
02110 544C 4F            LD     C,A
02120 544D 0600          LD     B,0             ;BC <-- Characters to move
02130 544F 11E400  PT2   LD     DE,DCB-OFST
02140 5452 C5            PUSH   BC
02150 5453 D5            PUSH   DE              ;Hold onto DCB
02160 5454 EDB0          LDIR
02170 5456 EB            EX     DE,HL
02180 5457 3603          LD     (HL),3          ;Posit Terminator
02190 5459 210402  PT3   LD     HL,LSTBUF-OFST
02200 545C 11C03F        LD     DE,3FC0H
02210 545F 014000        LD     BC,40H          ;Set up to restore bottom line
02220 5462 EDB0          LDIR                   ;Restore it
02230 5464 D1            POP    DE              ;Restore DCB
02240 5465 C1            POP    BC              ;Restore B=0
02250 5466 210401  PT4   LD     HL,BUFFER-OFST
02260 5469 CD2044        CALL   4420H           ;Open New or Existing File
02270 546C 2814          JR     Z,NOERR
02280
02290 546E 210402  ERROR LD     HL,ERRMES-OFST
02300 5471 11C03F        LD     DE,3FC0H
02310 5474 014000        LD     BC,40H          ;Set up Error Message
02320 5477 EDB0          LDIR                   ;Move it in
02330
02340 5479 CD5B03  ERRLOP CALL  INKEY
02350 547C FE0D          CP     13
02360 547E 20F9          JR     NZ,ERRLOP       ;If no <ENTER> keep looking
02370 5480 1822          JR     NOFILE
02380
02390 5482 21003C  NOERR LD     HL,SCREEN       ;Set up for dump
02400 5485 11E400  PT6   LD     DE,DCB-OFST
02410 5488 0E10          LD     C,10H           ;CRT Lines to write
02420
02430 548A 0640    LINLP1 LD    B,40H           ;Line Characters to write
02440
02450 548C 7E      LINLP2 LD    A,(HL)
02460 548D 23            INC    HL
02470 548E CD1B00        CALL   001BH           ;Write Byte to Disk File
02480
02490 5491 10F9          DJNZ   LINLP2
02500
02510 5493 3E0D          LD     A,13
02520 5495 CD1B00        CALL   001BH           ;Write Carriage Return to File
02530
02540 5498 0D            DEC    C
02550 5499 20EF          JR     NZ,LINLP1       ;Set up for next line
02560
02570 549B AF            XOR    A               ;Zero ACC
02580 549C CD1B00        CALL   001BH           ;Write Zero to File
02590
02600 549F CD2844        CALL   4428H           ;Close Dump File
02610 54A2 20CA          JR     NZ,ERROR
02620
02630 54A4 210402  NOFILE LD    HL,LSTBUF-OFST
02640 54A7 11C03F        LD     DE,3FC0H
02650 54AA 014000        LD     BC,40H          ;Set up to restore bottom line
02660 54AD EDB0          LDIR                   ;Restore it
02670 54AF E1            POP    HL
02680 54B0 D1            POP    DE
02690 54B1 C1            POP    BC
```

```
5482 F1                 POP   AF            ;Get back status

5483 C9                 RET                 ;Return to task

5484            DCB     DEFS  20H
5404            BUFFER  DEFS  100H
5504 20202020   LSTBUF  DEFM  '
     20202020202020202020202020202020
     202020202020202020202020
55F4 20202020           DEFM  '
     20202020202020202020202020202020
     202020202020202020202020
5614 20202044   TPPRMT  DEFM  '   Dump Filespec ( <ENTER> to abort ) ==>'
     756D702046696C6573706563202820 3C
     454E5445523E20746F2061626F727420
     29203D3D3E
5630 20202020   TYPEIN  DEFM  '
     20202020202020202020202020202020
     202020
5654 20202044   ERRMES  DEFM  '   DISK ERROR:  Press <ENTER> to'
     49534B204552524F523A202050726573
     73203C454E5445523E20746F
5674 20636F6E           DEFM  ' continue . . .
     74696E7565202E202E202E2020202020
     202020202020202020202020

5694            ENDING  EQU   $
5200                    END   LDSTRT
```

S Y M B O L   T A B L E

INKEY	035B	SCREEN	3C00	HIMEM1	4049	HIMEM3	4411	ENQUE1	4410
ENQUE3	447B	LDSTRT	5200	HM1	5215	HM2	521F	ENQ	5254
REPLCE	525D	ENTMSS	526A	OFST	5300	DMPSCN	5300	DUMPIT	530E
PT1	53E4	KEYLOP	5403	PROC	5425	BACKSP	542A	CONT	5435
NUNDER	5440	INSTAL	5442	DEXIT	5447	PT2	544F	PT3	5459
PT4	5466	ERROR	546E	ERRLOP	5479	NOERR	5482	PT6	5485
LINLP1	548A	LINLP2	548C	NOFILE	54A4	DCB	5484	BUFFER	5404
LSTBUF	5504	TPPRMT	5614	TYPEIN	5630	ERRMES	5654	ENDING	5694

ELECTRONIC MAIL REVISITED - This issue I'd like to present information on two new electronic mail services I've heard about. One is a new service in Canada, and the other is a new service from an old company that wants to take on MCI Mail (MCI Mail has already been featured in Northern Bytes Volume 5, Number 2).

I recently received a mailing from Western Union, describing their "EasyLink Instant Mail" service. My first thought was that perhaps they were going to try to compete with MCI Mail. After reading their material, however, I have come to the conclusion that MCI Mail needn't worry too much about the "competition".

Of course, if you were to read only Western Union's mailing, you might come away with a completely different impression. Part of the mailing is in "Question and Answer" format, and one of the Q&A's is a direct attack on MCI Mail. Question # 13 reads as follows:

"Question: What's the difference between EasyLink and MCI Mail?

"Answer: There are several differences. EasyLink message rates are generally more economical. You pay on a usage basis. MCI Mail's minimum rate is $1.00 per correspondence.

"EasyLink service offers a number of features not currently available with MCI Mail. If desired, you may request delivery notification for your outgoing mail. This will provide you with the address to which the message was sent and the actual time of delivery. Another example is EasyLink's Attention Line which permits you to direct your correspondence to a particular individual (e.g. ATTENTION: Mr. John Smith) or to highlight a particular portion of your correspondence.

"EasyLink service provides broader worldwide communication ability. Users in many overseas locations cannot reach you via MCI Mail. Also, with MCI Mail you cannot communicate directly with the majority of domestic U.S. Telex terminals."

(End of quoted material)

Unfortunately, many statements are made in the above paragraphs that might be considered half-truths at best. EasyLink's pricing structure seems to me to be rather difficult to interpret, but here's how I read it: Suppose I want to send a message from my personal computer to someone else's computer. I would have to pay 30 cents per minute at 300 baud (or 45 cents per minute at any other baud rate), BASED ON INPUT TIME, plus 15 cents per connection to EasyLink via WATS line because I don't live in one of the "400 major cities" served by EasyLink. My recipient would also have to pay the 15 cents WATS charge (unless he lives in one of the select major cities), plus 30 cents per minute for "mailbox scan". Since most letters are going to take more than one minute to send and receive at 300 baud, as near as I can tell it would cost 75 cents to send the message, and 75 cents for the recipient to read it (15 cents less in each case if a "local access" number is used, but more if a higher baud rate is used and/or it takes longer than two minutes to either send or read the message). As far as I can tell, Western Union would nearly always cost more than the $1.00 per message charged for an MCI Mail Instant Letter. On top of all of that, once you've used EasyLink for three months and are out of the trial period, there's a $25 per month usage minimum!

EasyLink lets you "request delivery notification for your outgoing mail." So what? You have the same capability with MCI Mail's RECEIPT option. However, Western Union charges 25 cents for each Notification of Delivery, while MCI Mail's receipts are FREE!

As far as the "Attention" line goes, this is really a "ho-hum" feature as far as I am concerned. If you need one, you can simply put one as the first line of the "text" portion of your letter, then skip a line or two and continue with the salutation and body of your letter. I suppose that if MCI Mail were to get many requests for such a feature, they could easily work it into their software - but, since they already give you four address lines to work with, I don't really see much need for the "Attention" line feature.

The Telex service comment is the one that really surprised me. The truth of the matter is that MCI Mail offers Telex delivery to EVERY Telex address in the world, at rates less than Easy Link. Furthermore, EVERY country, with the sole exception of Morocco, can generate messages through Telex addresses to MCI Mailboxes. And yes, you can send domestic Telex through MCI Mail, though if I really needed to communicate with a non-MCI Mail subscriber in a hurry, I would probably telephone, or use MCI Mail to send a Four-Hour or Overnight letter.

It's interesting to see the kind of statements that Western Union is making about MCI Mail in their literature, especially since MCI Mail has only been around for less than a year. The fact that they mention MCI Mail at all indicates to me that they consider the folks at MCI as serious competitors (they do not mention any other electronic mail services in their literature). In any event, I can't see myself switching to EasyLink as long MCI Mail continues the fine service that they now provide. Because I live in a somewhat remote area, I particularly appreciate MCI Mail's free WATS line access and the fact that they do not charge me to read my mail, nor for the time I spend online, but only for the actual sending of mail.

Turning to the Canadian scene, Canada now has an electronic mail service called Envoy 100. This service is offered by Telecom Canada (the Canadian equivalent of AT&T Long Lines in the days before the Bell System breakup). Envoy 100 can be accessed from almost anywhere, through regular phone lines, Telecom Canada's Datapac network, the TWX network, the U.S. Telenet and TYMNET networks, or through other foreign packet switched networks that connect with Datapac international.

Envoy 100 offers your choice of English or French command options. Messages can be delivered in two ways - to a "mailbox" (stored in the system until the recipient logs on and retrieves his messages) or autodelivery (Envoy 100 will dial up a terminal in Canada or the U.S. and deliver the message automatically). In the future, Envoy 100 users will also be able to send messages to Telex stations around the world. In addition, a service called EnvoyPost permits the sending of same-day or next-day hardcopy letters to addresses in the U.S. and Canada.

Envoy 100's rate structure is not as good a deal as MCI Mail's. There are two categories of service - corporate service, for businesses requiring more than one user accreditation, and individual user service for personal use or for businesses that require only a single user accreditation. Corporate users pay $20 per month which includes the organization administrator's account, plus $3 for each additional user I.D. accredited to the system. Individual users pay only $5 per month (there is no monthly charge with MCI Mail).

Usage charges are based upon kilocharacters (1000 characters) which are sent to and from Envoy 100 by the message originator, as well as kilocharacters which are sent to recipients of the message. The sender pays to input the message and also pays for the message to be delivered. The usage charge is 30

cents/kilocharacter for the first 15,000 kc/month/account, 25 cents/kilocharacter for the next 25,000 kc/month/account, and 22 cents/kilocharacter for more than 40,000 kc/month/account. In addition, there is a 5 cents/message/address charge. Confusing? Well, here's an example, straight from Telecom Canada's literature. A 1000 character message would cost the originator 30 cents to input into the system, 30 cents for each copy sent and 5 cents per addressee. Thus, a 1000 character (about 150 words) message sent to one addressee anywhere on the Datapac network would cost 65 cents. If sent to two addresses it would cost $1.00 (30 cents for input and 35 cents for each addressee). Users also pay for all characters required to interact with Envoy 100 (for example, to access or re-read messages).

Note the underlined sentence above - I added the underlining because you might otherwise miss the point that the recipient has to pay to read the message! By contrast, MCI Mail charges the sender of an "instant" letter $1.00 for the first 7500 characters, with no charge to the recipient to read the message, no charge to "interact with the system", no monthly fixed rate (unless you subscribe to their "advanced" service), and so on.

As for EnvoyPost, the rates for that service consist of Envoy 100 usage charges (the old 30 cents per kilocharacter, I assume), plus a flat rate of $1.10 for next day "basic" delivery, or $2.16 for same day "special" delivery. Delivery to the U.S. is also available, the flat rate for that is $1.60.

If you want to save money, you'd better make a habit of reading your electronic mail regularly on the Envoy 100 system, because you get nicked for one-half cent per kilocharacter per day for each message left over five days. You'd have to leave an awful lot of mail waiting for quite some time before you'd accumulate any substantial charge at that rate, but then MCI Mail users don't have to worry about such charges at all.

You might wonder why I keep comparing Envoy 100 with MCI Mail when MCI Mail isn't available in Canada. Well, actually, MCI Mail IS available to Canadians by calling a U.S. access number (see list in Northern Bytes Volume 5, Number 4). However, I am told that MCI Mail has been trying to expand their service to Canada in a more direct way (possibly through use of an 800 INWATS service number good throughout Canada), but has been met with some opposition to this plan. As one MCI Mail representative put it, "the problem is political, not technical."

In the meantime, Canadians interested in receiving more information about Envoy 100 service can call toll-free (800) 267-4747 for more information. Others may contact Telecom Canada at 410 Laurier Avenue West, Box 2410 Station D, Ottawa, Ontario K1P 6H5 - phone (613) 560-3000, TWX: 610-562-1911.

VERFILE/CMD file verification program - This program was written by Laurie Shields of Wingerworth, Chesterfield, England. It is intended to work on all Disk Operating Systems, but it may do funny things under Model III TRSDOS if the Logical Record Length of the file to be verified is something other than 256. To use the program, type:

VERFILE filespec

Where "filespec" is the filename and extension (with optional drive #) of the file to be verified. If the filespec is not specified in the command line, VERFILE will prompt for it. VERFILE will then "read" the indicated file, and report any disk errors that might occur during the read (such as parity errors, missing records, etc.). You will then be prompted for another filespec to verify, and if you hit <ENTER> only, you will exit the program.

The VERFILE program can be run from a DO file to automatically verify more than one file. Such a DO file would look like this:

```
VERFILE<enter>
FILEA<enter>
FILEB<enter>
FILEC<enter>
<enter>
```

In this way, VERFILE could be used to check all important files on a disk with a single command.

If you want to abort execution of VERFILE on an error, change line 85 to JR 4030H. The assembly language listing shown below was created by ZEN, a Z-80 Editor-Assembler-Debugger package written by Mr. Shields and sold by The Alternate Source. If another editor-assembler package is used, it may be necessary to omit the colons after labels in the source code, and to change DB, DW, and DS pseudo-opcodes to DEFB, DEFM, DEFW, and DEFS pseudo-ops as required by the context.

Both the object code version of VERFILE/CMD and an enhanced version of the program, called VERF/CMD, will soon be available on a NEWDOS/80 utility disk sold by The Alternate Source. VERF/CMD will read the directory of a NEWDOS/80 disk, and then verify each and every file on the disk, so that an entire NEWDOS/80 format disk can be verified with a single command.

```
 1              ;VERFILE/S
 2              VIDEO:   EQU  33H
 3              LINEIN:  EQU  40H
 4              CURSOR:  EQU  4020H
 5              OPEN:    EQU  4424H
 6              READ:    EQU  4436H
 7              ERROR:   EQU  4409H
 8              DOS:     EQU  402DH
 9                       ORG  5200H
10              START:   EXEC START
11 5200 21DC52           LD   HL,TITLE
12 5203 CD8152           CALL LINEOUT   ; Display title message
13              ; The program starts by prompting for a filespec.
14              ; If <Enter> keyed by itself then exit to Dos
15              ; otherwise the file is opened and read sequentially.
16              ; Each record number is displayed prior to attempting
17              ; to read the sector and any Dos error, except EOF
18              ; is reported.
19              ; On error or EOF then go back for the next filespec.
20
21 5206 212353  MAIN:    LD   HL,PROMPT
22 5209 CD8152           CALL LINEOUT   ; Display filespec prompt
23
24              ; Get filespec from keyboard
25
26 520C 214953           LD   HL,KEYBUF ; Set up buffer for input
27 520F 0617             LD   B,23      ; maximum = filespec/ext.password:d
28 5211 CD4000           CALL LINEIN    ; Rom routine common Models 1 & 3
29 5214 DA2D40           JP   C,DOS     ; <Break> key
30 5217 04               INC  B         ; test if B is zero
31 5218 05               DEC  B         ; i.e. just <Enter> keyed
32 5219 CA2D40           JP   Z,DOS     ; if so exit
33
34              ; Move filespec to DCB
35
36 521C 011800           LD   BC,24     ; max filespec + 0DH
37 521F 214953           LD   HL,KEYBUF ; prepare to move
38 5222 116154           LD   DE,DCB    ; to the DCB
39 5225 EDB0             LDIR           ; move it !
40
41              ; Set counter to zero and open file
42
43 5227 ED43CE52         LD   (SECTOR),BC ; conveniently BC = 0 after LDIR
44 522B 216153           LD   HL,INBUFF ; Input buffer for file
45 522E 116154           LD   DE,DCB
46 5231 0600             LD   B,0       ; for logical record length = 256
47 5233 CD2444           CALL OPEN
48 5236 202E             JR   NZ,DERROR ; report any errors
49              ; File opened successfully
50
51              ; Display reading message and save cursor location
52
53 5238 213153           LD   HL,READING
54 523B CD8152           CALL LINEOUT
55 523E CD8C52           CALL SAVECURS
56
57              ; This section is executed repetitively until error.
58              ; The cursor is restored and record number displayed.
59
60 5241 CDC552  READLOOP: CALL PUTCURS  ; cursor location restored
61 5244 2ACE52           LD   HL,(SECTOR) ; get sector count
62 5247 23               INC  HL        ; add one
63 5248 22CE52           LD   (SECTOR),HL ; save it and print it
64 524B CD7452           CALL PRINTSECT ; Basic record # = Dos sector + 1
65 524E 3A4038           LD   A,(3840H)
66 5251 CB57             BIT  2,A
67 5253 C22D40           JP   NZ,DOS
68
69 5256 116154           LD   DE,DCB    ; prepare for file read
70 5259 CD3644           CALL READ      ; read record
71 525C 28E3             JR   Z,READLOOP ; OK do it again
72 525E FE1C             CP   28        ; was the error = EOF
73 5260 28A4             JR   Z,MAIN    ; file finished
74 5262 FE1D             CP   29        ; was sector after EOF
```

```
5264 2840            JR   Z,MAIN    ; file finished

                ; Dos error condition reporting

5266 F5       DERROR:  PUSH AF       ; save error code
5267 3E0D              LD   A,0DH    ; print Carriage Return
5269 CD3300            CALL VIDEO
526C F1                POP  AF       ; recover error code
526D F6C0              OR   0C0H     ; set bits 6 & 7 for error message
526F CD0944            CALL ERROR    ; Dos error reporter
5272 1892              JR   MAIN     ; go for next file

                ; Print record number in decimal
                ; HL registers hold number for printing
                ; Conversion done by successive subtraction with
                ; answer stored in BUFFER for printing by LINEOUT

5274 114353   PRINTSECT: LD   DE,BUFFER ; DE is used as pointer
5277 FD21D252          LD   IY,DECIMALS ; so is IY to table of decimals
527B 0E00              LD   C,0       ; C = leading zero suppression flag

                ; Subtraction loop

527D D5       HD1:     PUSH DE        ; save pointer
527E 3E30              LD   A,'0'     ; counting up from Ascii 0
5280 FD5601            LD   D,(IY+1)  ; load DE registers with decimal
5283 FD5E00            LD   E,(IY+0)  ; values pointed to by IY

                ; Now do subtractions until Carry condition

5286 B7       HD2:     OR   A         ; clear Carry flag
5287 ED52              SBC  HL,DE     ; subtract decimal value from HL
5289 3003              JR   C,HD3     ; gone too far !
528B 3C                INC  A         ; keep count of subtractions
528C 18F8              JR   HD2       ; do it again

                ; correct for 1 too many & prepare for next position

528E 19       HD3:     ADD  HL,DE     ; add it back
528F FD23              INC  IY        ; move to next value in DECIMAL
5291 FD23              INC  IY        ; note each value is 2 bytes
5293 FE30              CP   '0'       ; is this answer a "0"
5295 2008              JR   NZ,HD4    ; if not no trouble
5297 0C                INC  C         ; if yes then check if leading zero
5298 0D                DEC  C         ; by testing C = 0 ?
5299 2005              JR   NZ,HD5    ; not leading zero so jump
529B 3E20              LD   A,' '     ; replace "0" by " "
529D 1801              JR   HD5       ; skip altering C to non-zero
529F 4F       HD4:     LD   C,A       ; cancel zero suppression flag

                ; Test if current value in E is 1, i.e. decimal units
                ; as if so then conversion complete, but do not react
                ; immediately as value has to be put in the buffer

52A0 1D       HD5:     DEC  E         ; if E=1 then set Zero flag
52A1 D1                POP  DE        ; recover pointer to buffer
52A2 12                LD   (DE),A    ; store decimal Ascii character
52A3 13                INC  DE        ; move pointer to next position
52A4 20D7              JR   NZ,HD1    ; go back for more subtractions

                ; Nearly finished but if leading zero suppression
                ; still on then answer all blank, so check and if so
                ; move pointer back to end of buffer and put in a "0"

52A6 0C                INC  C         ; leading zero suppr. ?
52A7 0D                DEC  C
52A8 2004              JR   NZ,HD6    ; no so skip next bit
52AA 1B                DEC  DE        ; backstep pointer
52AB 3E30              LD   A,"0"     ; put Ascii 0
52AD 12                LD   (DE),A    ; into end of buffer.

                ; All finished so prepare to print buffer

52AE 214353   HD6:     LD   HL,BUFFER

                ; Drop through to print line of Ascii characters
                ; terminated with binary 0.

52B1 7E       LINEOUT: LD   A,(HL)    ; get character pointed to by HL
52B2 23                INC  HL        ; bump the pointer
52B3 B7                OR   A         ; is it a zero

52B4 C8                RET  Z         ; if so then finished
52B5 D5                PUSH DE        ; save DE as the Rom alters value
52B6 CD3300            CALL VIDEO     ; call Rom output routine
52B9 D1                POP  DE        ; restore DE
52BA 18F5              JR   LINEOUT   ; go back for another character

                ; Routines used to save current cursor location
                ; & restore it before printing record number

52BC E5       SAVECURS: PUSH HL       ; save HL
52BD 2A2040            LD   HL,(CURSOR) ; get @Cursor position
52C0 22D052            LD   (CURSAVE),HL ; save it
52C3 E1                POP  HL         ; restore HL
52C4 C9                RET             ; done

52C5 E5       PUTCURS: PUSH HL         ; save again in reverse
52C6 2AD052            LD   HL,(CURSAVE)
52C9 222040            LD   (CURSOR),HL
52CC E1                POP  HL
52CD C9                RET

52CE 0000     SECTOR:  DW   0          ; Sector count
52D0 0000     CURSAVE: DW   0          ; Cursor save

                ; Table of decimal values
52D2 1027     DECIMALS: DW  10000
52D4 E803               DW  1000
52D6 6400               DW  100
52D8 0A00               DW  10
52DA 0100               DW  1

                ; Messages, 28 = Home cursor
                ;          31 = Clear to end of screen
                ;          13 = CR/LF
                ;          14 = Cursor on
52DC 1C1F2020 TITLE:    DB   28,31,"    V E R F I L E",13
52E0 20202056
52E4 20452052
52E8 20462049
52EC 204C2045
52F0 0D
52F1 2046696C           DB   " File verifying utility",13
52F5 65207665
52F9 72696679
52FD 696E6720
5301 7574696C
5305 69747900
5309 28632920           DB   "(c) 1983  Laurie Shields",13,0
530D 31393833
5311 20204C61
5315 75726965
5319 20536869
531D 656C6473
5321 0D00
5323 0E004669 PROMPT:   DB   14,13,"Filespec : ",0
5327 6C657320
532B 6563203A
532F 2000
5331 52656164 READING:  DB   "Reading Record : ",0
5335 696E6720
5339 5265636F
533D 7264203A
5341 2000

                ; Buffer for decimal output ending with 0
5343 20202020 BUFFER:   DB   "     ",0
5347 2000
                ; other buffers
                KEYBUF:  DS   24        ; keyboard input
                INBUFF:  DS   256       ; Dos input
                DCB:     DS   50        ; Model 3 TRSDOS needs 50 bytes
                         END

Exec Addr 5200
```

BASICOMP

by Dr. Dimitri P. Bertsekas
225 School Street
Belmont, Massachusetts 02178

This program is a more powerful version of the BASIC compiler for Models I and III that I presented in the October, 1982 issue of 80 Micro. Major improvements include four times faster compilation through the use of machine language routines, and essentially full compatibility of the compiler and the BASIC interpreter. Other improvements include expansion of the list of commands supported by the compiler, support of rectangular two dimensional array computations, and the option to call machine language routines from within the compiled code through the USR command. However the program now requires at least 32K of memory and Disk BASIC.

The compiler can translate into pure machine code an ordinary BASIC program containing virtually any type of single precision computations, a fair amount of integer computations and graphics, plus a limited amount of string handling. The machine code will run much faster than its BASIC equivalent (usually between 3 and 20 times faster depending on the nature of the program). Thus you can speed up your BASIC programs a great deal without rewriting them in assembly language.

The machine code generated by the compiler is, I believe, very efficient. It is generally comparable in speed to the one generated by Microsoft's Fortran compiler available from Radio Shack. However matrix computations are compiled more efficiently by this compiler. For example a matrix calculation program (published in my 80 Micro article) when compiled by this compiler runs roughly 11 times faster than in interpreted BASIC but runs only 7 times faster when compiled in Fortran. This is due to a special scheme used for storing two dimensional array addresses. Also the size of the compiled code produced is quite modest. This is due to the extensive use of ROM routines and absence of a separate subroutine library. For example the matrix program mentioned above compiled in 391 bytes but its Fortran equivalent is more than 10000 bytes long.

One of the most powerful features of this compiler is its capability to interface harmoniously with the BASIC interpreter through unlimited exchange of variables. It is possible to compile into machine code only a portion of a BASIC program (presumably the most computation intensive). The remaining uncompiled portion can be merged with the machine code into a single BASIC program and run as usual. Values of any variable computed by the BASIC code can be used by the machine code and vice versa. In this way you get the best of all possible worlds; the convenience and broad command repertoire of the BASIC interpreter together with the speed of pure machine language code.

USING THE COMPILER

Load BASICOMP and if you have 32K of memory change the statement MR=0 in lines 1000 and 6650 to MR=1. As an illustration we will compile the program:

```
10 CLS:FOR I%=15360 TO 16383:POKE I%,191:NEXT
20 A$="PRESS BREAK TO EXIT":PRINT A$;
30 A%=PEEK(14400):IF A%<>4GOTO30:' test for break key
500 END
```

Enter these lines and RUN. Line 10 will execute in interpreted BASIC as usual and will first clear and then white the screen. Execution will stop when you press BREAK and the END statement in line 500 is encountered.

Now type RUN 1000 and press <C>. After a short while the compiler will start translating line 10 into machine code and poke the code in memory starting at location -2000 (or -18384 if you have 32K of memory). It will then translate lines 20, 30, and the END statement in line 500 and stop compiling. The start and end of the compiled code will be displayed on your screen. You then have the option of executing the code directly by pressing ENTER. If you do so the screen will clear and turn white almost instantly. What happened is that statement 10 executed in machine code 63 times faster than in the usual interpreted BASIC mode. Press BREAK, type X=USR(0) and press ENTER. The compiled code will execute again exactly as the first time. You can now save the compiled program on disk using the Dump command and the start and end addresses displayed earlier.

The preceding example illustrates some of the basic rules for using the compiler. These are:

a) The BASIC program to be compiled should be entered or merged together with the compiler in lines ranging from 0 to 500. The command to compile it is RUN 1000. The compiler will successively translate into machine code all BASIC lines with numbers from 0 to 500 and store the machine code somewhere near the top of your memory.

b) An END statement is required at each point where you wish the compiled code to stop execution and return you to the BASIC interpreter. You can run the compiled code as many times as you wish, once it is stored in memory at the place where it was originally compiled, by executing the BASIC command X=USR(0) (the USR address is the start of the compiled code and should be first defined by a DEFUSR statement - for example if -2000 is the start of the compiled code you must first execute DEFUSR=-2000, see lines 1300 and 1340).

c) The program should contain only statements from Table 2. (See the end of the documentation. Table 1 lists the abbreviations used in the descriptions of Table 2). If your program includes some statement that is either not contained in Table 2 or is incorrectly coded the compiler will not recognize it and will warn you with an error message.

d) Multiple statements, remarks and blanks are allowed in each program line but blanks should not appear within a variable name and on either side of an = sign.

e) Only integer, single precision, and string variables are allowed in the program to be compiled. Furthermore their names are restricted as follows. Integer names allowed are A% to Z%. String names allowed are A$ to Z$. Single precision simple variable names allowed are A-Z, A0-Z0, ..., A9-Z9. Single precision one and two dimensional array names allowed are A-Z. No integer and string arrays are accepted by this compiler. Some additional restrictions on the use of variable names dictated by memory management considerations will be explained shortly.

There are a few more rules that you must observe before you can use the compiler to its full potential. These have to do with effective memory management, and interfacing harmoniously compiled code with ordinary BASIC code. We will take these up in sequence.

VARIABLE USE AND MEMORY MANAGEMENT

When the machine code generated by the compiler runs it uses variables similarly as usual BASIC programs and therefore it must store these variables somewhere in memory. In our case it stores the compiled code near the top of memory and stores variables right below the compiled code. Referring to lines 6690 and 6700, integers are stored starting at location VT, single precision simple variables are stored right after integers starting at location VF, then one dimensional arrays starting at VA, two dimensional arrays starting at VD, and finally string variables starting at VN.

Most existing compilers will allocate storage space only for the actual compiled code and only for those variables that you actually use in your program. This compiler differs in that it allocates storage space for compiled code and for some variables whether you use them or not. To avoid a serious potential memory waste it requires you to provide information as to how much space for compiled code and how many variables you need for your program. This is done by modifying the storage parameters CC and SS in line 6650 and IS,DO,DT,DC,SL,NO,NT,NS in line 6660.

More specifically CC is the number of bytes allocated for storage of compiled code. Set this to a close overestimate of the size of the compiled program. If you make a mistake and allocate too little space you will soon find out when the compiler tries to store machine code beyond the limits of your memory. If you allocate too much space your program will be compiled correctly, you will see its actual size, and you will have the chance to adjust the parameter CC and recompile the program should you wish to economize in memory space. The parameter SS sets the start of storage of variables and should always be greater or equal to CC. Ordinarily it should equal CC unless you wish to leave some space between compiled code and variable storage in which to put something else - for example a machine language routine or a second compiled program that shares the same variables with the first.

Returning to the use of variables the compiler will always allocate space for the integer variables A%-Z% and the single precision simple variables A-Z whether you use them or not (a total of 156 bytes). If you set all the parameters in line 6660 to zero you will not be allowed to use any additional variables and

the total amount of storage used other than for the code itself will be the minimum possible 156 for variables, and another 26 bytes for internal use of the compiled program - a total of 182 bytes.

You can tell the compiler that you need to use more variables than the minimum allowed by setting some of the parameters in line 6660 to nonzero values. The parameter IS specifies the number of single precision variables per letter in addition to the letter itself that you can use. If you set for example IS to one then only the variables A-Z, A0-Z0, are allowed, while if you set IS to ten all the variables A-Z, A0-Z0, ..., A9-Z9 are allowed.

The parameters NO and DO specify the number and dimension of one dimensional arrays that you can use. For example if you set NO=26 and DO=10 all the arrays A(.) through Z(.) can be used and their indexes must take values between 0 and 9. If NO=2 and DO=501 only the arrays A(.) and B(.) can be used and their indexes must take values between 0 and 500.

The parameters NT, DT, and DC specify the number of two dimensional arrays that can be used, and their row and column dimension. For example if NT=3, DT=15, and DC=10 only the matrices A(,,), B(,,), and C(,,) can be used and their first and second indexes must take values from 0 to 14, and 0 to 9 respectively.

In both one and two dimensional arrays the indexes should either be an integer within the dimension range of the array (this is checked by the compiler), or one of the integer variables A%-Z%.

For example if NS=5 and SL=40 then only the string variables A$-E$ can be used and each must have length no greater than 40 characters including blanks.

Simple single precision variables and elements of arrays take up 4 bytes of storage each so you can easily go beyond the limits of your memory if you don't control carefully their number. The compiler displays the storage parameters, so you have the chance to change them before starting to compile.

Finally when compiling programs you must protect the memory area where the compiled code and the assembly language routines used by the compiler are stored (see the DATA statements in lines 7380-7620). The rule is that the number of protected bytes should be at least 750 plus the allocated size for compiled code (the value of CC in line 6650). The number of protected bytes is set by adjusting the parameter PC in line 1000. You also have the chance to change this number on line when running the compiler. It is also possible to operate the compiler without memory protection (PC=0 in line 1000) as long as the the compiled code does not extend to the top 300 bytes or so of memory.

When executing the compiled code in conjunction with a BASIC program (see the next section), you must set memory size which is sufficient to protect both the compiled code and the area where it stores its variables. The compiler displays the required memory size at the end of compilation. Note that it is possible to set memory size from within a BASIC program. Line 6620 shows how this is done. The desired memory size is poked into locations 16561 and 16562 and this is followed by a CLEAR statement (any number of string bytes CLEARed will do).

If the preceding discussion seems confusing at first use the original parameters of BASICOMP for variable and code storage. This allows you 2000 bytes of compiled code, the integer variables A%-Z%, the simple single precision variables A-Z, the one dimensional arrays A(.)-C(.) with 20 elements each, and the string variables A$-D$ with maximum length 30 characters. The total size of a compiled program including variables with this parameter setting is 2551 bytes.

COMBINING COMPILED PROGRAMS WITH BASIC PROGRAMS

Because many normal functions of the BASIC interpreter cannot be compiled (disk I/O or PRINT USING for example), it is essential to be able to interface the compiled code harmoniously with ordinary BASIC code. As discussed earlier you can execute the compiled code from BASIC as many times as you want via the USR command. Similarly, by introducing END statements at the points where you wish the compiled code to return to BASIC, you can carry out the reverse process. However this switching back and forth from BASIC to compiled code would be of limited use if you didn't have the means of passing variable values in either direction. For this purpose the two special statements X=0+Y and X=1*Y have been provided. Here X is an integer or single precision variable of the compiled code and Y is an integer or single precision variable of the BASIC code. Note that X and Y must be both variables of the same type (both integer or both single precision).

As an illustration consider the statement A%=0+AB%. When compiled and executed it finds the location where the BASIC interpreter stores the variable AB% and transfers the current value of AB% into the location where the compiled code stores the variable A%. The statement A%=1*AB% performs the reverse process. It transfers the current value of the compiled code variable A% into the BASIC variable AB% (from where it can be picked up for disk I/O or further calculation for example). If the BASIC variable AB% has not yet been established the compiled code creates it automatically. As an example consider the following two lines:

```
10 FOR I%=0 TO 10:I%=1*I%:A(I%)=0+VECTOR(I%):NEXT
20 FOR I%=0 TO 10:I%=1*I1%:FOR J%=0 TO 10:J%=1*J1%:A(I%,J%
)=1*MATRIX(I1%,J1%):NEXT:NEXT
```

When compiled and executed, line 10 transfers the elements 0-10 of the BASIC one dimensional array VECTOR into the corresponding elements of the compiled code array A. Line 20 transfers the specified elements of the compiled code array A into the corresponding elements of the BASIC array MATRIX. If MATRIX or the integer variables I1% and J1% have not yet been defined they will be created.

If the names of X and Y are identical, the statements X=0+Y and X=1*Y when executed in BASIC while you are debugging will not interfere with the program's operation.

Let's discuss now the process of writing a combined BASIC and compiled code program. First, write your program in BASIC, test it and debug it as usual. During this stage try to use statements and variables that the compiler will accept, and structure the program so it requires few modifications later. Then decide which part of your program you are going to compile. Isolate that part and introduce interfaces for exchanging variables with the remaining parts using the statements X=0+Y and X=1*Y described earlier. Duplicate any subroutines used by both parts since the compiled code cannot call BASIC subroutines.

Renumber the part to be compiled so its line numbers are in the range 0-500 while the remaining part has numbers 10000 or higher. Test the program again by introducing GOTO statements to make sure it works after renumbering. Introduce a USR command at each point where you want the BASIC program to switch to the compiled code, and an END statement at each point where you wish the compiled code to switch to BASIC. At this point you should have a program such as the one in the Sample Listing (see the end of the documentation). This is a program that randomly generates N% numbers (lines 10020-10030), and calls the compiled code (line 10050) to compute their average. The compiled code first obtains from BASIC the value of N% (line 10), then obtains successively from BASIC all the numbers and gradually builds up their average A (lines 20-70). The value of A is passed to BASIC (line 80) and the compiled code returns to BASIC (line 500). BASIC now prints the average (line 70) and goes back to line 10020 to start again. Line 10010 sets memory size to protect the compiled code, and sets up the USR call.

Once you have written a combined program such as the one in the Sample Listing you should save it in ASCII on disk and load BASICOMP. Then merge the combined program with the compiler, adjust the storage parameters in lines 6650 and 6660 if necessary and RUN 1000. After compilation is completed you should dump the compiled code on disk and BREAK (or BREAK first and then Dump the code on disk if you don't use NEWDOS80). Then set memory size and set up the USR call (compare with line 10010), delete all lines up to 10000 if you wish, and your combined program can be executed via RUN 10000.

Once you have the combined program in its final form you may wish to translate the compiled code into DATA statements and merge it with the BASIC code in a single program. 80 Micro has published an excellent program for this purpose (Datagen, August 1981). Otherwise you must load the compiled code in memory via the DOS LOAD command before you can run the combined program.

TROUBLESHOOTING

The compiler checks for most syntax errors during compilation. However some errors can remain undetected until you find out the hard way during execution time.

If the compiled program runs but gives different results than its equivalent BASIC program the most likely problem is that some compiled code variable appears in the right hand side of an assignment statement without being initialized first (remember the BASIC interpreter automatically initializes all variables to zero but the compiler doesn't). Also the compiler does not check whether the result of an integer LET expression lies within the legal range (-32767 to 32767), another source of errors or program crashes depending on how the results are used later.

If the compiled program crashes check the For...Next loops; make sure they are set up properly (each For is matched by a Next) and that there is no jump out of the range of a For...Next loop.

Finally there are three more compiler parameters that you will rarely if ever need to adjust. These are the dimensions of the arrays L1, L2, and A in line 6630. The dimension of L1 and L2 is the maximum number of lines that can be compiled, and the dimension of A is the maximum numbers of jumps (GOTOs and GOSUBs) that can be compiled. Listing 1 allows 100 program lines and 50 jump statements. If you try to compile a program that is too long or has too many jumps, compilation will stop with a "subscript out of range" error. In that case you must increase appropriately the dimensions of L1 and L2 or A.

MODIFICATIONS

The compiler takes a lot of memory space which you may need if you want to compile a long program. One way to save memory is to delete all remarks. In fact this is essential if you have less than 48K of memory. If you don't need to see the current parameter settings before compiling delete lines 6710-6720. If you don't need all of the compiler's capabilities you can delete the corresponding lines and save space. For example if you don't need OUT delete line 1185 and lines 5500-5550. If you don't need SET, RESET, and POINT delete lines 2030, 6000-6110. If you don't need FIX, CINT, SGN and RND delete lines 2021-2024 etc. Another memory saving device is to write the DATA statements in lines 7380-7620 in a disk file and read instead the DATA in line 7200 from that file.

The option to Dump the compiled code to disk directly from the compiler (see line 1335) works for the NEWDOS operating system for both Models I and III. If you have TRSDOS you will have to forego this option and use the Dump command as described in your TRSDOS manual. You may, if you wish, then delete line 1335 and the portion "ANDA%<>68" in line 1330, and modify the Print statements in line 1320. If you have a different operating system that allows you to execute the Dump command directly from BASIC you should modify appropriately line 1335.

As an aid in understanding the program note that the functions of the four machine language routines used by the compiler are as follows:

USR0: Pokes P into memory location M and increments M.
USR1: Sets B to the next nonspace byte of the BASIC program.
USR2: Corrects the addresses of the compiled code jumps.
USR3: Sets E1 and D1 to the least and most significant bytes of the variable C respectively.

TABLE 1: Definitions and abbreviations

Integer Variable (IV): One of the variables A%-Z%

Single Precision Variable (SPV): A simple variable A-Z, A0-Z0, A9-Z9 or a one or two-dimensional array element

String Variable (SV): One of the variables A$-Z$

Constant (C): Any integer or decimal number

Possible Integer (PI): Any integer in the range -32767 to 32767

One Byte Integer (OBI): Any integer in the range 0-255

String (S): Any sequence of blanks or printable characters enclosed in quotation marks (the right quotation mark is optional if the string lies at the end of the line)

Integer Expression (IE): A sequence of the form X1sX2sX3... where X1, X2, X3,... is a positive integer less than or equal to 32767 or an integer variable, and s is plus or minus. The sequence may begin with a minus sign but not with a plus sign or a zero followed by a plus sign. Parentheses are not allowed (and not needed). The compiler evaluates integer expressions in the same way as the interpreter but if the result is outside the range -32767 to 32767 the compiled code will not indicate an error

Single Precision Expression (SPE): An arbitrarily parenthesized legal BASIC expression involving constants, single precision or integer variables, the operators +, -, *, /, ^, and the functions RND(0), SQR, ABS, LOG, EXP, COS, SIN, TAN, ATN. Here are some examples of SPE's:
a) -(1+SQR(1.2*A% + SIN(A(I%,J%)*2.5)))
b) LOG(EXP(B0+C)/2.3+.1)*B%
c) (-1.2+3.4)
d) (I%)

TABLE 2: Program statements that can be compiled
(Expressions in brackets are optional)

LET: There are several types of LET (assignment) statements. In each case the LET keyword is optional
1) X=Y (Integer LET)
X:IV
Y:IE
2)X=Y (Single Precision LET)
X:SPV
Y:SPE
3)X=Y (String LET)
X:SV
Y:S or CHR$(Z1)[+ CHR$(Z2)[+CHR$(Z3)[+ ... where Z1, Z2, ... are IVs or OBIs

PRINT: There are several types of PRINT statements:
1) PRINT (line feed and carriage return)
2) PRINT X1[;X2[;X3[;...
X1,X2,X3,...: SPEs or SVs or Ss
3) PRINT @ X,Y1[;Y2[;Y3[;...
X:PI in the range 0-1023 or IV
Y1,Y2,Y3,...: SPEs or IVs or Ss

LPRINT: Same format as PRINT statements 1) and 2) above.

IF...THEN...ELSE: The format of this statement is:
IF X rel Y THEN (line #)[:][ELSE...]
where
rel is any of =,<,>,<=,=<,>=,=>,<>,><
X is an IV or a SPE that does not begin with an IV
Y is a SPE, but if X is an IV then Y must be an IV or a PI
THEN can be replaced by GOTO or by THEN GOTO
ELSE can be followed by any other legal statements including another IF...THEN statement
Note: If X is an integer variable and Y is a SPE which is not an IV or a PI the compiler will give an error. If you compare an IV with a SPE other than the above type, code the statement as in (I%)relSPE or SPErelI%. Note that an IF...THEN statement involving integers will execute much faster than one involving SPEs.

GOTO (line #):

GOSUB (line #):

INPUT X:
X:IV or SPV. The constant keyed in should have no more than six digits. If X is an IV, it is not necessary that the constant keyed in is an integer. The code automatically truncates it to an integer. If the absolute value of the constant is larger than 32767 and X is an IV a fatal error will occur.

POKE X,Y:
X:IV or PI
Y:IV or OBI

SET(X,Y);RESET(X,Y):
X,Y:IV or OBI in the legal range of the SET and RESET commands.

CLS:

REM (or '):
The compiler skips over statements that begin with REM or '.

END:
It is mandatory to have an END statement at each point where you wish the USR to return to BASIC. This statement is translated as a Return to the point where the USR was called.

FOR X=Y1 TO Y2:
X:IV
Y1,Y2:IV or PI
Note: Y1 must have a value less than or equal to that of Y2 and the STEP option is not supported. If Y1 has a larger value than Y2, the program will get caught in an infinite loop, give incorrect results, or hang up. The compiler does not check for this error.

NEXT [X]:
X: IV
Notes: An unlimited number of FOR...NEXT loops is allowed but the loops must be matched (each FOR is matched by a NEXT). It is not legal to jump to a line outside the range of a FOR...NEXT loop from a line inside the loop. Code that violates this rule will be compiled with no indication of error but will very likely crash upon execution.

X=FN(Y):
X: IV
Y: SPE
FN: Any of INT, SGN, FIX, CINT

X=RND(Y):
X: IV
Y: Integer between 1 and 32767

X=PEEK(Y):
X: IV
Y: PI or IV

X=POINT(Z1,Z2):
X: IV
Z1, Z2: IV or OBI

X=USR(Y):
X: IV
Y: IV or PI
Notes: This statement calls a machine language routine with entry point Y. The integer variable X is dummy and its value is not affected by execution of this statement. This function works differently in BASIC (see the Level II manual).

X=VARPTR(Y):
X: IV
Y: Name of a BASIC integer, single precision, double precision, or string variable.

X=ASC(Y):
X: IV
Y: S
Note: When this function is executed X equals the ASCII value of the first character of the string Y. If Y="" then X will equal zero. By contrast in ordinary BASIC ASC("") gives an error.

X=INKEY$:
X=: S
Notes: The string X will equal the character of the key pressed while the statement was executed. If no key was pressed then X="". By contrast BASIC returns the character of the key latest pressed (see the Level II manual). This function together with the ASC function is useful in a program line such as:
10 A$=INKEY$:A%=ASC(A$):IF A%=0 THEN 10
The statement above will loop until a key is pressed. At that time A% will equal the ASCII value of the key.

OUT X,Y:
X: IV or OBI
Y: IV or OBI

SAMPLE LISTING

```
5 ' LINES 10-500 ARE A SAMPLE PROGRAM READY TO BE COMP
ILED AND RUN IN CONJUCTION WITH THE BASIC PROGRAM ST
ARTING AT LINE 10000
10 N%=0+N%' GET VALUE OF N% FROM BASIC
20 S=0:P%=N%-1
30 FOR I%=0 TO P%
40 I%=1*I%:N=0+ARRAY(I%)
50 S=S+N
60 NEXT
70 A=S/N% ' COMPUTE THE AVERAGE
80 A=1*AV:' PASS AVERAGE TO BASIC
500 END:' RETURN TO BASIC
10000 ' SAMPLE COMBINED PROGRAM THAT COMPUTES THE AV
ERAGE OF A SET OF RANDOMLY GENERATED NUMBERS
10010 MR=0:HS=-2551-MR*16384+65535:POKE 16562,HS/256:POK
E 16561,HS-INT(HS/256)*256:CLEAR 100:MR=0:DEFUSR=-2000-M
R*16384:' SET MR=1 FOR A 32K SYSTEM; THIS LINE SETS MEMO
RY SIZE AND SETS UP THE USR CALL
10020 DIM ARRAY(19)
10030 INPUT "HOW MANY NUMBERS (1 TO 20)";N%:FOR I=0 TO N
%-1:ARRAY(I)=RND(0):NEXT:S=0:FOR I=0 TO N%-1:S=S+ARRAY(I)
:NEXT:PRINT "AVERAGE =";S/N%
10040 PRINT "PRESS <ENTER> TO RUN THE COMPILED CODE":G
OSUB 11000
10050 X=USR(0)
10060 PRINT "AVERAGE COMPUTED BY THE COMPILED CODE ="
;AV
10070 PRINT STRING$(64,176):GOTO 10030
11000 A$=INKEY$
11010 A$=INKEY$:IF A$="" GOTO 11010 ELSE RETURN
```

END OF SAMPLE LISTING

MAIN PROGRAM

```
500 END
501 '*********** BASIC COMPILER VERSION 2 **********
*
502 '**      BY D. P. BERTSEKAS, BELMONT, MASS.
510 '************** AUXILIARY ROUTINES ****************
*
515 IFB<65ORB>90THEN517ELSERETURN
516 IFE<>37THEN517ELSEQ=Q+1:RETURN
517 PRINT:PRINT"ERROR LINE #"L1(L-1):END
518 PRINT:PRINTTAB(20)F$L1(L):RETURN
519 FORI=1TO64:PRINTCHR$(140);:NEXT:RETURN
520 IFA!>32767THENA%=A!-65536ELSEA%=A!
521 C$=FNH4$(A%):RETURN
522 A$=INKEY$
523 A$=INKEY$:IFA$=""THEN523ELSEA%=ASC(A$):RETURN
529 ' ******** SINGLE PRECISION ASSIGNMENT ROUTINES ****
****
530 ' ROUTINE TO FIND VAR. ADDRESS PARAMETERS
535 V1=B-65:IFE<48ANDE>57ANDE<>213ANDE<>40ANDCF<>1TH
EN517
538 IFE>47ANDE<58MI=E-47:X=USR1(0)ELSEMI=0
540 IFE=213OR(E<>40ANDCF=1)THENZ1=1:RETURN:' EXIT IF NEX
T BYTE IS AN = SIGN OR IS NOT A PARENTHESIS AND THIS IS A
N INPUT STATEMENT
545 IFE=40F2=0:F3=0:X=USR1(0):X=USR1(0):GOSUB804:IFC$<>""T
HENV2=CELSEIFB>64ANDB<91THENF2=1:V2=B-65:GOSUB516:X=
USR1(0)ELSEIFCF=1Z1=1:RETURN:ELSEGOTO517
550 IFB<>41ANDB<>44THEN517
555 IFB=41THENIF(F2=0AND(V2<0ORV2>=DO))OR(V1<0ORV1>=NO
)GOTO517ELSEZ1=2:RETURN: ' 1-D ARRAY
560 X=USR1(0):GOSUB804:IFC$<>""THENV3=C:ELSEF3=1:V3=B-65
:GOSUB516:X=USR1(0)
562 IFB<>41OR(F2=0AND(V2<0ORV2>=DT))OR(F3=0AND(V3<0ORV
3>=DC))OR(V1<0ORV1>=NT)GOTO517ELSEZ1=3:RETURN: ' 2-D AR
RAY
564 ' ADDRESS COMPUTATION ROUTINE
565 ONZ1GOSUB570,575,580:RETURN
570 IFMI>ISTHEN517ELSEC=VF+(V1+MI*26)*4:X=USR3(0):GOSUB9
02:RETURN
575 V7=V1:V8=V2:F8=F2:GOSUB610:RETURN
580 V7=V1:V8=V2:V9=V3:F8=F2:F9=F3:GOSUB620:RETURN
```

```
600 ' ARRAY PORTION OF ADDRESS ROUTINE
610 IFF8THENV0=V8:GOSUB912ELSEC=V8:X=USR3(0):GOSUB902
615 GOSUB906:C=VA+V7*DO*4:X=USR3(0):GOSUB900:GOSUB904:
RETURN:' 1-D ARRAY
620 IFF9THENV0=V9:GOSUB912ELSEC=V9:X=USR3(0):GOSUB902
625 P=41:X=USR(0):C=VD+4*NT*DT*DC+2*V7*DC:X=USR3(0):GOS
UB900:GOSUB904:P=94:X=USR(0):P=35:X=USR(0):P=86:X=USR(0)
630 IFF8=1THENV0=V8:GOSUB912ELSEC=V8:X=USR3(0):GOSUB90
2
635 GOSUB906:GOSUB904:RETURN:' 2-D ARRAY
699 ' ROUTINE TO EVALUATE SINGLE PRECISION EXPRESSIONS
700 X=USR1(0):IFB=205X=USR1(0):GOTO704ELSEIFB<>206GOTO70
4
702 E1=0:D1=0:GOSUB902:GOSUB926:GOSUB926:GOTO714:'TAKES
 CARE OF LEADING <-> SIGN
704 GOSUB730
706 X=USR1(0):IFB=0ORB=41ORB=58ORB=59ORB=212ORB=213OR
B=214ORB=141ORB=202THENRETURN:'PEEK NEXT BYTE; IF TER
MINATOR RETURN
708 GOSUB934:'MOVE INTERIM RESULT FROM 4121H STORAGE
 AREA TO STACK
710 ' MOVE NEW VARIABLE TO 4121H AREA; POP BCDE; OPERA
TE
712 IFB=205X=USR1(0):GOSUB730:GOSUB936:GOSUB940:GOTO706
:'ADD
714 IFB=206X=USR1(0):GOSUB730:GOSUB936:GOSUB942:GOTO706
:'SUBTRACT
716 IFB=207X=USR1(0):GOSUB730:GOSUB936:GOSUB944:GOTO706
:'MULTIPLY
718 IFB=208X=USR1(0):GOSUB730:GOSUB936:GOSUB946:GOTO706
:'DIVIDE
720 IFB=209X=USR1(0):GOSUB730:GOSUB936:GOSUB948:GOTO706
:'EXPONENTIATE
722 GOTO517:'ERROR TRAP
729 ' ROUTINE TO EVALUATE CONSTANTS, VARIABLES & FUNC
TIONS IN SINGLE PRECISION EXPRESSION & MOVE THEM TO 41
21H STORAGE AREA
730 IF(B<58ANDB>47)ORB=46GOSUB822:RETURN:'CONVERT CON
STANT TO 4-BYTE REPRESENTATION; MOVE IT & RETURN
732 IFB=222X=USR1(0):IFB<>40THEN517ELSEX=USR1(0):IFB<>46T
HEN517ELSEX=USR1(0):IFB<>41THEN517ELSEP=205:X=USR(0):P=2
40:X=USR(0):P=20:X=USR(0):RETURN:'RND(0)
734 IF(B>220ANDB<229)ORB=217ORB=40THEN756ELSEIFB<65OR
B>90THEN517:'IF FUNCTION GOTO 756
736 V4=B-65:IFE>47ANDE<58THENME=E-47:X=USR1(0):Z2=1:GOT
O746ELSEIFE<>40ANDE<>37THENME=0:Z2=1:GOTO746ELSEIFE=3
7THENZ2=4:X=USR1(0):GOTO746
738 F5=0:F6=0:X=USR1(0):X=USR1(0):GOSUB804:IFC$<>""THENV5
=C:ELSEF5=1:GOSUB515:V5=B-65:GOSUB516:X=USR1(0)
740 IFB<>41ANDB<>44THEN517
742 IFB=41THENIF(F5=0AND(V5<0ORV5>=DO))OR(V4<0ORV4>=NO
)GOTO517ELSEZ2=2:GOTO746: ' 1-D ARRAY
743 X=USR1(0):GOSUB804:IFC$<>""THENV6=C:ELSEF6=1:GOSUB5
15:V6=B-65:GOSUB516:X=USR1(0)
744 IFB<>41OR(F5=0AND(V5<0ORV5>=DT))OR(F6=0AND(V6<0ORV
6>=DC))OR(V4<0ORV4>=NT)GOTO517ELSEZ2=3:' 2-D ARRAY
746 ONZ2GOTO748,750,752,754
748 IFME>ISTHEN517ELSEC=VF+(V4+ME*26)*4:X=USR3(0):GOSUB
902:GOSUB932:RETURN
750 V7=V4:V8=V5:F8=F5:GOSUB610:GOSUB932:RETURN
752 V7=V4:V8=V5:V9=V6:F8=F5:F9=F6:GOSUB620:GOSUB932:RET
URN
754 V0=V4:GOSUB912:P=34:X=USR(0):P=33:X=USR(0):P=65:X=USR(
0):P=205:X=USR(0):P=204:X=USR(0):P=10:X=USR(0):RETURN:' CON
VERT INTEGER VAR. TO SINGLE PRECISION
756 IFB=40THENGOSUB700:GOTO794:'PARENTHESIS
758 Q=Q+1
760 IFB=221THENGOSUB700:GOSUB952:'SQR
762 IFB=217THENGOSUB700:GOSUB954:'ABS
764 IFB=223THENGOSUB700:GOSUB956:'LOG
766 IFB=224THENGOSUB700:GOSUB958:'EXP
768 IFB=225THENGOSUB700:GOSUB960:'COS
770 IFB=226THENGOSUB700:GOSUB962:'SIN
772 IFB=227THENGOSUB700:GOSUB964:'TAN
774 IFB=228THENGOSUB700:GOSUB966:'ATN
794 IFB<>41THEN517ELSERETURN
800 ' ************** CONVERSION ROUTINES ************
802 ' ROUTINE TO FIND LSB & MSB OF INTEGER NUMERIC STRI
NG
804 C$="":IFB=206THENB=45:GOSUB812ELSEGOSUB810:IFC$=""R
ETURN
806 C=VAL(C$):X=USR3(0):RETURN
810 IFB<48ORB>57THENRETURN
812 C$=C$+CHR$(B):X=USR1(0):GOTO810
814 ' ROUTINES TO FIND LSB & MSB OF ADDRESSES ABOVE 1ST
32K
816 Z=VARPTR(C):E1=PEEK(Z):D1=PEEK(Z+1):RETURN
818 C=VT+V1+V1:X=USR3(0):RETURN
820 ' ROUTINE TO CONVERT NUMERIC STRING TO 4-BYTE SING
LE PRECISION REPRESENTATION
822 C$=CHR$(B)
824 X=USR1(0):IF(B<58ANDB>47)ORB=46C$=C$+CHR$(B):GOTO824
826 R!=VAL(C$):Z=VARPTR(R!):E1=33:D1=65:GOSUB902:H=PEEK(Z):
GOSUB910:P=35:X=USR(0):H=PEEK(Z+1):GOSUB910:P=35:X=USR(0):
H=PEEK(Z+2):GOSUB910:P=35:X=USR(0):H=PEEK(Z+3):GOSUB910:Q
=Q-1:RETURN
828 ' ROUTINE TO PARSE SPE
829 Q1=Q:W=MC-625
830 A=PEEK(Q1):POKEW,A:IFA<>0ANDA<>58ANDA<>59AND(A<21
2ORA>214)ANDA<>202ANDA<>141Q1=Q1+1:W=W+1:GOTO830
831 PT=A:Q2=Q1+1:Q=MC-625:RS=Q+1
832 ' RESTRUCTURING THE EXPRESSION BY USING PARENTHES
ES
833 OP=0:FORI=RSTOW-1:A=PEEK(I):IFA=209OP=I:I=W-1
834 NEXT:IFOP=0THEN840
835 PR=0:R1=0:FORI=OP-1TOQSTEP-1:A=PEEK(I)
836 IFA=41PR=PR+1
837 IFA=40PR=PR-1:IFPR=-1I=Q:GOTO839
838 IFA>204ANDA<209ANDPR=0R1=I:I=Q
839 NEXT:IFR1=0RS=OP+1:GOTO833ELSEGOSUB850:RS=OP+1:GO
TO833
840 RS=Q+1
841 OP=0:FORI=RSTOW-1:A=PEEK(I):IFA=207ORA=208OP=I:I=W-1
842 NEXT:IFOP=0THENRETURN
843 PR=0:R1=0:FORI=OP-1TOQSTEP-1:A=PEEK(I)
844 IFA=41PR=PR+1
845 IFA=40PR=PR-1:IFPR=-1I=Q:GOTO847
846 IFA>204ANDA<207ANDPR=0R1=I:I=Q
847 NEXT:IFR1=0RS=OP+1:GOTO841ELSEGOSUB850:RS=OP+1:GO
TO841
850 PR=0:R2=W:FORI=OP+1TOW-1:A=PEEK(I):IFA=40PR=PR+1
852 IFA=41PR=PR-1:IFPR=-1R2=I:I=W-1
854 IFA>204ANDA<207ANDPR=0R2=I:I=W-1
856 NEXT:GOSUB864:RETURN
864 FORI=QTOR1:POKEI-1,PEEK(I):NEXT:POKER1,40:Q=Q-1:FORI
=WTOR2STEP-1:POKEI+1,PEEK(I):NEXT:POKER2,41:W=W+1:RETU
RN
879 ' ROUTINE TO POKE STRING IN TEMPORARY STORAGE ARE
A
880 C=MF:X=USR3(0):GOSUB900:NN=1
881 IFB=34GOSUB890:RETURN
882 BC=B:GOSUB889:P=19:X=USR(0):NN=NN+1:IFFP=1ANDPEEK(Q
)=32:Q=Q+1:B=32:GOTO882
884 X=USR1(0)
885 IFFP=1AND(B=34ORB=0)GOSUB890:RETURN:'STRING
886 IFFP=0AND(B=58ORB=0)GOSUB890:RETURN:'VAR. TRANSFE
R
887 IFFP=-1ANDB=41ANDE<>41GOSUB890:RETURN:'VARPTR
888 GOTO882
889 P=62:X=USR(0):P=BC:X=USR(0):P=18:X=USR(0):RETURN
890 BC=0:GOSUB889:IFNN>SLTHENPRINT:PRINT"STRING TOO L
ONG":GOTO517ELSERETURN
899 ' ************ FREQUENTLY USED MACHINE CODES ********
*****
900 P=17:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):RETURN:'LD D
E,E1D1
902 P=33:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):RETURN:'LD H
L,E1D1
904 P=25:X=USR(0):RETURN:'ADD HL,DE
906 P=41:X=USR(0):P=41:X=USR(0):RETURN:' ADD HL,HL; ADD HL
,HL
908 P=235:X=USR(0):RETURN:' EXC HL,DE
910 P=54:X=USR(0):P=H:X=USR(0):RETURN:'LD (HL),H
912 C=VT+V0+V0:X=USR3(0):P=42:X=USR(0):P=E1:X=USR(0):P=D1:
X=USR(0):RETURN:' LD HL,(C1)
914 P=42:X=USR(0):GOSUB818:P=E1:X=USR(0):P=D1:X=USR(0):RET
URN:'LD HL,(C)
916 P=34:X=USR(0):GOSUB818:P=E1:X=USR(0):P=D1:X=USR(0):RET
URN:'LD (C),HL
918 P=195:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):RETURN:'JP E
1D1
```

```
920 P=183:X=USR(0):P=237:X=USR(0):P=82:X=USR(0):RETURN:'OR
A; SBC HL,DE
922 P=40:X=USR(0):P=3:X=USR(0):RETURN:'JR Z,3
924 P=225:X=USR(0):RETURN:'POP HL
926 P=229:X=USR(0):RETURN:'PUSH HL
928 P=209:X=USR(0):RETURN:'POP DE
930 P=213:X=USR(0):RETURN:'PUSH DE
932 P=205:X=USR(0):P=177:X=USR(0):P=9:X=USR(0):RETURN:'MOV
E VARIABLE TO 4121H AREA
934 P=205:X=USR(0):P=164:X=USR(0):P=9:X=USR(0):RETURN:'MOV
E FROM 4121H TO STACK
936 P=193:X=USR(0):GOSUB928:RETURN:'POP BC & DE
938 ' ARITHMETIC OPERATION & FUNCTION ROUTINES
940 P=205:X=USR(0):P=22:X=USR(0):P=7:X=USR(0):RETURN
942 P=205:X=USR(0):P=19:X=USR(0):P=7:X=USR(0):RETURN
944 P=205:X=USR(0):P=71:X=USR(0):P=8:X=USR(0):RETURN
946 P=205:X=USR(0):P=162:X=USR(0):P=8:X=USR(0):RETURN
948 P=205:X=USR(0):P=247:X=USR(0):P=19:X=USR(0):RETURN
950 P=205:X=USR(0):P=12:X=USR(0):P=10:X=USR(0):RETURN
952 P=205:X=USR(0):P=231:X=USR(0):P=19:X=USR(0):RETURN
954 P=205:X=USR(0):P=239:X=USR(0):P=10:X=USR(0):P=205:X=USR(
0):P=119:X=USR(0):P=9:X=USR(0):RETURN
956 P=205:X=USR(0):P=9:X=USR(0):P=8:X=USR(0):RETURN
958 P=205:X=USR(0):P=57:X=USR(0):P=20:X=USR(0):RETURN
960 P=205:X=USR(0):P=65:X=USR(0):P=21:X=USR(0):RETURN
962 P=205:X=USR(0):P=71:X=USR(0):P=21:X=USR(0):RETURN
964 P=205:X=USR(0):P=168:X=USR(0):P=21:X=USR(0):RETURN
966 P=205:X=USR(0):P=189:X=USR(0):P=21:X=USR(0):RETURN
968 P=205:X=USR(0):P=167:X=USR(0):P=40:X=USR(0):RETURN
970 P=62:X=USR(0):P=4:X=USR(0):P=50:X=USR(0):P=175:X=USR(0):
P=64:X=USR(0):RETURN
972 P=205:X=USR(0):P=203:X=USR(0):P=9:X=USR(0):RETURN
974 P=205:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):X=USR1(0):RE
TURN:'USR
999 ' ***************** MAIN PROGRAM ********************
**
1000 MR=0:PC=2750:' MR IS 0 FOR 48K, 1 FOR 32K ; PC IS THE DE
FAULT # OF PROTECTED COMPILED CODE BYTES FROM TOP OF
 MEMORY
1002 CLS:PRINTCHR$(23):PRINTTAB(9)"BASIC COMPILER":PRINT
:PRINT:PRINT"<C> COMPILE":PRINT"<M> CHANGE MEMORY SIZ
E":PRINT"<S> SEE STORAGE PARAMETERS":PRINT"<Q> QUIT":G
OSUB522
1004 IFA%=67GOTO6620ELSEIFA%=77GOTO6600ELSEIFA%=83GO
TO6650ELSEIFA%=81THENCLS:ENDELSEGOTO1002
1005 ' **** MAIN LOOP STARTS HERE ****
1010 M1=PEEK(Q)+PEEK(Q+1)*256:L1(L)=PEEK(Q+2)+PEEK(Q+3)*25
6:IFL1(L)>500THEN1230ELSEGOSUB518:PRINTL1(L);M;":";L2(L)=
M:L=L+1:Q=Q+4:'START NEW LINE
1020 X=USR1(0):IFB=147Q=M1:GOTO1010:' PEEK FIRST BYTE OF
NEXT STATEMENT
1030 IFB=140X=USR1(0):' LET
1040 IFB>64ANDB<91ANDE=37GOSUB2000:GOTO1200:' INTEGER
LET
1050 IFB>64ANDB<91ANDE<>37ANDE<>36GOSUB535:X=USR1(0):X
=USR1(0):IF(B=49ANDE=207)OR(B=48ANDE=205)GOSUB4000ELSEQ
=Q-2:X=USR1(0):IFB<>213THEN517ELSEGOSUB829:GOSUB700:Q=Q
2:B=PT:GOSUB565:GOSUB972:GOTO1200:' S.P. LET
1060 IFB>64ANDB<91ANDE=36GOSUB4500:GOTO1200:' STRING L
ET
1070 IFB=178GOSUB2500:GOTO1200:' PRINT
1075 IFB=175P=62:X=USR(0):P=1:X=USR(0):P=50:X=USR(0):P=156:X
=USR(0):P=64:X=USR(0):GOSUB2500:P=62:X=USR(0):P=0:X=USR(0):
P=50:X=USR(0):P=156:X=USR(0):P=64:X=USR(0):GOTO1200:' LPRIN
T
1080 IFB=141X=USR1(0):GOSUB804:IFC<0ORC>500THEN517ELSED
=D1:F=E1:GOSUB3205:GOTO1200:' GOTO
1090 IFB=134P=205:X=USR(0):P=211:X=USR(0):P=1:X=USR(0):X=US
R1(0):GOTO1200:' RANDOM
1100 IFB=143GOSUB3000:GOTO1200:' IF ... THEN
1110 IFB=145X=USR1(0):GOSUB804:IFC<1ORC>500THEN517ELSED
=D1:F=E1:P=205:GOSUB3300:GOTO1200:' GOSUB
1120 IFB=146P=201:X=USR(0):X=USR1(0):GOTO1200:' RETURN
1130 IFB=132THENP=205:X=USR(0):P=201:X=USR(0):P=1:X=USR(0):
X=USR1(0):GOTO1200:' CLS
1140 IFB=137GOSUB2700:GOTO1200:'INPUT
1150 IFB=129GOSUB5000:GOTO1200:' FOR
1160 IFB=135V1=E(FC):GOSUB914:GOSUB928:GOSUB930:GOSUB92
6:GOSUB920:GOSUB924:P=35:X=USR(0):P=194:X=USR(0):C=D(FC):X
=USR3(0):P=E1:X=USR(0):P=D1:X=USR(0):FC=FC-1:GOSUB924:X=U
SR1(0):IFB=58ORB=0GOTO1200ELSEGOSUB515:GOSUB516:X=USR1
(0):GOTO1200:' NEXT
1170 IFB=130ORB=131GOSUB6000:GOTO1200:' SET & RESET
1180 IFB=177GOSUB6500:GOTO1200:' POKE
1185 IFB=160GOSUB5500:GOTO1200:' OUT
1190 IFB=128P=205:X=USR(0):P=157:X=USR(0):P=10:X=USR(0):P=20
1:X=USR(0):X=USR1(0):' END
1200 IFB=58ORB=149PRINTT$;:GOTO1020:' TEST FOR TERMINAT
OR & ELSE TOKEN
1210 IFB=0ORB=147Q=M1:GOTO1010ELSE517:' START NEW LINE
1215 ' **** END OF MAIN LOOP ****
1220 ' ********** ROUTINE TO ADJUST THE MACHINE CODE JU
MPS ******
1230 PRINT:PRINT"ADJUSTING JUMP ADDRESSES . . .":IFK=0G
OTO1300
1240 FORI=1TOK:DN=PEEK(A(I))+256*PEEK(A(I)+1):X=USR2(0):NE
XT
1299 ' ****** ROUTINE TO EXECUTE THE MACHINE CODE DIREC
TLY ********
1300 DEFUSR=MC:CLS:PRINT"PROGRAM COMPILATION COMPLE
TED":GOSUB519
1310 A!=65536+MC:GOSUB520:PRINT"START=";A!;" DECIMAL";"
OR  "!C$;" HEX":C1$=C$:A!=65536+M+1:GOSUB520:PRINT"END  ="
;A!;" DECIMAL";"  OR  "!C$;" HEX":C2$=C$:PRINT"IN A COMBIN
ED BASIC PROGRAM SET MEMORY SIZE";VN+65535:PRINT"OR PR
OTECT";-VN;"BYTES"
1320 GOSUB519:PRINT"PRESS <D> TO DUMP THE COMPILED CO
DE; <ENTER> TO RUN IT DIRECTLY":PRINT"PRESS <BREAK> TO
EXIT":PRINT"THE DUMP OPTION WORKS ONLY IF YOU USE NE
WDOS80"
1330 GOSUB522:IFA%<>13ANDA%<>68THEN1330
1335 IFA%=68THENINPUT"FILE NAME";F$:CMD"DUMP,"+F$+","+
C1$+"H,"+C2$+"H":GOTO1320:' DUMP COMMAND IS FOR NEWDOS
80; FOR TRSDOS MOD III USE INSTEAD ...
1340 CLS:X=USR(0):END
1400 ' ********** END OF MAIN PROGRAM **************
1999 ' INTEGER ASSIGNMENT ROUTINE
2000 V1=B-65:Q=Q+1:X=USR1(0):IFB<>213THEN517
2010 X=USR1(0):IF(B=49ANDE=207)OR(B=48ANDE=205)THEN2400
2020 IFB=216GOSUB829:GOSUB700:Q=Q2:B=PT:GOSUB970:P=205:
X=USR(0):P=61:X=USR(0):P=11:X=USR(0):GOSUB916:RETURN:' INT
2021 IFB=242GOSUB829:GOSUB700:Q=Q2:B=PT:GOSUB970:P=205:
X=USR(0):P=38:X=USR(0):P=11:X=USR(0):GOSUB916:RETURN:' FIX
2022 IFB=215GOSUB829:GOSUB700:Q=Q2:B=PT:GOSUB970:P=205:
X=USR(0):P=138:X=USR(0):P=9:X=USR(0):GOSUB916:RETURN:' SGN
2023 IFB=239GOSUB829:GOSUB700:Q=Q2:B=PT:P=205:X=USR(0):P=
138:X=USR(0):P=10:X=USR(0):GOSUB916:RETURN:' CINT
2024 IFB=222X=USR1(0):X=USR1(0):GOSUB804:IFC$=""THEN517:E
LSEGOSUB902:P=205:X=USR(0):P=204:X=USR(0):P=20:X=USR(0):P=
205:X=USR(0):P=127:X=USR(0):P=10:X=USR(0):GOSUB916:X=USR1(
0):RETURN:' RND
2025 IFB=193X=USR1(0):X=USR1(0):GOSUB804:IFC$=""THENV1=B
-65:GOSUB515:GOSUB516:X=USR1(0):GOSUB914:C=M+4:X=USR3(0)
:P=34:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):GOSUB974:RETUR
NELSEGOSUB974:RETURN:' USR
2026 IFB<>229THEN2030ELSEV3=V1:X=USR1(0):X=USR1(0):GOSUB
804:IFC$=""THENV1=B-65:GOSUB515:GOSUB516:X=USR1(0):GOSU
B914:P=126:X=USR(0)ELSEP=58:X=USR(0):P=E1:X=USR(0):P=D1:X=
USR(0):' PEEK
2027 P=38:X=USR(0):P=0:X=USR(0):P=111:X=USR(0):V1=V3:GOSUB9
16:X=USR1(0):RETURN:' PEEK
2030 IFB=198V3=V1:GOSUB6000:P=42:X=USR(0):P=33:X=USR(0):P=
65:X=USR(0):V1=V3:GOSUB916:RETURN:' POINT
2031 IFB=192X=USR1(0):X=USR1(0):MF=VN:FP=-1:GOSUB880:GOS
UB902:P=205:X=USR(0):P=13:X=USR(0):P=38:X=USR(0):P=235:X=US
R(0):GOSUB916:X=USR1(0):RETURN:'VARPTR
2032 IFB=246X=USR1(0):X=USR1(0):V2=B-65:IFV2<0ORV2>=NSOR
E<>36THEN517ELSEX=USR1(0):X=USR1(0):X=USR1(0):C=VS+V2*(SL
+1):X=USR3(0):P=58:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):GOS
UB818:GOSUB902:P=119:X=USR(0):P=35:X=USR(0):P=54:X=USR(0):
P=0:X=USR(0):RETURN:' ASC
2034 V2=V1:Q=Q-1:X=USR1(0):IFB=206ANDE>47ANDE<58GOSUB80
4:GOSUB902:GOTO2040ELSEIFB=206THENE1=0:D1=0:GOSUB902:G
OTO2040
2035 GOSUB804:IFC$<>""THENGOSUB902ELSEV1=B-65:GOSUB51
5:GOSUB516:GOSUB914:X=USR1(0)
2040 IFB=58ORB=0V1=V2:GOSUB916:RETURN
2045 IFB=206ANDE>47ANDE<58SG=205ELSESG=B:X=USR1(0)
2050 GOSUB2060:IFSG=205GOSUB904ELSEGOSUB920
2055 GOTO2040
2060 GOSUB804:IFC$<>""THENGOSUB900:RETURNELSEV1=B-65:
GOSUB515:GOSUB516:P=237:X=USR(0):P=91:X=USR(0):GOSUB818:
P=E1:X=USR(0):P=D1:X=USR(0):X=USR1(0):RETURN
```

```
2399 ' INTEGER VARIABLE TRANSFER ROUTINE
2400 IFB=49ANDE=207CM=1ELSECM=0
2410 Q=Q+1:X=USR1(0):MF=VN:FP=0:GOSUB880:GOSUB902:P=205:
X=USR(0):P=13:X=USR(0):P=38:X=USR(0)
2420 IFCM=0P=26:X=USR(0):P=111:X=USR(0):P=19:X=USR(0):P=26:
X=USR(0):P=103:X=USR(0):GOSUB916:RETURN:' TRANSFER BASIC
 VARIABLE INTO USR VARIABLE
2430 GOSUB914:P=125:X=USR(0):P=18:X=USR(0):P=19:X=USR(0):P=
124:X=USR(0):P=18:X=USR(0):RETURN:' TRANSFER USR VARIABL
E INTO BASIC VARIABLE
2499 ' PRINT ROUTINE
2500 X=USR1(0):IFB=58ORB=0P1=13:GOSUB2670:RETURN
2502 IFB<>64ANDB<>96THEN2600ELSEX=USR1(0)
2503 ' PRINT @
2504 IFB<58GOSUB804:C=C+15360:X=USR1(0):X=USR3(0):GOSUB90
2ELSEV1=B-65:GOSUB515:GOSUB516:X=USR1(0):X=USR1(0):GOSU
B914:D1=60:E1=0:GOSUB900:GOSUB904
2508 P=34:X=USR(0):P=32:X=USR(0):P=64:X=USR(0)
2599 ' SINGLE PRECISION EXPRESSION & STRING PRINT ROUTI
NE
2600 IFB>64ANDB<91ANDE=36Q=Q+1:V1=B-65:C=VS+V1*(SL+1):G
OSUB2680:X=USR1(0):GOTO2630
2610 IFB=34B=PEEK(Q):Q=Q+1:MF=VN:FP=1:GOSUB880:GOSUB268
0:GOTO2630
2620 Q=Q-1:GOSUB829:GOSUB700:Q=Q2:B=PT:GOSUB970:P=205:X=
USR(0):P=189:X=USR(0):P=15:X=USR(0):GOSUB968:IFB=59P1=32:G
OSUB2670:' SINGLE PRECISION PRINT
2630 IFB=44THEN517
2640 IFB=59X=USR1(0):IFB<>58ANDB<>0THEN2600ELSERETURN
2650 IFB=58ORB=0P1=13:GOSUB2670:RETURN
2660 GOTO517
2670 C=VN:X=USR3(0):GOSUB902:H=P1:GOSUB910:P=35:X=USR(0):
H=0:GOSUB910:P=43:X=USR(0):GOSUB968:RETURN:' PRINT BYTE
P1
2680 X=USR3(0):GOSUB902:GOSUB968:IFB=34X=USR1(0):RETURN
ELSERETURN:' PRINT STRING
2699 ' INPUT
2700 P=205:X=USR(0):P=179:X=USR(0):P=27:X=USR(0):P=35:X=USR(
0):P=205:X=USR(0):P=108:X=USR(0):P=14:X=USR(0):X=USR1(0)
2710 IFE=37P=205:X=USR(0):P=127:X=USR(0):P=10:X=USR(0):V1=B
-65:GOSUB818:GOSUB902:P=237:X=USR(0):P=75:X=USR(0):P=33:X=
USR(0):P=65:X=USR(0):P=113:X=USR(0):P=35:X=USR(0):P=112:X=US
R(0):X=USR1(0):X=USR1(0):RETURN
2720 CF=1:GOSUB535:CF=0:P=58:X=USR(0):P=175:X=USR(0):P=64:X
=USR(0):P=222:X=USR(0):P=4:X=USR(0):GOSUB922:P=205:X=USR(0)
:P=204:X=USR(0):P=10:X=USR(0):GOSUB565:GOSUB972:X=USR1(0):
RETURN
2999 ' INTEGER IF - THEN ROUTINE
3000 X=USR1(0):IFE<>37THEN3100ELSEGOSUB516:V1=B-65:GOSU
B914:GOSUB908:X=USR1(0)
3005 IFB=212ANDE=213ORB=213ANDE=212W1=1:Q=Q+1:GOTO3035
3010 IFB=214ANDE=213ORB=213ANDE=214W1=2:Q=Q+1:GOTO3035
3015 IFB=212ANDE=214ORB=214ANDE=212W1=3:Q=Q+1:GOTO3035
3020 IFB=212W1=4
3025 IFB=214W1=5
3030 IFB=213W1=6
3035 X=USR1(0):IFB<58ORB=206GOSUB804:GOSUB902ELSEGOSUB
515:GOSUB516:V1=B-65:GOSUB914:X=USR1(0)
3040 IFB<>202ANDB<>141THEN517ELSEX=USR1(0):IFB=141X=US
R1(0)
3045 GOSUB804:IFC<0ORC>500THEN517
3050 P=205:X=USR(0):P=57:X=USR(0):P=10:X=USR(0)
3055 GOTO3150
3099 ' SINGLE PRECISION IF - THEN ROUTINE
3100 Q=Q-1:GOSUB829:GOSUB700:Q=Q2:B=PT:E=PEEK(Q):GOSUB9
34
3105 IFB=212ANDE=213ORB=213ANDE=212W1=1:Q=Q+1:GOTO3135
3110 IFB=214ANDE=213ORB=213ANDE=214W1=2:Q=Q+1:GOTO3135
3115 IFB=212ANDE=214ORB=214ANDE=212W1=3:Q=Q+1:GOTO3135
3120 IFB=212W1=4
3125 IFB=214W1=5
3130 IFB=213W1=6
3135 GOSUB829:GOSUB700:Q=Q2:B=PT:E=PEEK(Q):GOSUB936:GOS
UB950
3140 IFB<>202ANDB<>141THEN517ELSEX=USR1(0):IFB=141X=US
R1(0)
3145 GOSUB804:IFC<0ORC>500THEN517
3150 D=D1:F=E1:ONW1GOTO3155,3160,3165,3170,3175,3180
3155 GOSUB3185:GOSUB3200:RETURN
3160 GOSUB3185:GOSUB3190:RETURN
3165 GOSUB3195:RETURN
3170 P=40:X=USR(0):P=3:X=USR(0):GOSUB3200:RETURN
3175 P=40:X=USR(0):P=3:X=USR(0):GOSUB3190:RETURN
3180 GOSUB3185:RETURN
3185 P=202:GOSUB3300:RETURN
3190 P=242:GOSUB3300:RETURN
3195 P=194:GOSUB3300:RETURN
3200 P=250:GOSUB3300:RETURN
3205 P=195:GOSUB3300:RETURN
3300 X=USR(0):K=K+1:A(K)=M:P=F:X=USR(0):P=D:X=USR(0):RETUR
N
3999 ' SINGLE PRECISION VARIABLE TRANSFER ROUTINE
4000 IFB=49ANDE=207THENCM=1ELSECM=0
4010 X=USR1(0):X=USR1(0):MF=VN:FP=0:GOSUB880:GOSUB902:P=
205:X=USR(0):P=13:X=USR(0):P=38:X=USR(0)
4020 IFCM=0GOSUB908:GOSUB932:GOSUB565:GOSUB972:RETURN
4030 GOSUB930:GOSUB565:GOSUB932:GOSUB924:GOSUB972:RET
URN
4499 ' STRING ASSIGNMENT ROUTINE
4500 V1=B-65:IFV1<0ORV1>=NSTHEN517ELSEMF=VS+V1*(SL+1):
X=USR1(0):X=USR1(0):IFB<>213THEN517
4530 X=USR1(0):IFB=247THEN4600ELSEIFB=201THEN4700ELSEIF
B<>34THEN517
4540 B=PEEK(Q):Q=Q+1:FP=1:GOSUB880:IFB=34X=USR1(0)ELSEB=
PEEK(Q-1)
4550 RETURN
4599 ' STRING ASSIGNMENT USING CHR$
4600 C=MF:X=USR3(0):GOSUB902
4610 X=USR1(0):IFB<>40THEN517
4615 X=USR1(0):IFB>64ANDB<91THENV1=B-65:GOSUB516:X=USR
1(0):IFB<>41THEN517ELSEP=58:X=USR(0):GOSUB818:P=E1:X=USR(
0):P=D1:X=USR(0):P=119:X=USR(0):GOTO4660
4620 GOSUB804:H=C:GOSUB910
4660 X=USR1(0):IFB=205X=USR1(0):IFB<>247THEN517ELSEP=35:X
=USR(0):GOTO4610
4670 P=35:X=USR(0):H=0:GOSUB910:RETURN
4699 ' INKEY$
4700 X=USR1(0):P=205:X=USR(0):P=91:X=USR(0):P=3:X=USR(0):C=M
F:X=USR3(0):GOSUB902:P=119:X=USR(0):P=35:X=USR(0):P=54:X=U
SR(0):P=0:X=USR(0):RETURN
4999 ' FOR ROUTINE
5000 FC=FC+1:D(FC)=M+7:X=USR1(0):GOSUB515:GOSUB516:V1=B-
65:E(FC)=V1:X=USR1(0):IFB<>213THEN517
5010 X=USR1(0):IFB<65ORB=206GOSUB804:J1=0:ID=D1:IE=E1ELS
EJ1=1:V2=B-65:GOSUB516:C=VT+V2*2:X=USR3(0):ID=D1:IE=E1:X=
USR1(0)
5020 IFB<>189THEN517
5030 X=USR1(0):IFB<65ORB=206GOSUB804:J2=0:FD=D1:FE=E1ELS
EJ2=1:V3=B-65:GOSUB516:X=USR1(0):C=VT+V3*2:X=USR3(0):FD=
D1:FE=E1
5040 IFJ2=0THENP=33ELSEP=42
5050 X=USR(0):P=FE:X=USR(0):P=FD:X=USR(0):GOSUB926
5060 IFJ1=0THENE1=IE:D1=ID:GOSUB902
5070 IFJ1=1THENP=42:X=USR(0):P=IE:X=USR(0):P=ID:X=USR(0)
5080 GOSUB916:RETURN
5499 ' OUT
5500 X=USR1(0):GOSUB804:IFC$=""THENGOSUB515:V2=B-65:Z2=1:
GOSUB516:X=USR1(0)ELSEV2=E1:Z2=0
5510 IFB=44THENX=USR1(0):GOSUB804ELSEGOTO517
5520 IFC$=""THENGOSUB515:V1=B-65:Z1=1:GOSUB516:X=USR1(0)
ELSEV1=E1:Z1=0
5530 IFZ1=0P=62:X=USR(0):P=V1:X=USR(0)ELSEGOSUB914:P=125:
X=USR(0)
5540 V1=V2:IFZ2=0P=211:X=USR(0):P=V1:X=USR(0)ELSEGOSUB914
:P=77:X=USR(0):P=237:X=USR(0):P=121:X=USR(0)
5550 RETURN
5999 ' POINT, SET & RESET
6000 IFB=130THENW=1ELSEIFB=131THENW=128ELSEIFB=198THE
NW=0
6020 MA=M:X=USR1(0):IFB<>40THEN517ELSEX=USR1(0):GOSUB80
4:IFC$=""THENGOSUB515:GOSUB516:V1=B-65:GOSUB818:D2=D1:
E2=E1:C2=1ELSEE2=E1:C2=0:IFB<>44THEN517
6030 IFC2=1X=USR1(0):IFB<>44THEN517
6040 X=USR1(0):GOSUB804:IFC$=""THENGOSUB515:GOSUB516:V1
=B-65:GOSUB818:D3=D1:E3=E1:C3=1ELSEE3=E1:C3=0:IFB<>41THE
N517
6050 IFC3=1X=USR1(0):IFB<>41THEN517
6070 X=USR1(0):C=MA+18+C2+C3:X=USR3(0):GOSUB902:GOSUB926
:E1=126:D1=7:GOSUB902:P=62:X=USR(0):P=W:X=USR(0):P=245:X=U
SR(0)
```

```
6080 IFC2=1THENP=58:X=USR(0):P=E2:X=USR(0):P=D2:X=USR(0)EL
SEP=62:X=USR(0):P=E2:X=USR(0)
6090 P=245:X=USR(0)
6100 IFC3=1THENP=58:X=USR(0):P=E3:X=USR(0):P=D3:X=USR(0)EL
SEP=62:X=USR(0):P=E3:X=USR(0)
6110 E1=80:D1=1:GOSUB918:RETURN
6499 ' POKE ROUTINE
6500 X=USR1(0):GOSUB804:IFC$=""THENGOSUB515:GOSUB516:V1
=B-65:GOSUB914:X=USR1(0)ELSEGOSUB902
6510 IFB<>44THEN517
6520 X=USR1(0):GOSUB804:IFC$=""THENGOSUB515:GOSUB516:V1
=B-65:GOSUB818:P=58:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):X
=USR1(0)ELSEP=62:X=USR(0):P=E1:X=USR(0)
6540 P=119:X=USR(0):RETURN
6599 ' ***** INITIALIZATION ROUTINE *****
6600 CLS:PRINTTAB(17)"BASIC COMPILER -"48-MR*16"K SYSTE
M":GOSUB519:PRINT"CURRENT # OF PROTECTED BYTES:"PC:PR
INT"MINIMUM IS 750 PLUS THE ALLOCATED SIZE FOR COMPILE
D CODE IN LINE 6650
6605 GOSUB519:PRINT"PRESS <C> TO CHANGE THAT NUMBER; A
NY OTHER KEY TO CONTINUE":GOSUB522
6610 IFA%=67THENINPUT"NEW #";PC:GOTO1002ELSEGOTO1002
6620 MC=-PC-MR*16384:HS=MC+65535:POKE16562,INT(HS/256):P
OKE16561,HS-INT(HS/256)*256:CLEAR150:S$="F":' HS IS MEMOR
Y SIZE
6630 DEFINTA-Z:DIML1(100),L2(100),A(50),D(25),E(25):P=0:Q=0:B=
0:E=0:X=0:M=0:C=0:A=0:V1=0:E1=0:D1=0:V0=0:Z=0:L=0:Z1=0:FC=0:T
$=CHR$(32)+CHR$(58)+CHR$(32):F$="COMPILING LINE"
6650 MR=0:CC=2000:SS=2000:IFSS<CCTHENPRINT"VARIABLE STO
RAGE OVERLAPS THE COMPILED CODE":END:' MR=0 FOR 48K; M
R=1 FOR 32K; CC IS THE ALLOCATED # OF BYTES FOR COMPILE
D CODE; SS IS THE # OF BYTES BELOW TOP OF MEMORY WHERE
 VARIABLE STORAGE BEGINS
6660 IS=0:DO=20:DT=10:DC=20:SL=30:NO=3:NT=0:NS=4:'IS=# OF A
DDITIONAL S.P. VAR. PER LETTER; DO=DIM OF 1-D ARRAYS; D
T,DC=ROW & COLUMN DIM OF 2-D ARRAYS; SL=LENGTH OF STR
INGS; NO=# OF 1-D ARRAYS ALLOWED; NT=# OF 2-D ARRAYS A
LLOWED; NS=# OF STRINGS ALLOWED
6670 S1=25:IFS1<SLTHENS1=SL
6680 MS=-SS-MR*16384:MC=-CC-MR*16384:Q=PEEK(16548)+256*P
EEK(16549):L=1:K=0:CF=0:M=MC
6690 VT=-2*26+MS:VF=-4*26*(1+IS)+VT:VA=-4*NO*DO+VF:VD=-
4*NT*DT*DC-2*NT*DC+VA:VS=-NS*(SL+1)+VD:VN=-(S1+1)+VS
6700 ' MS=END OF VARIABLE STORAGE AREA; MC=START OF M
ACHINE CODE; VT=START OF INTEGER STORAGE; VF=START OF
 SIMPLE VARIABLE STORAGE; VA=START OF 1-D ARRAY STORA
GE; VD=START OF 2-D ARRAY STORAGE; VS=START OF STRING
 STORAGE; VN=END OF STORAGE
6705 IFS$="F"GOTO6730
6707 CLS:PRINTTAB(19)"CURRENT STORAGE PARAMETERS":GOS
UB519
6710 PRINT"# OF BYTES ALLOCATED FOR COMPILED CODE:"CC:
PRINT"# OF SIMPLE SINGLE PRECISION VARIABLES:"26*(IS+1):
PRINT"# OF 1-D ARRAYS:"NO:PRINT"DIMENSION OF 1-D ARRA
YS:"DO:PRINT"# OF 2-D ARRAYS:"NT:PRINT"DIMENSION OF 2-
D ARRAYS:"DT"X"DC
6720 PRINT"# OF STRING VARIABLES:"NS:PRINT"LENGTH OF S
TRING VARIABLES:"SL:PRINT"TOTAL SIZE OF MACHINE CODE &
 STORAGE AREA:";-VN:GOSUB519:PRINT"STORAGE PARAMETER
S CAN BE CHANGED IN LINES 6650, 6660":PRINT"PRESS ANY KE
Y TO CONTINUE":GOSUB522:GOTO1002
6730 CLS:PRINTCHR$(23):PRINT@450,"B A S I C  C O M P I L E R"
;:PRINT@514,STRING$(27,176)
6740 GOSUB7200:CLS:IFNT>0GOSUB7000:' POKE MACHINE CODE
& GENERATE CODE TO STORE 2-D ARRAY ADDRESSES
6750 GOTO1010:' END OF INITIALIZATION
6999 ' ROUTINE FOR CODE TO STORE 2-D ARRAY ADDRESSES
7000 PRINT"CODE TO STORE 2-D ARRAY ADDRESSES":C=VD:X=
USR3(0):GOSUB902:C=VD+4*NT*DT*DC:X=USR3(0):P=221:X=USR(0
):GOSUB902:C=4*DT:X=USR3(0):GOSUB900:C=NT*DC:X=USR3(0):P=
1:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0)
7005 ' 4*DT IS THE MEMORY LENGTH OF A 2-D ARRAY COLUMN
; NT*DC IS THE TOTAL # OF COLUMNS
7010 P=221:X=USR(0):P=117:X=USR(0):P=0:X=USR(0):P=221:X=USR(
0):P=35:X=USR(0):P=221:X=USR(0):P=116:X=USR(0):P=0:X=USR(0):P
=221:X=USR(0):P=35:X=USR(0):GOSUB904:P=13:X=USR(0)
7020 C=M-12:X=USR3(0):P=194:X=USR(0):P=E1:X=USR(0):P=D1:X=U
SR(0):P=5:X=USR(0):P=14:X=USR(0):P=255:X=USR(0):C=M-18:X=US
R3(0):P=242:X=USR(0):P=E1:X=USR(0):P=D1:X=USR(0):PRINT:PRIN
T:PRINT"MAIN CODE BEGINS":RETURN
7199 ' ROUTINE TO POKE THE MACHINE CODE
7200 A=MC-233:FORP=ATOA+232:READB:POKEP,B:NEXT
7220 DEFUSR1=A+10:DEFUSR=A+49:DEFUSR2=A+105:DEFUSR3=A
+197
7230 C=A+225:GOSUB816:POKEA+198,E1:POKEA+199,D1
7240 C=A+227:GOSUB816:POKEA+205,E1:POKEA+206,D1
7250 C=A+230:GOSUB816:POKEA+215,E1:POKEA+216,D1
7260 C=A:X=USR3(0):POKEA+11,E1:POKEA+12,D1
7270 C=A+2:X=USR3(0):POKEA+41,E1:POKEA+42,D1
7280 C=A+4:X=USR3(0):POKEA+32,E1:POKEA+33,D1
7290 C=A+6:X=USR3(0):POKEA+57,E1:POKEA+58,D1
7300 C=A+8:X=USR3(0):POKEA+50,E1:POKEA+51,D1
7310 C=A+177:X=USR3(0):POKEA+113,E1:POKEA+114,D1
7320 C=A+183:X=USR3(0):POKEA+120,E1:POKEA+121,D1
7330 C=A+189:X=USR3(0):POKEA+127,E1:POKEA+128,D1
7340 C=A+192:X=USR3(0):POKEA+106,E1:POKEA+107,D1
7350 DEFFNH2$(A1%)=MID$("0123456789ABCDEF",INT(A1%/16)+1
,1)+MID$("0123456789ABCDEF",A1%-INT(A1%/16)*16+1,1)
7360 DEFFNH4$(A1%)=FNH2$(ASC(MID$(MKI$(A1%),2)))+FNH2$(AS
C(MKI$(A1%)))
7370 RETURN
7380 DATA81,0,66,0,69,0,80,0,77,0
7390 DATA33,0,224,205,13,38,235,94,35,86
7400 DATA26,19,254,32,40,250,114,43,115,245
7410 DATA213,33,4,224,205,13,38,225,126,18
7420 DATA33,2,224,205,13,38,241,18,201,33
7430 DATA8,224,205,13,38,213,33,6,224,205
7440 DATA13,38,213,26,50,33,65,62,0,50
7450 DATA34,65,62,2,50,175,64,205,189,15
7460 DATA205,167,40,62,2,50,175,64,209,225
7470 DATA26,94,35,86,18,43,126,60,119,254
7480 DATA0,192,35,24,247
7490 DATA33,192,224,205,13,38,213,33,177,224
7500 DATA205,13,38,213,33,183,224,205,13,38
7510 DATA213,33,189,224,205,13,38,26,111,19
7520 DATA26,103,235,221,225,253,225,221,110,2
7530 DATA221,102,3,183,237,82,40,10,221,35
7540 DATA221,35,253,35,253,35,24,235,253,126
7550 DATA2,225,94,35,86,18,19,253,126,3
7560 DATA18,201,76,50,40,48,41,0,76,49
7570 DATA40,48,41,0,68,78,0,65,40,73,41,0
7590 DATA33,225,224,205,13,38,213,33,228,224
7600 DATA205,13,38,225,229,126,18,33,231,224
7610 DATA205,13,38,225,35,126,18,201,67
7620 DATA0,69,49,0,68,49,0
7700 ' ************* END OF THE COMPILER ***************
7710 ' THIS VERSION, IN ADDITION TO THE FUNCTIONS SUPPO
RTED BY VERSION 1 (80 MICRO, OCT 82), SUPPORTS THE FOLLO
WING:
7720 ' LPRINT
7730 ' I%=SGN(SP EXPRESSION)
7740 ' I%=RND(INTEGER BETWEEN 1 & 32767)
7750 ' I%=FIX(SP EXPRESSION)
7760 ' I%=CINT(SP EXPRESSION)
7770 ' RANDOM
7775 ' Z%=VARPTR(BASIC INTEGER, STRING OR FLOATING POIN
T VARIABLE NAME)
7780 ' Z%=USR(IV,OR PI). CALL THE MACHINE LANGUAGE ROUT
INE W/ ENTRY POINT IV OR PI. THIS CAN BE USED WITH STRIN
G OR INTEGER ARRAY PACKED ROUTINES ENTERED FROM BAS
IC IN CONJUNCTION WITH THE VARPTR FUNCTION.
7790 ' SINGLE PREC. EXPRESSIONS ARE EVALUATED IDENTICA
LLY BY THE COMPILER & THE BASIC INTERPRETER
7800 ' ARRAYS CAN HAVE BOTH NUMERICAL & INTEGER VARIA
BLE INDEXES
7810 ' 2-D ARRAYS CAN HAVE DIFFERENT ROW & COLUMN DIM
ENSION (THESE ARE SET AT LINE 6660 AND ARE THE SAME FOR
 ALL MATRICES)
7820 ' PRINT @,CHR$ IS NO MORE SUPPORTED
```

CHRISTIAN-ORIENTED BULLETIN BOARD SYSTEMS

For Bulletin Board System callers, I've heard that the following BBS's have a Christian orientation: (215) 932-8829, (408) 997-2790, (515) 576-0591, and (714) 983-9923. The first and third listings are probably more Presbyterian oriented since they were listed in the EDP Network News.

ASCII FILE ENCODING/DECODING PROGRAM by Dave McGlumphy - While not secure enough for government secrets, this program will permit you to temporarily encode your ASCII disk files and later decode them. If others have access to your disks, you can keep them from reading your intimate love letters (you know - the "boilerplate" love letters you send to all the girls!). Instructions for operation are found in the source code comments, so without further ado, here's the program. Have fun!

```
7000            00010          ORG     7000H
7000 44         00020 DRM      DEFM    'Dave McGlumphy
     61 76 65 20 4D 63 47 6C 75 6D 70 68 79 09 09 20
     20 20
7013 43         00030          DEFM    'CODE/DECODE PROGRAM'
     4F 44 45 2F 44 45 43 4F 44 45 20 50 52 4F 47 52
     41 4D
7026 0A         00040          DEFB    10
7027 34         00050          DEFM    '4429 Paula Ln.'
     34 32 39 20 50 61 75 6C 61 20 4C 6E 2E
7035 0A         00060          DEFB    10
7036 43         00070          DEFM    'Chattanooga, Tn. 37415    02/26/84'
     68 61 74 74 61 6E 6F 6F 67 61 2C 20 54 6E 2E 20
     33 37 34 31 35 20 20 20 20 30 32 2F 32 36 2F 38
     34
7058 0A         00080          DEFB    10
7059 0D         00090 CR       DEFB    13
                00100 ;
                00110 ;This program codes/decodes a PENCIL (ASCII) file by
                00120 ;adding 1 to (or subtracting 1 from) the ASCII
                00130 ;value of each character in the file, stopping with
                00140 ;an occurrence of 00H which is used to indicate the
                00150 ;end of a PENCIL file.  It can also "reverse" part of
                00160 ;the ASCII character set.  For example, a space (20H)
                00170 ;becomes a lowercase z (7AH) while an exclamation point
                00180 ;(21H) becomes a lowercase y (79H), and so on.
                00190 ;
                00200 ;You can begin program execution by typing
                00210 ;CODE <ENTER>    or    CODE FILESPEC <ENTER>.
                00220 ;
                00230 ;Since I use NEWDOS80 almost exclusively, I don't know
                00240 ;if this will work with other DOSs, but if it doesn't,
                00250 ;it's probably just a matter of changing the EQUate
                00260 ;addresses to make it work.
                00270 ;
                00280 ;Caution! You can press <BREAK> to exit while the file
                00290 ;is being changed, but if you do, part of the file will
                00300 ;be changed and part won't.  And you'll have a $%&# of a
                00310 ;time trying to get it right again.  You may want to make
                00320 ;a copy of the file, update the copy, then delete the
                00330 ;original if you're satisfied.
                00340 ;
                00350 ;The inspiration for this program was John Krause's
                00360 ;program in the January '84 issue of the
                00370 ;VOICE OF THE '80 newsletter, 10 Richlee Rd.,
                00380 ;Norwalk, Ct., 06851.  Thank you John, Alan, and all
                00390 ;of the Fairfield County TRS-80 Users Group.
                00400 ;And to Jack Decker for his fearless leadership and
                00410 ;his book, "TRS-80 ROM ROUTINES DOCUMENTED",
                00420 ;$19.95 + FROM T.A.S., (517) 482-8270.
                00430 ;
4445            00440 BKUP     EQU     4445H          ;POS. FCB BACK 1 RECORD.
3840            00450 BRKROM   EQU     3840H          ;BREAK-KEY ROM.
0033            00460 BYTOUT   EQU     33H            ;DISPLAY SINGLE BYTE.
01C9            00470 CLS      EQU     01C9H          ;CLEAR THE SCREEN.
4428            00480 CLOSE    EQU     4428H          ;N80 CLOSE FILE.
4020            00490 CURSOR   EQU     4020H          ;CURSOR POSITION LSB.
402D            00500 DOS      EQU     402DH          ;GO BACK TO DOS.
4409            00510 DOSER    EQU     4409H          ;N80 DOS ERROR EXIT.
4467            00520 DSPLY    EQU     4467H          ;N80 DISPLAY A LINE.
441C            00530 EXTFL    EQU     441CH          ;N80 EXTRACT FILESPEC.
002B            00540 CKKEY    EQU     2BH            ;ROM CHECK FOR KEY-PRESS.
40A7            00550 LBUFPT   EQU     40A7H          ;LINE INPUT BUFFER PTR.
0361            00560 LIPT     EQU     361H           ;BASIC LINE INPUT RTN.
4436            00570 READ     EQU     4436H          ;N80 READ A RECORD.
  24            00580 OPEN     EQU     4424H          ;N80 OPEN EXISTING FL.
443C            00590 WRITE    EQU     443CH          ;N80 WRITE DISK RTN.
705A 00         00600 CONT     DEFB    0              ;IF 0, PAUSE.
705B 00         00610 EOFSW    DEFB    0              ;1=AT OR PAST EOF.
705C 1F         00620 EREOF    DEFB    1FH            ;ERASE EOF CHARACTER
705D 00         00630 FUNC     DEFB    0              ;01=INC BY 1, FF=DEC BY 1
                00640 ;                               ;0=REVERSE THE SET.
705E 63         00650 CODMSG   DEFM    'code/decode '  ;TO IDENTIFY FILE.
     6F 64 65 2F 64 65 63 6F 64 65 20
706A 31         00660 FLSPEC   DEFM    '12345678911234567892123456789312'
     32 33 34 35 36 37 38 39 31 31 32 33 34 35 36 37
     38 39 32 31 32 33 34 35 36 37 38 39 33 31 32
708A 0D         00670          DEFB    13             ;TERMINATE THE FLSPEC
                00680                                 ; FOR DISPLAYING IT.
0100            00690 IOBUF    DEFS    256            ;DISK I/O BUFFER.
718B 03         00700          DEFB    3              ;TERMINATE THE IOBUF
                00710                                 ; FOR DISPLAYING IT.
718C            00720 START    EQU     $
718C 3E1C       00730          LD      A,1CH          ;IF (HL) <> 1CH THEN
718E BE         00740          CP      (HL)           ;A FILESPEC WAS SUPPLIED
718F C40371     00750          CALL    NZ,ASKFL1      ;GET IT.
7192 21A740     00760          LD      HL,LBUFPT      ;SET UP THE
7195 36CA       00770          LD      (HL),0CAH      ; LINE
7197 23         00780          INC     HL             ;  INPUT
7198 3664       00790          LD      (HL),64H       ;  BUFFER.
719A 21A641     00800          LD      HL,41A6H       ;PLUG
719D 0615       00810          LD      B,15H          ; THE
719F 34C9       00820 BASLP    LD      (HL),0C9H      ;  DISK
71A1 23         00830          INC     HL             ;  BASIC
71A2 23         00840          INC     HL             ;  LINKS
71A3 23         00850          INC     HL             ;  WITH
71A4 10F9       00860          DJNZ    BASLP          ;  RETURNS (0C9H).
71A6 CDC901     00870          CALL    CLS            ;CLEAR SCREEN.
71A9 210070     00880          LD      HL,DRM         ;POINT AT MY NAME.
71AC CD6744     00890          CALL    DSPLY          ;IDENTIFY.
71AF CDC171     00900          CALL    ASKFL          ;WHICH FILE TO CHANGE?
71B2 CD0272     00910          CALL    GETFL          ;TRY TO OPEN IT.
71B5 CD1172     00920          CALL    ASKFN          ;INCREMENT, DECREMENT,
                00930 ;                               ; OR REVERSE?
71B8 CDB472     00940          CALL    CHGCOD         ;ALTER THE DATA.
71BB CD2844     00950          CALL    CLOSE          ;CLOSE THE FILE.
71BE C36573     00960          JP      DOSERR         ;IF ERROR, DISPLAY IT.
71C1            00970 ASKFL    EQU     $              ;WHICH FILE TO CHG? RTN.
71C1 216A70     00980          LD      HL,FLSPEC      ;POINT TO THE FCB.
71C4 3E31       00990          LD      A,'1'          ;DOES IT START
71C6 BE         01000          CP      (HL)           ; WITH A 1?
71C7 2010       01010          JR      NZ,DSPFL       ;NO. DISPLAY FILESPEC.
71C9 21E071     01020          LD      HL,FLMSG       ;POINT TO WHICH FILE MSG.
71CC CD6744     01030          CALL    DSPLY          ;PRINT IT.
71CF CD6103     01040          CALL    LIPT           ;GET KEYBOARD LINE.
71D2 23         01050          INC     HL             ;GET TO THE FILESPEC.
71D3 116A70     01060 ASKFL1   LD      DE,FLSPEC      ;POINT TO THE FCB.
71D6 CD1C44     01070          CALL    EXTFL          ;EXTRACT IT.
71D9 215E70     01080 DSPFL    LD      HL,CODMSG      ;TELL WHICH FILE
71DC CD6744     01090          CALL    DSPLY          ; WE'RE CHANGING.
71DF C9         01100          RET                    ;TO MAINLINE.
71E0 57         01110 FLMSG    DEFM    'What file do you want to change? '
     68 61 74 20 66 69 6C 65 20 64 6F 20 79 6F 75 20
     77 61 6E 74 20 74 6F 20 63 68 61 6E 67 65 3F 20
7201 03         01120          DEFB    3
7202            01130 GETFL    EQU     $
7202 116A70     01140          LD      DE,FLSPEC      ;POINT AT FILESPEC.
7205 218B70     01150          LD      HL,IOBUF       ;256-BYTE DISK I/O BUFFER
7208 0600       01160          LD      B,0            ;INDICATE 256-BYTE RECS.
720A CD2444     01170          CALL    OPEN           ;OPEN THE FILE.
720D C26573     01180          JP      NZ,DOSERR      ;IF ERROR, DISPLAY IT.
7210 C9         01190          RET                    ;TO MAINLINE.
7211            01200 ASKFN    EQU     $              ;INCREMENT OR DECREMENT?
7211 214C72     01210          LD      HL,CDOCMS      ;POINT TO INC/DEC MSG.
7214 CD6744     01220          CALL    DSPLY          ;PRINT IT.
7217            01230 GETFN1   EQU     $              ;INC, DEC, OR BREAK? RTN.
7217 CD1E73     01240          CALL    CKBRK          ;CHECK BREAK TO EXIT.
721A CD2B00     01250          CALL    CKKEY          ;KEY PRESSED?
721D B7         01260          OR      A              ;CONDITION THE FLAGS.
721E 28F7       01270          JR      Z,GETFN1       ;NOPE.
7220 FE5B       01280          CP      'Z'+1          ;LOWERCASE?
7222 F43372     01290          CALL    P,LWR          ;YES. FIX IT.
7225 FE44       01300          CP      'D'            ;IS IT A D?
7227 2816       01310          JR      Z,DEC          ;YES.
7229 FE49       01320          CP      'I'            ;IS KEY AN 'I'?
722B 2809       01330          JR      Z,INC          ;YES.
722D FE52       01340          CP      'R'            ;IS IT AN R?
722F 2817       01350          JR      Z,MINC         ;YES.
7231 18E4       01360          JR      GETFN1         ;TRY AGAIN FOR VALID IPT.
7233 D620       01370 LWR      SUB     20H            ;CHANGE LOWERCASE TO
7235 C9         01380          RET                    ; UPPER & GO BACK.
7236            01390 INC      EQU     $              ;DISPLAY KEY & SET INC.
7236 CD3300     01400          CALL    BYTOUT         ;DISPLAY THE KEY PRESSED.
```

```
7239 215D70   01410          LD     HL,FUNC          ;PUT IN +1
723C 3601     01420          LD     (HL),1           ; TO INCREMENT.
723E C9       01430          RET                     ;TO MAINLINE.
723F          01440 DEC      EQU    $                ;DISPLAY KEY & SET DEC.
723F CD3300   01450          CALL   BYTOUT           ;DISPLAY THE KEY PRESSED.
7242 215D70   01460          LD     HL,FUNC          ;PUT IN -1 (FFH)
7245 36FF     01470          LD     (HL),0FFH                ; TO DECREMENT.
7247 C9       01480          RET                     ;TO MAINLINE.
7248          01490 NDNC     EQU    $                ;NO INCREMENT - REVERSE.
7248 CD3300   01500          CALL   BYTOUT           ;DISPLAY THE KEY PRESSED.
              01510 ;FUNC HAS A 0 IN IT ALREADY TO INDICATE "REVERSE".
724B C9       01520          RET                     ;TO MAINLINE.
724C 0A       01530 CDOCMS   DEFB   10               ;LINEFEED.
724D 50       01540          DEFM   'Press <D> to decrement, <I> to incremen'
    72 65 73 73 20 3C 44 3E 20 74 6F 20 64 65 63 72
    65 6D 65 6E 74 2C 20 3C 49 3E 20 74 6F 20 69 6E
    63 72 65 6D 65 6E
7274 74       01550          DEFM   't, <R> to reverse'
    2C 20 3C 52 3E 20 74 6F 20 72 65 76 65 72 73 65
7285 0A       01560          DEFB   10
7286 6F       01570          DEFM   'or BREAK to exit. '
    72 20 42 52 45 41 4B 20 74 6F 20 65 78 69 74 2E
    20
7298 03       01580          DEFB   3                ;END MSG WITH NO C/R.
7299          01590 SHOW     EQU    $                ;DISPLAY THE RECORD.
7299 212040   01600          LD     HL,CURSOR        ;SET CURSOR POSITION.
729C 3600     01610          LD     (HL),0  ;SCREEN MEMORY 3E00H.
729E 23       01620          INC    HL               ;CURSOR MSB.
729F 363E     01630          LD     (HL),3EH         ;(LIKE PRINT@).
72A1 3A5C70   01640          LD     A,(EREOF)        ;CLEAR THE REST
72A4 CD3300   01650          CALL   BYTOUT           ; OF THE SCREEN.
72A7 21B870   01660          LD     HL,IOBUF         ;POINT TO THE RECORD.
72AA CD6744   01670          CALL   DSPLY            ;PRINT IT.
72AD 215970   01680          LD     HL,CR            ;PRINT A
72B0 CD6744   01690          CALL   DSPLY            ; CARRIAGE RETURN.
72B3 C9       01700          RET
72B4          01710 CHGCOD   EQU    $
72B4 CD1E73   01720          CALL   CKBRK            ;CHECK BREAK TO EXIT.
72B7 116A70   01730          LD     DE,FLSPEC        ;POINT AT FCB.
72BA 21B870   01740          LD     HL,IOBUF         ;PT TO DISK I/O BUFFER.
72BD CD3644   01750          CALL   READ             ;GET A RECORD.
72C0 C45473   01760          CALL   NZ,CKEOF         ;SEE IF AT OR PAST EOF.
72C3 215B70   01770          LD     HL,EOFSW         ;END
72C6 7E       01780          LD     A,(HL)           ; OF
72C7 FE01     01790          CP     1                ;  FILE?
72C9 C8       01800          RET    Z                ;YES. RET TO MAINLINE.
72CA CD1E73   01810          CALL   CKBRK            ;CHECK BREAK TO EXIT.
72CD 3A5D70   01820          LD     A,(FUNC)         ;WHAT TO DO?
72D0 B7       01830          OR     A                ;CONDITION THE FLAGS.
72D1 CC7873   01840          CALL   Z,REV            ;0 MEANS TO REVERSE SET.
72D4 3A5D70   01850          LD     A,(FUNC)         ;WHAT TO DO?
72D7 B7       01860          OR     A                ;RECONDITION THE FLAGS.
72D8 C40073   01870          CALL   NZ,CHGLP         ;TO INC. OR DEC.
72DB CD9972   01880          CALL   SHOW             ;DISPLAY IT.
72DE 116A70   01890          LD     DE,FLSPEC        ;POINT TO FCB.
72E1 CD4544   01900          CALL   BKUP             ;BACK UP TO CUR REC.
72E4 C26573   01910          JP     NZ,DOSERR        ;IF ERROR, DISPLAY IT.
72E7 CD1E73   01920          CALL   CKBRK            ;CHECK BREAK TO EXIT.
72EA 3A5A70   01930          LD     A,(CONT)         ;IF (CONT)=0, PAUSE
72ED FE00     01940          CP     0                ; TO CHECK RESULTS.
72EF CC9773   01950          CALL   Z,PAUSE          ;(JUST ONE TIME).
72F2 116A70   01960          LD     DE,FLSPEC        ;POINT TO FCB.
72F5 21B870   01970          LD     HL,IOBUF         ;PT TO DISK I/O BUFFER.
72F8 CD3C44   01980          CALL   WRITE            ;REWRITE THE RECORD.
72FB C26573   01990          JP     NZ,DOSERR        ;IF ERROR, DISPLAY IT.
72FE 18B4     02000          JR     CHGCOD           ;DO THE NEXT RECORD.
7300          02010 CHGLP    EQU    $                ;ALTER THE BYTES
7300 010000   02020          LD     BC,0             ;256 TIMES.
7303 21B870   02030          LD     HL,IOBUF         ;POINT AT DATA.
7306 CD0A73   02040          CALL   CHGLP1           ;DO IT.
7309 C9       02050          RET                     ;TO CHGCOD.
730A          02060 CHGLP1   EQU    $                ;ALTER DATA BY 1 ROUTINE.
730A 7E       02070          LD     A,(HL)           ;PUT THE BYTE IN A.
730B FE00     02080          CP     0                ;IF EOF PENCIL FILE,
730D C8       02090          RET    Z                ; RET TO CHGLP.
730E FEFF     02100          CP     0FFH             ;DON'T CREATE A
7310 CA2773   02110          JP     Z,WRNGFL         ; 0 BYTE.
7313 E5       02120          PUSH   HL               ;SAVE POINTER TO DATA.
7314 215D70   02130          LD     HL,FUNC          ;POINT TO INC/DEC VALUE.
7317 86       02140          ADD    A,(HL)           ;ALTER IT.
7318 E1       02150          POP    HL               ;RETRIEVE DATA POINTER.
7319 77       02160 PUTBK    LD     (HL),A           ;PUT IT BACK.
731A 23       02170          INC    HL               ;POINT TO NEXT BYTE.
731B 10ED     02180          DJNZ   CHGLP1           ;DO AGAIN IF NOT DONE.
731D C9       02190          RET                     ;TO CHGLP.
731E 3A4038   02200 CKBRK    LD     A,(BRKROW)       ;BREAK KEY
7321 E604     02210          AND    4                ; PRESSED?
7323 C22D40   02220          JP     NZ,DOS           ;YES.
7326 C9       02230          RET                     ;IF NOT.
7327          02240 WRNGFL   EQU    $                ;NOT A PENCIL FILE.
7327 213073   02250          LD     HL,BDFLMS        ;SO
732A CD6744   02260          CALL   DSPLY            ; SAY SO.
732D C32D40   02270          JP     DOS              ;GET OUT.
7330 54       02280 BDFLMS   DEFM   'This is not a PENCIL or ASCII file.'
    68 69 73 20 69 73 20 6E 6F 74 20 61 20 50 45 4E
    43 49 4C 20 6F 72 20 41 53 43 49 49 20 66 69 6C
    65 2E
7353 0D       02290          DEFB   13
7354          02300 CKEOF    EQU    $                ;CHECK FOR END OF FILE.
7354 FE1C     02310          CP     1CH              ;END OF FILE?
7356 2807     02320          JR     Z,SETEOF         ;YES.
7358 FE1D     02330          CP     1DH              ;PAST EOF?
735A 2803     02340          JR     Z,SETEOF         ;YES.
735C C36573   02350          JP     DOSERR           ;AN ERROR OCCURRED.
735F          02360 SETEOF   EQU    $                ;SET EOF FLAG.
735F 215B70   02370          LD     HL,EOFSW         ;SET EOF
7362 3601     02380          LD     (HL),1           ; SWITCH.
7364 C9       02390          RET                     ;TO CHGCOD.
7365          02400 DOSERR   EQU    $
7365 212040   02410          LD     HL,CURSOR        ;SET UP "PRINT@" CURSOR
7368 3600     02420          LD     (HL),00H         ; WITH SCREEN MEMORY
736A 23       02430          INC    HL               ; POSITION
736B 363E     02440          LD     (HL),3EH         ; 3E00H.
736D F5       02450          PUSH   AF               ;SAVE ERROR CODE.
736E 3A5C70   02460          LD     A,(EREOF)                ;EREOF CHARACTER.
7371 CD3300   02470          CALL   BYTOUT           ;DISPLAY IT.
7374 F1       02480          POP    AF               ;RESTORE THE ERROR CODE.
7375 C30944   02490          JP     DOSER            ;DISPLAY & EXIT.
7378          02500 REV      EQU    $
7378 010000   02510          LD     BC,0             ;TO LOOP 256 TIMES
737B 21B870   02520          LD     HL,IOBUF         ;POINT TO DATA.
737E          02530 REVLP    EQU    $
737E 7E       02540          LD     A,(HL)           ;GET A BYTE.
737F FE00     02550          CP     0                ;EOF PENCIL FILE?
7381 C8       02560          RET    Z                ;YES.
7382 FE20     02570          CP     ' '              ;SPACE?
7384 FA9373   02580          JP     M,REVLP1         ;LESS. GO TO NEXT BYTE.
7387 FE7B     02590          CP     'z'+1            ;GREATER THAN z?
7389 F29373   02600          JP     P,REVLP1         ;YES. GOTO NEXT BYTE.
738C C5       02610          PUSH   BC               ;SAVE LOOP COUNTER.
738D 47       02620          LD     B,A              ;MOVE THE BYTE TO B.
738E 3E9A     02630          LD     A,'z'+20H        ;TOP OF REVERSE SET.
7390 90       02640          SUB    B                ;GO TO OTHER END OF SET.
7391 77       02650          LD     (HL),A           ;PUT IT IN THE BUFFER.
7392 C1       02660          POP    BC               ;RESTORE LOOP COUNTER.
7393 23       02670 REVLP1   INC    HL               ;POINT TO NEXT BYTE.
7394 10E8     02680          DJNZ   REVLP            ;DO THE NEXT BYTE.
7396 C9       02690 REVLPX   RET                     ;WHEN 256 (OR EOF) DONE.
7397 21AC73   02700 PAUSE    LD     HL,CNTMSG        ;POINT TO CONTINUE MSG.
739A CD6744   02710          CALL   DSPLY            ;PRINT IT.
739D 215A70   02720          LD     HL,CONT          ;SET TO CONTINUE -
73A0 3601     02730          LD     (HL),1           ; (NO FURTHER PAUSES).
73A2 CD1E73   02740 PAULP    CALL   CKBRK            ;<BREAK> TO EXIT.
73A5 CD2B00   02750          CALL   CKKEY            ;ANY OTHER KEY
73A8 B7       02760          OR     A                ; TO CONTINUE.
73A9 28F7     02770          JR     Z,PAULP ;NO KEY YET.
73AB C9       02780          RET                     ;CONTINUE.
73AC 0A       02790 CNTMSG   DEFB   10               ;LINEFEED
73AD 50       02800          DEFM   'Press <BREAK> to exit or any other key'
    72 65 73 73 20 3C 42 52 45 41 4B 3E 20 74 6F 20
    65 78 69 74 20 6F 72 20 61 6E 79 20 6F 74 68 65
    72 20 6B 65 79
7303 20       02810          DEFM   ' to continue.'
    74 6F 20 63 6F 6E 74 69 6E 75 65 2E
73E0 03       02820          DEFB   3                ;END MSG WITHOUT C/R.
718C          02830          END    START
00000 TOTAL ERRORS
```

ASKFL	71C1	ASKFL1	71D3	ASKFN	7211	BASLP	719F	BDFLMS	7330
BKUP	4445	BRKRON	3040	BYTOUT	0033	CDDCMS	724C	CHGCOD	72B4
CHGLP	7300	CHGLP1	730A	CKBRK	731E	CKEOF	7354	CKKEY	002B
CLOSE	4428	CLS	01C9	CNTMSG	73AC	CODMSG	705E	CONT	705A
CR	7059	CURSOR	4020	DEC	723F	DOS	402D	DOSER	4409
DOSERR	7365	DRM	7000	DSPFL	71D9	DSPLY	4467	EOFSW	705B
EREOF	705C	EXTFL	441C	FLMSG	71E0	FLSPEC	706A	FUNC	705D
GETFL	7202	GETFN1	7217	INC	7236	IOBUF	709B	LBUFPT	40A7
LIPT	0361	LNR	7233	NINC	7248	OPEN	4424	PAULP	73A2
PAUSE	7397	PUTBK	7319	READ	4436	REV	7378	REVLP	737E
REVLP1	7393	REVLPX	7396	SETEOF	735F	SHOW	7299	START	718C
WRITE	443C	WRNGFL	7327						

ASSIGN THOSE LUN'S by John C. Adams, Jr., 208 Kaywood Avenue, Tullahoma, Tennessee 37388 [NOTE: This text file and both sample program files will appear on a TAS Public Domain Library disk in the near future]:

Introduction

In using the Microsoft FORTRAN package on the TRS-80 Model III computer the F80 compiler utilizes several different buffers, or logical units, for passing data to and from external devices such as disk drives, line printers, and the video display. Unless otherwise specified, logical unit numbers (LUN's) are assigned in Microsoft FORTRAN as follows:

LUN 1 and 3-5 to the screen or keyboard
LUN 2 to the line printer
LUN 6-10 to the disk drives.

In order to perform input and output (I/O) operations, you must first open a LUN between the computer and the I/O device. This is accomplished through an OPEN subroutine call using the following syntax:

CALL OPEN(logical unit number, 'filename', logical record length)

where the logical record length (LRL) is the length in bytes of each record. The LRL must be an integer constant or variable whose value lies between 1 and 256; it must be large enough to allow storage of all your data for that record. In practice the necessary length depends on your FORMAT statements and whether you are using direct or sequential disk accessing.

Most FORTRAN applications utilize an input file, an output file, and perhaps a data file for I/O communications between the user and the computer. It is up to the user to provide all necessary LUN and LRL information within his/her program, as well as all corresponding filenames. Enter INOUT, which is a general FORTRAN utility subroutine that can be INCLUDEd in your FORTRAN program to provide this capability.

Subroutine INOUT

Subroutine INOUT assigns LUN's (and corresponding LRL's and filenames) for input, output, and data files based on user keyboard (interactive) responses to screen prompt messages. With reference to the public domain disk file named INOUT/FOR, INOUT returns the LUN for the input file (LUNIN), the LUN for the output file (LUNOUT), and the LUN for a data file (LUNDAT) to the calling program as integer arguments in the subroutine call. If any LUN is entered by the user as a value greater than five (5), the user is asked to input the corresponding filename and LRL. In addition, if the output file LUN is entered as two (2), the user is prompted to ready the line printer for output. Any and all OPEN's are automatically generated within INOUT based upon the user-supplied responses. Note that all output WRITE statements in subroutine INOUT are assigned to unit three (3) which is the video display while all input READ statements are assigned to unit five (5) which also corresponds to the video.

Given in public domain disk file TESTIO/FOR is a simple test program called TESTIO which illustrates how subroutine INOUT can be easily INCLUDEd into any FORTRAN program. Once INOUT has been called at the beginning of your program, the LUN's for input, output, and data files are available for use. Any and all further I/O and associated file manipulations can be performed using these LUN's, e.g.,

```
READ( LUNIN, 100 ) AUTO,BIKE,WAGON
WRITE( LUNOUT, 200 ) HELP,TEMP
WRITE( LUNDAT ) (PLOT(I),I=1,10)
REWIND LUNDAT
ENDFILE LUNOUT
```

Program TESTIO contains examples of such LUN usage where a binary data file is written to LUNDAT, rewound, and read with output to LUNOUT using the number of data entries as controlled by input on LUNIN. It is suggested that the following be used as input to TESTIO:

```
LUN for Input File (LUNIN) = 5
LUN for Output File (LUNOUT) = 3
LUN for Data File (LUNDAT) = 8
Name of Data File:Drive = TESTIO/DAT:1
LRL for Data FIle (LRLDF) = 100
```

Experiment with other values, especially with regard to LRL for the data file.

It is necessary for the user to plan his/her usage of LUN's properly. For example, if there is no requirement for data file I/O in your program, simply answer the data file query with a file number between one and five (don't use two which is reserved for the line printer) that has not been used for either input or output files. Subroutine INOUT checks to be certain that you have not previously assigned a LUN to the current input value; if so, you are prompted to re-enter the current LUN using a different value. However, there is no checking for duplication of data file names or drives in INOUT so that it is the total responsibility of the user to assure that all data files are properly assigned. In a similar manner it is up to the user to properly assign the LRL's for each file based upon the data requirements for that file. If in doubt as to the LRL entry, use 256 which is the default value assigned by TRSDOS to its files.

[The following is the contents of public domain disk file INOUT/FOR:]

```
      SUBROUTINE INOUT( LUNIN, LUNOUT, LUNDAT )
C ==>    Input LUN's, Names, and LRL's
C ==>    for Input, Output, and Data Files
      BYTE IPRINT
      INTEGER*4 NAMEIF(6), NAMEOF(6), NAMEDF(6)
      DATA NAMEIF/6*' '/, NAMEOF/6*' '/, NAMEDF/6*' '/
      WRITE(3,7000)
  100 WRITE(3,9000)
      READ(5,8000) LUNIN
      IF ( LUNIN .EQ. 2 ) GO TO 100
      IF ( LUNIN .LT. 6 ) GO TO 200
      WRITE(3,9100)
      READ(5,8100) NAMEIF
  200 WRITE(3,9200)
      READ(5,8000) LUNOUT
      IF ( LUNOUT .EQ. LUNIN ) GO TO 200
      IF ( LUNOUT .LT. 6 ) GO TO 300
      WRITE(3,9300)
      READ(5,8100) NAMEOF
  300 IF ( LUNOUT .NE. 2 ) GO TO 400
      WRITE(3,9400)
      READ(5,8200) IPRINT
  400 WRITE(3,9500)
      READ(5,8000) LUNDAT
      IF ( LUNDAT .EQ. LUNIN .OR. LUNDAT .EQ. LUNOUT ) GO TO 400
      IF ( LUNDAT .LT. 6 ) GO TO 500
      WRITE(3,9600)
      READ(5,8100) NAMEDF
  500 IF ( LUNIN .LE. 5 ) GO TO 600
      WRITE(3,9700)
      READ(5,8300) LRLIF
      CALL OPEN( LUNIN, NAMEIF, LRLIF )
  600 IF ( LUNOUT .LE. 5 ) GO TO 700
      WRITE(3,9800)
      READ(5,8300) LRLOF
      CALL OPEN( LUNOUT, NAMEOF, LRLOF )
  700 IF ( LUNDAT .LE. 5 ) GO TO 800
      WRITE(3,9900)
      READ(5,8300) LRLDF
      CALL OPEN( LUNDAT, NAMEDF, LRLDF )
  800 CONTINUE
      RETURN
 7000 FORMAT('0','INPUT-OUTPUT-DATA FILE SPECIFICATIONS',/)
 8000 FORMAT(1I2)
 8100 FORMAT(6A4)
 8200 FORMAT(1A1)
 8300 FORMAT(1I3)
 9000 FORMAT('0','Logical Unit Number (1-10) for Input File? ')
```

```
9100  FORMAT(' Name:Drive for Input File? ')
9200  FORMAT(' Logical Unit Number (1-10) for Output File? ')
9300  FORMAT(' Name:Drive for Output File? ')
9400  FORMAT(' xx  Ready Printer for Output Print and Enter <P>  xx ')
9500  FORMAT(' Logical Unit Number (6-10) for Data File? ')
9600  FORMAT(' Name:Drive for Data File? ')
9700  FORMAT(' Logical Record Length for Input File? ')
9800  FORMAT(' Logical Record Length for Output File? ')
9900  FORMAT(' Logical Record Length for Data File? ')
      END
```

[The following is the contents of public domain disk file TESTIO/FOR:]

```
      PROGRAM TESTIO
C ==>   Test Program for Subroutine INOUT
      DIMENSION VAR(5), VALUE(5)
      DATA PI/3.141593/
      CALL INOUT( LUNIN, LUNOUT, LUNDAT )
      WRITE( 3, 100 ) LUNIN, LUNOUT, LUNDAT
10    WRITE( LUNOUT, 200 )
      READ( LUNIN, 300 ) NUM
      IF ( NUM .GT. 5 ) GO TO 10
      DO 20 I=1,NUM
      VAR(I) = SIN( I*PI/180.0 )
      WRITE( LUNOUT, 400 ) I, VAR(I)
20    CONTINUE
      WRITE( LUNDAT ) (VAR(I), I=1,NUM)
      REWIND LUNDAT
      READ( LUNDAT ) (VALUE(I), I=1,NUM)
      DO 30 I=1,NUM
      WRITE( LUNOUT, 400 ) I, VALUE(I)
30    CONTINUE
      ENDFILE LUNDAT
      STOP
100   FORMAT('0','LUNIN=',I3,5X,'LUNOUT=',I3,5X,'LUNDAT=',I3)
200   FORMAT(' NUMBER OF SIN TERMS ( <=5 )? ')
300   FORMAT(1I2)
400   FORMAT(1I4,10X,1F10.6)
      END
C ==>   INCLUDE Subroutine INOUT Stored as /FOR or /TXT File
      INCLUDE INOUT
```

TELEPHONE DIALER PROGRAM by Dave McGlumphy - For those of you too lazy to dial the phone yourself, here's a program to do it for you, courtesy of Dave McGlumphy:

```
                00010 ;DIALER - Dave McGlumphy  4429 Paula Lane
                00020 ;Chattanooga, Tennessee 37415   03/26/84
                00030 ;MCI# 181-7759
                00040 ;This program has model III defaults but also runs on a
                00050 ;model 1 without changes.  I don't know about model IV
                00060 ;It controls the "REMOTE" plug of the cassette cable
                00070 ;Use that plug to control a normally-closed relay which
                00080 ;is in series with the red or green wire of the phone
                00090 ;line to achieve pulse dialing.  If it dials too fast
                00100 ;for your phone system, increase the pause duration
                00110 ;at the two places in the code that say "WITHIN DIGIT"
7000            00120         ORG     7000H
3840            00130 BRKROW  EQU     3840H         ;KEYBOARD ROW WITH BREAK
033A            00140 BYTOUT  EQU     33AH          ;DISPLAY SINGLE BYTE
002B            00150 CKKEY   EQU     2BH           ;CHECK FOR KEY-PRESS
01C9            00160 CLS     EQU     1C9H          ;CLEAR SCREEN
4020            00170 CURSOR  EQU     4020H         ;CURSOR POSITION LSB
4405            00180 DODOS   EQU     4405H         ;EXIT & DO DOS COMMAND
402D            00190 DOS     EQU     402DH         ;DOS EXIT
4467            00200 DSPLY   EQU     4467H         ;DISPLAY MSG ON SCREEN
40A7            00210 LBUFPT  EQU     40A7H         ;LINE INPUT BUFFER PTR
0361            00220 LIPT    EQU     361H          ;BASIC LINE INPUT RTN
0060            00230 PAUSE   EQU     60H           ;(BC) DETERMINES DURATION
403D            00240 PORTF1  EQU     403DH         ;MODEL 1 CASS PORT FLAG
4210            00250 PORTF3  EQU     4210H         ;MODEL III CASS PORT FLAG
7000            00260 START   EQU     $
7000 21A641     00270         LD      HL,41A6H      ;PLUG THE DISK BASIC
7003 0615       00280         LD      B,15H         ; LINKS FROM 41A6H
7005 36C9       00290 BASLP   LD      (HL),0C9H     ; TO 41E2H
7007 23         00300         INC     HL
7008 23         00310         INC     HL
7009 23         00320         INC     HL
700A 10F9       00330         DJNZ    BASLP         ; WITH RETURNS
700C 3A5400     00340         LD      A,(54H)       ;CHECK MODEL 1/III
700F 3D         00350         DEC     A             ;Z IF MODEL 1
7010 CC5F71     00360         CALL    Z,MODEL1      ;INITIALIZE FOR MODEL 1
7013 21A740     00370         LD      HL,LBUFPT     ;SET UP THE
7016 36CA       00380         LD      (HL),0CAH     ; LINE
7018 23         00390         INC     HL            ; INPUT
7019 3664       00400         LD      (HL),64H      ; BUFFER
701B            00410 MENU    EQU     $             ;DISPLAY THE OPTIONS
701B 21ED71     00420         LD      HL,SCREEN     ;PT AT MENU SCREEN
701E CD6744     00430         CALL    DSPLY         ;PRINT IT
7021            00440 MENGET  EQU     $             ;WHAT FUNCTION?
7021 CDA871     00450         CALL    GETKEY        ;GET UPPERCASE REQUEST
7024 FE41       00460         CP      'A'
7026 2836       00470         JR      Z,ATC         ;YES. PLAY AIR TRAFFIC
7028 FE42       00480         CP      'B'           ;B?
702A 2838       00490         JR      Z,BASIC       ;BRING UP BASIC
702C FE43       00500         CP      'C'           ;C?
702E 283A       00510         JR      Z,MODM80      ;BRING UP MODEM80/CMD
7030 FE44       00520         CP      'D'           ;D?
7032 283C       00530         JR      Z,DIL68M      ;YES. CALL 6800 BOARD
7034 FE45       00540         CP      'E'           ;E?
7036 2840       00550         JR      Z,GETPHN      ;YES. GET KEYED PHONE#
7038 FE46       00560         CP      'F'           ;F?
703A 2852       00570         JR      Z,DILCRB      ;YES. CALL CRABAPPLE
703C FE47       00580         CP      'G'           ;G?
703E 2856       00590         JR      Z,DILMCI      ;YES. CALL MCI
7040 FE48       00600         CP      'H'           ;H?
7042 285A       00610         JR      Z,DILDCS      ;YES. CALL CHRIS SMITH
7044 FE49       00620         CP      'I'
7046 285E       00630         JR      Z,DILWSM      ;CALL BUTCH
7048 FE4A       00640         CP      'J'
704A 2862       00650         JR      Z,DILPET      ;CALL PETE
704C FE4B       00660         CP      'K'
704E 2866       00670         JR      Z,BOOT        ;A HALT INSTRUCTION
7050 FE4C       00680         CP      'L'
7052 2863       00690         JR      Z,DILWOM      ;DIAL WOMACK'S BBS
7054 FE4D       00700         CP      'M'
7056 2867       00710         JR      Z,HNG         ;HANG UP
7058 FE4E       00720         CP      'N'
705A 2868       00730         JR      Z,PIC         ;PICK UP THE PHONE
705C            00740 MENUXT  EQU     $             ;EXIT FROM MENU FUNCTIO
705C 18C3       00750         JR      MENGET        ;GO TO GET NEXT FUNCTIO
705E 217173     00760 ATC     LD      HL,ATCPGM     ;POINT AT ATC COMMAND
7061 C30544     00770         JP      DODOS         ;GO DO IT
7064 217573     00780 BASIC   LD      HL,BASPGM
7067 C30544     00790         JP      DODOS         ;GO INTO BASIC
706A 217B73     00800 MODM80  LD      HL,MODPGM
706D C30544     00810         JP      DODOS         ;EXECUTE MODEM80/CMD
7070 213C72     00820 DIL68M  LD      HL,M68PHN
7073 CDC970     00830         CALL    DIAL          ;DIAL 68MICRO JOURNAL
7076 18E4       00840         JR      MENUXT        ;START OVER
7078            00850 GETPHN  EQU     $             ;GET PHONE# FROM KEYBOARD
7078 21E371     00860         LD      HL,NBRMSG     ;POINT TO NUMBER? MSG
707B CD6744     00870         CALL    DSPLY         ;PRINT IT
707E 2AA740     00880         LD      HL,(LBUFPT)   ;POINT TO INPUT BUFFER
7081 E5         00890         PUSH    HL            ;SAVE IT
7082 CD6103     00900         CALL    LIPT          ;GET THE NUMBER
7085 E1         00910         POP     HL            ;POINT AT PHONE #
7086 CDC970     00920         CALL    DIAL          ;DIAL IT
7089 CDA871     00930         CALL    GETKEY        ;PAUSE FOR A KEYPRESS
708C 18CE       00940         JR      MENUXT        ;START OVER
708E 217772     00950 DILCRB  LD      HL,CRBPHN     ;POINT TO CRABAPPLE PHONE
7091 CDC970     00960         CALL    DIAL          ;DIAL IT
7094 18C6       00970         JR      MENUXT        ;START OVER
7096 219872     00980 DILMCI  LD      HL,MCIPHN     ;POINT TO MCI PHONE#
7099 CDC970     00990         CALL    DIAL          ;DIAL IT
709C 18BE       01000         JR      MENUXT        ;START OVER
709E 21BF72     01010 DILDCS  LD      HL,DCSPHN     ;POINT AT CHRIS' PHONE#
70A1 CDC970     01020         CALL    DIAL          ;DIAL IT
70A4 18B6       01030         JR      MENUXT        ;START OVER
70A6 21E072     01040 DILWSM  LD      HL,WSMPHN     ;POINT AT BUTCH'S PHONE
70A9 CDC970     01050         CALL    DIAL          ;DIAL IT
70AC 18AE       01060         JR      MENUXT        ;START OVER
70AE 210773     01070 DILPET  LD      HL,PETPHN     ;POINT AT PETE'S PHONE#
70B1 CDC970     01080         CALL    DIAL          ;DIAL IT
70B4 18A6       01090         JR      MENUXT        ;START OVER
70B6 76         01100 BOOT    HALT                  ;CAUSES BOOT
70B7 213873     01110 DILWOM  LD      HL,WOMPHN     ;POINT AT WOMACK'S PHN
70BA CDC970     01120         CALL    DIAL          ;DIAL IT
70BD 189D       01130         JR      MENUXT        ;START OVER
```

```
70BF CD7471   01140 HNG     CALL    HANGUP          ;HANG UP THE PHONE
70C2 1898     01150         JR      MENUXT          ;START OVER
70C4 CD8E71   01160 PIC     CALL    PICKUP          ;PICK UP THE PHONE
70C7 1893     01170         JR      MENUXT          ;START OVER
70C9          01180 DIAL    EQU     $               ;HL POINTS TO PHONE#
70C9 E5       01190         PUSH    HL              ;SAVE IT
70CA CD7471   01200         CALL    HANGUP          ;HANGUP THE PHONE
70CD 010000   01210         LD      BC,0            ;DO A LONG
70D0 CD6000   01220         CALL    PAUSE           ; PAUSE
70D3 CD8E71   01230         CALL    PICKUP          ;PICK IT UP. (DIALTONE)
70D6 010000   01240         LD      BC,0            ;DO A LONG
70D9 CD6000   01250         CALL    PAUSE           ; PAUSE
70DC E1       01260         POP     HL              ;RESTORE PTR TO PHONE#
70DD 2B       01270         DEC     HL              ;POINT TO BYTE BEFORE
70DE E5       01280         PUSH    HL              ; PHONE# & SAVE IT
70DF          01290 GETDIG  EQU     $               ;GET PHONE DIGIT
70DF CD0071   01300         CALL    CKBRK           ;BREAK EXITS TO DOS
70E2 E1       01310         POP     HL              ;RESTORE PHONE# PTR
70E3 23       01320         INC     HL              ;POINT TO NEXT DIGIT
70E4 E5       01330         PUSH    HL              ;SAVE PHON PTR
70E5 7E       01340         LD      A,(HL)          ;PUT IN A
70E6 FE00     01350         CP      0               ;END?
70E8 2873     01360         JR      Z,EXIT          ;YES
70EA FE0D     01370         CP      13              ;C/R?
70EC 286F     01380         JR      Z,EXIT          ;YES
70EE FE30     01390         CP      30H             ;< 0 ?
70F0 FA0F71   01400         JP      M,GETDIG        ;YES
70F3 FE3A     01410         CP      39H+1           ;> 9 ?
70F5 F20F71   01420         JP      P,GETDIG        ;YES
70F8 F5       01430         PUSH    AF              ;SAVE A
70F9 CD3A03   01440         CALL    BYTOUT          ;PRINT IT
70FC F1       01450         POP     AF              ;RESTORE IT
70FD D630     01460         SUB     30H             ;CHG ASCII# TO HEX#
70FF FE00     01470         CP      0               ;ZERO?
7101 2002     01480         JR      NZ,HAVDIG       ;NO
7103 C60A     01490         ADD     A,10

7105          01500 HAVDIG  EQU     $               ;HAVE DIGIT TO DIAL
7105 328773   01510         LD      (PULSCT),A      ;SAVE IT
7108 CD1371   01520         CALL    DIGIT           ;DIAL THE DIGIT
710B 01A861   01530         LD      BC,25000        ;PAUSE BETWEEN
710E CD6000   01540         CALL    PAUSE           ; DIGITS
7111 18CC     01550         JR      GETDIG          ;DO NEXT DIGIT

7113          01560 DIGIT   EQU     $
7113 3A8773   01570         LD      A,(PULSCT)      ;# PULSES LEFT
7116 FE00     01580         CP      0               ;DONE YET?
7118 C8       01590         RET     Z               ;YES. GOTO NEXT DIGIT
7119 3D       01600         DEC     A               ;SUBTRACT 1 &
711A 328773   01610         LD      (PULSCT),A      ; SAVE IT
711D 3A8673   01620         LD      A,(PORT)        ;PUT CASSETTE PORT #
7120 4F       01630         LD      C,A             ; IN C
7121 3A8473   01640         LD      A,(MODEL)       ;MODEL 1/III?
7124 FE01     01650         CP      1               ;1?
7126 2805     01660         JR      Z,DIGM1         ;YES
7128 3A1842   01670         LD      A,(PORTF3)      ;GET PORT SETTINGS
712B 1803     01680         JR      DIGXOR          ;TURN OFF CASSETTE
712D 3A3040   01690 DIGM1   LD      A,(PORTF1)
7130          01700 DIGXOR  EQU     $               ;RESET PROPER MODEL
7130 218373   01710         LD      HL,CASBIT       ; CASSETTE BIT
7133 AE       01720         XOR     (HL)            ;TURN OFF
7134 ED79     01730         OUT     (C),A           ; CASSETTE MOTOR
7136 019808   01740         LD      BC,2200         ;PUT DURATION IN BC
7139 CD6000   01750         CALL    PAUSE           ;WITHIN DIGIT
713C 3A8673   01760         LD      A,(PORT)        ;WHICH PORT TO CHANGE
713F 4F       01770         LD      C,A             ;PUT IT IN C
7140 3A8473   01780         LD      A,(MODEL)       ;MODEL 1/III?
7143 FE01     01790         CP      1               ;1
7145 2805     01800         JR      Z,DIGM1A
7147 3A1842   01810         LD      A,(PORTF3)      ;MODEL III PORT FLAGS
714A 1803     01820         JR      DIGAND          ;TO TURN ON CASSETTE
714C 3A3040   01830 DIGM1A  LD      A,(PORTF1)      ;MODEL 1 CASSETTE FLAGS
714F 218373   01840 DIGAND  LD      HL,CASBIT       ;MODEL CASS MTR BIT
7152 A6       01850         AND     (HL)            ; TO TURN IT ON
7153 ED79     01860         OUT     (C),A
7155 019808   01870         LD      BC,2200         ;PUT DURATION IN BC
7158 CD6000   01880         CALL    PAUSE           ;WITHIN DIGIT
715B 1886     01890         JR      DIGIT
715D          01900 EXIT    EQU     $               ;HOUSEKEEP & EXIT
715D E1       01910         POP     HL
715E C9       01920         RET
715F 3EFF     01930 MODEL1  LD      A,0FFH          ;LOAD PORT WITH

7161 328673   01940         LD      (PORT),A        ;MODEL 1 ADDRESS
7164 218473   01950         LD      HL,MODEL        ;PT TO MODEL BYTE
7167 3601     01960         LD      (HL),1          ;INDICATE MODEL1
7169 218373   01970         LD      HL,CASBIT       ;PT TO CASBIT
716C 3604     01980         LD      (HL),4          ;PUT 4 IN IT
716E 21E437   01990         LD      HL,14308        ;SELECT CASSETTE
7171 3601     02000         LD      (HL),1          ; #2
7173 C9       02010         RET
7174          02020 HANGUP  EQU     $               ;HANG UP THE PHONE
7174 3A8673   02030         LD      A,(PORT)        ;PUT CASSETTE PORT#
7177 4F       02040         LD      C,A             ;INTO C
7178 3A8473   02050         LD      A,(MODEL)       ;WHICH 'PUTER?
717B FE01     02060         CP      1               ;MODEL 1?
717D 2805     02070         JR      Z,HANG1         ;YES
717F 3A1842   02080         LD      A,(PORTF3)      ;GET CASS PORT FLAGS
7182 1803     02090         JR      HANGOR          ;GO TURN OFF
7184 3A3040   02100 HANG1   LD      A,(PORTF1)      ;GET MOD1 CASS PORT FLAGS
7187 218373   02110 HANGOR  LD      HL,CASBIT       ;MODEL CASS MTR BIT
718A AE       02120         XOR     (HL)            ; TO TURN OFF
718B ED79     02130         OUT     (C),A           ;TURN OFF MOTOR
718D C9       02140         RET                     ;FROM HANG-UP
718E          02150 PICKUP  EQU     $               ;PICK UP THE PHONE
718E 3A8673   02160         LD      A,(PORT)        ;PUT CASSETTE PORT#
7191 4F       02170         LD      C,A             ;INTO C
7192 3A8473   02180         LD      A,(MODEL)       ;WHICH 'PUTER?
7195 FE01     02190         CP      1               ;MODEL 1?
7197 2805     02200         JR      Z,PICK1         ;YES
7199 3A1842   02210         LD      A,(PORTF3)      ;GET CASS PORT FLAGS
719C 1803     02220         JR      PICAND          ;GO TURN ON
719E 3A3040   02230 PICK1   LD      A,(PORTF1)      ;GET MOD1 CASS PORT FLAGS
71A1 218373   02240 PICAND  LD      HL,CASBIT       ;MODEL CASS MTR BIT
71A4 A6       02250         AND     (HL)            ; TO TURN ON
71A5 ED79     02260         OUT     (C),A           ;TURN ON MOTOR
71A7 C9       02270         RET                     ;FROM PICKUP
71A8 CD0071   02280 GETKEY  CALL    CKBRK           ;<BREAK> EXITS TO DOS
71AB 3E00     02290         LD      A,0             ;CLEAR KEY BUFFER
71AD CD2B00   02300         CALL    CKKEY           ;KEY PRESSED?
71B0 B7       02310         OR      A               ;CONDITION THE FLAGS
71B1 28F5     02320         JR      Z,GETKEY        ;NO KEY PRESSED
71B3 F5       02330         PUSH    AF              ;SAVE A
71B4 DD212040 02340         LD      IX,CURSOR       ;SET UP PRINT@ POSITION
71B8 3E58     02350         LD      A,58H           ; AT
71BA DD7700   02360         LD      (IX),A          ; END
71BD 3E3F     02370         LD      A,3FH           ; OF
71BF DD7701   02380         LD      (IX+1),A        ; MENU
71C2 3E1F     02390         LD      A,31            ;ERASE EOF
71C4 CD3A03   02400         CALL    BYTOUT          ;DO IT
71C7 3E0E     02410         LD      A,14            ;TURN CURSOR
71C9 CD3A03   02420         CALL    BYTOUT          ; ON
71CC F1       02430         POP     AF              ;GET A AGAIN
71CD F5       02440         PUSH    AF              ;SAVE IT AGAIN
71CE CD3A03   02450         CALL    BYTOUT          ;PRINT IT
71D1 F1       02460         POP     AF              ;RESTORE A
71D2 FE5B     02470         CP      'Z'+1           ;LOWERCASE?
71D4 F8       02480         RET     M               ;NO
71D5 D620     02490         SUB     20H             ;MAKE IT UPPERCASE
71D7 C9       02500         RET                     ;WITH KEY IN A
71D8 F5       02510 CKBRK   PUSH    AF              ;SAVE A
71D9 3A4038   02520         LD      A,(BRKROW)      ;BREAK KEY
71DC E604     02530         AND     4               ; PRESSED?
71DE C22D40   02540         JP      NZ,DOS          ;YES
71E1 F1       02550         POP     AF              ;RESTORE A
71E2 C9       02560         RET                     ;NO
71E3 08       02570 NBRMSG  DEFB    8               ;BACKSPACE CHARACTER
71E4 4E       02580         DEFM    'NUMBER?        '
    55 4D 42 45 52 3F 09
71EC 03       02590         DEFB    3               ;END LINE WITH NO C/R
71ED 1C       02600 SCREEN  DEFB    1CH             ;HOME CURSOR
71EE 1F       02610         DEFB    1FH             ;EREOF (CLEAR SCREEN)
71EF 41       02620         DEFM    'A.     ATC
(<BREAK>      EXITS   '
    2E 09 41 54 43 09 09 09 09 28 3C 42 52 45 41 4B
    3E 09 45 58 49 54 53 09

7208 54       02630         DEFM    'TO     DOS.)
    4F 09 44 4F 53 2E 29
7210 0A       02640         DEFB    10
7211 42       02650         DEFM    'B.     BASIC'
    2E 09 42 41 53 49 43
7219 0A       02660         DEFB    10
```

```
721A 43        02670          DEFM    'C.     MODEM80'
     2E 09 4D 4F 44 45 4D 38 30
7224 0A        02680          DEFB    10
7225 44        02690          DEFM    'D.     68MICRO JOURNAL....     '
     2E 09 36 38 4D 49 43 52 4F 09 4A 4F 55 52 4E 41
     4C 2E 2E 2E 2E 09
723C 38        02700 M68PHN   DEFM    '842-6809'
     34 32 2D 36 38 30 39
7244 00        02710          DEFB    0
7245 0A        02720          DEFB    10
7246 45        02730          DEFM    'E.     ENTER   A       NUMBER  TO DIAL'
     2E 09 45 4E 54 45 52 09 41 09 4E 55 4D 42 45 52
     09 54 4F 09 44 49 41 4C
725F 0A        02740          DEFB    10
7260 46        02750          DEFM    'F.     CRABAPPLE.....          '
     2E 09 43 52 41 42 41 50 50 4C 45 2E 2E 2E 2E 2E
     2E 2E 2E 2E 2E 09
7277 38        02760 CRBPHN   DEFM    '875-6835'
     37 35 2D 36 38 33 35
727F 00        02770          DEFB    0
7280 0A        02780          DEFB    10
7281 47        02790          DEFM    'G.     MCI.................    '
     2E 09 4D 43 49 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E
     2E 2E 2E 2E 2E 09
7298 31        02800 MCIPHN   DEFM    '1-800-323-7751'
     2D 38 30 30 2D 33 32 33 2D 37 37 35 31
72A6 00        02810          DEFB    0
72A7 0A        02820          DEFB    10
72A8 48        02830          DEFM    'H.     CHRIS   SMITH.......   '
     2E 09 43 48 52 49 53 09 53 4D 49 54 48 2E 2E 2E
     2E 2E 2E 2E 2E 09
72BF 38        02840 DCSPHN   DEFM    '899-5377'
     39 39 2D 35 33 37 37
72C7 00        02850          DEFB    0
72C8 0A        02860          DEFB    10
72C9 49        02870          DEFM    'I.     BUTCH...............    '
     2E 09 42 55 54 43 48 2E 2E 2E 2E 2E 2E 2E 2E 2E
     2E 2E 2E 2E 2E 09
72E0 31        02880 MSMPHN   DEFM    '1-614-695-3056'
     2D 36 31 34 2D 36 39 35 2D 33 30 35 36
72EE 00        02890          DEFB    0
72EF 0A        02900          DEFB    10
72F0 4A        02910          DEFM    'J.     PETE................    '
     2E 09 50 45 54 45 2E 2E 2E 2E 2E 2E 2E 2E 2E 2E
     2E 2E 2E 2E 2E 09
7307 38        02920 PETPHN   DEFM    '870-1324'
     37 30 2D 31 33 32 34
730F 00        02930          DEFB    0
7310 0A        02940          DEFB    10
7311 4B        02950          DEFM    'K.     BOOT    THE     SYSTEM'
     2E 09 42 4F 4F 54 09 54 48 45 09 53 59 53 54 45
     4D
7323 0A        02960          DEFB    10
7324 4C        02970          DEFM    'L.     MOHACK  BBS.....      '
     2E 09 57 4F 4D 41 43 48 09 42 42 53 2E 2E 2E 2E
     2E 2E 2E 2E 2E 09
7338 38        02980 MOHPHN   DEFM    '891-8136'
     39 31 2D 38 31 33 36
7343 00        02990          DEFB    0
7344 0A        03000          DEFB    10
7345 4D        03010          DEFM    'M.     HANG    UP      THE     PHONE'
     2E 09 48 41 4E 47 09 55 50 09 54 48 45 09 50 48
     4F 4E 45
7359 0A        03020          DEFB    10
735A 4E        03030          DEFM    'N.     PICK    UP      THE     PHONE
     2E 09 50 49 43 4B 09 55 50 09 54 48 45 09 50 48
     4F 4E 45 09 09
7378 03        03040          DEFB    03
7371 41        03050 ATCPGM   DEFM    'ATC'
     54 43
7374 0D        03060          DEFB    13
7375 42        03070 BASPGM   DEFM    'BASIC'
     41 53 49 43
737A 0D        03080          DEFB    13
737B 4D        03090 MODPGM   DEFM    'MODEM80'
     4F 44 45 4D 38 30
7382 0D        03100          DEFB    13
7383 02        03110 CASBIT   DEFB    2               ;MIII CASSETTE MTR BIT
7384 03        03120 MODEL    DEFB    3               ;MIII (1=MODEL 1)
7385 FF        03130 PORT1    DEFB    0FFH            ;M1 CASSETTE PORT
7386 3C        03140 PORT     DEFB    03CH            ;MIII CASSETTE PORT
7387 00        03150 PULSCT   DEFB    0               ;PULSE COUNT
7000           03160          END     START
00000 TOTAL ERRORS

ATC     705E    ATCPGM  7371    BASIC   7064    BASLP   7005    BASPGM  7375
BOOT    7086    BRKRDM  3040    BYTOUT  033A    CASBIT  7383    CKBRK   710F
CKKEY   002B    CLS     01C9    CRBPHN  7277    CURSOR  4020    DCSPHN  72BF
DIAL    70C9    DIGAND  714F    DIGIT   7113    DIGM1   712D    DIGM1A  714C
DIGXOR  7130    DIL68M  7070    DILCRB  708E    DILDCS  709E    DILMCI  7096
DILPET  70AE    DILMOM  7087    DILMSM  70A6    DODOS   4405    DOS     402D
DSPLY   4467    EXIT    715D    GETDIG  700F    GETKEY  71A8    GETPHN  7078
HANG1   7104    HANGUP  7174    HANXOR  7187    HAVDIG  7105    HNC     708F
LBUFPT  40A7    LIPT    0361    M68PHN  723C    MCIPHN  7298    MENGET  7021
MENU    701B    MENUXT  705C    MODEL   7384    MODEL1  715F    MODMB0  706A
MODPGM  7378    NBRMSG  71E3    PAUSE   0060    PETPHN  7307    PIC     70C4
PICAND  71A1    PICK1   719E    PICKUP  718E    PORT    7386    PORT1   7385
PORTF1  403D    PORTF3  4210    PULSCT  7387    SCREEN  71ED    START   7000
MOMPHN  7338    MSMPHN  72E0
```

CHRISTIANS AND COMPUTERS - I have recently heard about several organizations which serve Christians using computers. In previous issues of Northern Bytes, I have mentioned Christian Computer-Based Communications, 44 Delma Drive, Toronto, Ontario M8W 4N6. In addition, I have learned of the existence of the EDP Network News, which is "a newsletter for Presbyterians interested in church uses for computers." To get the newsletter without charge for the balance of 1984, contact EDP Network News, Roberta Parrett, Synod of the Covenant, 6172 Busch Avenue, Suite 3000, Columbus, Ohio 43229. Finally, the Church Computer Users Network was begun by United Methodists a couple of years ago, but I hear that they welcome participation by Christians of other denominations. Memberships are $15 a year and they publish a newsletter. For information contact Clyde McDonald, 159 Ralph McGill Blvd., Atlanta, Georgia 30365.

EXPRESSION INPUT ROUTINE by Bill Coulter - This is a tiny routine that helps certain BASIC programs. It allows any valid BASIC expression to be entered at the INPUT prompt. For example, the dialogue with a function-plot program might go this way (keyboard entries underlined):

X value? LOG(7)
Y value? EXP(PI/2)

The program would proceed with the intended values. The reply for X will work in any case; the reply for Y will work provided PI has already been assigned a value by the program. I much prefer this to copying values from a $40 calculator over to a $2000 computer!

The BASIC code for handling the input to X (above) is:

```
10 LINE INPUT "X value? ";U$: X = USR(0)
```

See the program listing below for a complete example.

Two ROM routines are called up, both documented in Decker's fine book. Let LINE INPUT U$ take the user's reply; the text will be in BASIC's input buffer. The routine at 1BC0H is called to reduce the text to token form. Then the evaluation routine at 2337H is fired; if successful it deposits the result in BASIC's ACCUM at 4121H. Since the interpreter thinks it's processing program code, error trapping should be used to keep the game going.

The USR code - both POKE values and Z80 - is in lines 1000 through 1030 of the sample program listing below:

```
10 US$="0123456789": PI=3.14159265: E=2.7182818
20 U=VARPTR(U$): U=PEEK(U+1)+256*PEEK(U+2): DEFUSR=U
30 FOR I=U TO U+9: READ U: POKE I,U: NEXT
99 ON ERROR GOTO 2000
100 LINE INPUT "X value? ";U$: X=USR(0)
101 PRINT TAB(10+LEN(U$)) CHR$(27)"= "X
110 LINE INPUT "Y value? ";U$: Y=USR(0)
111 PRINT TAB(10+LEN(U$)) CHR$(27)"= "Y
120 PRINT STRING$(30,131): GOTO100
1000 DATA 42,167,64:'     LD   HL,(40A7H)    ;HL=>INBUF
1010 DATA 205,192,27:'    CALL 1BC0H         ;TOKENIZE
1020 DATA 35:'            INC  HL            ;FIX POINTER
1030 DATA 195,55,35:'     JP   2337H         ;EVAL & RET
2000 PRINT "*** INPUT ERROR - TRY AGAIN."
2010 IF ERL=100 THEN RESUME 100
2020 IF ERL=110 THEN RESUME 110
```

Piratable Software For Your TRS-80

(Models I, III and 4)

Double sided "flippy" diskettes filled with excellent public domain software.

Three volumes Available NOW! $10 per volume.

THE ALTERNATE SOURCE

704 North Pennsylvania
Lansing, MI 48906
(517) 482-8270

We'll Keep You Compatible

In today's competitive market, portability is the name of the game. If you have TRS-80 Model I/III software or Model 4 software, and you want to run it on other machines, The Alternate Source has tools that can help.

TRS-80 Model 4 and 2000 and Sanyo owners, our conversion programs are the most versatile on the market. We handle dozens of little time consuming "tweaks" for you – automatically.

Naturally we add spaces. All the conversion programs do that. We also allow you to "define" the video screen. Generally, after converting a program, you must change all the "PRINT @" statements, including those defined with variables, to accommodate the 80 x 24 screen. We take care of that, automatically, plus we allow you to specify an "offset".

Perhaps you want your screen "indented" eight spaces. Eighty characters (the TRSDOS 6.x and MSDOS screen width), minus sixty-four characters (the "old" TRS-80 screen width), is sixteen. Our program will let you put eight spaces on each side of the currently defined screen simply by typing,

X 8

And perhaps you want to move the existing screen down four spaces. Simply type:

Y 4

But you don't have to worry about a lot of commands. You can get the ball rolling this easily:

IN "oldfile"
OUT "newfile"
RUN

In just a short time (depending on length), your BASIC program will be ready to run on the new machine.

We fix several other "gotcha's" that will pop up when converting code, usually just about the time you figure everything's running right. We believe that we perform more of this drudgery than any similar program on the market. No need to bore you with many details because now you won't need to worry about them!

We now have three conversion utilities, with more on the way. Make sure you specify which computer you're using! Check out our low prices, too. Buy a copy for a friend and do your friend, TAS and yourself a favor!

BASANYO and BAS2000 are $39.95 until September 1, 1984. After that date, $49.95. BAS34 is $29.95 with no planned price increase. The two MSDOS versions include routines that will convert from I, III AND 4; BAS34 just includes the routines to convert from Models I and III to Model 4. Please include three bucks for our shipping department. They're threatning to go on welfare.

The Alternate Source is located at 704 North Pennsylvania Avenue, Lansing, MI, 48906, Phone (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)

POSTMASTER: If undeliverable return to:
The Alternate Source, 704 N. Pennsylvania, Lansing, MI 48906

To: