WISCONSIN COMPUTER SOCIETY

NEWSLETTER

Volume #3, Issue #1 January 1978 Don Stevens, Editor

MEETING NOTICE

Our meeting will be held at 1:30 p.m., Saturday, January 7, 1978, at the Waukesha Technical Institute (Room 202 - Administration Building).

PROGRAM AGENDA

Club members are invited to bring their computer systems to this meeting to demonstrate. There will be general discussions as to the scope of future club meetings as well as discussions on software and hardware exchange.

MEMBERSHIP DUES

Members are urged to send in their $3.00 for membership thru June 1978. Better yet, bring your money to the meeting!!!!!

OTHER INFO

Julian Jetzer of Sheboygan has answered the call and has agreed to become Associate Editor and will be charge of 6800 Software Information.

Future meetings - February 4, March 18, and April 1, 1978

Next month's issue will feature lengthy article on Bill Mack's TI-990 Executive Routine.

Anyone interested in S-100 16-K Boards (TMS-4044-25 chips) Fully Static 250ns for less than $350.00 - Group Purchase possible.

Richard Akeson (Phone 271-1840) is offerring Phi Deck Tape Systems for $30.00 each. Available the end of January - more info at meeting.

The Milwaukee Sentinel - Tuesday, Dec. 6, 1977 issue featured an article "Computer - the Maid of Future" which featured clubmember JIM WHITE.

Don Stevens

Don Stevens, Editor
Wisconsin Computer Society Newsletter
P.O. Box 159
Sheboygan Falls, Wisc. 53085

Received the following note from clubmember DOROTHY DEAN regarding the info she has researched on club incorporation:

There are basically two kinds of non-profit (for tax purposes) status: 501(c)(7) and 501(c)(3). The second one is the more desirable of the two since that is the one which allows the individual to deduct from his/her income taxes the amount of their donation. The (c)(7) status provides no such benefit to the donor.

501(c)(7) is non-profit status for "Clubs organized for pleasure, recreation, and other non profitable purposes, substantially all of the activities of which are for such purposes and no part of the net earnings of which inures to the benefit of any private shareholder."

The example that is usually used for the kinds of organizations that are eligible for 501(c)(7) is a flying club where the club owns expensive equipment and the members pay a specific amount each month to maintain the equipment and to use it. The benefit in having tax exemption is that the club (which must be a corporation) pays no taxes on the money that comes in from members. It might be a sizeable amount depending on the number of members and the type, quantity and cost of the equipment. Members do not get tax deductions for their contributions to the club.

501(c)(3) organizations are those "... organized and operated exclusively for religious, charitable, scientific, testing for public safety, literary or educational purposes,..." The computer club would probably not qualify for religious status or charitable status. The definition of "scientific" applies to the kind of work done at universities toward a specific goal. Cancer research research might qualify but from the description I don't see any way we could stretch the fefinition of what the club is to fit the IRS definition. Testing for public safety and literary are also not possible for the computer club. The only one that might be a possibility is the category "educational." In order to get this classification, it is necessary to show that the club does either of the following:

(a) The instruction or training of the individual for the purposes of improving or developing his(sic) capabilities; or
(b) The instruction of the public on subjects useful to the individual and beneficial to the community.

In order to get this designation, however, substantially all of the club's activity would have to be directed toward education in the form of lectures, forums, public discussion groups, etc. These would have to be on-going, not one-shot deals. I think that it would have to mean a major reorganization of the club.

I suggest that at this time the club incorporate as a non-profit, non-stock corporation in Wisconsin and forgo the attempt to get a tax-exemption from the federal government. If, at some point in the future, some generous soul wants to donate one of the 3000 series IBM's to the club and doesn't need a tax write-off, then the club could consider applying for a 501(c)(7) as a "social club" and house and maintain the computer for the use of members and not have to pay tax on any revenue whatever the form.

6800 Software Jan. 1978

Compiled by: Julian E. Jetzer Sheboygan

Our thanks for this months 6800 software goes out to Stephen Heinecke of Allenton, Wis. In fact, we will be featuring four of Steve's fine 6800 programs during the next few months. His first offering is based on the Thompson Lister which appeared in the October 1976 Byte. The modifications have been extensive, however, and the result is a program which not only lists, but also allows one to program sections of memory, make corrections, make insertions and much more. Some of the other features include the ability to calculate the address for branch instructions simply by entering FF as the address of the branch and then entering the correct location while listing the program back. It also has a read and write function which allows you to type in texts as simply as you would type out a letter and then read it back. A clear function clears memory from 0020 to 0Cff...a K function will KILL the clear function or the CLEAR function KILLS itself after a cycle so you can't erase your program. Also included are transfer memory and memory search functions. The other available routines are self explanatory as you will see.

The ASCII print out strings have not been included in the listing but the location at which they are called have been marked with an *. Just program the proper ASCII characters in the locations pointed to and add 00 for a delimiter and you'll be all set. Be careful..... program of memory is not contiguous so watch your addresses as you enter the code.

If you have a problem and would like the program on KC 300 Baud cassette or paper tape just let me know or ask Don the honorable editor and he can give me your name. For cassette, a blank tape would be appreciated.

Steve has done a good job on this program and once you learn how to use it you'll save time for other projects.

A sample run of the program is included to get you started. I have also included a Mikbug (R) format Punch Dump if you wish to load that way. All Hex is included.

Happy New Year and Happy Computing.....

Sample Run of Programming Aid

See Code Designations in Listing for Program.....

```
*L
*G
ENTER CODE. P
ADDRESS 0100

0100 CE  0000
0103 8D  FF
0105 C6  30
0107 17
0108 5C
0109 BD  E0[illegible] 6B
010C BD  E0CC
010F C1  39
0111 23  FF
0113 08
0114 BD  E07E
0117 39
0118 G
*G
ENTER CODE. L
ADDRESS 0100

0100 CE  0000
0103 8D  FF -- 0114-0F
0105 C6  30
0107 17
0108 5C
0109 BD  E0[illegible]6B
010C BD  E0CC
010F C1  39
0111 23  FF -- 0107-F4
0113 08
0114 BD  E07E
0117 39
0118 00
0119 00 esc
ENTER CODE. I
ADDRESS 0100

0100 CE  0000+    NOTE: change symbol
0100 CE  0118
0100 CE  0118esc
ENTER CODE. W
ADDRESS 0118

A COUNTER:crlfesc NOTE: carriage return and line feed
crlf
A COUNTER:crlf

0125
ENTER CODE. K
```

NOTE: all underlined characters are the ones I would have entered

```
ENTER CODE. W
ADDRESS 0126
crlf
THAT'S ALL.crlfesc
crlf
THAT'S ALL.crlf

0136
ENTER CODE. U
ADDRESS 0100

A COUNTER:
0 1 2 3 4 5 6 7 8 9
THAT'S ALL

ENTER CODE. A
A COUNTER:
0 1 2 3 4 5 6 7 8 9
THAT'S ALL.

ENTER CODE. S
BDFF
0109 BD  E0CA
010C BD  E0CC
0114 BD  E07E
ENTER CODE. T 0100 0200  0040
ENTER CODE. I
ADDRESS 0200

0200 CE  0118+
0200 CE  0218
0200 CE  0218esc
ENTER CODE. U
ADDRESS 0200

A COUNTER:
0 1 2 3 4 5 6 7 8 9
THAT'S ALL.

ENTER CODE. M
*
```

And that is how to use the programming aid

SET A048-A049 TO 0E5C

```
Locn  B1 B2 B3

0D00  86 10       SEARCH  LDAA I   10      Print a home up
0D02  BD E1 D1            JSR E    OUTEEE
0D05  86 16               LDAA I   16      Print an erase to end of frame
0D07  BD E1 D1            JSR E    OUTEEE

0D0A  CE A0 20            LDX I    A020    Set index to pattern start

0D0D  BD 0F 36    SRPATT  JSR E    BYTINN  Get a byte and store it
0D10  86 FF               CMPA I   FF      Is this a stop byte
0D12  26 F9               BNE      SRPATT  if not then continue building pattern

0D14  CE 00 00            LDX I    0000    Set index to start of area to be searched

0D17  7F 0D 21    SRCH2   CLR E    0D21    Clear area pointer
0D1A  86 20               LDAA I   20      Set pattern pointer to first byte
0D1C  B7 0D 27            STAA E   0D27    of pattern to be searched for

0D1F  B6 A0 20    SRCH3   LDAA E   A020    Get a pattern byte
0D22  81 FF               CMPA I   FF      Is this a stop byte
0D24  27 0C               BEQ      FOUND   if so then pattern has been found
0D26  A1 00               CMPA X   X+00    If not then does it match area of search
0D28  26 0B               BNE      NOTFND  if not then pattern not found

0D2A  7C 0D 27            INC E    0D27    Increment area pointer
0D2D  7C 0D 21            INC E    0D21    Increment pattern pointer
0D30  20 ED               BRA      SRCH3   Go check next byte

0D32  BD 0D 40    FOUND   JSR E    SRPRT   If found go print where found
0D35  08          NOTFND  INX              Increment index
0D36  8C 0C FF            CPX I    0CFF    Is this last byte of search
0D39  26 DC               BNE      SRCH2   if not continue search
0D3B  39                  RTS              if so the return to control

0D40  BD 0F 53    SRPRT   JSR E    LINE    Print the line number
0D43  E6 00               LDAB X   X+00    Save the instruction byte in B
0D45  BD E0 CA            JSR E    OUT2HS  Print the instruction by e and
0D48  BD E0 CC            JSR E    OUTS    and 2 spaces
0D4B  86 3D               LDAA I   3F      Pointer for return address
0D4F  BD 0F 70            JSR E    TDCDR   Go to Thompson decoder
0D50  FE A0 0C            LDX E    XHI     Get the old area address
0D53  39                  RTS              and return
```

```
Locn   B1 B2 B3

OD60   8D 24      TRNSFR  BSR      TADS    Get the from address
OD62   FF OD 6D           STX      OD6D    Store at from pointer
OD65   8D 1F              BSR      TADS    Get the to address
OD67   FF OD 70           STX      OD70    Store at to pointer
OD6A   8D 1A              BSR      TADS    Get the length
OD6C   B6 XX XX   TX2     LDAA E   ????    Get a byte to be transfered
OD6F   B7 XX XX           STAA E   ????    Store it in the new location
OD72   7C OD 6E           INC E    OD6E    Increment the low from pointer
OD75   26 03              BNE      TX3     If low pointer now equals zero
OD77   7C OD 6D           INC E    OD6D    Then increment high from pointer
OD7A   7C OD 71   TX3     INC E    OD71    Increment high topointer
OD7D   26 03              BNE      TX4     If low to pointer euqals zero
OD7F   7C OD 70           INC E    OD70    then increment high to pointer
OD82   09         TX4     DEX              Decrement the length counter
OD83   26 E7              BNE      TX2     If count not zero then continue transfers
OD85   39                 RTS              Otherwise return to control

OD86   BD EO CC   TADS    JSR E    OUTS    Print a space
OD89   7E EO 47           JSR E    DADDR   and then get an address


ODA0   CE OC EO   CLEAR   LDX I    OCEO    Clear out allmmemory below OD00
ODA3   6F 1F      CLR2    CLR X    X+20    except for the first 32 bytes
ODA5   09                 DEX
ODA6   26 FB              BNE      CLR2
ODA8   86 6D      KILLCL  LDAA I   6D      Kill the clear function
ODAA   B7 OD A3           STAA E   CLR2
ODAD   CE OF 30           LDX I    OF30    Get the address of a RTS instruction
ODB0   FF OE 84           STX E    OE84    Store it at C
ODB3   FF OE 94           STX E    OE94    Store it at K
ODB6   39                 RTS              Return to control
```

```
Locn   B1 B2 B3

0DC0   BD 0F 45  INSERT  JSR E   ADS1   Go get a starting address

0DC3   BD 0F 53  INS2    JSR E   LINE   Print the line number
0DC6   FF A0 18          STX E   A018   Save the line number at A018
0DC9   E6 00             LDAB X  X+00   Save the instruction byte in B
0DCB   BD 0F 42          JSR E   BYTOUT Print the instruction byte
0DCE   BD 0F 4E          JSR E   2SPACE and 2 spaces
0DD1   86 3D             LDAA I  3D     Pointer for return address
0DD3   BD 0F 70          JSR E   TDCDR  Go to Thompson decoder

0DD6   BD E1AAC  INS3    JSR E   INEEE  Go get a character
0DD9   81 1B             CMPA I  ASCIIesc Is it an ASCII escape
0DDB   26 01             BNE     INS4   if not continue
0DDD   39                RTS             if so return to control
0DDE   81 0D     INS4    CMPA I  ASCIIcr Is it an ASCII carriage return
0DE0   27 E1             BEQ     INS2   If so then go to the next line
0DE2   81 2B             CMPA I  ASCII+ Is it an ASCII "+"
0DE4   26 F0             BNE     INS3   if not then ignore it and get another

0DE6   FE A0 18          LDX E   A018   Get the line number we saved
0DE9   BD 0F 53          JSR E   LINE   Go reprint it
0DEC   BD 0F 36          JSR E   BYTINN Go get a new instruction byte
0DEF   16                TAB             and save it in B
0DF0   BD 0F 4E          JSR E   2SPACE Go print 2 spaces
0DF3   86 31             LDAA I  31     Pointer for return address
0DF5   BD 0F 70          JSR E   TDCDR  Go to Thompson decoder
0DF8   FE A0 18          LDX E   A018   Get the line number again
0DFB   20 C6             BRA     INS2   And go print it again
```

```
Locn   B1 B2 B3

0E00   BD 0F 45   WRITE   JSR E    ADS1     Get a starting address
0E03   BD E1 AC   WRT2    JSR E    INEEE    Get a character
0E06   8D 02              BSR      WRTPRC   Process the character
0E08   20 F9              BRA      WRT2     And the get another

0E0A   81 02      WRTPRC  CMPA I   02       Is this an ASCII control-B
0E0C   27 0C              BEQ      WRTBCK   if so the do backspace routine
0E0E   81 0D              CMPA I   ASCIIcr   Is this an ASCII carriage return
0E10   27 0E              BEQ      WRTCR    if so do carriage return routine
0E12   81 1B              CMPA I   ASCIIecs  Is this an ASCII escape character
0E14   27 12              BEQ      WTEXIT   if so do exit routin

0E16   A7 00      WRTSTR  STAA X   X+00      Store the character
0E18   08                 INX               Increment the index pointer
0E19   39                 RTS               And return

0E1A   09         WRTBCK  DEX               Back up the index pointer
0E1B   86 18              LDAA I   18       Print out a backspace
0E1D   7E E1 D1   WRTPRT  JMP E    OUTEEE

0E20   8D F4      WRTCR   BSR      WTRSTR   Store the carriage return
0E22   86 0A              LDAA I   ASCIIlf   Get a line feed character
0E24   8D F0              BSR      WTRSTR   Store it
0E26   20 F5      W       BRA               And the print it

0E28   86 04      WTEXIT  LDAA I   ASCIIeot  Get ASCII eot character
0E2A   A7 00              STAA I   X+00      and store it to end the string
0E2C   FE A0 0C           LDX E    XHI       Get the starting address
0E2F   31                 DES               Reset the stack to return to control
0E30   31                 DES
0E31   BD E0 7E           JSR E    PDATA1    Reprint the string
0E34   7E 0F 53           JMP E    LINE      Print the last address
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0E37   BD 0F 45   READ    JSR E    ADSL      Get the starting address
0E3B   BD E0 7E           JSR E    PDATA     Print out the string
0E3D   7E 0F 53           PMP E    LINE      Print the last addtess
```

```
Locn   B1 B2 B3

*0E40  CE 0E C0   CONTROL LDX I      0EC0     Print "crlfENTER CODE. "
 0E43  BD E0 7E            JSR E     PDATA1
 0E46  BD E1 AC            JSR E     INEEE    Get a character
 0E49  80 41               SUBA I    41       If it's less than A
 0E4B  25 F3               BCS       CONTRL   then go get another code
 0E4D  81 1A               CMPA I    1A       If it's greater than Z
 0E4F  24 EF               BCC       CONTRL   then go get another code
 0E51  8B 40               ADDA I    40       Compute code address
 0E53  48
 0E54  B7 0E 59            STAA E    0E59
 0E57  FE 0E 96            LDX E               Select a program from our vast file
 0E5A  AD 00               JSR X     X+00     go to a program
 0E5C  20 E2               BRA       CONTRL   Go get another code

 0E60  BD 0F 45            JSR E     ADS1     Get the address of the program
 0E63  6E 00               JMP X     X+00     and then go see if it does what you
                                              thought the thing would do.
```

Locn	B1	B2	B3	Letter	Function
0E80	01	00		A	Goes to usual beginning address of most programs
0E82	0F	30		B	The address of a RTS instruction (NOP)
0E84	0D	A0		C	Clears memory from 0CFF down to 0020
0E86	0F	30		D	NOP
0E88	0F	30		E	NOP
0E8A	0F	30		F	NOP
0E8C	0F	30		G	NOP
0E8E	0F	30		H	NOP
0E90	0D	C0		I	Insert: allows changes and insertinos
0E92	0F	30		J	NOP
0E94	0D	A8		K	Kill the clear function above to avoid mistakes
0E96	0F	0F		L	Lists a program in the memory
0E98	E0	D0		M	Returns to MIKBUG monitor
0E9A	0F	30		N	NOP
0E9C	0F	30		O	NOP
0E9E	0F	00		P	Programs a section of memory
0EA0	0F	30		Q	NOP
0EA2	0E	37		R	Reads back text to be printed and the last address
0EA4	0D	00		S	Searches for patterns
0EA6	0D	60		T	Transfers sections of memory
0EA8	0E	60		U	Goes to user selected program or subroutine for testing
0EAA	0F	30		V	NOP
0EAC	0E	00		W	Writes into memory text to be printed
0EAE	0F	30		X	NOP
0EB0	0F	30		Y	NOP
0EB2	0F	30		Z	NOP

```
Locn   B1 B2 B3

0F70   C1 8D      TDCDR   CMPB I    8D       If this is a BSR then
0F72   27 21              BEQ       RELTV    go to relative address exit
0F74   C1 8C              CMPB I    8C       If this is a CPX I then
0F76   27 1A              BEQ       3BYTE    go to 3 byte exit
0F78   C1 8E              CMPB I    8E       If this is a LDS I then
0F7A   27 16              BEQ       3BYTE    go to 3 byte exit
0F7C   C1 CE              CMPB I    CE       If this is a LDX I then
0F7E   27 12              BEQ       3BYTE    go to 3 byte exit
0F80   C4 F0              ANDB I    F0       Mask off bottom 4 bits
0F82   C1 20              CMPB I    20       If this is a branch then
0F84   27 0F              BEQ       RELTV    go to relative address exit
0F86   C1 60              CMPB I    60       If this is inherently addressed
0F88   25 09              BCS       1BYTE    go to 1 byte exit
0F8A   C4 30              ANDB I    30       Mask off top 2 bytes
0F8C   C1 30              CMPB I    30       If this is extendedly addressed then
0F8E   27 02              BEQ       3BYTE    go to 3 byte exit
0F90   4C         2BYTE   INCA               Here is the 2 byte exit
0F91   4C                 INCA
0F92   4C         3BYTE   INCA               Here is th 3 byte ex it
0F93   4C         1BYTE   INCA               Here is the 1 byte exit
0F94   4C                 INCA
0F95   C6 0F      RELTV   LDAB I    0F       Here is the relative exit ****
0F97   36                 PSHA               Push the return address onto
0F98   37                 PSHB               the stack and return to it
0F99   39                 RTS
                  **** NOTE: Location 0F96 is part of the return address
                  *          the lister/programer can be relocated anywhere
                  *          as long as this (and the other) pointer is
                  *          changed to represent the proper addresses
```

```
Locn   B1 B2 B3

OF9A   E6 00       RELCMP  LDAB X    X+00     Get the relative address byte
OF9C   BD E0 CA            JSR E     OUT2HS   Print it + a space, increment pointer
OF9F   FF A0 0E            STS E     TEMP     Store the pointer
OFA2   CE 0E F8            LDX I     0EF8     Print an arrow
OFA5   8D C6               BSR       PRINT
OFA7   CE A0 0A            LDX I     A00A     set pointer for computations
OFAA   5C                  INCB               Is the address  hex FF
OFAB   27 1F               BEQ       RELCAL   if so then go Calculate address
OFAD   5A                  DECB               Restore address
OFAE   2B 0A               BMI       RCMPNG   If negative go to subtract section
OFB0   EB 05               ADDB X    X+05     Otherwise add the address to the
OFB2   A6 04               LDAA X    X+04     location
OFB4   E7 01               STAB X    X+01     And store lower half
OFB6   89 00               ADCA I    00
OFB8   20 0A               BRA       RCMPOT   go store lower half and go out
OFBA   50          RCMPNG  NEGB               Subtract the address from the
OFBB   A6 05               LDAA X    X+05     location
OFBD   10                  SBA
OFBE   A7 01               STAA X    X+01     and store lower half
OFC0   A6 04               LDAA X    X+04
OFC2   82 00               SBCA I    00
OFC4   A7 00       RCMPOT  STAA X    X+00     Store upper half (either computation)
OFC6   BD E0 C8            JSR E     OUT4HS   Print the address
OFC9   EE 02               LDX X     X+02     Restore pointer
OFCB   39                  RTS                and return

OFCC   BD E0 47    RELCAL  JSR E     BADDR    Get the destination
OFCF   CE A0 0C            LDX I     A00C     Set pointer for calculations
OFD2   E6 01               LDAB X    X+01     Get the destination into the
OFD4   A6 00               LDAA X    X+00     accumulators and
OFD6   E0 03               SUBB X    X+03     subtract the location from it
FFD8   A2 02               SBCA X    X+02
OFDA   26 05               BNE       NEGBRA   If A ≠ 0 go check negative branch
OFDC   5D                  TSTB               If B is negative then
OFDE   2B 15               BMI       TOOFAR   Branch is too far positive
OFDF   20 06               BRA       RELSTR   Othewise go store it
OFE1   4C          NEGBRA  INCA               Is A = hex FF
OFE2   26 10               BNE       TOOFAR   If not branch is too far negative
OFE4   5D                  TSTB               Otherwise is B positive
OFE5   2A 0D               BPL       TOOFAR   if so the branch is still too negative
OFE7   86 3D       RELSTR  LDAA I    ASCII =  Othewise, Print an "="
OFE9   BE E1 D1            JSR E     OUTEEE
OFEC   EE 02               LDX X     X+02     Restore the pointer
OFEE   09                  DEX                Back it down one
OFEF   E7 00               STAB X    X+00     Store the address
OFF1   7E E0 CA            JMP E     OUT2HS   and Print it and return
* OFF4 CE 0E D0    TOOFAR  LDX I     0ED0     Print "TOO FAR..."
OFF7   31                  INS                and the return to control
OFF8   31                  INS
OFF9   7E E0 7E            JMP E     PDATA1
```

```
P
S1130D0086 10BDE1D18616BDE1D1CEA020BD0F363F
S1130D1081FF26F9CE00007F0D278620B70D21B66E
S1130D20A02081FF270CA100260B7C0D277C0D2120
S1130D3020EDBD0D40088C0CFF26DC3900000000BE
S1130D40BD0F53E600BDE0CABDE0CC863DBD0F70CB
S1130D50FEA00C3900000000000000000000000AC
S1130D608D24FF0D6D8D1FFF0D708D1AB60000B719
S1130D7000007C0D6E26037C0D6D7C0D7126037CBA
S1130D800D700926E739BDE0CC7EE0470000000085
S1130D900000000000000000000000000000004F
S1130DA0CE0CE06F1F0926FB866DB70DA3CE0F3066
S1130DB0FF0E84FF0E943900000000000000000C4
S1130DC0BD0F45BD0F53FFA018E600BD0F42BD0F78
S1130DD04E863DBD0F70BDE1AC811B26013981 0DEE
S1130DE027E1812B26F0FEA018BD0F53BD0F361648
S1130DF0BD0F4E8631BD0F70FEA01820C600000046
S1130E00BD0F45BDE1AC8D0220F98102270C810D97
S1130E10270E811B2712A70008390986187EE1D105
S1130E208DF4860A8DF020F58604A700FEA00C310F
S1130E3031BDE07E7E0F53BD0F45BDE07E7E0F5376
S1130E40CE0EC0BDE07EBDE1AC804125F3811A2405
S1130E50EF8B4048B70E59FE0E96AD0020E200001D
S1130E60BD0F456E000000000000000000000000FF
S1130E70BD0F45AD00BDE1AC20F600000000000050
S1130E8001000F300DA00F300F300F300F300F3036
S1130E900DC00F300DA80F0FE0D00F300F300F0032
S1130EA00F300E370D000D600E600F300E000F3046
S1130EB00F300F3000000000000000000000000B0
S1130EC00D0A454E5445522043 4F44452E200400FC
S1130ED00A0D0A544F4F20464152202021212104 5B
S1130EE01041444452455353203F2020160404002B
S1130EF00D0A040000000002D3E20043E200400E2
S1130F008D438D4F8D30168D4586318D6320F38D45
S1130F1034C60EF7A0208D3BE6008D268D30863D2D
S1130F208D4E7AA02026EFC610BDE1AC811B26E3CE
S1130F30392003398D00BDE055A700083920 5B39FD
S1130F408D007EE0BF8D1B20028D1A7EE0518D0046
S1130F507EE0CCFFA00C8D12CEA00CBDE0C8FEA09C
S1130F600C39CE0EE08D06BDE047CE0EF07EE07E5D
S1130F70C18D2721C18C271AC18E2716C1CE2712F5
S1130F80C4F0C120270FC1602509C430C130270235
S1130F904C4C4C4C4CC60F363739E600BDE0CAFF0A
S1130FA0A00ECE0EF88DC6CEA00A5C271F5A2B0ABF
S1130FB0EB05A604E7018900200A50A60510A70145
S1130FC0A6048200A700BDE0C8EE0239BDE047CE0A
S1130FD0A00CE601A600E003A2026055D2B152065
S1130FE0064C26105D2A0D863DBDE1D1EE0209E7CF
S1130FF0007EE0CACE0ED031317EE07E00000004D7
*
```

```
RUN
WEIGHT WATCHER'S RECORD

WEEK            WEIGHT          DIFFERENCE
 1               153             0
 2               149.5           -3.5
 3               147.5           -2
 4               147.5           0
 5               145             -2.5
 6               144.5           -.5
 7               141             -3.5
 8               141.5           .5
 9               139.25          -2.25
 10              139.5           .25
 11              137.5           -2
 12              138.5           1

AVG. WEEKLY LOSS  -1.20833 LBS.
LBS. TO GOAL   16.5
TOTAL POUNDS LOST SO FAR  -14.5

WEIGHT WATCHER'S GRAPH

      100  110  120  130  140  150  160  170  180  190  200
    I  +    +    +    +    +    +    +    +    +    +    +
 1  I                             *
 2  I                          *
 3  I                         *
 4  I                         *
 5  I                        *
 6  I                        *
 7  I                     *
 8  I                     *
 9  I                    *
 10 I                    *
 11 I                   *
 12 I                    *
```

THE SECOND GRAPH SHOWS THE SAME DATA AS THE FIRST, BUT IT IS SPREAD OUT OVER A BETTER RANGE.

```
WANT A CUSTOMIZED GRAPH? YES
WHAT IS THE SMALLEST NUMBER YOU WANT(INSTEAD OF 100)? 130
WHAT IS THE LARGEST NUMBER YOU WANT(INSTEAD OF 200)? 150
      130                  140                  150
    I  +                    +                    +
 1  I                                              *
 2  I                                           *
 3  I                                        *
 4  I                                        *
 5  I                                     *
 6  I                                   *
 7  I                              *
 8  I                               *
 9  I                          *
 10 I                          *
 11 I                     *
 12 I                        *
WANT ANOTHER GRAPH? NO
```

REPRINT FROM PITTSBURGH AREA NEWSLETTER

BINARY/OCTAL/DECIMAL/HEX CONVERSION TABLE

Below is a conversion table for numbers through 11111111_2, 377_8, 255_{10}, and ff_{16}. Thanks to the digital group who published this in their Clearinghouse

Examples:·····$01001010_2 = 112_8 = 74_{10} = 4A_{16}$·····$11000111_2 = 307_8 = 199_{10} = C7_{16}$·····

X \ Y	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F	BINARY HEX
0000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	DECIMAL
0	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17	OCTAL
0001	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
1	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	
0010	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
2	40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57	
0011	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
3	60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77	
0100	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	
4	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117	
0101	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	
5	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137	
0110	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	
6	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157	
0111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	
7	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177	
1000	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	
8	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217	
1001	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	
9	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237	
1010	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	
A	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257	
1011	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	
B	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277	
1100	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	
C	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317	
1101	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	
D	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337	
1110	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	
E	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357	
1111	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	
F	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377	

POWERS OF 2, 8 & 16

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
×1	1	2	4	8	16	32	64	128	256	512	1 024	2 048	4 096	8 192	16 384	32 768

	8^0	8^1	8^2	8^3	8^4	8^5
×1	1	8	64	512	4 096	32 768
×2	2	16	128	1 024	8 192	··
×3	3	24	192	1 536	12 288	··
×4	4	32	256	2 048	16 384	··
×5	5	40	320	2 560	20 480	··
×6	6	48	384	3 072	24 576	··
×7	7	56	448	3 584	28 672	··

	16^0	16^1	16^2	16^3
× 1	1	16	256	4 096
× 2	2	32	512	8 192
× 3	3	48	768	12 288
× 4	4	64	1 024	16 384
× 5	5	80	1 280	20 480
× 6	6	96	1 536	24 576
× 7	7	112	1 792	28 672
× 8	8	128	2 048	32 768
× 9	9	144	2 304	36 864
×10(A)	10	160	2 560	40 960
×11(B)	11	176	2 816	45 056
×12(C)	12	192	3 072	49 152
×13(D)	13	208	3 328	53 248
×14(E)	14	224	3 584	57 344
×15(F)	15	240	3 840	61 440

These tables show the powers of 2, 8 & 16 and their multiples far enough to allow conversion to decimal 65,535 (64K). This is equivalent to the 16-bit binary number (1111111111111111), the 6-digit octal number (177777) and the 4-digit hex number (FFFF). For examples, see the conversions used in Beginner-bits, page 6.

To improve the readability of the graph, we can also print the week in which each weighing was made. To do this we'll reserve six spaces on the left for printing X (the week number) in column 2, and the symbol "I" in column 4. Putting all these things together gives us:

```
PRINT X;TAB(4);"I";TAB((W-100)/2+6);"*"
```

Thus for X=21 and W=150 we'd have as part of our graph:

```
Col.   0123456789.....................31  (=25+6)
 21 I                      *
```

Automatic Scaling

We can generalize this idea by using a starting weight called A (instead of 100), and a final weight called B (instead of 200). This makes the scale factor 50/(B-A) spaces per pound. The translation is now A pounds (not 100), and the starting weight at the left edge of the graph is W-A (not W-100). This gives us as a generalized print statement:

```
PRINT X;TAB(4);"I";TAB((W-A)*(50/(B-A))+6);"*"
```

It will also be necessary to generalize the headings at the top of the graph, and this is done in a similar manner.

Let's look at a program that does all this "customized" scaling in a subroutine (lines 315 to 480). The first time the subroutine is used, the weights go from 100 to 200 (line 180). But then the user is asked to supply a more personalized set of minimum and maximum weights. These are input as A and B in lines 280 to 301. This program also contains the user's "goal" weight as the first number in DATA statement 900. This way the program can tell the dieter how many "pounds-to-goal" there are. The -1 at the end of the DATA is used to stop the READ loop (see line 80).

This program was written for a terminal with 70 columns. On a 40 column terminal, the number 50 in lines 340, 400, and 460 should be changed to 30. A listing and sample run of the program are given on the next two pages.

Note: This program is taken from the book, "BASIC and the Personal Computer" (Addison Wesley Co., Reading MA 01867). The book won't be out until December so some excerpts are being supplied to PACC for possible use in its BASIC course.

Another course that may be of interest to PACC members is a two-day workshop called "Personal Computing" to be given at Pitt on November 12 & 19 (Saturdays), from 8:30 AM to 5 PM. For further information call the Pitt Informal Program office at 624-6829, or inquire in person at room 407 Cathedral of Learning.

```
10 PRINT"WEIGHT WATCHER'S RECORD"
20 PRINT:PRINT"WEEK","WEIGHT","DIFFERENCE"
30 S=0
35 REM-----CALC. & PRINT TABLE----------
40 READ G
50 FOR I=1 TO 99
60 READ W
70 IF I=1 THEN 110
80 IF W<0 THEN 140
90 D=W-W1
100 S=S+D
110 PRINT I,W,D
120 W1=W
130 NEXT I
140 PRINT:PRINT"AVG. WEEKLY LOSS  ";S/(I-1);"LBS."
150 PRINT"LBS. TO GOAL  ";W1-G
160 PRINT"TOTAL POUNDS LOST SO FAR  ";S
170 PRINT:PRINT"WEIGHT WATCHER'S GRAPH":PRINT
175 REM-----STANDARD SCALE(100-200)----------
180 A=100:B=200
190 GOSUB 315
250 REM-----CUSTOMIZED SCALE----------
260 PRINT:PRINT"WANT A CUSTOMIZED GRAPH";:INPUT A$
270 IF A$="NO" THEN 999
280 PRINT"WHAT IS THE SMALLEST NUMBER YOU WANT(INSTEAD OF 100)";
290 INPUT A
300 PRINT"WHAT IS THE LARGEST NUMBER YOU WANT(INSTEAD OF 200)";
301 INPUT B
303 GOSUB 315
305 PRINT"WANT ANOTHER GRAPH";:INPUT A$
307 IF A$="YES" THEN 280
309 GOTO 999
315 REM-----GRAPH SUBROUTINE----------
316 X=0
317 REM-----HEADING(LINE 1)----------
330 FOR I=A TO B STEP 10
340 PRINT TAB(X*10*(50/(B-A))+5);I;
350 X=X+1
360 NEXT I
370 PRINT
375 REM-----HEADING(LINE 2)----------
380 PRINT"    I";
390 FOR I=0 TO (X-1)
400 PRINT TAB(I*50*(10/(B-A))+7);"+";
410 NEXT I
420 PRINT
425 RESTORE
426 READ G
428 REM-----PRINT GRAPH----------
430 FOR I=1 TO 99
440 READ W
450 IF W<0 THEN 480
460 PRINT I;TAB(4);"I";TAB((W-A)*(50/(B-A))+6);"*"
470 NEXT I
480 RETURN
900 DATA 122,153,149.5,147.5,147.5,145,144.5,141,141.5,139.25
910 DATA 139.5,137.5,138.5,-1
```

SUBROUTINE 315-480 IS USED FOR BOTH THE STANDARD AND CUSTOMIZED SCALES. THE VALUES OF A AND B MAKE THE DIFFERENCE.

Club incorporation - by Dorothy Dean - continued

It is possible to get the Postal Service to designate a group as a non-profit for postal purposes (lower postage rate, not first class service) without getting an IRS designation as 501(c)(3). Incorporating now will solve the problem of Don Stevens personal liability.

An even simpler procedure than incorporating is to open a club bank account. The officers are then responsible for the account and any transactions. The only thing that is needed is a Federal Employer's Number which the club can request from the IRS.

This is the extent of my research, at least as much as I think makes sense to put in the newsletter. I have more info but I can give that verbally at the meeting. Notice how the networking we discussed last time fits well into the social club status. If we wanted equipment that was too expensive or if we wanted to run part of the operation like a quasi-business with certain members receiving services of the network, then the club could consider the 501(c)(7). This can be brought up at the meeting.

****

GOOD THINGS TO READ

Mini-Micro Systems (Nov.-Dec. 1977 issue) is devoted completely to Microcomputer Systems.

EDN (Nov. 20, 1977 issue) is also devoted completely to Microcomputer Systems.

Microcomputer Interfacing - Sample & Hold Devices (American Lab - Nov.)

Microcomputer Interfacing - Analog Multiplexers (American Lab - Dec.)

Microcomputer Interfacing - Preparing Your Programs (Computer Design - November 1977)

****

PAY YOUR MEMBERSHIP FEE

PAY YOUR MEMBERSHIP FEE

PAY YOUR MEMBERSHIP FEE

LAST NEWSLETTER FOR UNPAID MEMBERS - - LAST NEWSLETTER FOR UNPAID MEMBERS

****

The Scaling of Graphs in BASIC

REPRINT FROM PITTSBURGH AREA NEWSLETTER

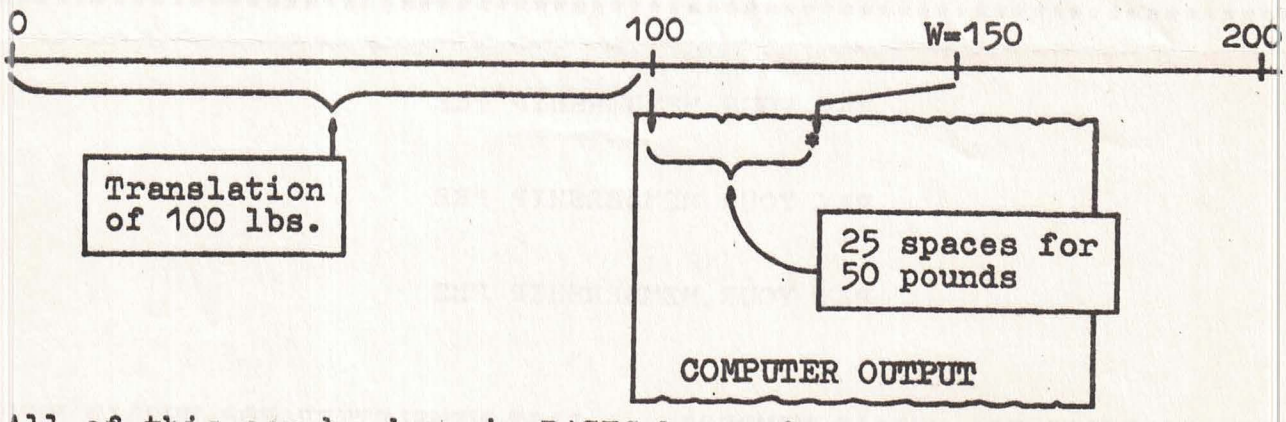
Margot Critchfield and Tom Dwyer

The best bet for making numerical output from a program useful is to show it in graphical form. Pictures make output easy to interpret and even easier to remember.

Most personal computer systems have an alphanumeric output device (either hard copy or CRT type) on which simple but useful graphs can be plotted by using the TAB(X) function in BASIC. The main difficulty is that the range of numbers to be plotted seldom matches the number of columns across the page (or screen) of the terminal. Also, terminals vary. Many CRT screens are limited to 40 columns, while hard copy terminals can handle 80 or even 132 columns. The solution to both these problems is to scale the numbers to be plotted to a range which fits on the terminal being used.

Scaling actually involves two operations. The first is an addition (or subtraction) of a number which translates all the data. The second is a multiplication by a number called a scale factor. For example, suppose you want to plot a graph of a dieter's weights, and the actual weights go from 100 to 200 pounds. But you want to squeeze the graph into 50 columns on a terminal. Terminals have columns numbered 0, 1, 2, 3, ... etc. So the first thing to do is translate (which here means move left) all the weights so that 100 pounds corresponds to column zero on the terminal. This is done by subtracting 100 from each weight W.

The next problem is to squeeze the weights from 100 to 200 into 50 terminal spaces. This can be done by multiplying each weight by a scale factor of 50/(200-100) = 1/2 terminal spaces per pound. Example: For a weight of 150 pounds, the program should first translate this weight by taking 150-100=50. It should then scale it by taking 50*1/2=25 terminal spaces. Here's a picture of what happens:



All of this can be done in BASIC by saying

```
PRINT TAB((W-100)*(1/2);"*"
```

But multiplying by 1/2 is the same as dividing by 2, so this can be written more simply as

```
PRINT TAB((W-100)/2);"*"
```