Ipso Facto

APRIL 1986 ISSUE 47

INDEX PAGE

A PUBLICATION OF THE ASSOCIATION OF THE COMPUTER-CHIP EXPERIMENTERS (ACE) 1981

No Man is fit to be entrusted with control of the "Present", who is ignorant of the "Past," and no People who are indifferent to their "Past" need Hope to Make Their "Future "Great.

Ancient "Inuit" saying

NOTE NEW MAILING ADDRESS: ACE INC.
P.O. BOX 6464
STATION "F"
HAMILTON, Ontario
CANADA L9C 7C7

ITS BARGAIN TIME SEE
PRODUCT PAGE FOR DETAILS

March 11, 1986 - NOTICE OF ELECTION OF OFFICERS for ACE INC. for 1986-1987 Term.

This NOTICE is to inform all members that Election of Officers for the 1986-1987 Term will be held at the next ACE meeting, April 8th, 1986 at Room 123 at Sheridan College.

John B. Norris

ARTICLE SUBMISSIONS:

The content of IPSO FACTO is voluntarily submitted by club members. While ACE assumes no responsibility for errors nor for infringement upon copyright, the Editors verify article content as much as possible. ACE can use article both hardware and software, of any level, relating to any micro computer components, peripherals, and products. Please specify the equipment and/or software to which the article applies. Typed articles are preferred and are printed first. Please send originals, not photocopy materials. We will return photocopies of original if required.

PUBLICATION POLICY:

The newsletter staff assume no responsibility for article errors nor for infringement upon copyright. The content of all articles will be verified, as much as possible, and limitations listed (i.e. Make of equipment, or author of software). The newsletter will be published every other month, Delays may be incurred as result of loss of staff, postal disruption, lack of articles, etc. We apologize for such inconvenience, however, they are generally caused by factors beyond the control of the Club.

THE SUBJECT WHO IS TRULY LOYAL TO THE CHIEF MAGISTRATE WILL NEITHER ADVISE NOR SUBMIT TO ARBITRARY MEASURES -- JUNIUS.

EDITORIAL

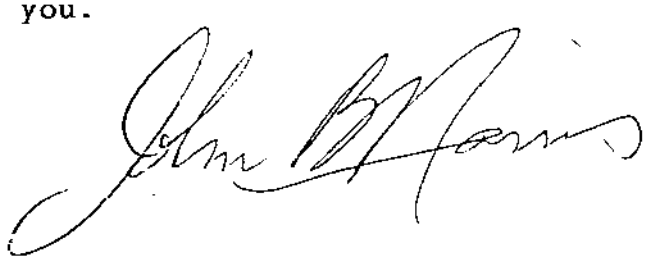
An "empty mail box" certainly reflects how interest has waned on 1802 Hardware, Firmware and Software experimentation.

Times have changed, the markets flooded with low cost "clones, pressure of business due to computerization certainly temps one to spend his leizure time doing other than what ACE stands for.

However, this time shall pass. The 32-64 bit "chips" are here with "3D", holography and computer Eye Ball command. Tass News Agency claims their new mini-computer is capable of performing up to one-billion operations per second. Certainly impressive. Europe boast more micros per family than any other country. Too bad we can't obtain more European Mag's etc. - perhaps we can catch up. Maybe the "Gold Curtain" is screening their new developments from us or we are developing a bad case of "Tunnel Vision" due to our smugness.

We accept articles on all types of computer hardware, software and firmware, and would be pleased to hear from you.

May the "FORCE" be with you.



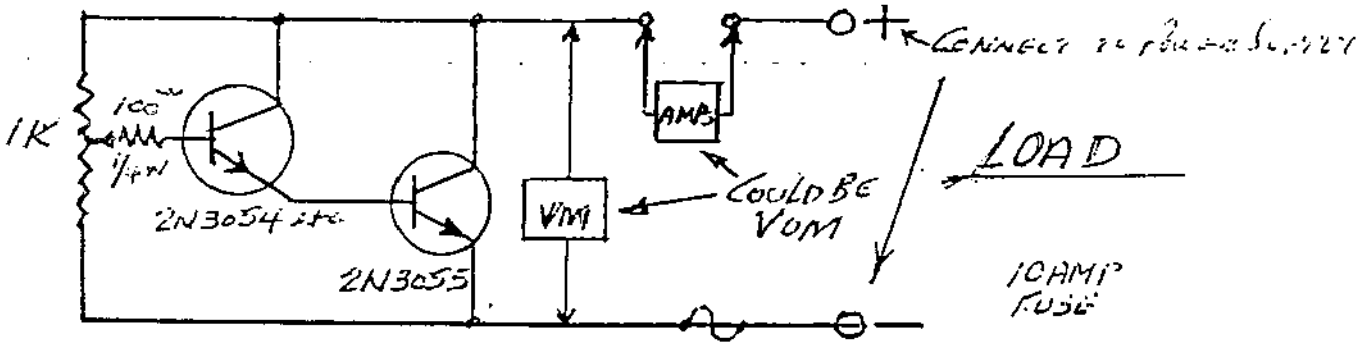
Electronic Power Supply Load Tester

– by Fred Feaver

How many of you have wished you knew **exactly how** much current your power supply could supply without collapsing, **or just** how good your voltage regulation was?

Here is a simple, inexpensive load **tester that** is very easy to build. The total cost should be much less than **$5.00.** With a 10 ampere capacity at up to 40 volts, this will handle **practically all** hobby computer supplies.

Note: The quoted price is for the **electronic components.**



The 2N3055 has a 60 volt rating at 15 amps. H_{FE} = 20 - 70 Design for 10 amps. Take worst case H_{FE} = 20

$$\text{Base Current } I_B = \frac{I_c}{H_{FE}} = \frac{10}{20} = 0.50 \text{ amps}$$

It is possible to connect the base directly to a potentiometer, but it is better to use a Darlington configuration to reduce the current through the pot.

Say the driver transistor has H_{FE} of 20 with 0.5 amp output:

$$\text{then} \qquad I_B = \frac{0.5}{20} = 25 \text{ ma.}$$

Almost any small or medium NPN transistor can be used. I used an obsolete 2N696 from my junk box, it had an output current rating of 500 ma and H_{FE} of 20.

18Ø2 EDITOR/ASSEMBLER

R. Irvine, 294 Book Rd. W. Ancaster Ont. L9G 3L1

Version 1.Ø

This program was written on a TRS-8Ø Model I using level 11 basic. There are no peeks or pokes. This should make the program adaptable to computers equipped with virtually any version of Microsoft basic.

When the program is run you are immediately in the command mode. Hitting [RETURN] or entering "HELP" will display all available commands and command syntax. They are as follows:

```
1         :Insert a line
C         :Change a line
ADD       :Add more code to the end of the buffer
DEL       :Delete a line
R         :Renumber program lines
A         :Assemble code
L         :List to screen
P         :List to printer
#XXXX     :Hex to decimal conversion
Dnnnn     :Decimal to hex conversion
```

Pressing shift C when starting a new line of code will jump to the command mode. When the buffer is empty, you must begin by entering I for insert. To add any subsequent lines the ADD command is used.

INPUT FORMAT

```
Line   Label   Mnem   Oper   Comments
 1ØØ   START   LDI    #FF    Initialization routine
       .
       .
```

The operand column will accept Hex entries as data or jump addresses in format #XXXX. It will also accept labels for addresses. The attached listing is from the RCA LSI Products - Applications book 1982, page 234. From the listing you can see the general convention used. This listing took 1 minute 25 seconds to assemble

There are several error traps in the program, but they are by no means extensive. As a result you may find a way to slide bad data past them. A symbol table is printed at the end of the listing with the corresponding addresses.

```
  10 REM            1802 EDITOR/ASSEMBLER

  11 REM            R. IRVINE  1985

  12 REM            VER  1.0

  13 REM

  14 REM

1000 CLS
     CLEAR 1000
     DEFINT A - Z
     DEFSNG A,D
     N = 500
     PC = 4
     CU$ =  CHR$ (95) +  CHR$ (24)
     NN = 1

1010 DIM  H$(15),J(99),A$(7,N),A(N),C$(99),N$(99),LB$(50,2),JJ(N)

1020 FOR X = 0 TO 15
     READ H$(X)
     NEXT

1030 DATA  0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

1040 FOR X = 1 TO 4
     READ D(X)
     NEXT

1050 DATA  4096,256,16,1

1060 FOR X = 1 TO 7
     READ TA(X)
     NEXT

1070 DATA  0,8,16,24,32,40,48

1080 FOR X = 1 TO 99
     READ N$(X),C$(X)
     NEXT

1090 FOR X = 1 TO 4
     READ P$(X)
     NEXT

1100 FOR X = 1 TO 99
     READ J(X)
     NEXT

1110 GOTO 1320

1120 INPUT "LINE NUMBER,INCREMENT";LN,IN
     IF LN + IN = 0 THEN LN = 100
     IN = 10
1130 LP = LN
```

```
IF = IN
1140  CLS
1150 PRINT  TAB( 16) USING "####":LN:
     GOSUB 1310

1160 I$ =  INKEY$
     IF I$ = "" THEN 1160

1170 I =  ASC (I$)

1180 IF I = 99 THEN  IF NN > 1 THEN NN = NN - 1
     GOTO 1320
     ELSE  GOTO 1320

1190 IF I = 13 THEN 1610

1200 IF I = 9 THEN  GOSUB 1290
     GOTO 1160

1210 IF I = 8 AND  LEN (W$) >  = 1 THEN W$ =  LEFT$ (W$, LEN (W$) - 1
          )
     PRINT " ";I$;I$;CU$;
     GOTO 1160

1220 IF I = 8 AND W$ = "" AND PC > 4 THEN PC = PC - 1
     W$ = A$(PC,NN)
     PRINT  STRING$ (8,24); CHR$ (30):W$;CU$;
     GOTO 1160

1230 IF I = 8 THEN 1160

1240 PRINT I$;CU$;
     W$ = W$ + I$
     GOTO 1160

1250 DN = 0
     FOR Z = 1 TO 4
     FOR Y = 0 TO 15
     IF  MID$ (HEX$,Z,1) <  > H$(Y) THEN  NEXT Y

1260 IF Y > 15 THEN  PRINT "ERROR IN HEX INPUT"
     GOTO 1320

1270 DN = (DN + (Y * D(Z)))
     NEXT Z
     RETURN

1280 IF DN > 65535 THEN  PRINT "MAXIMUM CONVERSION IS 65535"
     GOTO 1320
     ELSE HEX$ = ""
     FOR Z = 1 TO 4
     N(Z) =  INT (DN / D(Z))
     DN = DN - N(Z) * D(Z)
     HEX$ = HEX$ + H$(N(Z))
     NEXT
     RETURN

1290 IF PC = 7 THEN  RETURN

1300 A$(PC,NN) = W$
     W$ = ""
     PC = PC + 1

1310 PRINT  TAB( TA(PC))CU$;
```

```
     RETURN
1320 CO$ = "HELP"
     PRINT
     PRINT "COMMAND:";
     INPUT CO$

1330 IF CO$ = "DEL" THEN  FOR X = 1 TO NN
     A$(1,X) = ""
     A$(2,X) = ""
     A(X) = 0
     NEXT
     INPUT "LINE TO DELETE";LD
     GOTO 1990

1340 IF CO$ = "HELP" THEN  GOSUB 1870
     GOTO 1320

1350 IF  LEFT$ (CO$,1) = "D" THEN DN =  VAL ( RIGHT$ (CO$, LEN (CO$)
       - 1))
     GOSUB 1280
     PRINT "           "; CHR$ (27);CO$" > "HEX$
     PRINT
     GOTO 1320

1360 IF  LEFT$ (CO$,1) = "#" AND  LEN (CO$) = 5 THEN HEX$ =  RIGHT$ (
       CO$,4)
     GOSUB 1250
     PRINT "           "; CHR$ (27);CO$" >"DN
     PRINT
     GOTO 1320

1370 IF CO$ = "L" THEN  CLS
     FOR X = 1 TO NN
     FOR Y = 1 TO 7
     PRINT  TAB( TA(Y))A$(Y,X);
     NEXT
     PRINT
     NEXT
     PRINT
     FOR X = 1 TO LB
     PRINT LB$(X,2)"  "LB$(X,1)
     NEXT
     GOTO 1320

1380 IF CO$ = "P" THEN  CLS
     FOR X = 1 TO NN
     FOR Y = 1 TO 7
     LPRINT  TAB( TA(Y))A$(Y,X);
     NEXT
     LPRINT
     NEXT
     LPRINT
     FOR X = 1 TO LB
     LPRINT LB$(X,2)"  "LB$(X,1)
     NEXT
     GOTO 1320

1390 IF CO$ = "I" AND NN = 1 THEN 1120

1400 IF CO$ = "ADD" THEN NN = NN + 1
     GOTO 1140

1410 IF CO$ = "I" THEN  FOR X = 1 TO NN + 1
     A$(1,X) = ""
     A$(2,X) = ""
```

```
      INPUT "INSERT AT";I2
      GOSUB 1530
      FL:= 1
      LN = I2
      T = NN
      NN = II
      GOSUB 1140
      LN = LP
      IN = IP
      GOTO 1440

1420  IF CO$ = "A" THEN  GOSUB 1630
      GOTO 1320

1430  IF CO$ = "R" THEN  INPUT "START NUMBER,INCREMENT";LN,IN
      LP = LN
      IP = IN
      ELSE 1450

1440  FOR X = 1 TO NN
      A$(3,X) =  STR$ (LN)
      LN = LN + IN
      NEXT
      GOTO 1320

1450  IF CO$ = "C" THEN  INPUT "LINE TO CHANGE";ED
      GOTO 1470

1460  GOTO 1320

1470  FOR X = 1 TO NN
      IF  VAL (A$(3,X)) = ED THEN 1490

1480  NEXT
      PRINT "NO LINE NUMBER"ED
      GOTO 1320

1490  PRINT
      FOR Y = 1 TO 7
      PRINT  TAB( TA(Y))A$(Y,X);
      NEXT

1500  PRINT
      T1 = NN
      T2 = LN
      NN = X
      LN = ED
      FL = 1

1510  GOSUB 1150

1520  NN = T1
      LN = T2
      GOTO 1320

1530  II = 0
      FOR Z = 1 TO NN
      II = II + 1
      IF I2 <  > VAL (A$(3,Z)) THEN  NEXT

1540  FOR Z = NN TO II - 1 STEP  - 1

1550  FOR Y = 4 TO 6
```

```
1740  IF K < 12 THEN A$(2,X)  = A$(2,X) + A$(6,X)
1750  IF K > 11  AND  K < 32  AND   LEFT$ (A$(6,X),1) = "#" THEN A$(2,
      X) = A$(2,X) +  RIGHT$ (A$(6,X),2)
      GOTO 1780

1760  IF K > 31  AND  K < 40  AND   LEFT$ (A$(6,X),1) = "#" THEN A$(2,
      X) = A$(2,X) +  RIGHT$ (A$(6,X),4)
      GOTO 1780

1770  IF K > 39  AND  K < 50  AND   LEFT$ (A$(6,X),1) = "#" THEN A$(2,
      X) = A$(2,X) +  RIGHT$ (A$(6,X),2)
      GOTO 1780

1780 NEXT X

1790 FOR X = 2 TO NN - 1
     IF A$(6,X) = "" OR JJ(X) < 2 THEN 1820

1800 IF  LEFT$ (A$(6,X),1) = "#" THEN 1820

1810 FOR Z = 1 TO 50
     IF A$(6,X) = LB$(Z,1) THEN  GOSUB 1850
     GOTO 1820
     ELSE  NEXT
     PRINT
     PRINT "BAD LABLE ERROR IN LINE";A$(3,X)
     GOTO 1320

1820 NEXT X

1830 A$(1,NN) = "0000"

1840 RETURN

1850 IF JJ(X) = 2 THEN A$(2,X) = A$(2,X) +  RIGHT$ (LB$(Z,2),2)
     RETURN

1860 A$(2,X) = A$(2,X) + LB$(Z,2)
     RETURN

1870 CLS

1880 PRINT "I      :INSERT A LINE. (WHEN ASKED TO 'INSERT AT' IF THE L
          INE #
 ENTERED IS )20 THEN THE INSERTED LINE WILL BE
          AT 120 AND
 ALL SUBSEQUENT LINES WILL BE MOVED DOWN>"

1890 PRINT "C      :CHANGE A LINE"

1900 PRINT "ADD    :ADD MORE CODE TO THE END OF THE BUFFER"

1910 PRINT "DEL    :DELETE A LINE"

1920 PRINT "R      :RENUMBER PROGRAM LINES"

1930 PRINT "A      :ASSEMBLE CODE"

1940 PRINT "L      :LIST TO SCREEN"

1950 PRINT "P      :LIST TO PRINTER"

1960 PRINT "#XXXX :HEX TO DECIMAL CONVERSION"

1970 PRINT "DNNN   :DECIMAL TO HEX CONVERSION"
```

```
1740  IF K < 12 THEN A$(2,X)  = A$(2,X) + A$(6,X)
1750  IF K > 11  AND  K < 32  AND   LEFT$ (A$(6,X),1) = "#" THEN A$(2,
          X) = A$(2,X) +  RIGHT$ (A$(6,X),2)
      GOTO 1780

1760  IF K > 31  AND  K < 40  AND   LEFT$ (A$(6,X),1) = "#" THEN A$(2,
          X) = A$(2,X) +  RIGHT$ (A$(6,X),4)
      GOTO 1780

1770  IF K > 39  AND  K < 50  AND   LEFT$ (A$(6,X),1) = "#" THEN A$(2,
          X) = A$(2,X) +  RIGHT$ (A$(6,X),2)
      GOTO 1780

1780 NEXT X

1790 FOR X = 2 TO NN - 1
     IF A$(6,X) = "" OR JJ(X) < 2 THEN 1820

1800 IF  LEFT$ (A$(6,X),1) = "#" THEN 1820

1810 FOR Z = 1 TO 50
     IF A$(6,X) = LB$(Z,1) THEN  GOSUB 1850
     GOTO 1820
     ELSE  NEXT
     PRINT
     PRINT "BAD LABLE ERROR IN LINE";A$(3,X)
     GOTO 1320

1820 NEXT X

1830 A$(1,NN) = "0000"

1840 RETURN

1850 IF JJ(X) = 2 THEN A$(2,X) = A$(2,X) +  RIGHT$ (LB$(Z,2),2)
     RETURN

1860 A$(2,X) = A$(2,X) + LB$(Z,2)
     RETURN

1870 CLS

1880 PRINT "I      :INSERT A LINE. (WHEN ASKED TO 'INSERT AT' IF THE L
          INE #
 ENTERED IS )20 THEN THE INSERTED LINE WILL BE
          AT 120 AND
 ALL SUBSEQUENT LINES WILL BE MOVED DOWN>"

1890 PRINT "C      :CHANGE A LINE"

1900 PRINT "ADD    :ADD MORE CODE TO THE END OF THE BUFFER"

1910 PRINT "DEL    :DELETE A LINE"

1920 PRINT "R      :RENUMBER PROGRAM LINES"

1930 PRINT "A      :ASSEMBLE CODE"

1940 PRINT "L      :LIST TO SCREEN"

1950 PRINT "P      :LIST TO PRINTER"

1960 PRINT "#XXXX :HEX TO DECIMAL CONVERSION"

1970 PRINT "DNNN   :DECIMAL TO HEX CONVERSION"
```

```
1980 PRINT
     PRINT
     RETURN

1990 FOR X = 1 TO NN .

2000 IF  VAL (A$(3,X)) <  > LD THEN  NEXT

2010 IF X = NN + 1 THEN  PRINT "NO LINE NUMBER"LD
     GOTO 1320

2020 FOR Y = X TO NN - 1

2030 FOR Z = 1 TO 6

2040 A$(Z,Y) = A$(Z,Y + 1)

2050 NEXT Z

2060 NEXT Y

2070 NN = NN - 1

2080 GOTO 1320

2090 DATA  LDN,0,INC,1,DEC,2,LDA,4,STR,5,GLO,8,GHI,9,PLO,A,PHI,B,SEP,
          D,SEX,E

2100 DATA  BR,30,BQ,31,BZ,32,BDF,33,BPZ,33,BGE,33,B1,34,B2,35,B3,36,B
          4,37,NBR,38,BNQ,39,BNZ,3A,BNF,3B,BM,3B,BL,3B,BN1,3C,BN2,3D,
          BN3,3E,BN4,3F,LBR,C0,LBQ,C1,LBZ,C2,LBDF,C3,NLBR,C8,LBNQ,C9,
          LBNZ,CA,LBNF,CB

2110 DATA  ADDI,7C,SDBI,7D,SMBI,7F,LDI,F8,ORI,F9,ANI,FA,XRI,FB,ADI,FC
          ,SDI,FD,SMI,FF

2120 DATA  IDL,00,SKP,38,IRX,60,OUT1,61,OUT2,62,OUT3,63,OUT4,64,OUT5,
          65,OUT6,66,OUT7,67,INP1,69,INP2,6A,INP3,6B,INP4,6C,INP5,6D,
          INP6,6E,INP7,6F

2130 DATA  RET,70,DIS,71,LDXA,72,STXD,73,ADC,74,SDB,75,SHRC,76,RSHR,7
          6,SMB,77,SAV,78,MARK,79,SEQ,7B,REQ,7A,SHLC,7E,RSHL,7E,NOP,C
          4,LSNQ,C5,LSNZ,C6,LSKP,C8,LSIE,CC,LSQ,CD,LSZ,CE,LSDF,CF

2140 DATA  LSNF,C7

2150 DATA  LDX,F0,OR,F1,AND,F2,XOR,F3,ADD,F4,SD,F5,SHR,F6,SM,F7,SHL,F
          E

2160 DATA  EQU,ORG,END,DEFM

2170 DATA  1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
          2,2,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,2,2,2

2180 DATA  1,1,1,1,1,1,1,1,1,1

2190 DATA  1,1,1,1,1,1,1,1,1,1

2200 DATA  1,1,1,1,1,1,1,1,1,1

2210 DATA  1,1,1,1,1,1,1,1,1,1

2220 DATA  1,1,1,1,1,1,1,1,1,1
```

```
0000                100             ORG     #0000
0000  F880          110   PRMTST    LDI     #80
0002  B5            120             PHI     5
0003  F800          130             LDI     #00
0005  A5            140             PLO     5
0006  E5            150             SEX     5
0007  98            160             GHI     8
0008  55            170             STR     5
0009  F8AA          180             LDI     #AA
000B  F5            190             SD
000C  3A89          200             BNZ     COLDST
000E  88            210             GLO     8
000F  55            220             STR     5
0010  F8AA          230             LDI     #AA
0012  F5            240             SD
0013  3A89          250             BNZ     COLDST
0015  3044          260             BR      INIT
0017  F88F          270   MEMSIZ    LDI     #8F
0019  B5            280             PHI     5
001A  F8FF          290             LDI     #FF
001C  A5            300             PLO     5
001D  E5            310             SEX     5
001E  F8AA          320             LDI     #AA
0020  55            330             STR     5
0021  F5            340             SD
0022  3236          350             BZ      LOOP1
0024  7B            360   @FLASH    SEQ
0025  F820          370             LDI     #20
0027  B5            380             PHI     5
0028  25            390   @ON       DEC     5
0029  95            400             GHI     5
002A  3A28          410             BNZ     @ON
002C  7A            420             REQ
002D  F820          430             LDI     #20
002F  B5            440             PHI     5
0030  25            450   @OFF      DEC     5
0031  95            460             GHI     5
0032  3A30          470             BNZ     @OFF
0034  3024          480             BR      @FLASH
0036  95            490   LOOP1     GHI     5
0037  FC10          500             ADI     #10
0039  B5            510             PHI     5
003A  F8AA          520             LDI     #AA
003C  55            530             STR     5
003D  F5            540             SD
003E  3236          550             BZ      LOOP1
0040  95            560             GHI     5
0041  FF10          570             SMI     #10
0043  B5            580             PHI     5
0044  F800          590   INIT      LDI     #00
0046  B3            600             PHI     3
0047  A6            610             PLO     6
0048  F84C          620             LDI     #4C
004A  A3            630             PLO     3
004B  D3            640             SEP     3
004C  F880          650   START     LDI     #80
004E  B6            660             PHI     6
004F  B4            670             PHI     4
0050  F801          680             LDI     #01
0052  A4            690             PLO     4
0053  3583          700             B2      LOAD
```

```
0055    A0      710             PLO     0
0056    F880    720             LDI     #80
0058    B0      730             PHI     0
0059    F800    740             LDI     #00
005B    00      750             IDL
005C    E6      760     XMIT    SEX     6
005D    94      770             GHI     4
005E    56      780             STR     6
005F    90      790             GHI     0
0060    F3      800             XOR
0061    3A69    810             BNZ     OVRUN
0063    84      820             GLO     4
0064    56      830             STR     6
0065    80      840             GLO     0
0066    F3      850             XOR
0067    327D    860             BZ      SAVE
0069    95      870     OVRUN   GHI     5
006A    56      880             STR     6
006B    94      890             GHI     4
006C    F3      900             XOR
006D    3A75    910             BNZ     URTCHK
006F    85      920             GLO     5
0070    56      930             STR     6
0071    84      940             GLO     4
0072    F3      950             XOR
0073    328F    960             BZ      OVFLOW
0075    3675    970     URTCHK  B3      URTCHK
0077    3C77    980     PRTCHK  BN1     PRTCHK
0079    E4      990             SEX     4
007A    61      1000            OUT1
007B    305C    1010            BR      XMIT
007D    90      1020    SAVE    GHI     0
007E    B7      1030            PHI     7
007F    80      1040            GLO     0
0080    A7      1050            PLO     7
0081    304C    1060            BR      START
0083    97      1070    LOAD    GHI     7
0084    B0      1080            PHI     0
0085    87      1090            GLO     7
0086    A0      1100            PLO     0
0087    305C    1110            BR      XMIT
0089    F8AA    1120    COLDST  LDI     #AA
008B    B8      1130            PHI     8
008C    A8      1140            PLO     8
008D    3017    1150            BR      MEMSIZ
008F    7B      1160    OVFLOW  SEQ
0090    308F    1170            BR      OVFLOW
0000            1180            END

0000  PRMTST
0017  MEMSIZ
0024  QFLASH
0028  QON
0030  QOFF
0036  LOOP1
0044  INIT
004C  START
005C  XMIT
0069  OVRUN
0075  URTCHK
0077  PRTCHK
007D  SAVE
0083  LOAD
0089  COLDST
008F  OVFLOW
```

USING A BAUDOT TELETYPE ON THE 1802
- BY B. MILLER, I.F. #6, p. 38

The low cost of a CPU card, such as those offered by Tektron or Netronics makes the 1802 chip a natural for anyone who wishes to ease into the microprocessor field without too great a cash outlay. However, before any serious programming can be done some form of terminal must be obtained, as loading via a hex keypad and reading data from LEDs or a 7 Segment Display is rather slow.

At this point the cost of the system will likely soar as a Video Display or a printer will cost upwards of five hundred dollars. Even a reconditioned Model 33 Teletype which is an ASCII machine and will interface with the 1802 hardware and software available, costs over one thousand dollars.

A low cost alternative for those who can get their hands on one, is a Model 15 or 19 Teletype unit. These units have long been removed from commercial service and can often be obtained from ham radio operators for a very modest amount. As they are Baudot machines they require some software modifications if they are to be used on a system that communicates in the ASCII code. This, however is not difficult to implement.

A Baudot machine is however lacking in two areas that cannot easily be overcome:

1) The keyboard is a rather slow mechanical one that is awkward to use; i.e. it must be shifted for letters-figures changes. Also there are no provisions for many of the arithmetic signs used in BASIC, for example.

2) Some useful ASCII characters that might be found in the output string, are not available on the printer. Examples of this are arithmetic signs such as plus, multiplication sign, equals, less than, greater than, etc.

As a result of the above, it was felt that it would be better to use a low cost ASCII keyboard-Encoder chip set configured as a parallel input rather than using the Teletype keyboard as a Serial input device. That looked after 1) above.

The second limitation can be tackled by assigning some of the extra characters such as ! . and dollar sign to the missing arithmetic signs. This usage will soon become quite natural to the user.

The ASCII to Baudot conversion can be performed by a hardware circuit, and indeed many hams have gone this route. However it is more natural to do this in software in a computer system as no additional parts are required and the memory that it takes is minimal.

THE HARDWARE

The first thing that must be done is make sure that your Teletype is working well. Details of the operation of the machine are available from many sources. The Sept. 1977 issue of 73 Magazine is largely devoted to TTY, and although there are more comprehensive sources available, I was able to gain insight into the operation of the machine well enough with this issue to get the machine running.

The Selector magnets (on the L.H.S. of machine) should be wired in parallel, rather than in series as this will allow them to work on a low voltage "loop" supply. I am using a simple 10 volt D.C. supply, as shown in figure 1.

Current to run the magnets in this configuration is roughly 60 M.A. This can be handled by any Plastic Tab type transistor with a dissipation of about 5 watts and a breakdown voltage more than 50 or 60 volts. The Reverse-biased diode from coll.-emitter is to suppress the spike produced by the magnets as the field collapses when they are de-energized. I have configured this circuit to be in the "marking" state (machine locked up when no data present) when Q is equal to "0" which is the state that the 1802 will be in after a Reset. Therefore, whatever program is running, it is not necessary to initialize the Q line to lock up the TTY.

I have not encountered any problems in running the magnets directly: i.e. no opto-isolator between the CPU and TTY.

Note that most references to TTY in ham publications will show a high voltage (in the vicinity of 120 volts D.C.) loop supply. This is not required in this case and would only complicate the switching design.

The speed of the Teletype machine that you have must be determined before the software delay loop is written. This can be done by wiring the keyboard contacts into the loop and running the machine in the "local" mode. If an oscilloscope is connected across the keyboard contacts and the space bar is depressed, the single "1" data bit in the middle of the 5 bit data stream will be easy to distinguish and measure. A 22 ms pulse corresponds to a 60 WPM machine, 20 ms equals 66 WPM, 17.57 ms equals 75 WPM.

While the TTY is connected in the local loop check to see whether the machine unshifts (goes from figs-to lets) when the space bar is depressed, or for that matter after a carriage return. I am not familiar with other machines but the model 19 that I have will unshift on space (UOS). This must be taken into account in the software program or the computer will "lose track" of the shift condition of the TTY machine, and print incorrectly.

THE SOFTWARE

The software program to run the Teletype was developed to be compatible with Pittman Tiny BASIC 1802. It is in the form of SCRT type subroutines and can of course be utilized by other programs, such as machine language programs developed by the user.

There are several considerations that arise in developing the necessary software program. The important ones are as follows:

1) The ASCII output of the BASIC program (or users program) must be converted to Baudot. The ASCII output is parallel and is

converted as such to Baudot. This is most easily accomplished by a look-up table. The parallel Baudot data could then be serialized by a UART but it is rather easy to implement this in software so that is the method that I have chosen to use. In the Baudot data I have imbedded a "1" in the L.S.B. in all cases. This is a loop sustaining bit and eliminates the need for a register allocated for use as a counter. (Tiny Basic uses most of the registers, making this a consideration)

2) The software routine must keep track of the nature of the last character printed: i.e. whether it was a letter or a figure. This is done by looking at the M.S.B. of the ASCII char. i.e. the 7th bit. If it is a "0" thechar. is a figure; a "1" indicates a letter. Therefore upon receipt of an ASCII character, the program determines what the present char. is and compares it with the data stored in Register C which contains the nature of the last character sent. If a difference exists a routine which outputs the correct "Figs" or "letters" character takes over, and then returns to the main print program to output the character given it by the main program. After this character is printed, its nature is stored in the Register C for use by the next character that is inputted. The character "space" is trapped before it enters the above routine as it looks like a figure and stores a code in Register C as such, but when it is printed it will unshift the TTY carriage. This has the effect of printing letters in place of numbers if there is a space between them. The "space" character is handled differently in the routine and always results in a letters code being placed in Register C after it is printed.

Certain control codes outputted by Tiny Basic as well as pad characters will also cause the output routine to lose track of the shift condition unless taken care of by the output routine. By setting location 0111 in Tiny to 02 as opposed to the 82 code in the original program, the pads will come out as nulls. The look-up table should code ASCII nulls to a "Figs" character (hex 37). I have set all unused ASCII code locations to the code for Baudot "blank" (hex 01) It appears that Tiny outputs some control characters that are not mentioned in the text, and which will confuse the output routine at times if this coding is not done.

3) This software program uses several registers to store temporary data during the I/O routines. I have chosen to use registers not used by Tiny BASIC 1802. While the text that comes with Tiny states that Register Eo is not used by the interpreter, I have been unable to have any luck using this register for a purpose such as storing the nature of the last character, which requires that the register is left unaltered between I/O calls. It is therefore used for ASCII character storage through the I/O routine.

If the program is loaded exactly as given it may be tested as follows

Address	Data	
0000	C0 0E 00	Long jump to Initialization (after initialization program will jump to 0004)
0003	XX Don't care	
0004	D4 0E 53	Call Ineee
0007	30 04	Repeat

This will print out the characters typed into the ASCII keyboard.

PARALLEL ASCII KEYBOARD INPUT

I have used a standard ASCII KEYBOARD, G.I. AY5-3600 Encoder and two 4016 quad transmission gates to form the input system.

The Keyboard Encoder Chip's "data ready strobe" is connected to the 1802's $\overline{EF\ 3}$ line and is tested in software. The ASCII data is gated to the bus through the two 4016 quad transmission gates. These are enabled by the N 1 line of the 1802. A "6A" instruction will cause the keyboard entry to be loaded into the Accumulator and M (R X).

System Initialization For Input-Output Subroutines

The following is a system initialization which will be required if the Monitor in your system does not support the SCRT (Standard Call and Return Technique) system of calling sub-routines as set out in the R.C.A. 1802 User's Manual. This is only required if the user wishes to use these routines as part of machine language programs of his own; initialzation is already present in Pittman Tiny Basic and it will run properly without this initialization being entered in memory.

Assuming a 4K memory, maximum space will be left for user's Basic programs if the following scheme is used;

0E00-0E52	Inilialization (if required, per above)
0E53-0EDB	Ineee, Outee, Break subroutines
0F00-0FFF	ASCII-BAUDOT look up Table, Stack

Basically, more than ¼ of the last page is not required by the look-up table and could be used for other purpose. I have chosen to leave it filled with the code for a Baudot "blank".

System Initialization

```
0E00  F8 0E       LDI  0E      ; Set this program to run with
  02  B6          PHI  R6      ; R6 as the program counter
  03  B7          PHI  R7
  04  F8 08       LDI  08
  06  A6          PLO  R6
  07  D6          SEP  R6
  08  F8 26       LDI  26      ; Set R7 to point to table of
  0A  A7          PLO  R7      ; initial values.
  0B  47          LDA  R7
  0C  B5          PHI  R5      ; Initialize R5 (SCRT Return
  0D  47          LDA  R7      ; routine pointer)
  0E  A5          PLO  R5
  0F  47          LDA  R7      ; Initialize R4 (SCRT Call
  10  B4          PHI  R4      ; routine pointer)
  11  47          LDA  R7
  12  A4          PLO  R4
  13  47          LDA  R7      ; Initialize R3 (System program
  14  B3          PHI  R3      ; Counter)
  15  47          LDA  R7
  16  A3          PLO  R3
0E17      47      LDA  R7      ; Initialize R2 (System stack
  18      B2      PHI  R2      ; pointer)
  19      47      LDA  R7
  1A      A2      PLO  R2
```

```
1B      47          LDA  R7      ; Initialize R1 (Interrupt Service
1C      B1          PHI  R1      ; pointer)
1D      47          LDA  R7
1E      A1          PLO  R1
1F      47          LDA  R7      ; Initialize R0 (DMA pointer)
20      B0          PHI  R0
21      47          LDA  R7
22      A0          PLO  R0
23      E2          SEX  R2      ; Set X Register to R2
24      7A          REQ          ; Put serial output in
                                 ; Mark state.
                                   Convention Mark equal 0
                                              Space equal 1
                                         -on Q line
25      D3          SEP  R3      ; Go to 0004 At this location at
                                   the beginning of memory user
                                   can select either a long jump
                                   to Basic (0100) or to a Moniter
                                   or other machine language program.
```

Table of Initial Register Values

```
26      0E45        TAB 1        ; SCRT Return routine start address
28      0E33                     ; SCRT Call Routine start address
2A      0004                     ; Jump to Basic or Monitor address
2C      0FFA                     ; Stack location
2E      0000                     ; Interrupt service routine addr.
30      0FFA                     ; DMA storage initial value
32      D3          SEP R3       ; Go to Called subroutine; leave
                                 ; R4 pointing to entry of Call
                                 ; subroutine
CALL
33      E2          SEX  R2      ; Point to Stack
34      BF          PHI  RF      ; Save D
35      96          GHI  R6      ; Save R6 on Stack
36      73          STXD         ;
37      86          GLO  R6      ;
38      73          STXD
39      93          GHI  R3      ; Copy R3 into R6 to save return
3A      B6          PHI  R6      ; address
0E3B    83          GLO  R3      ;
3C      A6          PLO  R6      ;
3D      46          LDA  R6      ; Load the subroutine address
3E      B3          PHI  R3      ; into R3
3F      46          LDA  R6
40      A3          PLO  R3
41      9F          GHI  RF      ; Restore D
42      30 32       BR   32      ; Branch to 32, Exit point for
                                 ; call subroutine
44      D3          SEP  R3      ; Return to main program; leave
                                 ; R5 pointing to entry of Return
                                   subroutine
RETURN
0E45    BF          PHI  R7      ; Save D
46      96          GHI  R6      ; Copy R6 into R3 (R6 contains
47      B3          PHI  R3      ; return address)
48      86          GLO  R6
49      A3          PLO  R3
```

```
  4A  E2          SEX  R2        ; Point to stack.
  4B  12          INC  R2        ; Pop stack
  4C  72          LDXA           ; Restore the saved old R6
  4D  A6          PLO  R6        ; into R6
  4E  F0          LDX
  4F  B6          PHI  R6
  50  9F          GHI  RF        ; Restore D
  51  30 44       BR   44        ; Branch to 44, exit point for
                                 ; return subroutine
```

Input-Output subroutines for Tiny Basic 1802

```
INEEE
0E53  36 53       B3   53        ; Wait til ASCII Key pressed
  55  3E 55       BN3  55        ; (Keyboard strobe to EF3)
  57  6A          INP  2         ; ASCII data to D, m (R2)
  58  FE FE       SHL,SHL        ; Shift out 2 M.S.B.
  5A  3B 63       BNF  63        ; If DF equal to 0 go to 63
  5C  FE          SHL            ; Shift out M.S.B.
  5D  3B 63       BNF  63        ; If DF equal 0 go to 63
  5F  F8 20       LDI  20        ;
  61  F5          SD   R2        ; Subtract 20 from ASCII char.
                                 ; (Change L.C. to U.C.)
  62  38          SKP            ; Skip
  63  F0          LDX  R2        ; ASCII char. on stack to D.
  64  D4 0E 6B    CALL OUTEE
  67  D5          RETURN         ; Return to caller
  68  C4 C4 C4    NOP            ; NOP's for spacing

OUTEE
0E6B  AF          PLO  RF        ; Save ASCII char at RFo
  6C  FB 20       XRI  20        ; Compare char, "space"
  6E  32 96       BZ   96        ; Same? Yes-go to 96
  70  8F          GLO  RF        ; Get ASCII char.
  71  FE FE       SHL,SHL        ; Shift out 2 M.S.B.
  73  3B 8A       BNF  8A        ; If DF equal to 0, go to 8A
  75  8C          GLO  RC        ; Get RC (let-figs register)
  76  FB FF       XRI  FF        ; Compare RC, FF
  78  32 96       BZ   96        ; Same? yes-go to 96
  7A  F8 7A       LDI  7A        ; Load 7A to D
  7C  D4 0E 9F    CALL PRINT
  7F  30 96       BR   96        ; Go to 96
  81  8F          GLO  RF        ; Get ASCII char.
  82  D4 0E 9F    CALL PRINT
  85  F8 00       LDI  00        ; Load 00
  87  AC          PLO  RC        ; to RC0
  88  8F          GLO  RF        ; Get ASCII char.
  89  D5          RETURN         ; Return to caller
  8A  8C          GLO  RC        ; Get RC (let-figs register)
  8B  FB 00       XRI  00        ; Compare RC, 00
  8D  32 81       BZ   81        ; Same? yes-go to 81
  8F  F8 6D       LDI  6D        ; Load 6D to D.
  91  D4 0E 9F    CALL PRINT
0E94  30 81       BR   81        ; Go to 81
  96  8F          GLO  RF        ; Get ASCII char.
  97  D4 0E 9F    CALL PRINT
  9A  F8 FF       LDI  FF        ; Load FF
  9C  AC          PLO  RC        ; to RCo
  9D  30 88       BR   88        ; Go to 88
```

```
PRINT
OE9F  AE          PLO  RE      ; Save ASCII char at REo
  AO  F8 OF       LDI  OF      ; Load OF
  A2  BE          PHI  RE      ; to RE 1
  A3  C4 C4       NOP  NOP     ; NOP's -extra space
  A5  OE          LDN  RE      ; Load Baudot char. from look-
                               ; up Table-pointed to by RE.
  A6  FE FE       SHL  SHL     ; Shift out 2 M.S.B.
  A8  7B          SEQ          ; Set Q "start bit"
                               ; Convention Mark-"0"
                               ;            Space-"1"
                               ; on Q line
  A9  D4 OE C5    CALL DELAY
  AC  FE          SHL          ; Shift out M.S.B.
  AD  32 BA       BZ   BA      ; If D equals o then go to BA
  AF  3B B4       BNF  B4      ; If DF equals o then go to B4
  Bl  7A          REQ          ; Reset Q
  B2  30 B5       BR   B5      ; Go to B5
  B4  7B          SEQ          ; Set Q
  B5  D4 OE C5    CALL DELAY
  B8  30 AC       BR   AC      ; Go to AC
  BA  7A          REQ          ; Reset Q "Stop bit"
  BB  D4 OE C5    CALL DELAY   ; Stop bit two units duration
  BE  D4 OE C5    CALL DELAY   ;
  C1  D5          RETURN       ; Return to caller
  C2  C4 C4 C4    NOP          ; NOP's-extra spaces
DELAY
OEC5  BC          PHI  RC      ; Save Baudot char at RC
  C6  F8 C8       LDI  C8      ; Delay constant for 75 WPM
                               ; Teletype and 1.79 MHZ clock
  C8  FF 01       SMI  01      ; Subtract 1 from D
  CA  C4 C4 C4    NOP          ; NOP's for timing
  CD  C4 C4       NOP          ; NOP's for timing
  CF  3A C8       BNZ  C8      ; D equals 0, No-go to C8
  D1  9C          GHI  RC      ; Restore Baudot char to D
  D2  D5          RETURN       ; Return to caller
  D3  C4          NOP          ; NOP-extra space
BREAKSUB
OED4  3F D9       BN4  D9      ; If EF4 equals 0 go to D9
                               ; (Break switch connected to
                                 1802 EF 4 line)
  D6  FD 00       SDI  00      ; Set DF equal to 1 (Break cond.
                               ; in Tiny Basic)
  D8  D5          RETURN       ; Return to caller
  D9  FC 00       ADI  00      ; Set DF equal to 0 (No Break)
  DB  D5          RETURN       ; Return to caller
```

ASCII TO BAUDOT LOOK-UP TABLE

Address	Data	(ASCII)	(Baudot)
0F00	37	nul	"figs"
0F07	29	bel	Bell
0F0A	11	LF	line Feed
0F0D	05	CR	Carriage Ret.
0F20	09	SP	Space
0F21	2D	!	!
0F22	23	"	"
0F28	3D	(	(
0F29	13	)	)
0F2A	0F	*	. (no * avail)
0F2B	2D	+	! (no + avail)
0F2C	0D	,	,
0F2D	31	-	-
0F2E	0F	.	.
0F2F	2F	/	/
0F30	1B	0	0
0F31	3B	1	1
0F32	33	2	2
0F33	21	3	3
0F34	15	4	4
0F35	03	5	5
0F36	2B	6	6
0F37	39	7	7
0F38	19	8	8
0F39	07	9	9
0F3A	1D	:	:
0F3B	1F	;	;
0F3C	0F	<	. (no < avail)
0F3D	25	=	(no = avail)
0F3E	0F	>	. (no > avail)
0F3F	27	?	?
0F41	31	A	A
0F42	27	B	B
0F43	1D	C	C
0F44	25	D	D
0F45	21	E	E
0F46	2D	F	F
0F47	17	G	G
0F48	0B	H	H
0F49	19	I	I
0F4A	35	J	J
0F4B	3D	K	K
0F4C	13	L	L
0F4D	0F	M	M
0F4E	0D	N	N
0F4F	07	O	O
0F50	1B	P	P
0F51	3B	Q	Q
0F52	15	R	R
0F53	29	S	S
0F54	03	T	T
0F55	39	U	U
0F56	1F	V	V
0F57	33	W	W
0F58	2F	X	X
0F59	2B	Y	Y
0F5A	23	Z	Z

Control Characters

OF13	37	DC 1	"figs"
OF91	37	DC 3	"figs"

NOTE: Other locations in memory not listed (in page OF that is) should be coded with the code for Baudot "blank" i.e. 01.

This insures that any ASCII characters that Tiny Basic might output that are not available on a Model 15 Teletype keyboard will not result in illegal codes being presented to the Teletype printer.

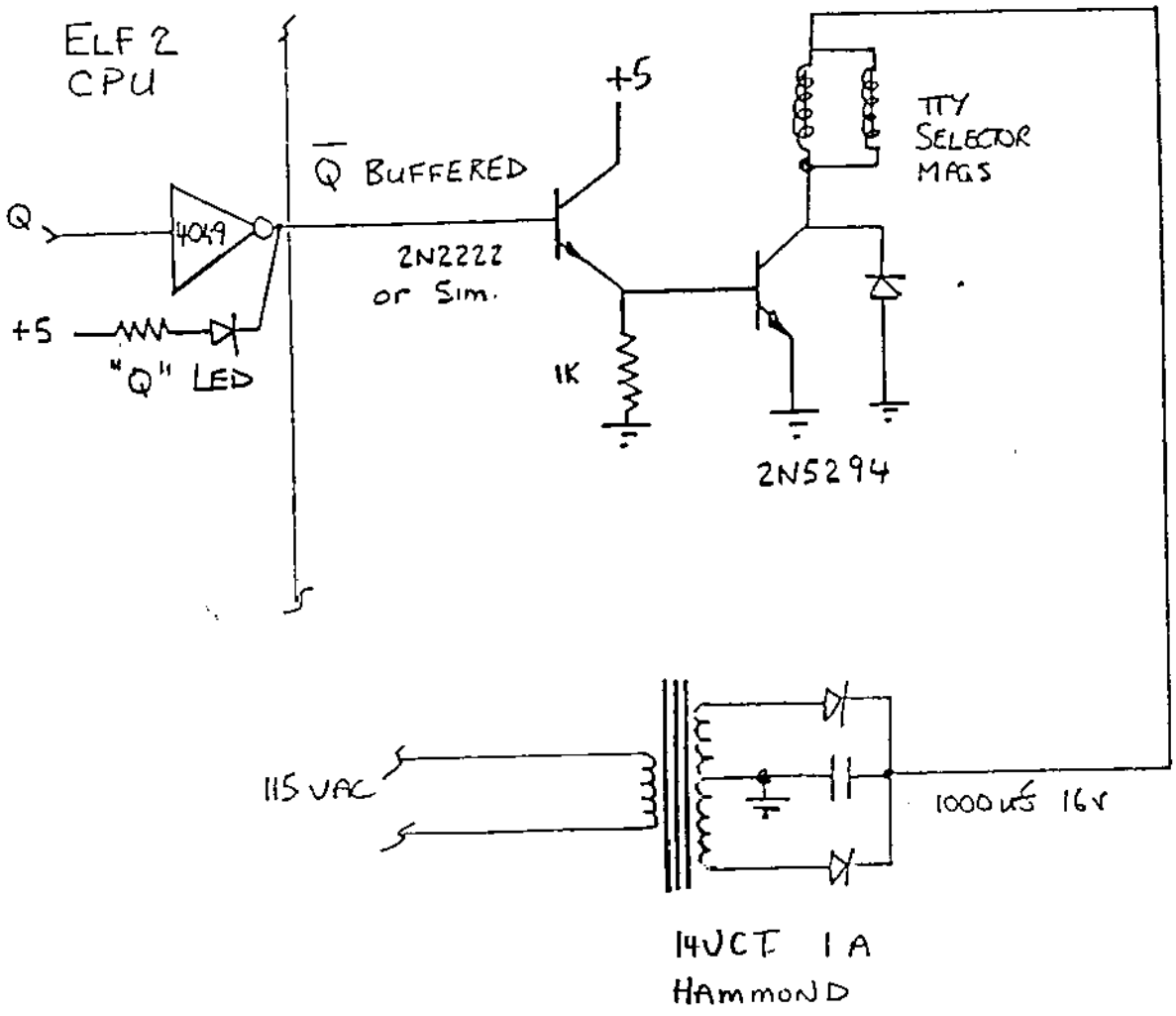


FIG 1

LOGIC PROBE
- BY M. PUPEZA, I.F. 8, p. 36

One of the easiest to use and most useful trouble shooting aids for problems in your micro system is a good logic probe. After playing around with quite a few circuits, I finalized on this relatively simple circuit modified from one lifted from Popular Electronics March 1974.

I made several mods, mainly to increase the input impedance to not load down CMOS circuits excessively.

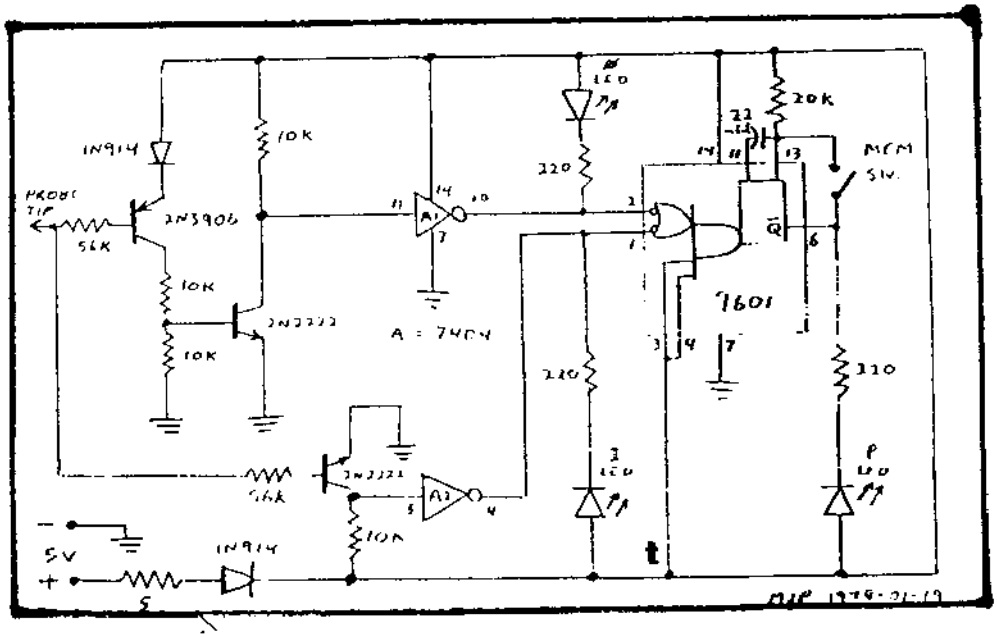
The circuit actually works best on TTL since the leds turn on at +2.6V and below +0.8V. A floating input does not turn on either led and any change of state causes the pulse led to come on for a moment. A pulse train causes the pulse led to remain lit and will record single pulses down to a 50 ns one.

The memory switch allows the pulse led to come on and stay on at any change in state of the input. This is handy when tracing a single narrow pulse that is not repetitive and can also be used to detect power interruptions.

For CMOS applications, this probe will detect the problem almost all the time if the power supply is at 5 Volts for the circuits and the probe. The only time you might fail to get a proper indication on the probe is when the signals do not reach the full high or low potentials due to loading. This hasn't happened to me yet.

Construction of my model was on some perf board and slipped into a short length of 1" copper tubing in which I had drilled 3 holes for the red, green, and amber leds. I installed a test probe into a plastic crutch tip and slipped it on one end and the switch and short length of coax power cable on a crutch tip slipped onto the other end. Any type of holder such as a penlite case, a cigar case, or similar could be used.

This probe works very well (a lot better than most I've tried) and has become indispensible for work on my system and other projects. While it can't replace a good scope, it certainly beats my Radio Shack $9 multimeter.



4.7V

LEVEL HI > 2.6 V

LO < .8 V

PROBE HI ≈ .07ma
LOAD LO ≈ .04ma

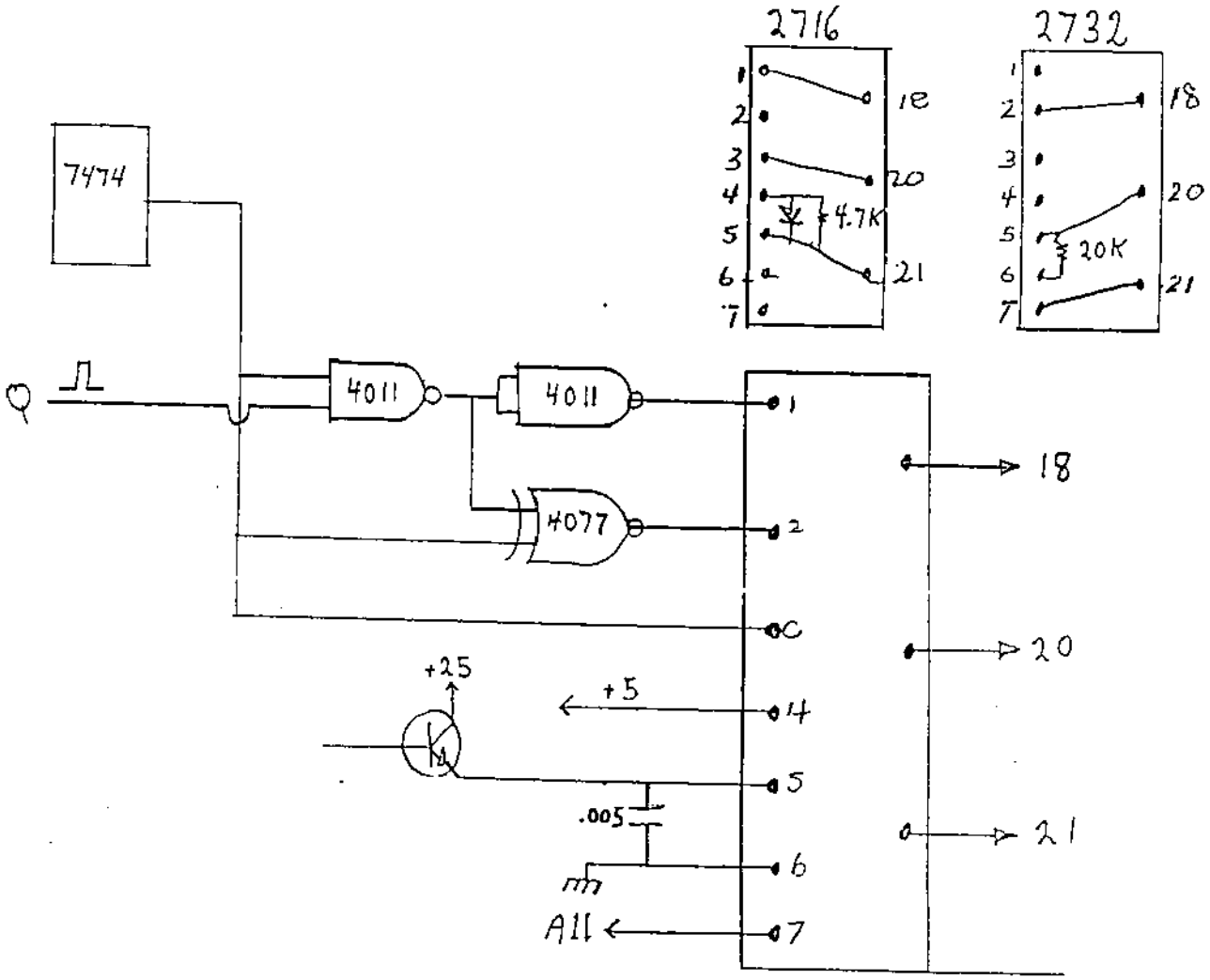
EPROM BURNER WITH PERSONALITY MODULE

BY D. WARNUK

THUNDER BAY ONTARIO P7B 5E3

I WIREWRAPPED THE "EPROM BURNER" IN I.F. #36 AND USED IT WITH 2716 CHIPS. TO INCREASE ITS VERSATILITY I MODIFIED IT SOME WHAT AND ADDED A "PERSONALITY MODULE" MADE FROM A 14 PIN DIP SOCKET. I CAN NOW ALSO USE 2732 CHIPS. SEE PARTIAL DIAGRAM OF HOW I DID IT.

P.S. SOOM TO COME: ELFA - OMEGA.



RS-232C INTERFACE
- BY B. MURPHY, I.F. #5, p. 41

If you are fortunate enough to have an RS-232 device such as a terminal, digital cassette deck, et., and wonder how to plug that funny 25 pin connector into your micro, this article should be of interest to you.

By using the circuit in figure 1, I was able to connect a 300 BPS ASCII terminal to my TEC-1802. The CD4049 was included mainly as a buffer, as I was a little nervous having a chip that has +/- 12 volts on its input line tied directly to the 1802 CPU! Perhaps a 10K pullup resistor should exist between the MC1489L and the CD4049 - my interface works fine without it.

The MC1489L is a RS-232-C line receiver and the MC1488L is the matching line driver chip. These chips are easy to use and only cost $1.75 each. There are actually 4 receivers/drivers in each chip so that other devices can use the free sections. If you are not using all inverters in the CD4049, you should tie all input lines to ground.

In order to test out the interface, the program found in listing 1 was written. Note that the input and output routines are written as subroutines, so that they can be "lifted" for other programs. If your system clock is running at a speed other than 1 MHZ, then the constants in statements 73 and 104 will have to be adjusted accordingly. The values in the program listing are for 300 BPS, and can be changed to run at other speeds if required.

References:

1. User manual for the CD1802 Cosmac Microprocessor, MPM201A, RCA Corporation, p. 73.
2. Application note for MC1489, DS9173, Motorola Semiconductor Products Inc.
3. Application note for MC1488L, DS9162, Motorola Semiconductor Products Inc.
4. EIA Standard RS-232-C, Electronic Industries Association, Washington, D.C. 20006.
5. Lancaster 'SERIAL INTERFACE', Byte Magazine, September 1975, p. 22.
6. RCA COS/MOS, SSD-203C, RCA Corporation.

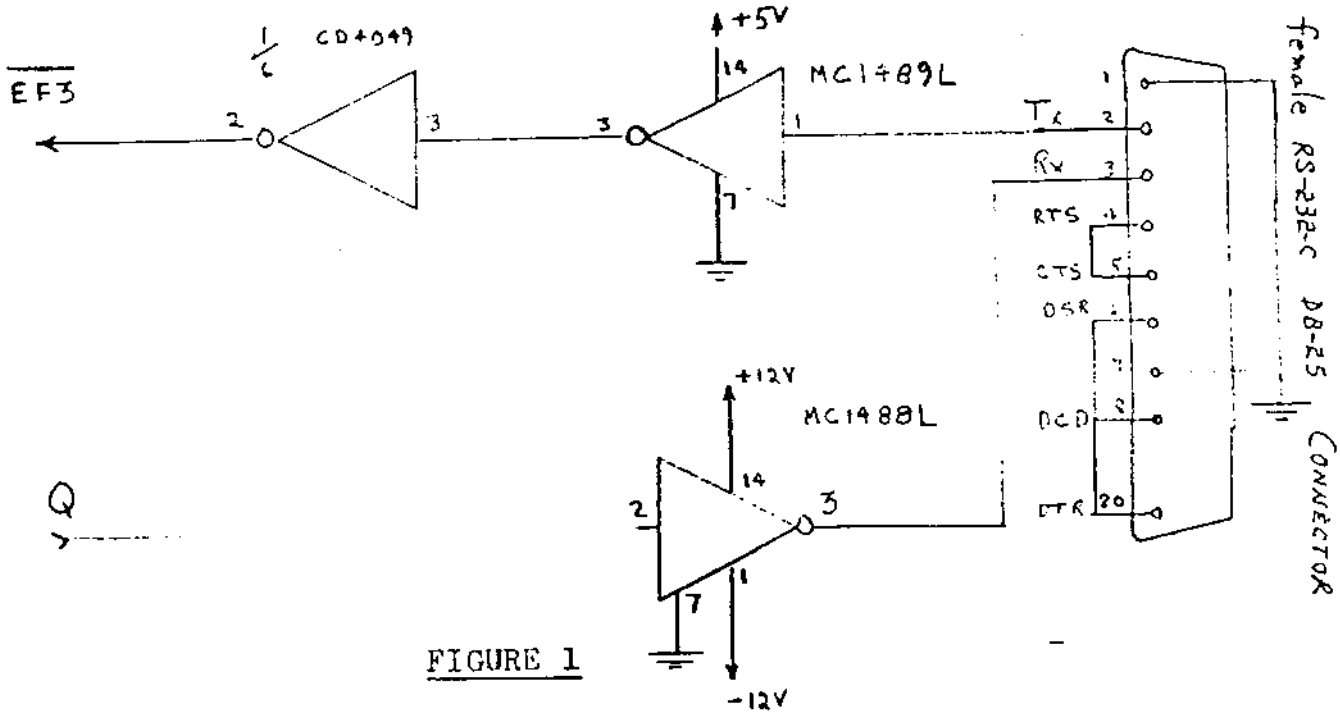


FIGURE 1

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                    1802 VER 1.3

                 1 * * * * * * * * * * * * * * * * * * * * * * * * *
                 2 *
                 3 * ASCIITST: TEST PROGRAM
                 4 *       THIS MAINLINE PROGRAM ECHOS CHARACTERS
                 5 *       ENTERED BY THE KEYBOARD
                 6 *
                 7 *         REGISTER USAGE:     R0=INITIAL P.C.
                 8 *                             R2=STACK
                 9 *                             R3=MAINLINE P.C.
                10 *                             R9=TX/RX SUBR P.C.
                11 *                             R15=DELAY P.C.
                12 *
                13 * * * * * * * * * * * * * * * * * * * * * * * * *
0000 E2         14 †        SEX   R2            STACK IS R2
0001 F8 00      15           LDI   0             D=0
0003 B2         16           PHI   R2            CLEAR R2 HIGH
0004 B3         17           PHI   R3            CLEAR R3 HIGH
0005 B9         18           PHI   R9            CLEAR R9 HIGH
0006 BF         19           PHI   R15           CLEAR R15 HIGH
0007 F8 11      20           LDI   MAINLOOP      ADDR OF TESTLOOP
0009 A3         21           PLO   R3            P.C. FOR MAINLINE
000A F8 FF      22           LDI   $FF           STACK ADDR FOR NOW
000C A2         23           PLO   R2            SAVE IN R2,
000D F8 39      24           LDI   DELAY         SET UP R15 AS DELAY
000F AF         25           PLO   R15           SUBROUTINE P.C.
0010 D3         26           SEP   R3            SET P.C. TO R3
0011 F8 43      27 MAINLOOP  LDI   RXENTRY       SET UP R9 AS RX
0013 A9         28           PLO   R9            SUBROUTINE P.C.
0014 D9         29           SEP   R9            CALL RX ROUTINE
0015 52         30           STR   R2            STORE CHAR ON STACK
0016 64         31           OUT4                DISPLAY CHAR
0017 22         32           DEC   R2            BECAUSE OF OUT4
0018 F8 20      33           LDI   TXENTRY       SET UP R9 AS TX
001A A9         34           PLO   R9            SUBROUTINE P.C.
0018 02         35           LDN   R2            GET CHAR FROM STACK
001C D9         36           SEP   R9            CALL TX ROUTINE
001D 30 11      37           BR    MAINLOOP      DO IT AGAIN
                38 * * * * * * * * * * * * * * * * * * * * * * * * *
                39 * TXTTY: TTY OUTPUT ROUTINE
                40 *
                41 *         REGISTER USAGE :    R15=PC FOR DELAY SUBR
                42 *                             R14=BIT COUNTER
                43 *                             R13=CHAR INPUT
                44 *                             R12=LOOP COUNTER
                45 *                             R9=PC FOR TXTTY
                46 *
                47 *         INPUTS:             D=CHARACTER TO BE
                48 *                             SENT
                49 *
                50 *         OUTPUTS:            Q LINE...SERIAL OUT
                51 *
                52 * * * * * * * * * * * * * * * * * * * * * * * * *
001F D3         53 TXRETURN SEP   R3            RETURN TO CALLER
0020 AD         54 TXENTRY  PLO   R13           SAVE CHAR IN D
0021 7B         55          SEQ                 SET MARK (ON)
0022 F8 08      56          LDI   8             8 BITS PER CHARACTER
0024 AE         57          PLO   R14           NUMBER OF BITS TO GO
0025 7A         58          REQ                 SET START BIT
```

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                    1802 VER 1.3

0026 DF         59          SEP   R15           DELAY ONE BIT TIME
0027 8D         60 TXSHIFT  GLO   R13           GET BITS TILL NOW
0028 76         61          SHRC                GET HIGH BIT INTO DF
0029 AD         62          PLO   R13           SAVE SHIFTED VALUE
002A CF         63          LSDF                TEST IF BIT WAS THERE
002B 7A         64          REQ                 NO...SO SET SPACE
002C 38         65          SKP                 SKIP THE SEQ
002D 7B         66          SEQ                 SET TO MARK
002E DF         67          SEP   R15           DELAY ONE BIT TIME
002F 2E         68          DEC   R14           NUMBER OF BITS TO GO
0030 8E         69          GLO   R14           INTO D
0031 3A 27      70          BNZ   TXSHIFT       BRA IF NOT DONE YET
0033 7B         71          SEQ                 SET STOP BIT (MARK)
0034 DF         72          SEP   R15           DELAY FOR 1 STOP BIT
0035 DF         73          SEP   R15           ANOTHER ONE...
0036 30 1F      74          BR    TXRETURN      RETURN TO CALLER
                75 *
0038 D9         76 DLYRTN   SEP   R9            BACK UP 1 LEVEL
0039 F8 3C      77 DELAY    LDI   60            1 BIT TIME AT 1MGHZ
                78 * NOTE:  ABOVE VALUE WORKS ONLY FOR 1 MGHZ CLOCK
003B AC         79          PLO   R12           SAVE AWAY
003C 2C         80 DECR     DEC   R12           COUNT DOWN
003D 8C         81          GLO   R12           RESULT INTO D
003E 3A 3C      82          BNZ   DECR          DONE ?
0040 30 38      83          BR    DLYRTN        RETURN TO CALLER
                84 *
                85 * * * * * * * * * * * * * * * * * * * * * * * * *
                86 * RXTTY: TTY INPUT ROUTINE
                87 *        USES FLAG 3 AS SERIAL INPUT PORT
                88 *
                89 *        REGISTER USAGE :    R15=PC FOR DELAY SUBR
                90 *                            R14=CHAR,BIT COUNTER
                91 *                            R9=PC FOR RXTTY
                92 *
                93 *        INPUTS:             FLAG EF3
                94 *
                95 *        OUTPUTS:            RESULT IN D REG
                96 *
                97 * * * * * * * * * * * * * * * * * * * * * * * * *
0042 D3         98 RXRETURN SEP   R3            RETURN TO CALLER
0043 F8 00      99 RXENTRY  LDI   0             ZERO D REG
0045 BE        100          PHI   R14           CLEAR R14 HIGH
0046 F8 08     101          LDI   8             EXPECT 8 BITS
0048 AE        102          PLO   R14           NUMBER OF BITS TO GO
0049 36 49     103 RXSTART  B3    *             WAIT FOR START BIT
0048 F8 1E     104          LDI   30            1/2 BIT TIME
               105 * NOTE:  ABOVE VALUE WORKS ONLY FOR 1 MGHZ CLOCK
004D 1F        106          INC   R15           15=R15+2 TO MISS
004E 1F        107          INC   R15           LDI 60 AT 'DELAY'
004F DF        108          SEP   R15           DELAY 1/2 BIT TIME
0050 36 49     109          B3    RXSTART       GOOD START BIT?
0052 DF        110 RXLOOP   SEP   R15           DELAY ONE BIT TIME
0053 FC 00     111          ADI   0             RESET DF
0055 3E 59     112          BN3   RXBITON       BIT ON
0057 FF 00     113          SMI   0             SET DF
0059 9E        114 RXBITON  GHI   R14           GET CHAR
005A 76        115          SHRC                SHIFT DF INTO CHAR
005B BE        116          PHI   R14           SAVE RESULT TILL NOW
```

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                    1802 VER 1.3

005C 2E          117          DEC   R14          NUMBER OF BITS TO GO
005D 8E          118          GLO   R14          INTO D
005E 3A 52       119          BNZ   RXLOOP       BRA IF NO DONE YET
0060 DF          120          SEP   R15          DELAY FOR STOP BIT
0061 9E          121          GHI   R14          RESULT INTO D
0062 30 42       122          BR    RXRETURN     RETURN TO CALLER
                 123 *
0064             124          END

   0 DIAGNOSTICS GENERATED
  12 SYMBOLS

SYMBOL TABLE:
MAINLOOP    0011          DECR        003C
TXRETURN    001F          RXRETURN    0042
TXENTRY     0020          RXENTRY     0043
TXSHIFT     0027          RXSTART     0049
DLYRTN      0038          RXLOOP      0052
DELAY       0039          RXBITDN     0059
```

Super Elf 7-segment Display Replacements
by Steven S. Coles

Easy interfacing and the ability to run on battery power have brought several 1802-based single board computers out of retirement at the Seattle Robotics Society. Super Elves and Super Elf Adapter Boards obtained secondhand are frequently lacking the FND 500/503/560 half-inch-high common cathode 7-segment displays for high and low address. These numbers are no longer stocked by any supplier I contacted. However, two days of phone calls and digging through surplus bins located the following substitutes: Radio Shack 276-1647, Heath/Zenith 411-819, Heath/Zenith 411-884, William J. Purdy Company AND 362R, General Instrument MAN 6780, Hewlett Packard HDSP 5503 and Texas Instruments TIL 322A. The Radio Shack part is usually found on the discontinued item table, if at all. Those stocking the TIL 322A said that they would not be obtaining new stock after April, 1986. A large inventory of TIL 322A was found at the Chicago, Illinois warehouse of Marshall Electronic Industries. While I have not tried the Fairchild FND 530/540/550, they do have the correct pinout and provide green, yellow and amber light. As these require slightly higher voltage, D19 and/or D20 (CR1 and/or CR2 on the adapter board) may have to be jumpered, in which case display types should not be mixed. Prices for the displays I did obtain ranged from $1.39 to $3.15 (Yanklandish dollars).

The 4368/9368 display driver chips (and the MAN 6780's) were found at Active Electronics.

A SIMPLE 'CAPS-LOCK' CIRCUIT

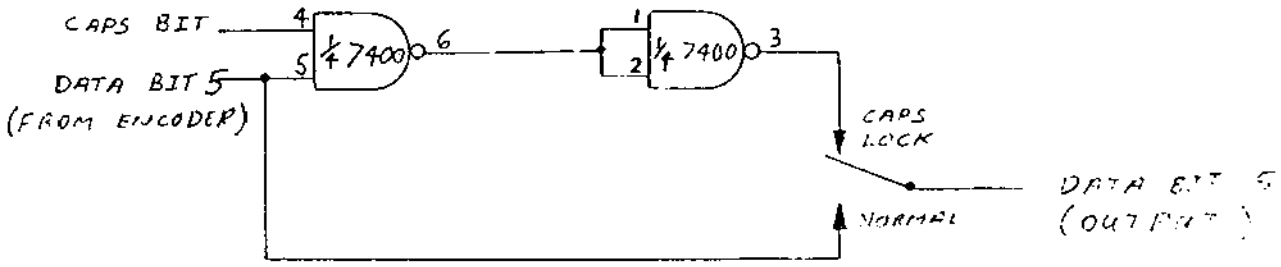
W. Bowdish

Some time ago I purchased a used ASCII keyboard for my system. The keyboard has a single chip keyboard encoder which is capable of generating the full 128 ASCII character set. Unfortunately it is a typewriter style keyboard. The unshifted output consists of lower case alphabetic characters, digits and some special characters. The shift key causes the upper case characters to be generated. This setup is fine for many applications but most monitors and languages only accept upper case letters. What to do?

When the keyboard was purchased (several people got them at about the same time) Frank Ditomaso spent many hours determining how the keyboards worked (you don't get much documentation for $20). Frank found that the encoder chip generated a rather interesting bit (I'll call it the CAPS bit) along with the usual data and parity bits. The following table shows how this bit is related to the ASCII codes.

ASCII code (hex)	CAPS bit	ASCII data bit 5
00 - 1F	0	0
20 - 3F	1	1
40 - 5F	0	0
60 - 7F	0 for lower case (61-7F) 1 for all other characters	1

In order to convert lower case to upper case, all that has to be done is to logically AND data bit 5 and the CAPS bit and use the result as data bit 5 of the character. The following schematic shows a circuit which will do this. I had a 7400 in the junk box so that is what the circuit is based on. The switch was mounted on the keyboard case and labeled appropriately.



CLUB COMMUNIQUE

NAME:_____ DATE:_____

PRODUCT ORDER	QTY.	UNIT PRICE	TOTAL
CPU BOARD	____	~~50.00~~ 30.00	___.__
BACKPLANE & I/O BOARD VER.2	____	~~40.00~~ 25.00	___.__
FRONT PANEL WITH EPROM PROG., CLOCK	____	~~35.00~~ 25.00	___.__
VDU BOARD VER.2 "SPECIAL"	____	~~25.00~~ 25.00	___.__
EPROM/RAM (BYTEWIDE & EPROM) "SPECIAL"	____	~~25.00~~ 25.00	___.__
80 X 25 VIDEO BOARD	____	~~50.00~~ 25.00	___.__
DISK CONTROLLER BOARD VER.2	____	~~50.00~~ 25.00	___.__

BOARDS AVAILABLE WILL BE SHIPPED. EXCESS PAYMENT WILL BE RETURNED.
WITH THE PURCHASE OF A COMPLETE SET OF BOARDS WE WILL GIVE A 50% DISCOUNT
TOTAL PRICE WITH DISCOUNT=137.50 + 5.00 SHIPPING & HANDLING

SOFTWARE SUPPLIED IN NETRONICS CASSETTE FORMAT

Fig. FORTH - 6K - 0000H	____	$10.00	___.__
Tiny PILOT - 2K - 0000H	____	10.00	___.__
SYMON - 2K - 0000H	____	10.00	__.__
SYDOS - 4K - D000H	____	10.00	__.__
CHIP 8AE - 1.5K-1000H	____	10.00	__.__
SPACE WAR - disk - 32k & tape - 32k	____	15.00	__.__
COSMIC CONQUEST - tape - 32k	____	10.00	__.__

PRICE NOTE

Prices listed are in local funds. Americans and overseas pay in U.S. funds Canadians in Canadian funds. Overseas orders: for all items add $10.00 for air mail postage. Please use mony orders or bank draft for promp shipment. Personal cheques require up to six weeks for bank clearence prior to shipping orders.

SALES POLICY

ALL OF THE ABOVE PRODUCTS WORK IN A ACE configured microcomputer. We will endever to assist in custom applications, but assume no liability for such use. Orders will be shipped as promptly as payment is guaranteed.