This version of Forth was designed for the SIMH PDP-8 emulator. An EAE is assumed for multiply and divide. A screen editor (8 lines of 63 chars) is provided. The buffer (packed 3 chars per double word) for the screen editor takes three pages (7000-7577) near the end of 4K memory. Thus the Forth interpreter plus the screen buffer leaves about 1450 words for user defined Forth words (about 5½ pages).

```
                / FORTH8



                     / Definitions
       7200  ZERO=   CLA             /    0
       7201  ONE=    CLA IAC         /    1
       7326  TWO=    CLA CLL CML RTL /    2
       7041  NEG=    CMA IAC         /    Two's complement
       7300  CAL=    CLA CLL

       7240  MINUS1= CLA CMA         /   -1 or 4095
       7344  MINUS2= CLA CMA CLL RAL /   -2 or 4094
       7346  MINUS3= CLA CMA CLL RTL /   -3 or 4093

       0000  *0
00000  7200          CLA
00001  6046          TLS             / send null character to terminal to prepare it
00002  5134  JMP     EXNV
00003  0040  C40,    40              / Space
00004  0100  C100,   100
00005  7000  C7000,  7000
00006  7753  M25,    -25

       0020  *20
00020  4022  IP,     START   / Instruction pointer
00021  4160  SP,     S       / Stack pointer
00022  4300  RSP,    RS      / Return stack pointer
00023  0000  EVR,    0       / Execution vector
00024  4103  VL,     CARRYH  / Last dictionary link
00025  4300  DP,     DICTIONARY / Next dictionary addr

00026  0000  REG6,   0
00027  0000  REG8,   0
00030  0000  REG10,  0
00031  0000  REG12,  0
00032  0000  REG14,  0

00033  0000  STATE,  0       / -1 if compiling, zero otherwise
00034  0000  TOIN,   0       / Pointer into text input
00035  0040  FENCE,  40      / Default fence is space
00036  0000  ROWNO,  0
00037  0000  COLNO,  0
00040  7777  INSOVR, -1      / Start edit with insert

00041  0221  PUSH8,  PUSH8R
00042  0206  POP2,   POP2R   / Pop Reg10 then Reg8 (was popA8)
00043  0226  PUSH2,  PUSH2R  / Push Reg8 then Reg10 (was push8A)
00044  0214  POP3,   POP3R   / Pop Reg 12 then Reg 10 then Reg 8 (was popCA8)
00045  0234  PUSH3,  PUSH3R  / Push Reg 8 then Reg 10 then Reg 12 (was push8AC)
00046  1461  DOTQAUX, DOTQR
00047  1006  TWIXTAUX, TWIXTR

                             / Initialization constants
00050  4160  SI,     S
00051  4300  RSI,    RS
                             / These constants appear as AND instructions
00052  0777  C777,   777
00053  0007  C7,     7
00054  1000  C1000,  1000

                             / Inc IP
00055  0000  INCIP,  0
00056  2020          ISZ     IP
```

```
00057  7200          CLA
00060  5455          JMP I   INCIP   / Return with 0 in Acc

                             / Inc SP
00061  0000  INCSP,  0
00062  2021          ISZ     SP
00063  7200          CLA
00064  5461          JMP I   INCSP   / Return with 0 in Acc

                             / Dec SP
00065  0000  DECSP,  0
00066  7240          MINUS1          / Load -1
00067  1021          TAD     SP      / SP=SP-1
00070  3021          DCA     SP
00071  7020          CML             / Set link to previous
00072  5465          JMP I   DECSP   / Return with 0 in Acc

                             / Inc RSP
00073  0000  INCRSP, 0
00074  2022          ISZ     RSP
00075  7200          CLA
00076  5473          JMP I   INCRSP  / Return with 0 in Acc

                             / Dec SP
00077  0000  DECRSP, 0
00100  7240          MINUS1          / Load -1
00101  1022          TAD     RSP
00102  3022          DCA     RSP
00103  5477          JMP I   DECRSP  / Exit with 0 in Acc

                             / Pop stack to Acc
00104  0000  POP,    0
00105  7200          CLA
00106  1421          TAD I   SP      / Leave result in Acc
00107  3113          DCA     PUSH    / Save arg in push return for tmp
00110  4061          JMS     INCSP
00111  1113          TAD     PUSH
00112  5504          JMP I   POP

                             / Push Acc to stack
00113  0000  PUSH,   0
00114  3104          DCA     POP     / What to push is in Acc
00115  4065          JMS     DECSP   / Save arg in pop return for tmp
00116  1104          TAD     POP
00117  3421          DCA I   SP      / Push Acc to stack
00120  5513          JMP I   PUSH    / Return with 0 in Acc

                             / Serout
00121  0000  SEROUT, 0               / Routine for sending a single character
00122  6041          TSF             / Argument is in Acc
00123  5122          JMP .-1
00124  6046          TLS
00125  7200          CLA             / Returns with 0 in Acc
00126  5521          JMP I   SEROUT

                             / SerIn
00127  0000  SERIN,  0               / Routine for reading a single character
00130  6031          KSF
00131  5130          JMP     .-1
00132  6036          KRB             / Result of input is in Acc
00133  5527          JMP I   SERIN

                             / ExNV
00134  7200  EXNV,   CLA
00135  1420          TAD I   IP      / Address at IP to EVR
00136  3023          DCA     EVR
00137  2020          ISZ     IP      / Point IP to next word address
00140  5423          JMP I   EVR     / Go to word address
                                     /   Acc is zero after jump

                             / ExColon
```

```
00141  4077  COLON,  JMS     DECRSP  / Push current IP to RS
00142  1020          TAD     IP
00143  3422          DCA I   RSP
00144  1023          TAD     EVR     / Point IP to word after Colon
00145  7001          IAC
00146  3020          DCA     IP
00147  5134          JMP     EXNV

                             / Jump: Jump routine
00150  7200  JUMP,   CLA             / Get the addr to jump to
00151  1420          TAD I   IP
00152  3020          DCA     IP      / Store the addr to jump to in IP
00153  5134          JMP     EXNV

                             / Jump TOS zero routine
       0154  JUMPF=  .               / Jump false
00154  4104  JUMPZ,  JMS     POP
00155  7450          SNA
00156  5150          JMP     JUMP
00157  4055  JUMPZ1, JMS     INCIP
00160  5134          JMP     EXNV

                             / Jump TOS nonzero routine
       0161  JUMPT=  .
00161  4104  JUMPNZ, JMS     POP
00162  7440          SZA
00163  5150          JMP     JUMP
00164  5157          JMP     JUMPZ1

                             / Jump TOS minus routine
00165  4104  JUMPMI, JMS     POP
00166  7510  JMPMI1, SPA
00167  5150          JMP     JUMP
00170  5157          JMP     JUMPZ1

                             / Jump TOS GE 0 routine
00171  4104  JUMPNM, JMS     POP
00172  7040          CMA
00173  5166          JMP     JMPMI1

             PAGE

00200  5000          JMP     0
                             / NR: Number runner routine: Number at IP to stack
00201  7200  NR,     CLA
00202  1420          TAD I   IP
00203  4113          JMS     PUSH
00204  4055          JMS     INCIP
00205  5134          JMP     EXNV

                             / PopA8: Pop Reg10 then Reg8
00206  0000  POP2R, 0
00207  4104          JMS     POP
00210  3030          DCA     REG10   / Top of stack to R10
00211  4104          JMS     POP     / Result is in Acc
00212  3027          DCA     REG8    / Under top of stack to R8
00213  5606          JMP I   POP2R

                             / PopCA8: Pop Reg12 then Reg10 then Reg8
00214  0000  POP3R, 0
00215  4104          JMS     POP     / Top of stack to R12
00216  3031          DCA     REG12
00217  4442          JMS I   POP2
00220  5614          JMP I   POP3R

                             / Push8
00221  0000  PUSH8R, 0
00222  7200          CLA
00223  1027          TAD     REG8
00224  4113          JMS     PUSH
00225  5621          JMP I   PUSH8R
```

```

                             / Push8A: Push Reg8 then Reg10
00226  0000  PUSH2R, 0
00227  7200          CLA
00230  4441          JMS I   PUSH8
00231  1030          TAD     REG10
00232  4113          JMS     PUSH    / R10 will be top of stack
00233  5626          JMP I   PUSH2R

                             / Push8AC: Push Reg8 then Reg10 then Reg12
00234  0000  PUSH3R, 0
00235  4443          JMS I   PUSH2   / Push Reg8 then Reg10
00236  1031          TAD     REG12   / R12 will be top of stack
00237  4113          JMS     PUSH
00240  5634          JMP I   PUSH3R

                             / SemiColon
00241  7200  SEMIC,  CLA             / Return address to IP
00242  1422          TAD I   RSP
00243  3020          DCA     IP      / Store return address in IP
00244  4073          JMS     INCRSP  / Update RSP
00245  5134          JMP     EXNV

                             / Store
00246  1001  STOREH, 1001
00247  1241          "!+1000         / ! plus end of text flag
00250  0000          0               / Link to prior head
00251  4442  STORE,  JMS I   POP2    / Top of stack in R10, value to store in R8
00252  1027          TAD     REG8    / Value to store into Acc
00253  3430          DCA I   REG10   / Store via R10
00254  5134          JMP     EXNV

                             / At
00255  1001  ATH,    1001
00256  1300          "@+1000
00257  0246          STOREH          / Link to prior head
00260  4104  AT,     JMS     POP     / Address to fetch from
00261  3027          DCA     REG8
00262  1427          TAD I   REG8    / Fetch the value
00263  4113          JMS     PUSH    / The value to stack
00264  5134          JMP     EXNV

                             / Swap
00265  1004  SWAPH,  1004
00266  0323          "S;"W;"A;"P+1000
00267  0327
00270  0301
00271  1320
00272  0255          ATH             / Link to prior head
00273  4104  SWAP,   JMS     POP
00274  3027          DCA     REG8    / Top of stack to R8
00275  4104          JMS     POP
00276  3030          DCA     REG10
00277  4443          JMS I   PUSH2   / Push first R8 then R10
00300  5134          JMP     EXNV

                             / Dup
00301  1003  DUPH,   1003
00302  0304          "D;"U;"P+1000
00303  0325
00304  1320
00305  0265          SWAPH
00306  4104  DUP,    JMS     POP
00307  3027          DCA     REG8
00310  1027          TAD     REG8
00311  4113          JMS     PUSH
00312  1027          TAD     REG8
00313  4113          JMS     PUSH
00314  5134          JMP     EXNV

                             / Over
```

```
00315  1004  OVERH,  1004
00316  0317          "O;"V;"E;"R+1000
00317  0326
00320  0305
00321  1322
00322  0301          DUPH            / Link to prior head
00323  4442  OVER,   JMS I   POP2    / Top of stack to R10
00324  4443          JMS I   PUSH2   / Push R8 then R10
00325  4441          JMS I   PUSH8   / Push a copy of R8
00326  5134          JMP     EXNV

                             / Rot
00327  1003  ROTH,   1003
00330  0322          "R;"O;"T+1000
00331  0317
00332  1324
00333  0315          OVERH
00334  4444  ROT,    JMS I   POP3    / Top of stack to R12
00335  1030          TAD     REG10   / Stack is R8 R10 R12
00336  4113          JMS     PUSH
00337  1031          TAD     REG12
00340  4113          JMS     PUSH
00341  1027          TAD     REG8
00342  4113          JMS     PUSH
00343  5134          JMP     EXNV

                             / Reverse ROT
00344  1004  RROTH,  1004
00345  0322          "R;"R;"O;"T+1000
00346  0322
00347  0317
00350  1324
00351  0327          ROTH
00352  5141  RROT,   JMP COLON
00353  0334          ROT
00354  0334          ROT
00355  0241          SEMIC

                             / Drop
00356  1004  DROPH,  1004
00357  0304          "D;"R;"O;"P+1000
00360  0322
00361  0317
00362  1320
00363  0344          RROTH
00364  4061  DROP,   JMS     INCSP
00365  5134          JMP     EXNV

                             / And
00366  1003  ANDH,   1003
00367  0301          "A;"N;"D+1000
00370  0316
00371  1304
00372  0356          DROPH
00373  4442  FAND,   JMS I   POP2    / Note: in this program use FAND
00374  1027          TAD     REG8
00375  0030          AND     REG10
00376  4113          JMS     PUSH
00377  5134          JMP     EXNV

                             / Or
00400  1002  ORH,    1002
00401  0317          "O;"R+1000
00402  1322
00403  0366          ANDH
00404  4442  OR,     JMS I   POP2    / Pop TOS to REG10
00405  1027          TAD     REG8
00406  0030          AND     REG10
00407  7040          CMA
00410  0030          AND     REG10
00411  1027          TAD     REG8
```

```
00412  4113          JMS     PUSH
00413  5134          JMP     EXNV

                             / Plus
00414  1001  PLUSH,  1001
00415  1253          "++1000
00416  0400          ORH
00417  4442  PLUS,   JMS I   POP2    / Pop TOS to REG10
00420  7100          CLL
00421  1027          TAD     REG8
00422  1030          TAD     REG10
00423  4113          JMS     PUSH
00424  5134          JMP     EXNV

                             / Minus: TOS subtracted from under TOS
00425  1001  MINUSH, 1001
00426  1255          "-+1000
00427  0414          PLUSH
00430  4442  MINUS,  JMS I   POP2    / Pop TOS to REG10
00431  1030          TAD     REG10
00432  7041          NEG
00433  1027          TAD     REG8
00434  4113          JMS     PUSH
00435  5134          JMP     EXNV

                             / OntoR: stack to Rstack
00436  1002  ONTORH, 1002
00437  0276          ">;"R+1000
00440  1322
00441  0425          MINUSH
00442  4077  ONTOR,  JMS     DECRSP  / Make a hole in RS to put it in
00443  4104          JMS     POP
00444  3422          DCA I   RSP     / Store the value in the hole
00445  5134          JMP     EXNV

                             / ROnto: Rstack to stack
00446  1002  RONTOH, 1002
00447  0322          "R;">+1000
00450  1276
00451  0436          ONTORH
00452  1422  RONTO,  TAD I   RSP     / Get the current entry in RS
00453  3104          DCA     POP     / Store return in push return from POP
00454  4073          JMS     INCRSP  / Increment RSP to prune RS
00455  1104          TAD     POP
00456  4113          JMS     PUSH    / Push the value onto stack
00457  5134          JMP     EXNV

                     / FindAux: Addr of word at TOS
                     /  Returns address of parameter part of word or zero if not found

00460  4104  FINDAUX, JMS    POP     / Get the address of word
00461  3031          DCA     REG12   / R10 & R12 point to word
00462  1031          TAD     REG12   / R12 saves the address of the word
00463  3030          DCA     REG10   / R10 works on it
00464  7200          CLA
00465  1024          TAD     VL      / Last defined head in dictionary
00466  3027          DCA     REG8    / REG8 point to dictionary
00467  1027  FA1,    TAD     REG8    / Save the start of head inn R6
00470  3026          DCA     REG6
00471  1427          TAD I   REG8    / Get entry from dictionary
00472  0052          AND     C777    / Get the count
00473  7041          NEG
00474  1430          TAD I   REG10   / Get the word count
00475  7440          SZA
00476  5325          JMP     FA5B    / Character counts didn't match

                                     / Compare chars in the pair of words
00477  2027  FA2,    ISZ     REG8    / Increment dictionary pointer to first char
00500  2030          ISZ     REG10   / Increment word pointer to first char
00501  7200          CLA             / Loop to match characters
00502  1427          TAD I   REG8    / Fetch a char in dictionary
```

```
00503  0052          AND     C777    / Mask out the stop bit (if there)
00504  7041          NEG
00505  1430          TAD I   REG10   / Get the character in word
00506  7440          SZA
00507  5326          JMP     FA5     / They're different, skip to next dictionary word

00510  7200  FA3,    CLA             / They're the same - keep going
00511  1427          TAD I   REG8    / Get the current chracter
00512  0054          AND     C1000   / Get the stop flag
00513  7450          SNA             / Jump to exit with word found
00514  5277          JMP     FA2     / Keep going -  go back for the next character

00515  2027  FA4,    ISZ     REG8    / Found a match - exit with found
00516  2027          ISZ     REG8    / Point R8 to Ex addr
00517  7200          CLA
00520  1027          TAD     REG8    / Get the execution address
00521  4113          JMS     PUSH    / Put it on the stack
00522  1026          TAD     REG6    / Get the start of the head
00523  4113          JMS     PUSH
00524  5134          JMP     EXNV
                                     / Current word wasn't it, need to go to next word
00525  2027  FA5B,   ISZ     REG8    / Point dictionary pointer to next char if from count
00526  7200  FA5,    CLA             / Entry for working with current char
00527  1427          TAD I   REG8    / Get the current word
00530  0054          AND     C1000   / Get the stop flag
00531  7450          SNA             / Jump on 0 flag to next dictionary character
00532  5325          JMP     FA5B
                                     / Current character is last in word
00533  2027          ISZ     REG8    / Go to the next word - that's the link
00534  7200          CLA
00535  1427          TAD I   REG8    / Get the link to next word in dictionary
00536  7450          SNA             / If zero were at the end of dictionary
00537  5344          JMP     FA6
00540  3027          DCA     REG8    / Store the new link in R8
00541  1031          TAD     REG12   / Restore the pointer to the word
00542  3030          DCA     REG10
00543  5267          JMP     FA1     /  and do it again

00544  7200  FA6,    CLA             / End of dictionary and we didn't find it
00545  4113          JMS     PUSH    / Push a zero on the stack
00546  5134          JMP     EXNV

                             / UStar Multiply two numbers at TOS
00547  1002  USTARH, 1002
00550  0325          "U;"*+1000
00551  1252
00552  0446          RONTOH
00553  4104  USTAR,  JMS     POP     / Multiplicand to MQ
00554  7421          MQL
00555  4104          JMS     POP     / Get Multiplier
00556  3360          DCA     USTAR1  /  and store in multiplier
00557  7405          7405            / MUY
00560  0000  USTAR1, 0
00561  3030          DCA     REG10   / Save upper part in R10
00562  7501          MQA
00563  3027          DCA     REG8    / Lower part to R8
00564  4443          JMS I   PUSH2   / Push R8 then R10 (upper in TOS)
00565  5134          JMP     EXNV    / High part (TOS), low part (TOS-1)

00566  0000  SPARE,  0
                             / UDiv: Divide double length by TOS
       7407  DVI=    7407
00567  1002  UDIVH, 1002
00570  0325          "U;"/+1000
00571  1257
00572  0547          USTARH
00573  4104  UDIV,   JMS     POP     / Divisor is at top of stack
00574  3026          DCA     REG6
00575  4442          JMS I   POP2    / TOS (high part) in R10
00576  1027          TAD     REG8    / Get low part
00577  7421          MQL             /  and put it in MQ
```

```
00600  1026          TAD     REG6
00601  3204          DCA     UDIV1
00602  1030          TAD     REG10   / High part to Acc
00603  7407          DVI             / DVI
00604  0000  UDIV1,  0               / Divisor
00605  4113          JMS     PUSH    / Remainder to stack
00606  7200          CLA
00607  7501          MQA             / Or in the quotient
00610  4113          JMS     PUSH    / Push remainder from MQ (TOS)
00611  5134          JMP     EXNV    / Remainder (TOS), quotient (TOS-1)

                             / Zero equals: -1 to stack if TOS zero
00612  1002  ZEQH,   1002
00613  0260          "0;"=+1000
00614  1275
00615  0567          UDIVH
00616  4104  ZEQ,    JMS     POP
00617  7440          SZA             / Skip if zero Acc
00620  7201          ONE             / Make it 1 if non-zero
00621  7041          NEG             / Change 1 to -1 but leave zero unchanged
00622  7040          CMA
00623  4113          JMS     PUSH
00624  5134          JMP     EXNV

                                     / (TOS-1) >= (TOS) ?
00625  3777  MAXNO,  3777
00626  1002  GEH,    1002
00627  0276          ">;"=+1000
00630  1275
00631  0612          ZEQH
00632  4442  GE,     JMS I   POP2    / Low in REG8, high in REG10
00633  7200          CLA
00634  1030          TAD     REG10
00635  7041          NEG
00636  1027          TAD     REG8
00637  7500          SMA
00640  5244          JMP     GE1
00641  7200          CLA
00642  4113  GE0,    JMS     PUSH
00643  5134          JMP     EXNV
00644  7240  GE1,    MINUS1
00645  5242          JMP     GE0

                                     / TOS = TOS-1
00646  1001  EQH,    1001
00647  1275          "=+1000
00650  0626          GEH
00651  4442  EQ,     JMS I   POP2    / TOS in R10
00652  3031          DCA     REG12   / Store 0 in REG12
00653  1027          TAD     REG8
00654  7041          NEG
00655  1030          TAD     REG10
00656  7450          SNA
00657  2031          ISZ     REG12   / REG12 is 1 if 0 result
00660  7200          CLA
00661  1031          TAD     REG12   / 1 if equal 0 if not
00662  7041          NEG
00663  4113          JMS     PUSH
00664  5134          JMP     EXNV

                             / Exit to OS8
00665  1004  STOPH,  1004
00666  0323          "S;"T;"O;"P+1000
00667  0324
00670  0317
00671  1320
00672  0646          EQH
00673  5674  STOP,   JMP I   STOPA
00674  7600  STOPA,  7600

                             / O.: (OhDot) print TOS in octal
```

```
00675  7774  MINUS4, 7774
00676  0260  C260,   260     / Char 0
00677  1002  ODOTH,  1002
00700  0317          "O;".+1000
00701  1256
00702  0665          STOPH
00703  7200  ODOT,   CLA
00704  1275          TAD     MINUS4
00705  3031          DCA     REG12
00706  4104          JMS     POP     / Get the value to print to R8
00707  7004          RAL             / Shift left into link
00710  5313          JMP     ODOT2
00711  7200  ODOT1,  CLA
00712  1104          TAD     POP     / Use Pop return for tmp
00713  7006  ODOT2,  RTL             / Cycle 3 bits into low of Acc
00714  7004          RAL
00715  3104          DCA     POP
00716  1104          TAD     POP
00717  0053          AND     C7
00720  1276          TAD     C260    / Make it a char by adding char0
00721  4121          JMS     SEROUT
00722  2031          ISZ     REG12
00723  5311          JMP     ODOT1
00724  5134          JMP     EXNV

                             / Compile word to dictionary
00725  1001  COMMAH, 1001
00726  1254          ",+1000
00727  0677          ODOTH
00730  4104  COMMA,  JMS     POP     / Get the value to store
00731  3425          DCA I   DP      / Save Acc in dictionary
00732  2025          ISZ     DP      / Update the dictionary ptr
00733  5134          JMP     EXNV

                             / Increment TOS
00734  1002  PLUS1H, 1002
00735  0261          "1;"++1000
00736  1253
00737  0725          COMMAH
00740  4104  PLUS1,  JMS     POP
00741  7001          IAC
00742  4113          JMS     PUSH
00743  5134          JMP     EXNV


                             / CMOVE: Move from bottom of source to target
                             /  TOS: count; next: target; next: source

00744  1005  CMOVEH, 1005
00745  0303          "C;"M;"O;"V;"E+1000
00746  0315
00747  0317
00750  0326
00751  1305
00752  0775          HEREH
00753  4444  CMOVE,  JMS I   POP3
00754  4356          JMS     CMOVER
00755  5134          JMP     EXNV

00756  0000  CMOVER, 0
00757  7240          MINUS1          / Count R12; target R10; source R8
00760  1027          TAD     REG8
00761  3010          DCA     10      / Source
00762  7240          MINUS1
00763  1030          TAD     REG10
00764  3011          DCA     11      / Target
00765  1031          TAD     REG12
00766  7041          NEG
00767  3031          DCA     REG12   / -Count
00770  1410  CMOVE1, TAD I   10
00771  3411          DCA I   11
```

```
00772  2031           ISZ     REG12
00773  5370           JMP     CMOVE1
00774  5756           JMP I   CMOVER

                              / Here: current dictionary pointer
00775  1004  HEREH,   1004
00776  0310           "H;"E;"R;"E+1000
00777  0305
01000  0322
01001  1305
01002  0734           PLUS1H
01003  1025  HERE,    TAD     DP      / Enter with 0 in Acc
01004  4113           JMS     PUSH
01005  5134           JMP     EXNV

                              / PAL version of TWIXT
                              / Arg in REG12
                              / Low, High in Rtn+1 and RTN+2
01006  0000  TWIXTR,  0
01007  7300           CAL
01010  1606           TAD I   TWIXTR  / Compare arg to low
01011  2206           ISZ     TWIXTR
01012  7041           NEG
01013  1031           TAD     REG12
01014  7500           SMA             / Skip if Arg >= low limit
01015  5221           JMP     TWIXT3
01016  2206  TWIXT1,  ISZ     TWIXTR
01017  7200  TWIXT2,  CLA             / Exit with false
01020  5606           JMP I   TWIXTR
01021  7300  TWIXT3,  CAL             / Compare to high limit
01022  1031           TAD     REG12
01023  7041           NEG
01024  1606           TAD I   TWIXTR
01025  2206           ISZ     TWIXTR
01026  7510           SPA             / Skip if high limit >= arg
01027  5217           JMP     TWIXT2
01030  7240           MINUS1          / Exit with true
01031  5606           JMP I   TWIXTR

                      / WORDAUX: Comes from FIND and other places
                      /       TOS: fence; input area
                      /       On exit: TOS: count; destination; source
                      /       Count is zero if no word found
                      /       TOIN is updated with count

01032  4442  WORDAUX, JMS I   POP2    / Get the fence (TOS R10) and input addr(R8)
01033  3026           DCA     REG6    / Initialize count
01034  1427  WA0,     TAD I   REG8    / If the leading char not the fence?
01035  7041           NEG
01036  1030           TAD     REG10
01037  7440           SZA
01040  5243           JMP     WA0B    / Wasn't a blank
01041  2027           ISZ     REG8    / Was the fence keep on going
01042  5234           JMP     WA0
01043  7200  WA0B,    CLA
01044  1027           TAD     REG8    / Save the input addr in REG12
01045  3031           DCA     REG12
01046  1427  WA1,     TAD I   REG8    / Compare char in buffer with fence
01047  7041           NEG
01050  1030           TAD     REG10
01051  7450           SNA
01052  5257           JMP     WA2     / Found it
01053  2027           ISZ     REG8    / Not the fence yet, go for another
01054  2026           ISZ     REG6    / Bump the char count
01055  7200           CLA             / Go back for another character
01056  5246           JMP     WA1
01057  7200  WA2,     CLA             / Found a word, get ready to CMOVE it
01060  1026           TAD     REG6    / Put the char cocunt at HERE
01061  3425           DCA I   DP
01062  1031           TAD     REG12   / Source to stack first
01063  4113           JMS     PUSH
```

```
01064  1025          TAD     DP      / Then dictionary pointer
01065  7001          IAC             / Pt the characters after the count
01066  4113          JMS     PUSH
01067  1026          TAD     REG6    / TOS: word count for CMOVE
01070  4113          JMS     PUSH
01071  1050          TAD     SI
01072  7041          NEG
01073  1027          TAD     REG8
01074  7001          IAC             / Point to the char after fence
01075  3034          DCA     TOIN
01076  5134          JMP     EXNV

01077  4001  NULLH,  4001            / High bit set means this is null
01100  1000          1000            / 0 plus flag bit
01101  0744          CMOVEH          / Skip over TWIXTH
01102  0000  NULL,   0

                     / Number: Address of word at TOS
                     / Simplified version of number - only does octal numbers
                     / REG6 is the character counter, REG8/10 the accumulator
                     / REG12 is the char being added, Reg14 is the pointer to buffer
01103  4000  INVALID, 4000
01104  0004  C4,     4
01105  3027  NUMBER, DCA     REG8    / Initialize result
01106  3030          DCA     REG10   / Accomodate a 24 bit result
01107  4104          JMS     POP     / Get address of word
01110  3032          DCA     REG14   / R14 is the index into the source
01111  1432          TAD I   REG14   / Get count
01112  4113          JMS     PUSH    /   and save it on the stack
01113  1432          TAD I   REG14   / Get the count again
01114  7041          NEG
01115  3026          DCA     REG6    / This is the negative of the count
01116  2032  NUMX1,  ISZ     REG14   / Point to next (or first) char of number
01117  1432          TAD I   REG14   / Character to examine
01120  3031          DCA     REG12   / Digit to examine in R12
01121  4447          JMS I   TWIXTAUX / Check against char 0 and char 7
01122  0260          260
01123  0267          267
01124  7450          SNA
01125  5365          JMP     NUMX2   / Wasn't a valid number
01126  7200          CLA
01127  1031          TAD     REG12
01130  0053          C7              / Make it a digit
01131  3031          DCA     REG12   / Save it as a digit
01132  1027          TAD     REG8    / This is the low part of current accumulation
01133  7421          MQL
01134  1030          TAD     REG10   / this is the high part of the result
01135  7413          7413            / Shift AD/MQ 3 places left (get this with 2)
01136  0002          2               / The result is AC/MQ
01137  3030          DCA     REG10   / Save the high part of the accumulation
01140  7501          MQA             / Get the lower part
01141  7100          CLL             / Clear the link
01142  1031          TAD     REG12   / Add the current digit to lower part
01143  3027          DCA     REG8    /  and save the updated accumulation
01144  2026          ISZ     REG6    / Inc count (R6) and get another digit
01145  5316          JMP     NUMX1
                                     / Finished
01146  4104          JMS     POP     / Put the count in R12
01147  3031          DCA     REG12
01150  1027          TAD     REG8    / Push the low part
01151  4113          JMS     PUSH
01152  7240          MINUS1          / Make 4 a three
01153  1031          TAD     REG12   / Get the count back
01154  0304          AND     C4      / Get the high bit
01155  7450          SNA
01156  5134          JMP     EXNV
01157  7200          CLA
01160  1030          TAD     REG10
01161  4113          JMS     PUSH
01162  1033          TAD     STATE
01163  7450          SNA
```

```
01164  5134          JMP     EXNV
01165  4442  NUMX2,  JMS I   POP2
01166  1303          TAD     INVALID / Wasn't a valid number, push invalid to stack
01167  4113          JMS     PUSH
01170  5134          JMP     EXNV

                     / Key:
01171  1003  KEYSUBH, 1003
01172  0313          "K;"E;"Y+1000
01173  0305
01174  1331
01175  1077          NULLH
01176  4127  KEY,    JMS     SERIN
01177  4113          JMS     PUSH    / Returns with 0 in Acc
01200  1421          TAD I   SP      / Echo the character to print
01201  4121          JMS     SEROUT
01202  5134          JMP     EXNV

                     / Emit Sub:
01203  1004  EMITH, 1004
01204  0305          "E;"M;"I;"T+1000
01205  0315
01206  0311
01207  1324
01210  1171          KEYSUBH
01211  4104  EMIT,   JMS     POP     / Get char to print
01212  4121          JMS     SEROUT
01213  5134          JMP     EXNV    / Returns with 0 n Acc

                                     / Print out a space
01214  1002  SPH,    1002            /  Note: Space is SP on command line
01215  0323          "S;"P+1000      /   but is spelled SPACE in assembly
01216  1320
01217  1203          EMITH
01220  1003  SPACE,  TAD     C40
01221  4121          JMS     SEROUT
01222  5134          JMP     EXNV

                     / No arguments, returns with args for CMOVE
                     / Comes from FIND and other places
01223  5141  WORD,   JMP     COLON
01224  0201          NR              / Add TOIN to source origin
01225  0034          TOIN
01226  0260          AT
01227  1600          LDSI            / Assume only input from TIB (stack origin)
01230  0417          PLUS
01231  0201          NR              / Load a blank for a fence
01232  0035          FENCE
01233  0260          AT
01234  1032          WORDAUX         / Returns arguments to move from
01235  0753          CMOVE           /    TIB to Here in Dictionary
01236  0241          SEMIC

                                     / No arguments, result: param (TOS) and head addr
01237  5141  FIND,   JMP     COLON
01240  1223          WORD            / Returns address of counted text at HERE
01241  2234          UPCASE
01242  1003          HERE            / Requires address of dictionary
01243  0460          FINDAUX         / Returns addr of head and params
01244  0241          SEMIC


                                     / A bunch of arguments for EXPECT
01245  0010  CBS,    10              / Ctrl-h (backspace)
01246  7763  MCR,    -15             / minus CR
01247  7601  MRO,    -177            / minus backspace
01250  7740  MSPACE, -40             / - space

                                     / Comes from QUIT
                                     /  Input char from keyboard to TIB (stack origin)
                                     /   limit to buffer size of 56 (70 octal)
                                     /  No arguments - no results
                                     / places null space at end of input
01251  1004  EXPECT, TAD     C100
01252  7041          NEG
01253  3027          DCA     REG8    / Initialize count
01254  1050          TAD     SI      / Initialize buffer ptr to stack origin
01255  3030          DCA     REG10
01256  4127  EX0,    JMS     SERIN
01257  3031          DCA     REG12   / REG8 has the char read in
01260  1031          TAD     REG12
01261  1247          TAD     MRO     / Compare to back space
01262  7440          SZA
01263  5306          JMP     EX1     / Wasn't a backspace
01264  1027          TAD     REG8    / Got BS, check if beginning of buffer
01265  1004          TAD     C100
01266  7450          SNA
01267  5256          JMP     EX0     / Was at the beginning, just start over
01270  7200          CLA
01271  1245          TAD     CBS     / Output a BS
01272  4121          JMS     SEROUT
01273  1003          TAD     C40
01274  4121          JMS     SEROUT
01275  1245          TAD     CBS
01276  4121          JMS     SEROUT
01277  7240          MINUS1          / Back up the buffer pointer
01300  1030          TAD     REG10
01301  3030          DCA     REG10
01302  7240          MINUS1          / Back up the count
01303  1027          TAD     REG8
01304  3027          DCA     REG8
01305  5256          JMP     EX0
01306  7200  EX1,    CLA             / Wasn't a BS - Check for CR
01307  1031          TAD     REG12
01310  1246          TAD     MCR     / Negative carriage return
01311  7450          SNA
01312  5335          JMP     EX3     / Was a CR
                                     / Check to make sure it's valid
01313  4447          JMS I   TWIXTAUX
01314  0040          40
01315  0172          172             / 172 is lower case z
01316  7450          SNA
01317  5256          JMP     EX0
01320  7200          CLA             / Store the character
01321  1031          TAD     REG12
01322  3430          DCA I   REG10
01323  1031          TAD     REG12   / And echo it to output
01324  4121          JMS     SEROUT
01325  2030          ISZ     REG10
01326  2027          ISZ     REG8    / Check to see if at end of buffer
01327  5256          JMP     EX0     / And go back to another
01330  1346          TAD     EXMSGA  / Ran out of buffer
01331  3030          DCA     REG10   / Print the "Too long" message
01332  4446          JMS I   DOTQAUX
01333  1050          TAD     SI
01334  3030          DCA     REG10
01335  7200  EX3,    CLA
01336  1003          TAD     C40     / At end store a space
01337  3430          DCA I   REG10
01340  2030          ISZ     REG10   / then a null
01341  3430          DCA I   REG10
01342  2030          ISZ     REG10   / And a space
01343  1003          TAD     C40
01344  3430          DCA I   REG10
01345  5134          JMP     EXNV
01346  1347  EXMSGA, EXMSG
01347  0011  EXMSG,  11
01350  0240          " ;"T;"o;"o;" ;"l;"o;"n;"g
01351  0324
01352  0357
01353  0357
01354  0240
```

```
01355  0354
01356  0357
01357  0356
01360  0347

01361  0015  CCR,    15              / Bunch of constants for CR
01362  0012  CLF,    12
01363  1002  CRH,    1002
01364  0303          "C;"R+1000
01365  1322
01366  1214          SPH
01367  1361  CR,     TAD     CCR
01370  4121          JMS     SEROUT
01371  1362          TAD     CLF
01372  4121          JMS     SEROUT
01373  5134          JMP     EXNV
                                     / Print a word whose addr is at TOS
01374  1002  WDOTH,  1002
01375  0327          "W;".+1000
01376  1256
01377  1363          CRH
01400  4104  WDOT,   JMS     POP     / Get the addr of the word to print
01401  3030          DCA     REG10
01402  4446          JMS I   DOTQAUX
01403  5134          JMP     EXNV

01404  1005  EXWOSH, 1005
01405  0305          "E;"X;"W;"O;"S+1000
01406  0330
01407  0327
01410  0317
01411  1323
01412  1374          WDOTH
01413  4104  EXWOS,  JMS     POP
01414  3023          DCA     EVR     / Go to this word without incrementing IP
01415  5423          JMP I   EVR

                                     / Print the word following call in word
01416  1020  DOTQ,   TAD     IP
01417  3030          DCA     REG10
01420  4446          JMS I   DOTQAUX / Returns 0 in Acc
01421  1030          TAD     REG10
01422  3020          DCA     IP
01423  2020          ISZ     IP      / IP points to next word
01424  5134          JMP     EXNV

                             / Dot Quote: Print following string
01425  3002  DOTQH,  3002            / Set immediate bit
01426  0256          ".;""+1000
01427  1242
01430  1404          EXWOSH
01431  5141          JMP     COLON
01432  1605          LDSTATE         / If compiling, state not 0
01433  0154          JUMPZ
01434  1440          NEWDQ1
01435  0201          NR              / Compiling, put in call to DOTQ
01436  1416          DOTQ
01437  0730          COMMA
01440  3011  NEWDQ1, STOREAT         / Store quote at fence
01441  0035          FENCE
01442  0042          42
01443  1223          WORD
01444  1605          LDSTATE         / If compiling, adjust dictionary
01445  0154          JUMPZ
01446  1452          NEWDQ2
01447  3020          INCDP
01450  0150          JUMP
01451  1455          NEWDQ3
01452  1644  NEWDQ2, LOADAT          / Section for immediate
01453  0025          DP              / Address to print is in DP
01454  1400          WDOT
```

```
01455  3011  NEWDQ3, STOREAT         / Return the fence to a space
01456  0035          FENCE
01457  0040          40
01460  0241          SEMIC

             / Print the string at REG10
01461  0000  DOTQR, 0
01462  1430          TAD I   REG10   / Count
01463  0052          C777            / And out the high bits
01464  7041          NEG
01465  3027          DCA     REG8
01466  2030  DOTQA,  ISZ     REG10
01467  1430          TAD I   REG10   / Get the char to print
01470  4121          JMS     SEROUT  / Returns 0 in Acc
01471  2027          ISZ     REG8
01472  5266          JMP     DOTQA
01473  5661          JMP I   DOTQR


                                     / Check for an invalid number (4000)
01474  4104  VALIDN, JMS     POP
01475  7104          CLL RAL
01476  7440          SZA
01477  7240          MINUS1          / Non-zero in low 11 bits
01500  7420          SNL
01501  7240          MINUS1          / Zero in link
01502  4113          JMS     PUSH
01503  5134          JMP     EXNV

01504  1515  NEADDR, NE1
01505  1025  NUMERR, TAD     DP      / Print out offending word
01506  3030          DCA     REG10
01507  4446          JMS I   DOTQAUX / It's at the dictionary pointer
01510  1003          TAD     C40     / Print a space
01511  4121          JMS     SEROUT
01512  1304          TAD     NEADDR  / Print out WHAT?
01513  3030          DCA     REG10
01514  4446          JMS I   DOTQAUX
01515  0005  NE1,    5
01516  0327          "W;"H;"A;"T;"?
01517  0310
01520  0301
01521  0324
01522  0277
01523  5134          JMP     EXNV

                     / Interpreter
01524  5141  INTERP, JMP COLON
01525  1237  INT0,   FIND
01526  0306          DUP             / Zero if we couldn't find it
01527  0154          JUMPZ           / Couldn't find it - try number
01530  1553          INT3
01531  0260  INT1,   AT              / Found the word in dictionary
01532  0306          DUP             / Returns with head addr (TOS) and ex addr
01533  0165          JUMPMI          / Minus at head addr means end of text
01534  1573          INT4            / Quietly exit
01535  0201          NR              / Dig out the immediate bit
01536  2000          2000            / Bit 2 set means immediate
01537  0373          FAND
01540  0161          JUMPNZ          / Not set - go execute it
01541  1550          INT2
01542  1605          LDSTATE         / Ask if we are compiling
01543  0154          JUMPZ
01544  1550          INT2            / Not compiling - go execute it
01545  0730          COMMA           / Compile the word's ex addr
01546  0150          JUMP
01547  1525          INT0            / And go back for another word
01550  1413  INT2,   EXWOS           / Exec word on stack
01551  0150          JUMP            / And go back for another word
01552  1525          INT0
01553  0364  INT3,   DROP            / Drop the zero we duped
```

```
01554  1003          HERE
01555  1105          NUMBER          / Didn't find it, try number
01556  0306          DUP
01557  1474          VALIDN          / If valid, put on the stack
01560  0154          JUMPZ           /  if zero - an invalid number
01561  1576          INT5
01562  1605          LDSTATE         / Are we compiling?
01563  0154          JUMPZ
01564  1525          INT0            / Not compiling - just leave No. on stack
01565  0201          NR              / Compile a NR address
01566  0201          NR
01567  0730          COMMA
01570  0730          COMMA           / Then the number
01571  0150          JUMP            / Go back for another word
01572  1525          INT0
01573  0364  INT4,   DROP            / Drop the zero duped from FIND
01574  0364          DROP            / Drop Null's execution address
01575  0241          SEMIC
01576  1505  INT5,   NUMERR
01577  1610          QUIT

01600  1050  LDSI,   TAD     SI
01601  4113          JMS     PUSH
01602  5134          JMP     EXNV

01603  4113  LDZERO, JMS     PUSH
01604  5134          JMP     EXNV

01605  1033  LDSTATE, TAD    STATE
01606  4113          JMS     PUSH
01607  5134          JMP     EXNV


                                     / QUIT: Loops calling for keyboard input
                                     / Calls EXPECT then INTERP repeatedly
                                     /  executing or compiling each word from input
01610  5141  QUIT,   JMP     COLON
01611  1637          INIT
01612  3011          STOREAT         / Set STATE to immediate
01613  0033          STATE
01614  0000          0
01615  1367  QUIT1,  CR
01616  1416          DOTQ            / Put out CR, OK message
01617  0002          2
01620  0317          "O;"K
01621  0313
01622  1367  QUIT2,  CR
01623  1603          LDZERO          / Initialize buffer pointer
01624  0201          NR
01625  0034          TOIN
01626  0251          STORE
01627  1251          EXPECT          / No arguments, no results
01630  1367          CR
01631  1524          INTERP          / Process words up to a null
01632  1605          LDSTATE         / If compiling, just ask for another line
01633  0154          JUMPZ           / Zero means not compiling
01634  1615          QUIT1           / Not compiling - put out OK message
01635  0150          JUMP
01636  1622          QUIT2           / Compiling, ask for another line

01637  1050  INIT,   TAD     SI
01640  3021          DCA     SP
01641  1051          TAD     RSI
01642  3022          DCA     RSP
01643  5134          JMP     EXNV
                                     / Alternate for NR xxx @
                                     / Assumes address to load follows LOADAT
01644  1420  LOADAT, TAD I   IP
01645  3027          DCA     REG8
01646  1427          TAD I   REG8
01647  4113          JMS     PUSH
```

```
01650  2020          ISZ     IP
01651  5134          JMP     EXNV

                                     / ?S: Prints the current status of the stack
01652  1002  QSH,    1002
01653  0277          "?;"S+1000
01654  1323
01655  1425          DOTQH
01656  5141  QSTACK, JMP COLON
01657  1644          LOADAT          / Get the start of the stack
01660  0050          SI
01661  2126          LOAD1           / Load one on the stack
01662  0430          MINUS
01663  1644          LOADAT          / Get the current stack pointer
01664  0021          SP              / @SI @SP
01665  0273          SWAP            / @SP @SI
01666  0323          OVER
01667  0323          OVER            / @SP @SI @SP @SI
01670  0651          EQ
01671  0161          JUMPNZ          / Returns true if equal
01672  1722          QS3
01673  1367          CR
01674  1220  QS1,    SPACE           / @SP @S
01675  0306          DUP             / @SP @S @S
01676  0260          AT              / @SP @S Char
01677  0703          ODOT            / @SP @S
01700  2126          LOAD1           / @SP @S 1
01701  0430          MINUS           / @SP @S-1
01702  0323          OVER
01703  0323          OVER            / @SP @S-1 @SP @S-1
01704  0651          EQ              /
01705  0154          JUMPZ           / Jump if false
01706  1674          QS1
01707  1416          DOTQ
01710  0006          6
01711  0240          240;"(;"T;"O;"S;")
01712  0250
01713  0324
01714  0317
01715  0323
01716  0251
01717  0364  QS2,    DROP            / Drop the @SI and @SP
01720  0364          DROP
01721  0241          SEMIC
01722  1416  QS3,    DOTQ
01723  0006          6
01724  0305          "E;"M;"P;"T;"Y;215
01725  0315
01726  0320
01727  0324
01730  0331
01731  0215
01732  0150          JUMP
01733  1717          QS2

                                     / Compiled verson of TWIXT
                                     / Arg on stack
                                     / Low and high follow call to TWIXT
                                     / See example in DEC
01734  4104  TWIXT,  JMS     POP
01735  3031          DCA     REG12
01736  1420          TAD I   IP
01737  3345          DCA     .+6
01740  4055          JMS     INCIP
01741  1420          TAD I   IP
01742  3346          DCA     .+4
01743  4055          JMS     INCIP
01744  4447          JMS I   TWIXTAUX
01745  0000          0
01746  0000          0
01747  4113          JMS     PUSH
```

```
01750  5134          JMP     EXNV

01751  1004  STODH,  1004
01752  0323          "S;"-;">;"D+1000
01753  0255
01754  0276
01755  1304
01756  1652          QSH
01757  4104  STOD,   JMS     POP
01760  3026          DCA     REG6
01761  1026          TAD     REG6
01762  4113          JMS     PUSH    / Push the low part
01763  1026          TAD     REG6
01764  7500          SMA
01765  5370          JMP     STOD1
01766  7240          MINUS1
01767  5371          JMP     STOD2
01770  7200  STOD1,  CLA
01771  4113  STOD2,  JMS     PUSH    / Push the high part
01772  5134          JMP     EXNV

01773  1002  DDOTH,  1002
01774  0304          "D;".+1000
01775  1256
01776  1751          STODH
01777  5141  DDOT,   JMP     COLON
02000  0703          ODOT
02001  0703          ODOT
02002  0241          SEMIC



02003  1002  DPLUSH, 1002
02004  0304          "D;"++1000
02005  1253
02006  1773          DDOTH
02007  4444  DPLUS,  JMS I   POP3
02010  4104          JMS     POP     / 8,10 as one double number
02011  3032          DCA     REG14   / 12,14 as the other
02012  7100          CLL
02013  1030          TAD     REG10
02014  1032          TAD     REG14
02015  4113          JMS     PUSH
02016  7430          SZL
02017  7201          ONE
02020  7100          CLL
02021  1027          TAD     REG8
02022  1031          TAD     REG12
02023  4113          JMS     PUSH
02024  5134          JMP     EXNV

02025  4442  DCOM,   JMS I   POP2
02026  1027          TAD     REG8
02027  7040          CMA
02030  3027          DCA     REG8
02031  1030          TAD     REG10
02032  7040          CMA
02033  3030          DCA     REG10
02034  4443          JMS I   PUSH2
02035  5134          JMP     EXNV

02036  1004  DNEGH,  1004
02037  0304          "D;"N;"E;"G+1000
02040  0316
02041  0305
02042  1307
02043  2003          DPLUSH
02044  5141  DNEG,   JMP     COLON
02045  2025          DCOM
02046  2126          LOAD1
02047  2133          LOAD0
```

```
02050  2007          DPLUS
02051  0241          SEMIC
                                     / Print the signed double word at TOS
                                     / DEC 123456 DNEG OVER OVER D. SP .
                                     /  Gives: 77416700 -123456
02052  1001  DOTH,   1001
02053  1256          ".+1000
02054  2036          DNEGH
02055  5141  DOT,    JMP COLON
02056  0306          DUP
02057  0171          JUMPNM
02060  2065          DOTA
02061  0201          NR
02062  0055          55
02063  1211          EMIT
02064  2044          DNEG
02065  2133  DOTA,   LOAD0
02066  0352          RROT            / Stack: 0 al ah
02067  2133  DOT1,   LOAD0           / Divide Arg(high) by 10
02070  0201          NR
02071  0012          12
02072  0573          UDIV
02073  0442          ONTOR           / Save qh on RS
02074  0201          NR              / Divide rem low part by 10
02075  0012          12
02076  0573          UDIV            / Stack: rem ql (TOS)
02077  0452          RONTO           / Get back qh. Stack: rem ql qh (TOS)
02100  0334          ROT             / Stack: ql qh rem
02101  0201          NR              / Make it a printable digit
02102  0060          60
02103  0417          PLUS
02104  0352          RROT            / Stack: 0 d1 ql qh
02105  0323          OVER
02106  0323          OVER
02107  0404          OR
02110  0161          JUMPNZ
02111  2067          DOT1
02112  0364  DOT2,   DROP            / Stack: 0,d1,d2,ql,qh
02113  0364          DROP
02114  1211  DOT3,   EMIT            / Stack: 0,d1,d2
02115  0306          DUP
02116  0161          JUMPNZ
02117  2114          DOT3
02120  0364          DROP            / Drop the sentinel
02121  0241          SEMIC

02122  4104  NEGATE, JMS     POP
02123  7041          NEG
02124  4113          JMS     PUSH
02125  5134          JMP     EXNV

                                     / Load 1 on the stack
02126  7201  LOAD1,  ONE
02127  4113  L1A,    JMS     PUSH
02130  5134          JMP     EXNV
                                     / Load -1 on stack
02131  7240  LOADM1, MINUS1
02132  5327          JMP L1A
                                     / Load 0 on stack
02133  4113  LOAD0,  JMS     PUSH
02134  5134          JMP     EXNV

02135  4444  BCMOVE, JMS I   POP3    / R8 source, R10 target, R12 count
02136  1031          TAD     REG12
02137  7041          NEG
02140  3031          DCA     REG12
02141  1427  BC1,    TAD I   REG8
02142  3430          DCA I   REG10
02143  7240          MINUS1
02144  1027          TAD     REG8
02145  3027          DCA     REG8
```

```
02146  7240          MINUS1
02147  1030          TAD     REG10
02150  3030          DCA     REG10
02151  2031          ISZ     REG12
02152  5341          JMP     BC1
02153  5134          JMP     EXNV

02154  1425  CREAUX, TAD I   DP      / Save char count in R10
02155  3030          DCA     REG10
02156  1425          TAD I   DP      / Add 1000 to char count
02157  1054          TAD     C1000
02160  3425          DCA I   DP
02161  1030          TAD     REG10   / Add 1000 to last char
02162  1025          TAD     DP
02163  3027          DCA     REG8    / R8 = addr of last char
02164  1427          TAD I   REG8
02165  1054          TAD     C1000
02166  3427          DCA I   REG8
02167  1024          TAD     VL      / Add link to prior in new def
02170  2027          ISZ     REG8
02171  3427          DCA I   REG8
02172  1025          TAD     DP      / VocLnk = DP
02173  3024          DCA     VL
02174  1027          TAD     REG8    / DP = DP + CC + 1
02175  7001          IAC
02176  3025          DCA     DP
02177  5134          JMP     EXNV
                                     / Create Head
02200  1002  CREATH, 1002
02201  0303          "C;"H+1000
02202  1310
02203  2052          DOTH
02204  5141  CREATE, JMP     COLON
02205  1237          FIND
02206  0161          JUMPT
02207  2212          CR1
02210  2154          CREAUX
02211  0241          SEMIC
02212  0364  CR1,    DROP
02213  2225          CRX
02214  1416          DOTQ
02215  0006          6
02216  0240          240;"R;"E;"D;"E;"F
02217  0322
02220  0305
02221  0304
02222  0305
02223  0306
02224  1610          QUIT

02225  1025  CRX,    TAD     DP      / Print out offending word
02226  3030          DCA     REG10
02227  4446          JMS I   DOTQAUX / It's at the dictionary pointer
02230  5134          JMP     EXNV

                                     / Upper case word at HERE
02231  7640  MLCA,   -140
02232  7740  M40,    -40
02233  0340  C340,   340
02234  1425  UPCASE, TAD I   DP      / Get the word count
02235  7041          NEG
02236  3027          DCA     REG8
02237  1025          TAD     DP      / Put the DP pointer in REG10
02240  3030          DCA     REG10
02241  2030  UC1,    ISZ     REG10   / Point to the next word
02242  1430          TAD I   REG10
02243  7450          SNA             / Leave zero alone
02244  5252          JMP     UC2
02245  1231          TAD     MLCA    / Compare to (lower case) a
02246  7500          SMA
02247  1232          TAD     M40     / And out the lower case bit
```

```
02250  1233          TAD     C340    / Add in the overpunch
02251  3430          DCA I   REG10   / Store back in dictionary
02252  2027  UC2,    ISZ     REG8
02253  5241          JMP     UC1
02254  5134          JMP     EXNV

02255  1001  COLONH, 1001
02256  1272          ":+1000
02257  2200          CREATH
02260  5141          JMP     COLON
02261  2204          CREATE
02262  0201          NR
02263  5141          JMP     COLON
02264  0730          COMMA
02265  3005          TOGST
02266  0241          SEMIC

02267  3001  SEMICH, 3001
02270  1273          ";+1000         / Set to immediate
02271  2255          COLONH
02272  5141          JMP     COLON
02273  0201          NR
02274  0241          SEMIC
02275  0730          COMMA
02276  3005          TOGST
02277  2301          INITSP
02300  0241          SEMIC

02301  1050  INITSP, TAD     SI
02302  3021          DCA     SP
02303  5134          JMP     EXNV

02304  1006  FGH,    1006
02305  0306          "F;"O;"R;"G;"E;"T+1000
02306  0317
02307  0322
02310  0307
02311  0305
02312  1324
02313  2267          SEMICH
02314  5141  FORGET, JMP     COLON
02315  1237          FIND            / Get the argument
02316  0306          DUP
02317  0154          JUMPZ
02320  2334          FG1
02321  0273          SWAP            / Get the exec addr
02322  2126          LOAD1           / Point back to link
02323  0430          MINUS
02324  0260          AT              / Get the link
02325  0201          NR              / Store the link in VL
02326  0024          VL
02327  0251          STORE
02330  0201          NR              / Store the head addr in DP
02331  0025          DP
02332  0251          STORE
02333  0241          SEMIC
02334  2225  FG1,    CRX
02335  1416          DOTQ
02336  0006          6
02337  0240          240;"N;"O;"D;"E;"F
02340  0316
02341  0317
02342  0304
02343  0305
02344  0306
02345  1610          QUIT

                                     / Stack entries for DO LEAVE LOOP
                                     /  and IF ELSE THEN
                                     /
                                     / >0 IF ELSE THEN
                                     /  0 LEAVE
                                     / -1 Loop-end (addr at DO)
02346  3002  DOH,    3002            / Set immediate bit
02347  0304          "D;"O+1000
02350  1317
02351  2304          FGH
02352  5141          JMP     COLON
02353  0201          NR              / Put in a call to Doaux
02354  2361          DOAUX
02355  0730          COMMA
02356  1003          HERE            / Address of loop head to stack
02357  2133          LOAD0           / Zero to stack to indicate position of DO
02360  0241          SEMIC

02361  5141  DOAUX,  JMP     COLON
02362  0452          RONTO
02363  0352          RROT            / Put return at bottom
02364  0442          ONTOR
02365  0442          ONTOR           / Limit at top of RS
02366  0442          ONTOR           / Put the return back
02367  0241          SEMIC

02370  3004  LOOPH,  3004            / Make it immediate
02371  0314          "L;"O;"O;"P+1000
02372  0317
02373  0317
02374  1320
02375  2346          DOH
02376  5141          JMP     COLON
                                     / Process LEAVEs and look for DO addr
02377  0306  L0,     DUP             / Is this the sentinel for the DO?
02400  0154          JUMPF
02401  2421          L3
02402  0306  L1,     DUP             / Get candidate hole addr
02403  0260          AT              / Stack: hole addr, (hole)
02404  0154          JUMPF           / Go because
02405  2411          L2              /  it's a LEAVE
02406  0364          DROP            / It was an IF-ELSE_THEN hole
02407  0150          JUMP            / Drop current entry, go to next
02410  2377          L0
02411  1003  L2,     HERE            / Process the LEAVE hole
02412  0201          NR
02413  0003          3
02414  0417          PLUS
02415  0273          SWAP
02416  0251          STORE
02417  0150          JUMP            / Go back looking for loop-end
02420  2377          L0
                                     / Process the loop-end
02421  0364  L3,     DROP            / Drop the sentinel
02422  0201          NR              / Put in call to LOOPAUX
02423  2432          LOOPAUX
02424  0730          COMMA
02425  0201          NR              / Put in a JUMPZ inst
02426  0154          JUMPZ
02427  0730          COMMA
02430  0730          COMMA           / Put in the branch addr that's on the stack
02431  0241          SEMIC

02432  5141  LOOPAUX, JMP    COLON
02433  0452          RONTO           / Get the return
02434  0452          RONTO           / This is the limit
02435  0452          RONTO           / This is the index
02436  0740          PLUS1           / Increment index
02437  0323          OVER
02440  0323          OVER
02441  0273          SWAP
02442  0632          GE
02443  0154          JUMPF
02444  2452          LA1
02445  0364          DROP            / Drop the limit and index
```

```
02446  0364          DROP
02447  0442          ONTOR           / Put the return back
02450  2126          LOAD1           / Return true to skip out of loop
02451  0241          SEMIC
02452  0442  LA1,    ONTOR           / This is the index
02453  0442          ONTOR           / This is the limit
02454  0442          ONTOR           / This is the return
02455  2133          LOAD0           / Return false to stay in loop
02456  0241          SEMIC

02457  1001  IH,     1001            / Get the index off of RS
02460  1311          "I+1000
02461  2370          LOOPH
02462  1022          TAD     RSP
02463  7001          IAC
02464  3027          DCA     REG8
02465  1427          TAD I   REG8
02466  4113          JMS     PUSH
02467  5134          JMP     EXNV

02470  5141  IETAUX, JMP     COLON
02471  1644          LOADAT
02472  0021          SP
02473  0306  IET1,   DUP
02474  0260          AT
02475  0260          AT              / Get the value at the hole
02476  0740          PLUS1
02477  0154          JUMPF
02500  2505          IET2            / If not -1
02501  2126          LOAD1           / It wasn't an IET hole
02502  0417          PLUS
02503  0150          JUMP
02504  2473          IET1
02505  0260  IET2,   AT              / Get the hole addr
02506  0241          SEMIC           / Leave with the IET addr on stack

02507  3002  IFH,    3002
02510  0311          "I;"F+1000
02511  1306
02512  2457          IH
02513  5141  IF,     JMP     COLON
02514  0201          NR              / Put in a call to JUMPZ
02515  0154          JUMPZ
02516  0730          COMMA
02517  1003          HERE            / Addr of the hole
02520  2131          LOADM1          / Load the hole with -1
02521  0730          COMMA           / Make a hole
02522  0241          SEMIC

02523  3004  ELSEH,  3004
02524  0305          "E;"L;"S;"E+1000
02525  0314
02526  0323
02527  1305
02530  2507          IFH
02531  5141  ELSE,   JMP     COLON   / Enters looking for hole addr
02532  2470          IETAUX          / Get the hole addr
02533  0201          NR              / Put in a jump before the THEN
02534  0150          JUMP
02535  0730          COMMA
02536  1003          HERE            / Stack: IF addr, New hole addr
02537  2131          LOADM1          / Make a hole for the JUMP
02540  0730          COMMA           / Load the new hole with -1
02541  0273          SWAP            / Stack: New hole addr, IF addr
02542  1003          HERE            / Stack: New hole addr, IF addr, here
02543  0273          SWAP            / Stack: Hole addr, here, If addr
02544  0251          STORE           / Stack: hole addr
02545  0241          SEMIC

02546  3004  THENH,  3004
02547  0324          "T;"H;"E;"N+1000
```

```
02550  0310
02551  0305
02552  1316
02553  2523          ELSEH
02554  5141  THEN,   JMP     COLON   / Enters looking for hole addr
02555  2470          IETAUX          / Get the hole addr
02556  1003          HERE            / Put in the addr of the JUMP
02557  0273          SWAP            /  that got us here
02560  0251          STORE
02561  0241          SEMIC

02562  3630  HEAD,   TEXT    "^X EXIT, ^B CLEAR, ^N NEW LINE, ^Y DEL LINE, ^G GO, ^O INS/OVR: "
02563  4005
02564  3011
02565  2454
02566  4036
02567  0240
02570  0314
02571  0501
02572  2254
02573  4036
02574  1640
02575  1605
02576  2740
02577  1411
02600  1605
02601  5440
02602  3631
02603  4004
02604  0514
02605  4014
02606  1116
02607  0554
02610  4036
02611  0740
02612  0717
02613  5440
02614  3617
02615  4011
02616  1623
02617  5717
02620  2622
02621  7240
02622  0000

                                     / Get three char from pair
                                     / Addr of pair in REG14
02623  0000  GET3,   0
02624  1432          TAD I   REG14   / Get first one
02625  0252          AND     C377
02626  3027          DCA     REG8    / Store the first one in REG8
02627  1432          TAD I   REG14   / Get the low part of second
02630  7006          RTL
02631  7006          RTL
02632  7004          RAL
02633  0251          AND     C17
02634  3030          DCA     REG10   / Store the low part in REG14
02635  2032          ISZ     REG14   / Point to the high of the pair
02636  1432          TAD I   REG14
02637  7012          RTR
02640  7012          RTR
02641  0253          AND     C360
02642  1030          TAD     REG10
02643  3030          DCA     REG10   / Store middle char in REG10
02644  1432          TAD I   REG14   / Get the last char
02645  0252          AND     C377    / The last char in AC
02646  3031          DCA     REG12   / Store last in REG12
02647  2032          ISZ     REG14   / Point REG14 to next pair
02650  5623          JMP I   GET3

                                     / Unpack 21 (dec) pairs (one row) of triplets
```

```
02651  0017  C17,    17
02652  0377  C377,   377
02653  0360  C360,   360
02654  7400  C7400,  7400
02655  4157  SM1,    S-1
02656  4104  UNPACK, JMS     POP     / Get the row address
02657  3032          DCA     REG14   / Row addr in REG14
02660  1006          TAD     M25     / Do 21 (dec) triplets
02661  3026          DCA     REG6    / Count in REG6
02662  1255          TAD     SM1     / Start of the text buffer-1
02663  3010          DCA     10      / Text buffer addr in 10 autoindex
02664  4223  UP1,    JMS     GET3    / REG8, REG10 and REG12 will be the chars
02665  1027          TAD     REG8    /  in the pair at REG12
02666  3410          DCA I   10
02667  1030          TAD     REG10
02670  3410          DCA I   10
02671  1031          TAD     REG12
02672  3410          DCA I   10
02673  2026          ISZ     REG6
02674  5264          JMP     UP1
02675  5134          JMP     EXNV

02676  4104  PRINTL, JMS     POP     / Get the row address
02677  3032          DCA     REG14   / Row addr in REG14
02700  1006          TAD     M25     / Do 21 (dec) triplets
02701  3026          DCA     REG6    / Count in REG8
02702  4223  PL1,    JMS     GET3    / Put the triplet on the stack
02703  1027          TAD     REG8
02704  4121          JMS     SEROUT
02705  1030          TAD     REG10
02706  4121          JMS     SEROUT
02707  1031          TAD     REG12
02710  4121          JMS     SEROUT
02711  2026          ISZ     REG6
02712  5302          JMP     PL1
02713  5134          JMP     EXNV

                                     / Pack 21 (dec) pairs (one row) of triplets
02714  4104  PACK,   JMS     POP     / Get the row address
02715  3030          DCA     REG10
02716  1006          TAD     M25     / Do 21 (dec) triplets
02717  3027          DCA     REG8
02720  1050          TAD     SI      / Start of the text buffer
02721  3031          DCA     REG12
02722  1431  PACK1,  TAD I   REG12   / Get first char from text buffer
02723  3430          DCA I   REG10   / Store it in the low part of W1
02724  2031          ISZ     REG12   / Point to the next char in text buffer
02725  1431          TAD I   REG12   /  and get that char
02726  7012          RTR             / Rotate the low 4 bits to high part
02727  7012          RTR
02730  7010          RAR
02731  0254          AND     C7400   / And out the four high bits
02732  1430          TAD I   REG10
02733  3430          DCA I   REG10   / And store the result in W1
02734  2030          ISZ     REG10   / Go to W2
02735  1431          TAD I   REG12   / Get the 2nd char back
02736  7006          RTL             / Rotate the high bits left 4 places
02737  7006          RTL
02740  0254          AND     C7400
02741  3430          DCA I   REG10   / Store it in the high 4 bits of W2
02742  2031          ISZ     REG12   / Point to the third char
02743  1431          TAD I   REG12   / Get the third char
02744  1430          TAD I   REG10   / Add it to the high bits already there
02745  3430          DCA I   REG10
02746  2030          ISZ     REG10   / Point to next char in text buffer
02747  2031          ISZ     REG12   / Point to next pair in packed buffer
02750  2027          ISZ     REG8
02751  5322          JMP     PACK1
02752  5134          JMP     EXNV

02753  5141  LEAUX,  JMP     COLON
```

```
02754  0452          RONTO           / Save the return
02755  0452          RONTO           / Get rid of the limit and index
02756  0364          DROP
02757  0452          RONTO
02760  0364          DROP
02761  0442          ONTOR
02762  0241          SEMIC

02763  3005  LEAVEH, 3005            / Set immediate bit
02764  0314          "L;"E;"A;"V;"E+1000
02765  0305
02766  0301
02767  0326
02770  1305
02771  2546          THENH
02772  5141  LEAVE,  JMP     COLON
02773  0201          NR
02774  2753          LEAUX
02775  0730          COMMA
02776  0201          NR
02777  0150          JUMP
03000  0730          COMMA
03001  1003          HERE            / Address of the hole
03002  2133          LOAD0           / Fill the hole with 0
03003  0730          COMMA
03004  0241          SEMIC

03005  1033  TOGST,  TAD     STATE
03006  7040          CMA
03007  3033          DCA     STATE
03010  5134          JMP     EXNV
                                     / Alternative to NR arg NR addr STORE
                                     / Call: STOREAT Addr Arg
03011  1420  STOREAT, TAD I  IP      / Address of store at next word
03012  3027          DCA     REG8
03013  4055          JMS     INCIP
03014  1420          TAD I   IP      / Thing to store at second word
03015  3427          DCA I   REG8
03016  4055          JMS     INCIP
03017  5134          JMP     EXNV
                                     / Inc DP by inc at DP
                                     / Inc on stack
03020  1425  INCDP,  TAD I   DP      / (DP) aka char count
03021  1025          TAD     DP
03022  7001          IAC
03023  3025          DCA     DP
03024  5134          JMP     EXNV

03025  1036  LOADRN, TAD     ROWNO
03026  4113          JMS     PUSH
03027  5134          JMP     EXNV

03030  1037  LOADCN, TAD     COLNO
03031  4113          JMS     PUSH
03032  5134          JMP     EXNV

03033  7770  M10,    -10
03034  1036  LOADMRN, TAD    ROWNO
03035  1233          TAD     M10
03036  4113          JMS     PUSH
03037  5134          JMP     EXNV

03040  5141  ROWADD, JMP     COLON
03041  3025          LOADRN
03042  0201          NR
03043  0052          52
03044  0553          USTAR
03045  0364          DROP
03046  0201          NR
03047  7000          7000
03050  0417          PLUS
```

```
03051  0241          SEMIC

03052  5141  PAINT,  JMP     COLON
03053  1416          DOTQ
03054  0004          4
03055  0033          33;133;62;112
03056  0133
03057  0062
03060  0112
03061  1416          DOTQ
03062  0006          6
03063  0033          33;133;61;73;61;110
03064  0133
03065  0061
03066  0073
03067  0061
03070  0110
03071  0201          NR
03072  0100          100
03073  0201          NR
03074  2562          HEAD
03075  3302          PLINE
03076  3327          PIO
03077  1367          CR
03100  0201          NR
03101  7770          -10
03102  0201          NR
03103  7000          7000            / -Row count, buff addr (TOS)
03104  1416  PAINT1, DOTQ
03105  0001          1
03106  0333          "[
03107  0323          OVER            / Print line from text buf?
03110  3034          LOADMRN         / Load LineNo -10
03111  0651          EQ
03112  0154          JUMPF
03113  3136          PAINT3
03114  0201          NR              / Print from buffer
03115  7701          -77
03116  1644          LOADAT
03117  0050          SI
03120  0306  PAINT2, DUP
03121  0260          AT
03122  1211          EMIT
03123  0740          PLUS1
03124  0273          SWAP
03125  0740          PLUS1
03126  0273          SWAP
03127  0323          OVER
03130  0161          JUMPNZ
03131  3120          PAINT2
03132  0364          DROP            / Drop the addr and count
03133  0364          DROP
03134  0150          JUMP
03135  3140          PAINT4
03136  0306  PAINT3, DUP
03137  2676          PRINTL
03140  1416  PAINT4, DOTQ
03141  0001          1
03142  0335          "]
03143  1367          CR
03144  0323          OVER            / Print the CLine?
03145  3034          LOADMRN         / Load LineNo -10
03146  0651          EQ
03147  0154          JUMPF
03150  3166          PAINT6
03151  3030          LOADCN          / Yes - print the CLine
03152  0740          PLUS1
03153  2122          NEGATE
03154  1220  PAINT5, SPACE
03155  0740          PLUS1
03156  0306          DUP
```

```
03157  0161          JUMPNZ
03160  3154          PAINT5
03161  0364          DROP
03162  1416          DOTQ
03163  0001          1
03164  0336          "^
03165  1367          CR
03166  0201  PAINT6, NR
03167  0052          52      / Add 42 (dec) to go to next line
03170  0417          PLUS
03171  0273          SWAP
03172  0740          PLUS1
03173  0273          SWAP
03174  0323          OVER
03175  0161          JUMPNZ
03176  3104          PAINT1
03177  0364          DROP
03200  0364          DROP
03201  0241          SEMIC
                                     / Get a char with no echo
03202  4127  GETCHR, JMS     SERIN
03203  4113          JMS     PUSH
03204  5134          JMP     EXNV

03205  5141  LOADIO, JMP     COLON
03206  0201          NR
03207  0040          INSOVR
03210  0260          AT
03211  0241          SEMIC

03212  5141  GO,     JMP     COLON
03213  1367          CR
03214  3040          ROWADD
03215  2714          PACK
03216  3011          STOREAT
03217  0036          ROWNO
03220  0000          0
03221  3040  GO1,    ROWADD
03222  2656          UNPACK
03223  3011          STOREAT
03224  4257          S+77
03225  0040          40
03226  3011          STOREAT
03227  4260          S+100
03230  0000          0
03231  3011          STOREAT
03232  4261          S+101
03233  0040          40
03234  3011          STOREAT
03235  0034          TOIN
03236  0000          0
03237  1524          INTERP
03240  3325          INCRN
03241  3025          LOADRN
03242  0201          NR
03243  0010          10
03244  0651          EQ
03245  0154          JUMPF
03246  3221          GO1
03247  1610          QUIT
                                     / Blank a line
                                     / Argument is row addr
03250  1040  C1040,  1040
03251  4104  BLANK,  JMS     POP
03252  3030          DCA     REG10
03253  1006          TAD     M25     / Do 25 pairs
03254  3027          DCA     REG8
03255  1003  BLANK1, TAD     C40
03256  3430          DCA I   REG10
03257  2030          ISZ     REG10
03260  1250          TAD     C1040
```

```
03261  3430          DCA I   REG10
03262  2030          ISZ     REG10
03263  2027          ISZ     REG8
03264  5255          JMP     BLANK1
03265  5134          JMP     EXNV


03266  0077  C77,    77
03267  0300  C300,   300
03270  0000  PLAUX,  0
03271  0266          AND     C77
03272  3031          DCA     REG12
03273  1031          TAD     REG12
03274  1031          TAD     REG12
03275  7040          CMA             / Results in either 11 or 10
03276  0267          AND     C300    / Get the bits to add
03277  1031          TAD     REG12
03300  4121          JMS     SEROUT
03301  5670          JMP I   PLAUX

03302  4442  PLINE,  JMS I   POP2    / Chr count in R8, Addr in R10
03303  1027          TAD     REG8    / Chr count
03304  7100          CLL
03305  7010          RAR             / Divide by two
03306  7041          NEG
03307  3027          DCA     REG8    / (Chr count)/2
03310  3031          DCA     REG12   / Line count
03311  1430  PLINE1, TAD I   REG10   / Get first or next char
03312  7002          BSW             / Put left char at right
03313  4270          JMS     PLAUX   / Dig out char in the left part
03314  1430          TAD I   REG10   / Now do the left part
03315  4270          JMS     PLAUX
03316  2030          ISZ     REG10
03317  2031          ISZ     REG12   / Line count
03320  2027          ISZ     REG8
03321  5311          JMP     PLINE1
03322  5134          JMP     EXNV

03323  2037  INCCN,  ISZ     COLNO
03324  5134          JMP     EXNV

03325  2036  INCRN,  ISZ     ROWNO
03326  5134          JMP     EXNV

03327  5141  PIO,    JMP     COLON
03330  3205          LOADIO
03331  0171          JUMPNM          / Jump GE to OVR print
03332  3341          PIO1
03333  1416          DOTQ
03334  0003          3
03335  0311          "I;"N;"S
03336  0316
03337  0323
03340  0241          SEMIC
03341  1416  PIO1,   DOTQ
03342  0003          3
03343  0317          "O;"V;"R
03344  0326
03345  0322
03346  0241          SEMIC

03347  1004  EDITH,  1004
03350  0305          "E;"D;"I;"T+1000
03351  0304
03352  0311
03353  1324
03354  2763          LEAVEH
03355  5141  EDIT,   JMP     COLON
03356  3011          STOREAT
03357  0037          COLNO
03360  0000          0
```

```
03361  3011          STOREAT
03362  0036          ROWNO
03363  0000          0
03364  0201          NR
03365  7000          7000
03366  0260          AT
03367  0154          JUMPZ
03370  3407          EDIT0A
03371  0201  EDITI,  NR
03372  7000          7000
03373  2656          UNPACK
03374  3052  EDIT0,  PAINT
03375  3202          GETCHR  / Read the input
03376  0306          DUP
03377  1734          TWIXT
03400  0040          40      / Space
03401  0172          172     / z
03402  0161          JUMPT
03403  3754          EDIT10
03404  4026          CHECK
03405  0002          2
03406  3432          EDIT1
03407  0201  EDIT0A, NR
03410  7770          -10
03411  0201          NR
03412  7000          7000
03413  0306  EDIT0B, DUP
03414  3251          BLANK
03415  0201          NR
03416  0052          52      / 21 (dec) pairs
03417  0417          PLUS
03420  0273          SWAP
03421  0740          PLUS1
03422  0273          SWAP
03423  0323          OVER
03424  0161          JUMPNZ
03425  3413          EDIT0B
03426  0364          DROP
03427  0364          DROP
03430  0150          JUMP
03431  3371          EDITI

03432  4026  EDIT1,  CHECK
03433  0030          30
03434  3440          EDIT2
03435  3040          ROWADD
03436  2714          PACK
03437  1610          QUIT

03440  4026  EDIT2,  CHECK           / Check for a ^O
03441  0017          17
03442  3452          EDIT3
03443  3205          LOADIO
03444  2122          NEGATE
03445  0201          NR
03446  0040          INSOVR
03447  0251          STORE
03450  0150          JUMP
03451  3374          EDIT0

03452  4026  EDIT3,  CHECK           / Check for arrow key
03453  0033          33
03454  3572          EDIT4
03455  3202          GETCHR          / Get the left bracket
03456  0364          DROP            / Drop the right bracket
03457  3202          GETCHR          / This is the letter code: A-D
03460  1367          CR              / Make sure the [ gets rubbed out
03461  4026          CHECK
03462  0101          101
03463  3503          EDIT3B
03464  3025          LOADRN          / Make sure ROWNO is not 0
```

```
03465  0154          JUMPZ
03466  3501          EDIT3A          / ROWNO is 0, just leave
03467  3040          ROWADD
03470  2714          PACK
03471  3025          LOADRN
03472  2126          LOAD1
03473  0430          MINUS
03474  0201          NR
03475  0036          ROWNO
03476  0251          STORE
03477  3040          ROWADD
03500  2656          UNPACK
03501  0150  EDIT3A, JUMP
03502  3374          EDIT0

03503  4026  EDIT3B, CHECK           / Is it a B?
03504  0102          102
03505  3530          EDIT3C
03506  3025  EDIT3X, LOADRN          / Make sure ROWNO is not 7
03507  0201          NR
03510  0007          7
03511  0651          EQ
03512  0161          JUMPT
03513  3501          EDIT3A          / ROWNO is 7, just leave
03514  3040          ROWADD
03515  2714          PACK
03516  3025          LOADRN
03517  2126          LOAD1
03520  0417          PLUS
03521  0201          NR
03522  0036          ROWNO
03523  0251          STORE
03524  3040          ROWADD
03525  2656          UNPACK
03526  0150          JUMP
03527  3374          EDIT0

03530  4026  EDIT3C, CHECK           / Is it a C?
03531  0103          103
03532  3551          EDIT3D
03533  3030          LOADCN          / Make sure COLNO is not 63 (dec)
03534  0201          NR
03535  0076          76
03536  0651          EQ
03537  0161          JUMPT
03540  3501          EDIT3A          / COLNO is 37, just leave
03541  2126          LOAD1
03542  3030  EDIT3Y, LOADCN
03543  0417          PLUS
03544  0201          NR
03545  0037          COLNO
03546  0251          STORE
03547  0150          JUMP
03550  3374          EDIT0

03551  4026  EDIT3D, CHECK           / Is it a D?
03552  0104          104
03553  3567          EDIT3E
03554  3030          LOADCN
03555  0154          JUMPZ           / Make sure COLNO is not 0
03556  3501          EDIT3A          / COLNO is 0, just leave
03557  3030  EDIT3Z, LOADCN
03560  2126          LOAD1
03561  0430          MINUS
03562  0201          NR
03563  0037          COLNO
03564  0251          STORE
03565  0150          JUMP
03566  3374          EDIT0
03567  0364  EDIT3E, DROP
03570  0150          JUMP
```

```
03571  3501          EDIT3A

03572  4026  EDIT4,  CHECK           / Is it a CR?
03573  0015          15
03574  3603          EDIT5
03575  2133          LOAD0
03576  0201          NR
03577  0037          COLNO
03600  0251          STORE
03601  0150          JUMP
03602  3506          EDIT3X

03603  4026  EDIT5,  CHECK           / Is it a BS?
03604  0177          177
03605  3637          EDIT6
03606  3030          LOADCN          / If COLNO Zero, just exit
03607  0154          JUMPZ
03610  3374          EDIT0
03611  3030          LOADCN
03612  1644          LOADAT
03613  0050          SI
03614  0417          PLUS
03615  0306          DUP
03616  2126          LOAD1
03617  0430          MINUS
03620  0201          NR
03621  0077          77
03622  3030          LOADCN
03623  0430          MINUS
03624  0753          CMOVE
03625  1644          LOADAT
03626  0003          C40
03627  0201          NR
03630  0076          76
03631  1644          LOADAT
03632  0050          SI
03633  0417          PLUS
03634  0251          STORE
03635  0150          JUMP
03636  3557          EDIT3Z

03637  4026  EDIT6,  CHECK           / Is it a ^N?
03640  0016          16
03641  3675          EDIT7
03642  0201          NR              / if row no is 7 then no move
03643  0007          7
03644  3025          LOADRN
03645  0651          EQ
03646  0161          JUMPT
03647  3663          EDIT6A
03650  3040          ROWADD
03651  2714          PACK
03652  0201          NR
03653  7445          7445            / Source
03654  0201          NR
03655  7517          7517            / Target
03656  0201          NR
03657  7446          7446
03660  3040          ROWADD          / Get row addr
03661  0430          MINUS
03662  2135          BCMOVE
03663  3040  EDIT6A, ROWADD          / Set new row to blanks
03664  3251          BLANK
03665  3040          ROWADD
03666  2656          UNPACK
03667  2133  EDIT6B, LOAD0
03670  0201          NR
03671  0037          COLNO
03672  0251          STORE
03673  0150          JUMP
03674  3374          EDIT0
```

```

03675  4026  EDIT7,  CHECK           / Is it a ^Y?
03676  0031          31
03677  3732          EDIT8
03700  0201          NR              / if row no is 7 then no move
03701  0007          7
03702  3025          LOADRN
03703  0651          EQ
03704  0161          JUMPT
03705  3723          EDIT7A
03706  3040          ROWADD          / Put the text back in packed buffer
03707  2714          PACK
03710  3040          ROWADD          / Target
03711  0306          DUP
03712  0201          NR
03713  0052          52
03714  0417          PLUS            / Source
03715  0273          SWAP
03716  0201          NR
03717  7446          7446
03720  3040          ROWADD          / Get row addr
03721  0430          MINUS
03722  0753          CMOVE
03723  0201  EDIT7A, NR
03724  7446          7446
03725  3251          BLANK
03726  3040          ROWADD
03727  2656          UNPACK
03730  0150          JUMP
03731  3667          EDIT6B

03732  4026  EDIT8,  CHECK           / Is it a TAB?
03733  0011          11
03734  3750          EDIT9
03735  0201          NR
03736  0070          70
03737  3030          LOADCN          / Check if gt 56
03740  0273          SWAP
03741  0632          GE
03742  0161          JUMPT
03743  3374          EDIT0
03744  0201          NR
03745  0006          6
03746  0150          JUMP
03747  3542          EDIT3Y

03750  4026  EDIT9,  CHECK
03751  0007          07
03752  3567          EDIT3E
03753  3212          GO              / Execute the screen

                                     / Process an ordinary char
03754  3205  EDIT10, LOADIO          / Load Ins/Ovr flag
03755  0171          JUMPNM          / -1 if insert
03756  4004          EDIT11
03757  3030          LOADCN          / Are we at end of line?
03760  0201          NR
03761  0077          77
03762  0651          EQ
03763  0161          JUMPT
03764  4004          EDIT11
03765  1644          LOADAT
03766  0050          SI
03767  0201          NR              / Target
03770  0077          77
03771  0417          PLUS
03772  0306          DUP             / Source
03773  0201          NR
03774  0001          1
03775  0430          MINUS
03776  0273          SWAP
```

```
03777  0201          NR
04000  0077          77
04001  3030          LOADCN
04002  0430          MINUS
04003  2135          BCMOVE
04004  1644  EDIT11, LOADAT          / Place char at col in text buffer
04005  0050          SI
04006  3030          LOADCN
04007  0417          PLUS
04010  0251          STORE
04011  3030          LOADCN          / Check for end of line
04012  0201          NR
04013  0076          76
04014  0651          EQ
04015  0161          JUMPT           / True at end of line
04016  4020          EDIT12
04017  3323          INCCN
04020  0150  EDIT12, JUMP
04021  3374          EDIT0

04022  3011  START,  STOREAT
04023  7000          7000
04024  0000          0
04025  1610          QUIT

04026  1420  CHECK,  TAD I   IP
04027  7041          NEG
04030  1421          TAD I   SP
04031  7450          SNA             / Exit if equal
04032  5237          JMP     CHECK1
04033  4055          JMS     INCIP   / Not equal, get jump addr
04034  1420          TAD I   IP
04035  3020          DCA     IP
04036  5134          JMP     EXNV
04037  4055  CHECK1, JMS     INCIP   / Skip over the test char
04040  4055          JMS     INCIP   / Skip over the jump addr
04041  4104          JMS     POP     / Prune the stack
04042  5134          JMP     EXNV

                                     / Shift right n unsigned double
04043  1003  SRNH,   1003
04044  0323          "S;"R;"N+1000
04045  0322
04046  1316
04047  3347          EDITH
04050  4444  SRN,    JMS I   POP3    / TOS in REG12, double in R8/R10
04051  1031          TAD     REG12
04052  7041          NEG
04053  3031          DCA     REG12
04054  7100  SRN1,   CLL
04055  1030          TAD     REG10
04056  7010          RAR
04057  3030          DCA     REG10
04060  1027          TAD     REG8
04061  7010          RAR
04062  3027          DCA     REG8
04063  2031          ISZ     REG12
04064  5254          JMP     SRN1
04065  4443          JMS I   PUSH2
04066  5134          JMP     EXNV

04067  1004  PICKH,  1004
04070  0320          "P;"I;"C;"K+1000
04071  0311
04072  0303
04073  1313
04074  4043          SRNH
04075  4104  PICK,   JMS     POP
04076  1021          TAD     SP
04077  3027          DCA     REG8
04100  1427          TAD I   REG8
```

```
04101  4113          JMS     PUSH
04102  5134          JMP     EXNV

04103  1005  CARRYH, 1005
04104  0303          "C;"A;"R;"R;"Y+1000
04105  0301
04106  0322
04107  0322
04110  1331
04111  4067          PICKH
04112  7430  CARRY,  SZL             / Link (carry) to stack
04113  7201          ONE
04114  4113          JMS     PUSH
04115  5134          JMP     EXNV


04116  7777          7777
04117  7777          7777

       4160  S=  .+40                / 32 words of stack should be enough
       4300  RS= S+120               / Input and text buffer is at bottom of RS
       4300  DICTIONARY= RS

             $

No detected errors
No links generated
```