```
00001                           NAM     DEBUG:
00002                           OPT     ABS,LLE=100

00004                   *       6809 INTERACTIVE DEBUG PROGRAM
00005                   * FOR CREATIVE MICRO SYSTEMS 9609 MODULE

00007                   * (C) 1980 MICROWARE SYSTEMS CORPORATION
00008                   *       DES MOINES, IOWA
00009                   *       ALL RIGHTS RESERVED



00011                   * HARDWARE REQUIREMENTS:
00012                   *
00013                   *   THIS ROM AT $F800 - $FFFF
00014                   *   RAM AT $E700 - $E7FF
00015                   *   ACIA AT $E3C0
00016                   *   PTM AT $E3E0  (ENABLED TO NMI)



00018                           TTL     6809 UTILITY SUBROUTINE PACKAGE


00021                   * SYMBOLIC DEFINITIONS

00023                   * ASCII SPECIAL CHARACTERS
00024           0020  A SPACE  EQU   $20
00025           000D  A EOL    EQU   $0D       (CARRIAGE RETURN)
00026           002C  A COMMA  EQU   $2C


00028                   * CONDITION CODE REGISTER VALUES

00030           0001  A CBIT   EQU   1
00031           0002  A VBIT   EQU   2
00032           0004  A ZBIT   EQU   4
00033           00FE  A NOTC   EQU   $FF-CBIT
00034           00FD  A NOTV   EQU   $FF-VBIT
00035           00FB  A NOTZ   EQU   $FF-ZBIT
00036A F800                    ORG   $F800
00038                   * Input Conversion Subroutines

00040                   * These subroutines convert an unsigned binary
00041                   * number (passed in B or D) to an ASCII character
00042                   * string starting at the address passed in the
    3                   * X register, which will return with the address
                        * following the string.  All other registers
00045                   * except CC are preserved.



00047                   * Subroutine BIN4HS - Convert word in D reg to
00048                   * four-character hexadecimal followed by a space.

00050A F800 8D   0C     F80E BIN4HS BSR     BIN4HX   PERFORM CONVERSION
00051A F802 20   02     F806        BRA     PUTSPC   GO OUTPUT SPACE AND RTS


00053                   * Subroutine BIN2HS - Convert byte in B reg to
00054                   * two-character hexadecimal followed by a space.

00056A F804 8D   0I     F814 BIN2HS BSR     BIN2HX   PERFORM CONVERSION
00057                   * FALL THROUGH TO PUTSPC



00059                   * Subroutine PUTSPC - Put a space character in
00060                   * the output buffer and increment pointer (X)

00062A F806 34   02       A PUTSPC PSHS    A
00063A F808 86   20       A        LDAA    #SPACE
00064A F80A A7   80       A        STA     ,X+
00065A F80C 35   82       A        PULS    A,PC



00067                   * Subroutine BIN4HX - convert word in D reg to
00068                   * four-character hexadecimal.

00070A F80E 1E   89       A BIN4HX EXG     A,B      SWAP BYTES
00071A F810 8D   02     F814       BSR     BIN2HX   CONVERT HI ORDER BYTE
00072A F812 1F   89       A        TFR     A,B      MOVE LO ORDER BYTE TO B
00073                   * FALL THROUGH TO CONVERT LOW BYTE



00075                   * subroutine BIN2HX - convert byte in B reg to
   76                   * two-character hexadecimal.

00078A F814 34   04       A BIN2HX PSHS    B        SAVE BYTE
00079A F816 C4   F0       A        ANDB    #$F0     MASK HI NYBBLE
00080A F818 54                    LSRB             SHIFT IT TO RIGHT
00081A F819 54                    LSRB
00082A F81A 54                    LSRB
00083A F81B 54                    LSRB
00084A F81C 8D   04     F822      BSR     HEXCHR   THEN CONVERT IT
00085A F81E 35   04       A       PULS    B        RESTORE BYTE
00086A F820 C4   0F       A       ANDB    #$0F     MASK LOW BYTE
00088A F822 C1   09       A HEXCHR CMPB   #9       RANGE 0-9?
00089A F824 23   02     F828       BLS     HXCHR2   IF SO SKIP CORRECTION
00090A F826 CB   07       A       ADDB    #7       ADJUST FOR A-F
00091A F828 CB   30       A HXCHR2 ADDB    #$30     MAKE IT ASCII
00092A F82A E7   80       A       STB     ,X+      PUT IN BUFFER
00093A F82C 39                    RTS


00095                   * subroutine BINDEC - convert word in d REG TO
00096                   * five-character decimal.

00098A F82D 34   64       A BINDEC PSHS    B,Y,U    SAVE LOCAL REGISTERS
00099A F82F 33   8D 0020          LEAU    TNSTBL,PCR GET CONSTANTS TABLE ADDR
00100A F833 108E 0005     A       LDY     #5       Y IS LOOP COUNTER
00101                   * DIGIT CONVERSION LOOP
00102A F837 6F   E4       A BINDC2 CLR     ,S       CLEAR DIGIT TEMP
00103A F839 A3   C4       A BINDC3 SUBD    ,U       SUBTRACT POWER-OF-TEN
00104A F83B 25   04     F841       BCS     BINDC4   EXIT IF UNDERFLOW
00105A F83D 6C   E4       A       INC     ,S       ELSE BUMP COUNT,
00106A F83F 20   F8     F839      BRA     BINDC3   AND DO IT AGAIN
00107A F841 E3   C1       A BINDC4 ADDL    ,U++     RESTORE AND BUMP PTR
00108A F843 34   04       A       PSHS    B        SAVE LOW BYTE
00109A F845 E6   61       A       LDB     1,S      GET DIGIT COUNTER
00110A F847 CB   30       A       ADDB    #$30     FORM ASCII CHAR
00111A F849 E7   80       A       STB     ,X+      STORE IT
00112A F84B 35   04       A       PULS    B        RESTORE LO BYTE
00113A F84D 31   3F       A       LEAY    -1,Y     DECR LOOP COUNT
00114A F84F 26   E6     F837      BNE     BINDC2   LOOP IF MORE TO DO
00115A F851 35   E4       A       PULS    B,Y,U,PC POP REGS + RTS

00117                   * Power-of-Tens conversion table
00118A F853      2710   A TNSTBL FDB    10000
00119A F855      03E8   A        FDB    1000
00120A F857      0064   A        FDB    100
00121A F859      000A   A        FDB    10
00122A F85B      0001   A        FDB    1
00124                   * Input Conversion Subroutines

00126                   * These subroutines convert ASCII character strings
00127                   * which represent decimal, hexadecimal or binary
00128                   * notation to binary numbers.  The input strings
00129                   * can be of any legal length and terminate when
00130                   * an illegal character for the specific radix
00131                   * being processed is encountered.

00133                   * Upon entry, the X register is assumed to point to
00134                   * the present position in the buffer to be scanned.
00135                   * The routine will return the converted value in the
00136                   * D register, status in the CC register, and the
00137                   * updated pointer in the X register.

00139                   * If the conversion was performed correctly, the C
00140                   * bit of CC reg will be cleared, otherwise an error
00141                   * occured: if a legal string was not found the Z bit
00142                   * will be set.  If the Z bit returned clear (and the
00143                   * C bit was set) a legal number was encountered but
00144                   * a conversion overflow occurred.



00146                   * Subroutine NUMBIN - Parse Radix Specification
00147                   * Character and Perform Conversion to Binary

00149                   * Skip leading spaces, then look for radix sign:
00150                   *   # for decimal
00151                   *   % for binary
00152                   *   $ for hexadecimal
00153                   * The default if no specification is given is hex.


00155                   * Definition of stack local variables
00156           0000  A DIGTMP EQU   0       DIGIT TEMPT
00157           0001  A DIGCNT EQU   1       DIGIT COUNTER
00158           0002  A NUMTMP EQU   2       CONVERSION TEMP (2 BYTES)


00160A F85D 17   00A9 F909 NUMBIN LBSR    SKIPSP   SKIP LEADING SPACES
00161A F860 30   01       A        LEAX    1,X      BUMP POINTER
00162A F862 81   23       A        CMPA    #'#      DECIMAL?
00163A F864 27   36     F89C       BEQ     DECBIN   IF SO GO CONVERT IT
00164A F866 81   25       A        CMPA    #'%      BINARY?
00165A F868 27   5A     F8C2       BEQ     ASCBIN   IF SO GO CONVERT
00166A F86A 81   24       A        CMPA    #'$      HEX?
00167A F86C 27   02     F870       BEQ     HEXBIN   IF SO GO CONVERT
00168A F86E 30   1F       A        LEAX    -1,X     BACKUP POINTER
00169                   * ASSUME DEFAULT, FALL THROUGH TO HEXBIN


00171                   * Subroutine HEXBIN - convert hexadecimal
00172                   * ASCII string to binary

00174A F870 32   7C       A HEXBIN LEAS    -4,S     RESERVE TEMPS
00175A F872 8D   65     F8D9       BSR     NSETUP   CALL TEMP INITIALIZATION
00176A F874 8D   6A     F8E0 HEXBN2 BSR     DIGIT    DECIMAL DIGIT?
00177A F876 24   0A     F882       BCC     HEXBN3   IF SO GO PROCESS IT
00178A F878 C1   41       A        CMPB    #'A      ELSE CHECK FOR A - F
00179A F87A 25   76     F8F2       BLO     NUMEND   TERMINATE CONV IF < A
00180A F87C C1   46       A        CMPB    #'F
00181A F87E 22   72     F8F2       BHI     NUMEND   TERMINATE IF > F
00182A F880 C0   37       A        SUBB    #$37     ADJUST TO BINARY
00183                   * ADD THIS DIGIT TO WORKING SUM
00184A F882 E7   E4       A HEXBN3 STB     DIGTMP,S SAVE THIS DIGIT
00185A F884 EC   62       A        LDD     NUMTMP,S GET CURRENT SUM
00186A F886 65   F0       A        BITA    #$F0     WILL MULTIPLY OVERFLOW?
00187A F888 26   7B     F905       BNE     NUMOVR   ERROR EXIT IF IT WILL
00188                   * MULTIPLY WORKING SUM BY 16
00189A F88A 58                     ASLB
00190A F88B 49                     ROLA
00191A F88C 58                     ASLB             *
00192A F88D 49                     ROLA             * MULTIPLY BY 16
00193A F88E 58                     ASLB
00194A F88F 49                     ROLA
00195A F890 58                     ASLB
00196A F891 49                     ROLA
00197A F892 EB   E4       A        ADDB    DIGTMP,S ADD NEW DIGIT TO SUM
00198A F894 89   00       A        ADCA    #0
00199A F896 ED   62       A        STD     NUMTMP,S SAVE RESULT
00200A F898 6C   61       A        INC     DIGCNT,S BUMP DIGIT COUNT
00201A F89A 20   D8     F874       BRA     HEXBN2   GO TRY ANOTHER DIGIT


00203                   * Subroutine DECBIN - convert decimal ASCII
00204                   * character string to binary

00206A F89C 32   7C       A DECBIN LEAS    -4,S     RESERVE LOCAL STORAGE
00207A F89E 8D   39     F8D9       BSR     NSETUP   AND INIT TEMP
00208A F8A0 8D   3F     F8E2 DECBN2 BSR     DIGIT    IS IT A DIGIT?
00209A F8A2 25   4E     F8F2       BCS     NUMEND   EXIT CONVERSION IF NOT
00210A F8A4 E7   E4       A        STB     DIGTMP,S SAVE NEW DIGIT
00211A F8A6 EC   62       A        LDD     NUMTMP,S
00212                   * MULTIPLY WORKING SUM BY 10
00213A F8A8 58                     ASLB
00214A F8A9 49                     ROLA             S=S*2
00215A F8AA ED   62       A        STD     NUMTMP,S SAVE S*2
00216A F8AC 58                     ASLB
00217A F8AD 49                     ROLA             NOW S*4
00218A F8AE 58                     ASLB
00219A F8AF 49                     ROLA             NOW S*8
00220A F8B0 25   53     F905       BCS     NUMOVR   EXIT IF OVERFLOW
00221A F8B2 E3   62       A        ADDD    NUMTMP,S (S*8)+(S*2)=S*10
00222A F8B4 25   4F     F905       BCS     NUMOVR   MAY ALSO OVERFLOW HERE
00223A F8B6 EB   E4       A        ADDB    DIGTMP,S ADD NEW DIGIT
00224A F8B8 89   00       A        ADCA    #0       PROPOGATE CARRY
00225A F8BA 25   49     F905       BCS     NUMOVR   LAST OVERFLOW TEST
00226A F8BC ED   62       A        STD     NUMTMP,S SAVE NEW SUM
00227A F8BE 6C   61       A        INC     DIGCNT,S BUMP DIGIT COUNT
00228A F8C0 20   DE     F8A0       BRA     DECBN2   GO TRY ANOTHER CHAR


00230                   * Subroutine ASCBIN - convert ASCII string
00231                   * in binary notation to binary number

00233A F8C2 32   7C       A ASCBIN LEAS    -4,S     RESERVE LOCAL STORAGE
00234A F8C4 8D   13     F8D9       BSR     NSETUP   AND INITIALIZE
00235A F8C6 E6   80       A ASCBN2 LDB     ,X+      GET CURRENT CHAR, BUMP X
00236A F8C8 C0   30       A        SUBB    #'0      ZERO CHAR?
00237A F8CA 25   26     F8F2       BLO     NUMEND   EXIT CONVERSION IF LOWER
00238A F8CC 54                     LSRB             SHIFT TO C BIT
00239A F8CD 26   23     F8F2       BNE     NUMEND   IF <>0 IT WASN'T A ONE
```

```
00240A F8CF 69   63        A         ROL     NUMTMP+1,S MULT SUM*2, SHIFT NEW BIT IN
00241A F8D1 69   62        A         ROL     NUMTMP,S
00242A F8D3 25   30     F905         BCS     NUMOVR    EXIT IF OVERFLOW
00243A F8D5 6C   61        A         INC     DIGCNT,S INCR DIGIT COUNT
00244A F8D7 20   ED     F8C6         BRA     ASC2N2    GO DO MORE

00246                    * Input Conversion Auxiliary Subroutines

00248                    * Subroutine NSETUP - set up locals on stack
00249A F8D9 4F           NSETUP CLRA
00250A F8DA 5F                  CLRB
00251A F8DB ED   62        A         STD     2,S
00252A F8DD ED   64        A         STD     4,S
00253A F8DF 39                  RTS

00255                    * Subroutine DIGIT - check for decimal character
00256                    * 0 - 9 and convert to binary

00258A F8E0 E6   80        A DIGIT  LDB     ,X+       GET NEXT CHAR, INCR X
00259A F8E2 C1   30        A        CMPB    #'0       TEST LOWER BOUND
00260A F8E4 25   04     F8EA        BLO     RETNO     EXIT IF NOT DIGIT
00261A F8E6 C1   39        A        CMPB    #'9       TEST HIGH BOUND
00262A F8E8 23   03     F8ED        BLS     DIGIT2    PROCEED IF OK
00263                    * RETURN ERROR STATUS
00264A F8EA 1A   01        A RETNO  ORCC    #CBIT     SET C BIT
00265A F8EC 39                  RTS
00266A F8ED C0   30        A DIGIT2 SUBB    #$30      MAKE BINARY
00267                    * RETURN NOT ERROR STATUS
00268A F8EF 1C   FE        A RETYES ANDCC   #NOTC     CLEAR C BIT
00269A F8F1 39                  RTS

00271                    * Subroutine NUMEND - conversion cleanup
00272                    * and exit for all main routines

00274A F8F2 30   1F        A NUMEND LEAX    -1,X      BACKUP INPUT POINTER
00275A F8F4 6D   61        A        TST     DIGCNT,S DID WE FIND DIGITS?
00276A F8F6 27   06     F8FE        BEQ     NOTFND    ERROR IF NOT
00277A F8F8 EC   62        A        LDD     NUMTMP,S
00278A F8FA 1C   FE        A        ANDCC   #NOTC     CLEAR C BIT
00279A F8FC 20   04     F902        BRA     GOBACK    GO EXIT
00280A F8FE 1A   04        A NOTFND ORCC    #ZBIT     SET C BIT
00281                    * RETURN ERROR IN NUMBER
00282A F900 1A   01        A NUMERR ORCC    #CBIT     SET C BIT

00284                    * CLEAN UP STACK AND RETURN TO CALLER
00285A F902 32   64        A GOBACK LEAS    4,S
00286A F904 39                  RTS
00287                    * RETURN OVERFLOW ERROR
00288A F905 1C   FB        A NUMOVR ANDCC   #NOTZ     CLEAR Z BIT
00289A F907 20   F7     F900        BRA     NUMERR

00291                    * Subroutine SKIPSP - skip leading spaces

00293A F909 A6   80        A SKIPSP LDA     ,X+
00294A F90B 81   20        A        CMPA    #SPACE
00295A F90D 27   FA     F909        BEQ     SKIPSP
00296A F90F 30   1F        A        LEAX    -1,X      BACKUP POINTER
00297A F911 39                  RTS
00299                    * Arithmetic subroutines

00301                    * Subroutine MULT - 16 bit unsigned multiply.
00302                    * Contents of D and X registers are multiplied
00303                    * and a 16 bit product is returned in D.  If
00304                    * an overflow occurs the C bit is set and the
00305                    * high order 16 bits are returned in X (which
00306                    * is cleared otherwise).

00308A F912 34   16        A MULT   PSHS    D,X       SAVE OPERANDS
00309A F914 A6   63        A        LDA     3,S       GET XL
00310A F916 3D                  MUL
00311A F917 34   06        A        PSHS    D         D=LSW OF PRODUCT
00312A F919 A6   62        A        LDA     2,S       GET DH
00313A F91B E6   64        A        LDB     4,S       GET XH
00314A F91D 3D                  MUL
00315A F91E 34   06        A        PSHS    D         D=MSW OF PRODUCT
00316                    * NOW COMPUTE MIDDLE PARTIAL PRODUCTS
00317A F920 A6   64        A        LDA     4,S       GET DH
00318A F922 E6   67        A        LDB     7,S       GET XL
00319A F924 8D   13     F939        BSR     MULSUB    MULT AND ADD TO PRODUCT
00320A F926 A6   65        A        LDA     5,S       GET DL
00321A F928 E6   66        A        LDB     6,S       GET XH
00322A F92A 8D   0D     F939        BSR     MULSUB    MULT AND ADD TO PRODUCT
00323                    * CLEAN UP AND RETURN
00324A F92C 1C   FE        A        ANDCC   #NOTC     ASSUME NOT ERROR
00325A F92E EC   62        A        LDD     2,S       GET LOW WORD OF PRODUCT
00326A F930 A4   A4        A        LDX     ,S        GET HI WORD OF PRODUCT
00327A F932 27   02     F936        BEQ     MULT2     SKIP IF NO OVERFLOW
00328A F934 1A   01        A        ORCC    #CBIT     RETURN ERROR STATUS
00329A F936 32   68        A MULT2  LEAS    8,S       CLEAN UP STACK
00330A F938 39                  RTS

00332                    * Partial product multiply/add
00333A F939 3D           MULSUB MUL
00334A F93A E3   63        A        ADDD    3,S       ADD TO MIDDLE BYTES OF PRODUCT
00335A F93C ED   63        A        STD     3,S       STORE RESULT
00336A F93E 24   02     F942        BCC     MULSB2    SKIP IF NO CARRY
00337A F940 6C   62        A        INC     2,S       ELSE BUMP NEXT BYTE
00338A F942 39           MULSB2 RTS

00340                    * Subroutine DIVIDE - 16 bit unsigned divide
00341                    * Divide contents of X register by D register.
00342                    * Return quotient in D register and remainder
00343                    * in X register.  Returns C bit set if a divide
00344                    * by zero was attempted.

00346A F943 34   36        A DIVIDE PSHS    D,X,Y     PUSH OPERANDS + Y FOR TEMP
00347A F945 EC   E4        A        LDD     ,S        GET DIVISOR
00348A F947 26   04     F94D        BNE     DIV20     SKIP IF NONZERO
00349A F949 1A   01        A        ORCC    #CBIT     RETURN ERROR
00350A F94B 20   20     F96D        BRA     DIV60
00351A F94D CC   0010      A DIV20  LDD     #16       SHIFT COUNT=16
00352A F950 A7   64        A        STB     4,S       SAVE IT
00353A F952 5F                  CLRB                  CLEAR D
00354A F953 68   63        A DIV40  ASL     3,S       SHIFT MSB OF DIVIDEND
00355A F955 69   62        A        ROL     2,S       INTO D
00356A F957 59                  ROLB
00357A F958 49                  ROLA
00358A F959 A3   E4        A        SUBD    ,S        SUBTRACT DIVISOR
00359A F95B 2F   04     F961        BMI     DIV45     EXIT IF UNDERFLOW
00360A F95D 6C   63        A        INC     3,S       ELSE SET QUOTIENT LSB
00361A F95F 20   02     F963        BRA     DIV50     AND PROCEED
00362A F961 E3   E4        A DIV45  ADDD    ,S        RESTORE DIVIDEND
00363A F963 6A   64        A DIV50  DEC     4,S       DECR SHIFT COUNT
00364A F965 26   EC     F953        BNE     DIV40     DO ANOTHER BIT IF NOT=0
00365A F967 1F   01        A        TFR     D,X       COPY REMAINDER TO X
00366A F969 EC   62        A        LDD     2,S       GET QUOTIENT
00367A F96B 1C   FE        A        ANDCC   #NOTC     SET NO ERR STATUS
00368A F96D 32   66        A DIV60  LEAS    6,S       CLEAN UP STACK
00369A F96F 39                  RTS                   ...DONE
00371                    * Move and Compare Subroutines
00372                    *
00373                    * In general, the Y register is the "from" address
00374                    * X is the "to" address.  Subroutines using variabl
00375                    * length string terminate on a "zero byte" - a byte
00376                    * having a value of zero.  This byte is NOT
00377                    * included in the compare.

00379                    * Subroutine STRMOV - move a variable length
00380                    * string terminated by a zero byte from (Y)
00381                    * to (X)

00383                    * >>> warning - enter at STRMOV ! <<<
00384A F970 A7   80        A SMLOOP STA     ,X+       STORE BYTE
00385A F972 A6   A0        A STRMOV LDA     ,Y+       GET NEXT BYTE
00386A F974 26   FA     F970        BNE     SMLOOP    STORE IT NOT STR END
00387A F976 39                  RTS                   ELSE DONE

00389                    * Subroutine BLKMOV - move a block of memory
00390                    * whose size is given in D from (Y) to (X)

00392A F977 34   46        A BLKMOV PSHS    D,U       SAVE U REGISTERS
00393A F979 1F   13        A        TFR     X,U       COPY DESTINATION ADDRESS
00394A F97B AE   E4        A        LDX     ,S        GET BYTE COUNT IN X
00395A F97D 27   08     F987        BEQ     BLKMV3    EXIT IF ZERO
00396A F97F A6   A0        A BLKMV2 LDA     ,Y+       FETCH NEXT BYTE
00397A F981 A7   C0        A        STA     ,U+       STORE IT
00398A F983 30   1F        A        LEAX    -1,X      DECR BYTE COUNT
00399A F985 26   F8     F97F        BNE     BLKMV2    LOOP TIL ALL MOVED
00400A F987 1F   31        A BLKMV3 TFR     U,X       MOVE DEST BACK TO X
00401A F989 35   C6        A        PULS    D,U,PC    THATS IT



00404                              TTL     EXPRESSION INTERPRETER

00407                    *
00408                    * These routines interpret expressions using
00409                    * a top-down recursive descent parser.
00410                    * All arithmetic is unsigned binary.
00411                    *
00412                    * Operators and Operands - in precedence
00413                    * order:
00414                    *
00415                    *  + -
00416                    *  * /
00417                    *  & (and)  ! (or)
00418                    *  ^ (not)  - (negate)
00419                    *  (expr)  [expr]  <expr>
00420                    *  constants:
00421                    *     nex
00422                    *     $aex
00423                    *     #decimal
00424                    *     %binary
00425                    *     'char
00426                    *     "double char
00427                    *     :register
00428                    *     .  Dot
00429                    *

00431            F98B    A SAVORG SET   *

00433                    * Error code definitions
00434                    *
00435A 0000                       ORG     0
00436A 0000      0001    A E$ILNM RMB     1         ILLEGAL CONSTANT
00437A 0001      0001    A E$DVZR RMB     1         DIVIDE BY ZERO
00438A 0002      0001    A E$MLOV RMB     1         MULTIPLY OVERFLOW
00439A 0003      0001    A E$OPMS RMB     1         OPERAND MISSING
00440A 0004      0001    A E$PRMS RMB     1         RIGHT PAREN MISSING
00441A 0005      0001    A E$BRMS RMB     1         RIGHT BRACKET MISSING
00442A 0006      0001    A E$GTMS RMB     1         GREATER THAN MISSING
00443A 0007      0001    A E$RGNM RMB     1         ILLEGAL REGISTER NAME
00444            0008    A ERRORG SET     *

00447A F98B                       ORG     SAVORG
00448                    * Main entry point
00449                    *
00450                    * Enter with X pointing to input string
00451                    *
00452                    * Exits with result in D, X updated
00453                    * past expression.  If error occured,
00454                    * C bit is set and B will contain
00455                    * error code.

00457A F98B 34   60        A EVAL   PSHS    U,Y
00458A F98D 1F   43        A        TFR     S,U       COPY STACK PTR
00459A F98F 8D   0C     F99D        BSR     EXPR      GO EVALUATE
00460A F991 1C   FE        A        ANDCC   #NOTC
00461A F993 35   E0        A        PULS    Y,U,PC    POP REGS +RTS

00463                    * Error exit - restore sp and
00464                    * return
00465A F995 1F   34        A EXPERR TFR     U,S       RESTORE SP
00466A F997 1A   01        A        ORCC    #CBIT     SET ERROR
00467A F999 35   E0        A        PULS    U,Y,PC

00469A F99B 30   01        A XEXPR  LEAX    1,X

00471A F99D 8D   20     F9BF EXPR   BSR     TERM      PROCESS LEFT NODE
00472A F99F 34   06        A        PSHS    D         SAVE RESULT
00473A F9A1 8D   70     FA13 EXPR2  BSR     SKIP      SKIP SPACES
00474A F9A3 81   2D        A        CMPA    #'-       SUBTRACT OPERATOR?
00475A F9A5 26   08     F9AF        BNE     EXPR3     SKIP IF NOT
00476A F9A7 8D   14     F9BD        BSR     XTERM     GET RIGHT OPERAND
00477A F9A9 40                     NEGA                COMPLIMENT RESULT
00478A F9AA 50                     NEGB
00479A F9AB 82   00        A        SBCA    #0
00480A F9AD 20   06     F9B5        BRA     EXPR4     GO FINISH UP
00481A F9AF 81   2B        A EXPR3  CMPA    #'+       ADD OPERATOR?
00482A F9B1 26   08     F9BB        BNE     EXPR5     SKIP IF NOT
00483A F9B3 8D   08     F9BD        BSR     XTERM     GET RIGHT OPERAND
00484A F9B5 E3   E4        A EXPR4  ADDD    ,S        ADD TO LEFT OPERAND
00485A F9B7 ED   E4        A        STD     ,S        REPLACE TOS
00486A F9B9 20   E6     F9A1        BRA     EXPR2     GO LOOK FOR MORE
```

```
00487A F9BB 35   86      A EXIT5  PULS   D,PC

00489                             * Subroutine TERM - process multiply and
00490                             * divide precedence level

00492A F9BD 30   E1      A XTERM  LEAX   1,X       INCR X ENTRY POINT

00494A F9BF 8D   32      F9F3 TERM   BSR    FACT     GET LEFT NODE
00495A F9C1 34   06        A         PSHS   D        SAVE RESULT
00496A F9C3 8D   4E      FA13 TERM2  BSR    SKIP
00497A F9C5 81   2A        A         CMPA   #'*      MULTIPLY OPERATOR?
00498A F9C7 26   0F      F9D8        BNE    TERM3    SKIP IF NOT
00499A F9C9 8D   26      F9F1        BSR    XFACT    GET RIGHT NODE
00500A F9CB 34   10        A         PSHS   X        SAVE TEXT PTR
00501A F9CD AE   62        A         LDX    2,S      GET LEFT OPERAND VALUE
00502A F9CF (17) FF40 F912           LBSR   MULT     CALL MULTIPLY
00503A F9D2 24   17      F9EB        BCC    TERM4    GO FINISH IF NO ERROR
00504A F9D4 C6   02        A         LDB    #E$MLOV  MULT OVERFLOW
00505A F9D6 20   BD      F995        BRA    IXPERR
00506A F9D8 81   2F        A TERM3   CMPA   #'/      DIVIDE OPERATOR?
00507A F9DA 26   DF      F9BB        BNE    EXPR5    EXIT IF NOT
00508A F9DC 8D   13      F9F1        BSR    XFACT    GET RIGHT NODE
00509A F9DE 34   10        A         PSHS   X        SAVE TEXT PTR
00510A F9E0 AE   62        A         LDX    2,X      GET LEFT OPRAND RESULT
00511A F9E2 17   FF5E F942           LBSR   DIVIDE   CALL DIVIDE
00512A F9E5 24   04      F9EB        BCC    TERM4    CONT IF NO ERROR
00513A F9E7 C6   01        A         LDB    #E$DVZR  DIVIDE BY 0 ERROR
00514A F9E9 20   AA      F995        BRA    IXPERR
00515A F9EB 35   10        A TERM4   PULS   X        RESTORE TEXT PTR
00516A F9ED ED   F4        A         STD    ,S       PUT RESULT ON STACK
00517A F9EF 20   D2      F9C3        BRA    TERM2    GO LOOK FOR MORE

00519                             * Subroutine FACT - process logical
00520                             * operator precedence level.

00522A F9F1 30   01      A XFACT  LEAX   1,X       INCR POINTER

00524A F9F3 8D   23      FA18 FACT   BSR    UNARY    GET LEFT NODE
00525A F9F5 34   06        A         PSHS   D        SAVE RESULT
00526A F9F7 8D   1A      FA13 FACT2  BSR    SKIP
00527A F9F9 81   26        A         CMPA   #'&      AND OPERATOR?
00528A F9FB 26   08      FA05        BNE    FACT3    SKIP IF NOT
00529A F9FD 8D   17      FA16        BSR    XUNARY   GET RIGHT NODE
00530A F9FF E4   61        A         ANDB   1,S      PERFORM AND OPERATION
00531A FA01 A4   E4        A         ANDA   ,S
00532A FA03 20   0A      FA0F        BRA    FACT4
00533A FA05 81   21        A FACT3   CMPA   #'!      OR OPERATOR?
00534A FA07 26   B2      F9BB        BNE    EXPR5    DONE IF NOT
00535A FA09 8D   0B      FA16        BSR    XUNARY   GET RIGHT NODE
00536A FA0B EA   61        A         ORB    1,S      PERFORM OR OPERATION
00537A FA0D AA   E4        A         ORA    ,S
   38A FA0F ED   E4        A FACT4   STD    ,S       STORE RESULT
   39A FA11 20   E4      F9F7        BRA    FACT2    CHECK FOR MORE

00541                             * Subroutine SKIP - call skip space to
00542                             * next char subroutine
00543A FA13 16   FEF2 F909 SKIP     LBRA   SKIPSP

00545                             * Subroutine UNARY - process unary precedence
00546                             * level

00548A FA16 30   01      A XUNARY LEAX   1,X

00550A FA18 8D   F9      FA13 UNARY  BSR    SKIP
00551A FA1A 81   5E        A         CMPA   #'^      NOT OPERATOR?
00552A FA1C 26   06      FA24        BNE    UNARY2   IF NOT .. CONT
00553A FA1E 8D   07      FA27        BSR    XOPRND   GET OPERAND
00554A FA20 53                       COMB            COMPLIMENT IT
00555A FA21 43                       COMA
00556A FA22 20   0A      FA2E        BRA    UNARY3   DONE...

00558A FA24 81   2D        A UNARY2  CMPA   #'-      NEGATIVE OPERATOR?
00559A FA26 26   09      FA31        BNE    OPRND    TRY OPERAND IF NOT
00560A FA28 8D   05      FA2F        BSR    XOPRND   ELSE GET ITS OPERAND
00561A FA2A 40                       NEGA            AND NEGATE IT
00562A FA2B 50                       NEGB
00563A FA2C 82   00        A         SBCA   #0
00564A FA2E 39                UNARY3 RTS

00566                             * Subroutine OPRAND - process operand
00567                             * operands may be numeric or character
00568                             * constants, registers, dot, parenthetic
00569                             * expressions, or byte/word indirect
00570                             * expressions.

00572A FA2F 30   01      A XOPRND LEAX   1,X

00574                             * Scan for paren expression
00575A FA31 8D   E0      FA13 OPRND  BSR    SKIP
00576A FA33 81   28        A         CMPA   #'(      OPEN PAREN?
   577A FA35 26   0F      FA46        BNE    OPRND2
   578A FA37 17   FF61 F99B           LBSR   XEXPR    PROCESS EXPRESSION
   579A FA3A 34   06        A         PSHS   D        SAVE RESULT
00580A FA3C 8D   D5      FA13        BSR    SKIP
00581A FA3E 81   29        A         CMPA   #')
00582A FA40 27   36      FA78        BEQ    OPRND4   GO FINISH IF OK
00583A FA42 D6   04        A         LDB    #E$PRMS  ELSE ERROR
00584A FA44 20   15      FA5B        BRA    OPRER1

00586                             * Scan for word indirect
00587A FA46 81   5B        A OPRND2  CMPA   #'[      OPEN BRACKET?
00588A FA48 26   16      FA60        BNE    OPRND5   CONTINUE SCAN IF NOT
00589A FA4A 17   FF4E F99B           LBSR   XEXPR    GET EXPRESSION
00590A FA4D 1F   02        A         TFR    D,Y      RESULT IS ADDR
00591A FA4F EC   A4        A         LDD    ,Y       GET CONTENTS
00592A FA51 34   06        A         PSHS   D
00593A FA53 8D   BE      FA13        BSR    SKIP
00594A FA55 81   5D        A         CMPA   #']      CLOSE THERE?
00595A FA57 27   1B      FA78        BEQ    OPRND4   GO FINISH IF OK
00596A FA59 C6   05        A         LDB    #E$BRMS  ELSE ERROR

00598                             * OPERAND ERROR EXITS
00599A FA5B 32   62        A OPRER1 LEAS   2,S       POP OLD RESULT
00600A FA5D 16   FF35 F995 OPRER2 LBRA    IXPERR

00602                             * Scan for byte indirect
00603A FA60 81   3C        A OPRND3  CMPA   #'<      OPEN INDIR?
00604A FA62 26   18      FA7C        BNE    OPRND5   CONTIN SCAN IF NOT
00605A FA64 17   FF34 F99B           LBSR   XEXPR    GET EXPRESSION
00606A FA67 1F   02        A         TFR    D,Y      RESULT IS ADDRESS
00607A FA69 4F                       CLRA            HI BYTE=0
00608A FA6A E6   A4        A         LDB    ,Y       GET CONTENTS
00609A FA6C 34   06        A         PSHS   D        SAVE RESULT
00610A FA6E 8D   A3      FA13        BSR    SKIP
00611A FA70 81   3E        A         CMPA   #'>      CLOSE INDIR?
00612A FA72 27   04      FA78        BEQ    OPRND4   SKIP IF OK
00613A FA74 C6   06        A         LDB    #E$GTMS  ELSE ERROR
00614A FA76 20   I3      FA5B        BRA    OPRER1
00615A FA78 (30) 01        A OPRND4  LEAX   1,X       SKIP CLOSING CHAR
00616A FA7A 35   86        A         PULS   D,PC      POP RESULT AND RETURN

00618                             * Scan for constants
00619A FA7C 81   2E        A OPRND5  CMPA   #'.      DOT?
00620
00621A FA7E 26   05      FA85        BNE    OPRND6   CONTINUE SCAN IF NOT
00622A FA80 DC   00        A         LDD    <DOT
00623A FA82 30   01        A OPR5A   LEAX   1,X
00624A FA84 39                       RTS

00626A FA85 81   27        A OPRND6  CMPA   #''      SGL CHAR?
00627A FA87 26   04      FA8D        BNE    OPRND7   IF NOT SCAN MORE
00628A FA89 EC   81        A         LDD    ,X++     FETCH IT
00629A FA8B 4F                       CLRA
00630A FA8C 39                       RTS

00632A FA8D 81   22        A OPRND7  CMPA   #'"      DBL CHAR?
00633A FA8F 26   05      FA96        BNE    OPRND8   CONT IF NOT
00634A FA91 30   01        A         LEAX   1,X
00635A FA93 EC   81        A         LDD    ,X++     FETCH CHARS
00636A FA95 39                       RTS

00638A FA96 81   3A        A OPRND8  CMPA   #':      REGISTER?
00639A FA98 26   10      FAAA        BNE    OPRND9   CONTINUE IF NOT
00640A FA9A 30   01        A         LEAX   1,X      SKIP :
00641A FA9C 8D   1B      FAB9        BSR    RPARSE   SCAN REG NAME
00642A FA9E 25   BD      FA5D        BCS    OPRER2   EXIT IF ERROR
00643A FAA0 4D                       TSTA            BYTE OR WORD REG?
00644A FAA1 2B   04      FAA7        BMI    OPR8B    BRA IF WORD
00645A FAA3 4F                       CLRA
00646A FAA4 E6   A4        A         LDB    ,Y       GET BYTE
00647A FAA6 39                       RTS             DONE...
00648A FAA7 EC   A4        A OPR8B   LDD    ,Y       GET WORD REG
00649A FAA9 39                OPR8C  RTS

00651A FAAA 17   FDB0 F85D OPRND9  LBSR   NUMBIN    TRY NUMBER
00652A FAAD 24   FA      FAA9        BCC    OPR8C    DONE IF NO ERROR
00653A FAAF 27   04      FAB5        BEQ    OPR9A    BRA IF NOT FOUND
00654A FAB1 C6   03        A         LDB    #E$OPMS  MISSING OPERATOR
00655A FAB3 20   A8      FA5D        BRA    OPRER2
00656A FAB5 C6   00        A OPR9A   LDB    #E$ILNM  ILLEGAL NUMBER
00657A FAB7 20   A4      FA5D        BRA    OPRER2


00659                             * Subroutine RPARSE - parse register name
00660                             * Enter with X = input string.
00661                             * exits with C bit set if legal name
00662                             * not found,
00663                             * else Y = pointer to register on user
00664                             * stack, A has sign bit set if word
00665                             * size register, and X advanced past
00666                             * name in input string.

00668A FAB9 C6   09        A RPARSE  LDB    #9       REGISTER COUNT
00669A FABB 34   04        A         PSHS   B        SAVE IT
00670A FABD EC   84        A         LDD    ,X       GET NEXT 2 CHARS
00671A FABF 1083 5350      A         CMPD   #$5350   D= 'SP' ?
00672A FAC3 26   0D      FAD2        BNE    RPARS@   CONT IF NOT
00673                             * SP IS THE FUNNY CASE
00674A FAC5 30   02        A         LEAX   2,X      SKIP REG NAME
00675A FAC7 CC   0002      A         LDD    #USERSP  ADDR OF SP VALUE
00676A FACA 1F   B8        A         TFR    DP,A     MAKE A REAL ADDRESS
00677A FACC 1F   02        A         TFR    D,Y      COPY TO Y
00678A FACE 86   80        A         LDA    #$80     RETURN WORD SIZE
00679A FAD0 20   24      FAF6        BRA    RPAROK   ..DONE

00681                             * DO A TABLE SEARCH
00682A FAD2 31   8D 002E   RPARS@  LEAY   REGTBL,PCR GET TABLE ADDR
00683A FAD6 A6   A4        A RPARS1  LDA    ,Y       GET 1ST TBL CHAR
00684A FAD8 E6   21        A         LDB    1,Y      GET 2ND
00685A FADA 26   06      FAE2        BNE    RPLONG   EXIT IF TWO CHARS
00686A FADC A1   84        A         CMPA   ,X       COMPARE TO TEXT
00687A FADE 27   09      FAE9        BEQ    RPARS3   EXIT IF MATCH
00688A FAE0 20   18      FAFA        BRA    RPARS4   ELSE LOOP
00689A FAE2 10A3 84        A RPLONG  CMPD   ,X       COMPARE TO TEXT
00690A FAE5 26   13      FAFA        BNE    RPARS4   LOOP IF NO MATCH
00691A FAE7 30   01        A         LEAX   1,X      INCR INPUT PTR
00692A FAE9 30   01        A RPARS3  LEAX   1,X      AGAIN
00693A FAEB A6   22        A         LDA    2,Y      GET OFFSET/LEN BYTE
00694A FAED 1F   89        A         TFR    A,B      COPY IT
00695A FAEF C4   0F        A         ANDB   #$0F     MASK OFFSET
00696A FAF1 109E 02        A         LDY    <USERSP  GET USER SPA ADDR
00697A FAF4 31   A5        A         LEAY   B,Y      GET REG ADDR
00698A FAF6 1C   FE        A RPAROK  ANDCC  #NOTC    RETURN FOUND
00699A FAF8 35   84        A         PULS   B,PC
00700A FAFA 31   23        A RPARS4  LEAY   3,Y      BUMP TABLE PTR
00701A FAFC 6A   E4        A         DEC    ,S       DECR LOOP COUNT
00702A FAFE 26   D6      FAD6        BNE    RPARS1   LOOP TIL DONE
00703A FB00 1A   01        A         ORCC   #CBIT    RETURN ERROR
00704A FB02 35   84        A         PULS   B,PC


00706                             * register name/offset lookup table

00708                FB04     A REGTBL EQU    *
00709A FB04   43           A         FCB    'C,'C,6
00710A FB07   44           A         FCB    'D,'P,3
00711A FB0A   50           A         FCB    'P,'C,$8A
00712A FB0D   41           A         FCB    'A,0,1
00713A FB10   42           A         FCB    'B,0,2
00714A FB13   44           A         FCB    'D,0,$81
00715A FB16   58           A         FCB    'X,0,$84
00716A FB19   59           A         FCB    'Y,0,$86
00717A FB1C   55           A         FCB    'U,2,$86

00719                             TTL    DEBUGGER COMMAND ROUTINES
00721                             * Interactive Debugger Command Routines
00722                             *
00723                             * Request, accept and execute commands
00724                             * from keyboard

00726                FB1F     A SAVORG SET    *

00728A 0000                          ORG    0
00729                             * DIRECT PAGE STORAGE ASSIGNMENTS
00730A 0000        0002      A DOT     RMB    2        DOT ADDRESS
00731A 0002        0002      A USERSP  RMB    2        USER STACK PTR
00732A 0004        0002      A DBUGSP  RMB    2        DEBUG STACK POINTER
00733A 0006        0002      A IOBUF   RMB    2        ADDRESS OF I/O BUFFER
00734A 0008        0002      A OLDDOT  RMB    2        LAST DOT VALUE
00735A 000A        0002      A BKPTBL  RMB    2        BREAKPOINT TABLE
00736              000C      A DPORG   SET    *

00738                             * ERROR CODE DEFINITIONS
00739A 0008                          ORG    ERRORG
```

```
00740A 0008          0001   A E$BTOV RMB    1              BYTE OVERFLOW
00741A 0009          0001   A E$ILCM RMB    1              UNRECOG. COMMAND
00742A 000A          0001   A E$NOCH RMB    1              MEMORY DIDNT CHANGE
00743A 000B          0001   A E$BKFL RMB    1              BREAKPOINT TABLE FULL
00744A 000C          0001   A E$BKNF RMB    1              BREAKPOINT NOT FOUND
00745A 000D          0001   A E$ILBK RMB    1              ILLEGAL BREAKPOINT
00746                000E   A ERRORG SET    *

00748A FB1F                          ORG    SAVORG

00750                        * Initialization Entry Point

00752A FB1F A6   8D 04C3    START   LDA    DIRPAG,PCR GET DIRECT PAGE ADDR
00753A FB23 1F   8B       A         TFR    A,DP      GET DP ADDRESS
00754A FB25 C6   FF       A         LDB    #$FF      START AT TOP OF DP
00755A FB27 1F   04       A         TFR    D,S       ALLOCATE USER STACK
00756A FB29 32   74       A         LEAS   -12,S     MAKE ROOM FOR REGS
00757A FB2B 10DF 02       A         STS    USERSP    SAVE INITIAL USER SP
00758A FB2F 31   8D 0373            LEAY   NOGO,PCR
00759A FB32 10AF 6A       A         STY    10,S      DEFAULT USER PC
00760A FB35 86   80       A         LDA    #$80
00761A FB37 A7   E4       A         STA    ,S        INITIAL USER CC REG
00762A FB39 32   E8 CC    A         LEAS   -52,S     GIVE TOTAL OF 64 BYTES
00763A FB3C 10DF 04       A         STS    DBUGSP    SET DEBUG TOP OF STACK
00764A FB3F 1F   41       A         TFR    S,X       COPY MEM PTR TO X
00765A FB41 30   88 88    A         LEAX   -(72+48),X ALLOCATE I/O BUFFER
00766A FB44 9F   06       A         STX    IOBUF     SAVE ITS ADDRESS
00767A FB46 30   88 DC    A         LEAX   -(12*3),X ALLOCATE BKPT TABLE
00768A FB49 9F   0A       A         STX    BKPTBL    SAVE ITS ADDRESS
00769A FB4B 0F   00       A         CLR    DOT       RESET DOT
00770A FB4D 0F   01       A         CLR    DOT+1
00771A FB4F 6F   80       A INIT2   CLR    ,X+       CLEAR BKPT TABLE
00772A FB51 9C   06       A         CPX    IOBUF
00773A FB53 25   FA   FB4F          BLO    INIT2     LOOP TIL CLEAR
00774A FB55 31   8D 02BC            LEAY   BKPT,PCR GET ADDR OF BKPT SERVICE ROUTINE
00775A FB59 17   047F FFDB          LBSR   SETSWI    GO SETUP SWI VECTOR
00776A FB5C 31   8D 02FA            LEAY   SSBK,PCR GET ADDR OF SS SERVICE ROUTINE
00777A FB60 17   047C FFDF          LBSR   SETNMI    GO SETUP NMI VECTOR
00778A FB63 17   03BD FF23          LBSR   IOINIT    INITIALIZE I/O
00779A FB66 9E   06       A         LDX    IOBUF
00780A FB68 17   03E3 FF4E          LBSR   WRTLIN    OUTPUT BLANK LINE
00781A FB6B 9E   06       A         LDX    IOBUF
00782A FB6D 31   8D 033B            LEAY   TITLE,PCR GET PGM NAME STRING ADDR
00783A FB71 8D   43   FBB6          BSR    CPYSTR    MOVE TO BUFFER
00784A FB73 17   03D8 FF4E INIT3    LBSR   WRTLIN
00785                        * FALL THROUGH TO COMMAND LOOP

00787                        * Command Loop

00789A FB76 31   8D 0341    COMAND  LEAY   PROMPT,PCR GET PROMPT STR ADDR
00790A FB7A 17   041B FF96          LBSR   WRTSTR    PRINT IT
00791A FB7D 17   03E4 FF64          LBSR   RDLIN     INPUT COMMAND LINE
00792A FB80 25   F1   FB73          BCS    INIT3     IGNORE IF DELETED LINE
00793                        * COMMAND TABLE SEARCH
00794A FB82 31   8D 0370            LEAY   CMDTBL-2,PCR GET TABLE ADDR
00795A FB86 31   22       A CMD1    LEAY   2,Y       INCR TABLE PTR
00796A FB88 A6   A0       A         LDA    ,Y+       GET MATCH CHAR
00797A FB8A 27   10   FB9C          BEQ    ILLCMD    EXIT IF TABLE END
00798A FB8C A1   84       A         CMPA   ,X        COMPARE TO TEXT
00799A FB8E 26   F6   FB86          BNE    CMD1      LOOP MORE IF NO MATCH
00800A FB90 30   01       A         LEAX   1,X       BUMP TEXT PTR
00801A FB92 EC   A4       A         LDD    ,Y        GET ROUTINE OFFSET FROM TABLE
00802A FB94 33   8D 0360            LEAU   CMDTBL,PCR GET TABLE ADDRESS
00803A FB98 AD   CB       A         JSR    D,U       CALL COMMAND ROUTINE
00804A FB9A 20   DA   FB76          BRA    COMAND    GO FOR ANOTHER

00806                        * UNRECOGNIZED COMMAND HANDLER
00807A FB9C C6   09       A ILLCMD  LDB    #E$ILCM
00808A FB9E 8D   02   FBA2          BSR    CMDERR    REPORT IT...
00809A FBA0 20   D4   FB76          BRA    COMAND

00811                        * COMMAND ERROR PROCESSOR
00812A FBA2 9E   06       A CMDERR  LDX    IOBUF
00813A FBA4 31   8D 034A            LEAY   ERRMSG,PCR
00814A FBA8 8D   0C   FBB6          BSR    CPYSTR    MOVE TO BUFFER
00815A FBAA 17   FC80 F82D          LBSR   BINDEC    CONVERT IT IT
00816A FBAD 30   1D       A         LEAX   -3,X      BACKUP OUTPUT PTR
00817A FBAF EC   01       A         LDD    1,X       GET LAST 2 CHARS OF NUMBER
00818A FBB1 ED   1E       A         STD    -2,X      MOVE THEM BACK
00819A FBB3 16   0398 FF4E          LBRA   WRTLIN    PRINT IT

00821                        * CALL UTILITY "STRMOV"
00822A FBB6 16   FDB9 F972 CPYSTR   LBRA   STRMOV

00825                        * Command Processing Routines

00827                        * Dot Commands

00829A FBB9 A6   84       A DOTCMD  LDA    ,X
00830A FBBB 81   2E       A         CMPA   #'.       DOUBLE DOT?
00831A FBBD 26   04   FBC3          BNE    DOTCM2    SKIP IF NOT
00832A FBBF DC   08       A         LDD    OLDDOT    GET LAST VALUE
00833A FBC1 20   0D   FBD0          BRA    DSPDOT    GO DISPLAY IT

00835A FBC3 81   0D       A DOTCM2  CMPA   #$D       EOL?
00836A FBC5 26   04   FBCB          BNE    DOTCM3    SKIP IF NOT
00837A FBC7 DC   00       A CURDOT  LDD    DOT       GET CURRENT DOT
00838A FBC9 20   05   FBD0          BRA    DSPDOT    GO DISPLAY IT

00840A FBCB 17   FDBD F98B DOTCM3   LBSR   EVAL      TRY AN EXPRESSION
00841A FBCE 25   D2   FBA2          BCS    CMDERR    EXIT IF ERROR

00843                        * UPDATE AND DISPLAY DOT
00844A FBD0 9E   00       A DSPDOT  LDX    DOT       GET PRESENT DOT
00845A FBD2 9F   08       A         STX    OLDDOT    MOVE TO OLD DOT
00846A FBD4 DD   00       A         STD    DOT       PUT NEW DOT
00847A FBD6 34   06       A         PSHS   D         SAVE IT
00848A FBD8 8D   2F   FC09          BSR    INDENT    PRINT INDENT
00849A FBDA EC   E4       A         LDD    ,S        RESTORE DOT
00850A FBDC 17   FC21 F800          LBSR   BIN4HS    CONVERT TO HEX
00851A FBDF 35   20       A         PULS   Y         POP DOT AS ADDRESS
00852A FBE1 E6   A4       A         LDB    ,Y        GET MEM CONTENTS
00853A FBE3 17   FC2E F814          LBSR   BIN2HX    CONVERT IT
00854A FBE6 16   0365 FF4E          LBRA   WRTLIN    PRINT LINE

00856                        * "-" COMMAND - BACKUP DOT AND PRINT
00857A FBE9 DC   00       A DOTBAK  LDD    DOT       GET DOT
00858A FBEB 83   0001     A         SUBD   #1        DECRMENT IT
00859A FBEE 20   E0   FBD0          BRA    DSPDOT    GO PRINT IT

00861                        * "=" COMMAND - CHANGE MEMORY
00862A FBF0 8D   3C   FC2E CNGMEM   BSR    BYTVAL
00863A FBF2 25   AE   FBA2          BCS    CMDERR    EXIT IF ERROR
00864A FBF4 9E   00       A         LDX    DOT       GET DOT ADDR
00865A FBF6 E7   84       A         STB    ,X        STORE IT
00866A FBF8 E1   84       A         CMPB   ,X        VERIFY IT CHANGED
00867A FBFA 27   06   FC02          BEQ    DOTFWD    PRINT NEXT IF OK
00868A FBFC C6   0A       A         LDB    #E$NOCH   ELSE ERROR
00869A FBFE 8D   A2   FBA2          BSR    CMDERR    REPORT IT
00870A FC00 20   C5   FBC7          BRA    CURDOT    REPRINT IT

00872                        * "<EOL>" COMMAND - ADVANCE DOT AND PRINT
00873A FC02 DC   00       A DOTFWD  LDD    DOT       GET DOT
00874A FC04 C3   0001     A         ADDD   #1        INCREMENT IT
00875A FC07 20   C7   FBD0          BRA    DSPDOT

00877                        * SUBROUTINE INDENT - OUTPUT LINE INDENT

00879A FC09 9E   06       A INDENT  LDX    IOBUF     GET BUFFER ADDRESS
00880A FC0B 8D   00   FC0D          BSR    IND2      PUT 1 SPACE
00881A FC0D 16   FBF6 F806 IND2     LBRA   PUTSPC    PUT 1 SPACE

00883                        * <SP> COMMAND - EVALUATE EXPRESSION AND
00884                        * DISPLAY RESULT IN HEX AND DECIMAL

00886A FC10 17   FD78 F98B CNVCMD   LBSR   EVAL      EVALUATE EXPR
00887A FC13 25   8D   FBA2          BCS    CMDERR    EXIT IF ERROR
00888A FC15 8D   F2   FC09          BSR    INDENT
00889A FC17 34   06       A         PSHS   D         SAVE RESULT
00890A FC19 86   24       A         LDA    #'$       OUTPUT "$"
00891A FC1B A7   80       A         STA    ,X+
00892A FC1D A6   E4       A         LDA    ,S        RESTORE A
00893A FC1F 17   FBDE F800          LBSR   BIN4HS    CONVERT TO HEX
00894A FC22 86   23       A         LDA    #'#       OUTPUT "#"
00895A FC24 A7   80       A         STA    ,X+
00896A FC26 35   06       A         PULS   D         RESTORE RESULT
00897A FC28 17   FC02 F82D          LBSR   BINDEC    CONVERT IT
00898A FC2B 16   0320 FF4E          LBRA   WRTLIN    GO PRINT IT

00900                        * SUBROUTINE TO EVALUATE EXPRESSION TO
00901                        * BYTE VALUE - RETURN ERROR IF RESULT
00902                        * IS LARGER THAN 255

00904A FC2E 17   FD5A F98B BYTVAL   LBSR   EVAL      CALL EVAL
00905A FC31 25   07   FC3A          BCS    BYTVL2    EXIT IF EXPR ERROR
00906A FC33 4D                      TSTA             RESULT IN RANGE?
00907A FC34 27   04   FC3A          BEQ    BYTVL2    RETURN IF SO
00908A FC36 C6   08       A         LDB    #E$BTOV   ELSE RETURN ERROR
00909A FC38 1A   01       A         ORCC   #1
00910A FC3A 39              BYTVL2  RTS

00912                        * ":" REGISTER EXAMINE/CHANGE COMMAND

00914A FC3B 17   00A8 FCE6 REGCMD   LBSR   CHKEOL
00915A FC3E 27   3F   FC7F          BEQ    DSPREG    IF SO PRINT REGISTERS
00916A FC40 17   FE76 FAB9          LBSR   RPARSE    ELSE GET REG NAME, ADDR
00917A FC43 1025 FF5B FBA2          LBCS   CMDERR    EXIT IF ERROR
00918A FC47 34   22       A         PSHS   A,Y       SAVE ADDR, LEN CODE
00919A FC49 17   009A FCE6          LBSR   CHKEOL
00920A FC4C 26   16   FC64          BNE    REGCM3    IF NOT GO PROCESS
00921                        * DISPLAY REGISTER CONTENTS
00922A FC4E 8D   B9   FC09          BSR    INDENT    START OUTPUT LINE
00923A FC50 35   22       A         PULS   A,Y       RESTORE ADDR, LENGTH
00924A FC52 4D                      TSTA             BYTE OR WORD?
00925A FC53 2A   07   FC5C          BPL    REG2A     SKIP IF BYTE REG
00926A FC55 EC   A4       A         LDD    ,Y        ELSE GET WORD
00927A FC57 17   FBB4 F80E          LBSR   BIN4HX    CONVERT IT
00928A FC5A 20   05   FC61          BRA    REG2B
00929A FC5C E6   A4       A REG2A   LDB    ,Y        GET BYTE REG
00930A FC5E 17   FBB3 F814          LBSR   BIN2HX    CONVERT TO HEX
00931A FC61 16   02EA FF4E REG2B    LBRA   WRTLIN    OUTPUT LINE + RETURN

00933                        * CHANGE REGISTER CONTENTS
00934A FC64 A6   E0       A REGCM3  LDA    ,S+       POP REG LEN CODE
00935A FC66 2A   0C   FC74          BPL    REG3A     SKIP IF BYTE REG
00936A FC68 17   FD20 F98B          LBSR   EVAL      PROCESS EXPRESSION
00937A FC6B 35   20       A         PULS   Y         POP REGISTER ADDR
00938A FC6D 1025 00B1 FD22          LBCS   BCMDER    EXIT IF ERROR
00939A FC71 ED   A4       A         STD    ,Y        ELSE PUT RESULT IN REG
00940A FC73 39                      RTS              ...DONE
00941A FC74 8D   B8   FC2E REG3A    BSR    BYTVAL    PROCESS BYTE EXPR
00942A FC76 35   20       A         PULS   Y         POP REG ADDRESS
00943A FC78 1025 00A6 FD22          LBCS   BCMDER    EXIT IF ERROR
00944A FC7C E7   A4       A         STB    ,Y        PUT RESULT IN REG
00945A FC7E 39                      RTS              ...DONE

00947                        * SUBROUTINE DSPREG - DISPLAY
00948                        * USER REGISTER CONTENTS

00950A FC7F 8D   88   FC09 DSPREG   BSR    INDENT    GET LINE READY
00951A FC81 31   8D 0239            LEAY   REGSTR,PCR GET HEADER STRING ADDR
00952A FC85 17   FF2E FBB6          LBSR   CPYSTR    COPY TO OUTPUT BUFFER
00953A FC88 17   02C3 FF4E          LBSR   WRTLIN    OUTPUT IT
00954A FC8B 17   FF7B FC09          LBSR   INDENT    GET SECOND LINE READY
00955A FC8E DC   02       A         LDD    USERSP    GET SP CONTENTS
00956A FC90 8D   27   FCB9          BSR    C4HEX2    CONVERT IT
00957A FC92 109E 02       A         LDY    USERSP    GET SP VAL IN Y
00958A FC95 8D   2A   FCC1          BSR    DSPCC     CONV CC
00959A FC97 8D   23   FCBC          BSR    C2HEX     CONV A
00960A FC99 8D   21   FCBC          BSR    C2HEX     CONV B
00961A FC9B 8D   1F   FCBC          BSR    C2HEX     CONV DP
00962A FC9D 8D   18   FCB7          BSR    C4HEX     CONV X
00963A FC9F 8D   16   FCB7          BSR    C4HEX     CONV Y
00964A FCA1 8D   14   FCB7          BSR    C4HEX     CONV U
00965A FCA3 8D   12   FCB7          BSR    C4HEX     CONV PC
00966A FCA5 10AE 3E       A         LDY    -2,Y      GET PROGRAM COUNTER
00967A FCA8 8D   12   FCBC          BSR    C2HEX     DISPLAY OPCODE TO EXEC NEXT
00968A FCAA 109E 00       A         LDY    DOT       GET CURRENT MEM ADDR
00969A FCAD 8D   0D   FCBC          BSR    C2HEX     PRINT IT
00970A FCAF 109E 08       A         LDY    OLDDOT    GET PREVIOUS MEM ADDR
00971A FCB2 8D   08   FCBC          BSR    C2HEX     PRINT IT
00972A FCB4 16   0297 FF4E          LBRA   WRTLIN    OUTPUT LINE + RTS

00974                        * SUBROUTINES C2HEX AND C4HEX -
00975                        * LOAD AUTOINCRMENT FROM Y AND
00976                        * CONVERT TO HEX

00978A FCB7 EC   A1       A C4HEX   LDD    ,Y++
00979A FCB9 16   FB44 F800 C4HEX2   LBRA   BIN4HS

00981A FCBC E6   A0       A C2HEX   LDB    ,Y+
00982A FCBE 16   FB43 F804          LBRA   BIN2HS
00983                        *
00984                        *       SUBROUTINE TO PRINT CONDITION CODES REGISTER
00985                        *
00986A FCC1 E6   A0       A DSPCC   LDB    ,Y+       GET CONDITION CODES
00987A FCC3 86   08       A         LDA    #8        NUMBER OF FLAGS
00988A FCC5 34   22       A         PSHS   A,Y       SAVE REGISTER AND COUNT
```

```
00989A FCC7 31   8D 0013          LEAY   SFLGS,PCR ADDR OF FLAG STRING

00991A FCCB A6   A0      A DSPCC1 LDA    ,Y+       INIT CHAR TO WRITE
00992A FCCD 58                    ASLB              CHECK FLAG VIA CARRY BIT
00993A FCCE 25   02   FCD2        BCS    DSPCC2    PRINT NAME IF SET
00994A FCD0 8E   2D      A        LDA    #'-       OTHERWISE HYPHEN

00996A FCD2 A7   80      A DSPCC2 STA    ,X+       STORE IN BUFFER
00997A FCD4 6A   E4      A        DEC    ,S        DECREMENT COUNTER
00998A FCD6 26   F3   FCCB        BNE    DSPCC1    BRANCH IF NOT DONE

01000A FCD8 86   20      A DSPCC3 LDA    #SPACE    PRINT A SPACE
01001A FCDA A7   80      A        STA    ,X+
01002A FCDC 35   A2      A        PULS   A,Y,PC    RETURN
01003A FCDE              45      A SFLGS  FCC    /EFHINZVC/CONDITION CODE LABELS

01005                             * SUBROUTINE CHKEOL
01006                             * SKIP SPACES IN INPUT BUFFER
01007                             * THEN CHECK IF NEXT CHARACTER
01008                             * IS A CARRIAGE RETURN

01010A FCE6 17   FC20 F909 CHKEOL LBSR   SKIPSP
01011A FCE9 81   0D      A        CMPA   #$D
01012A FCEB 39                    RTS

01014                             * "B" COMMAND - SET OR DISPLAY BREAKPOINTS

01016A FCEC 8D   F8   FCE6 BCMD   BSR    CHKEOL    IS NEXT CHAR RETURN?
01017A FCEE 26   1C   FD0C        BNE    BCMD2     IF NOT CONTINUE
01018A FCF0 17   FF16 FC09        LBSR   INDENT    ELSE GET OUTPUT READY
01019A FCF3 109E 0A      A        LDY    BKPTBL    GET BKPT TABLE ADDRESS
01020A FCF6 C6   0C      A        LDB    #12       MAX NUMBER OF BREAKPOINTS
01021A FCF8 34   04      A        PSHS   B         PUSH COUNT
01022                             * BREAKPOINT DISPLAY LOOP
01023A FCFA EC   A4      A BCMD1  LDD    ,Y        GET NEXT ADDR
01024A FCFC 27   03   FD01        BEQ    BCMD1A    SKIP IF NOT ACTIVE
01025A FCFE 17   FAFF F800        LBSR   BIN4HS    ELSE CONVERT TO HEX
01026A FD01 31   23      A BCMD1A LEAY   3,Y       MOVE TO NEXT ENTRY
01027A FD03 6A   E4      A        DEC    ,S        DECREMENT COUNT
01028A FD05 26   F3   FCFA        BNE    BCMD1     LOOP IF MORE TO DO
01029A FD07 32   61      A        LEAS   1,S       CLEAN UP STACK
01030A FD09 16   0242 FF4E        LBRA   WRTLIN    OUTPUT LINE +RTS

01032                             * SET BREAKPOINT
01033A FD0C 17   FC7C F98B BCMD2  LBSR   EVAL      PROCESS EXPR
01034A FD0F 25   11   FD22        BCS    BCMDER    EXIT IF ERROR
01035A FD11 34   06      A        PSHS   D         SAVE RESULT
01036A FD13 8D   15   FD2A        BSR    FNDBKP    SEARCH TABLE FOR THIS ADDR
01037A FD15 27   0E   FD25        BEQ    BCMD2A    SKIP IF ALREADY IN TABLE
01038A FD17 CC   0000    A        LDD    #0        SEARCH FOR EMPTY ENTRY
01039      A 8D   0E   FD2A        BSR    FNDBKP
01040.     C 27   07   FD25        BEQ    BCMD2A    SKIP IF FOUND
01041A .01E C6   0B      A        LDB    #E$BKFL   ELSE TABLE FULL ERROR
01042A FD20 32   62      A        LEAS   2,S       CLEAN UP STACK
01043A FD22 16   FE7D FBA2 BCMDER LBRA   CMDERR

01045A FD25 35   06      A BCMD2A PULS   D         POP ADDR
01046A FD27 ED   A4      A        STD    ,Y        PUT IN TABLE
01047A FD29 39                    RTS               DONE

01049                             * SUBROUTINE FNDBKP - SEARCH BREAKPOINT
01050                             * TABLE FOR ADDRESS
01051                             *  D = ADDRESS OF BREAKPOINT
01052                             * RETURNS TABLE ADDRESS IN Y AND Z BIT SET
01053                             * FOUND, ELSE Z BIT CLEARED

01055A FD2A 34   40      A FNDBKP PSHS   U         U IS LOCAL
01056A FD2C 1F   03      A        TFR    D,U       COPY BKPT ADDRESS
01057A FD2E C6   0C      A        LDB    #12       MAX TABLE ENTRIES
01058A FD30 109E 0A      A        LDY    BKPTBL    GET TABLE ADDRESS
01059A FD33 11A3 A4      A FNDBK2 CMPU   ,Y        DOES THIS ONE MATCH?
01060A FD36 27   09   FD41        BEQ    FNDBK3    IF SO, WE'RE DONE
01061A FD38 31   23      A        LEAY   3,Y       ELSE MOVE Y TO NEXT ENTRY
01062A FD3A 5A           A        DECB              DECR COUNT
01063A FD3B 26   F6   FD33        BNE    FNDBK2    LOOP IF NOT DONE
01064A FD3D D6   0C      A        LDB    E$BKNF    BKPT NOT FOUND ERROR
01065A FD3F 1C   FB      A        ANDCC  #NOTZ     ELSE CLEAR Z BIT
01066A FD41 35   C0      A FNDBK3 PULS   U,PC

01068                             * "K" COMMAND: KILL ONE OR ALL
01069                             * BREAKPOINTS

01071A FD43 8D   A1   FCE6 KCMD   BSR    CHKEOL    NEXT CHAR=RET?
01072A FD45 27   0E   FD55        BEQ    KCMD2     IF SO, GO CLEAR ALL
01073A FD47 17   FC41 F98B        LBSR   EVAL      ELSE GET EXPRESSION
01074A FD4A 25   D6   FD22        BCS    BCMDER    EXIT IF ERROR
01075A FD4C 8D   DC   FD2A        BSR    FNDBKP    FIND BREAKPOINT
01076A FD4E 26   D2   FD22        BNE    BCMDER    EXIT IF NOT FOUND
01077A FD50 4F           A        CLRA
01078A FD51 5F           A        CLRB
01079      52 ED   A4      A      STD    ,Y        CLEAR THIS ONE
01080      A4 39                    RTS               DONE
01081A FD55 109E 0A      A KCMD2  LDY    BKPTBL    CLEAR ALL BREAKPOINTS
01082A FD58 C6   24      A        LDB    #12*3     TABLE SIZE
01083A FD5A 6F   A0      A KCMD2A CLR    ,Y+       CLEAR LOOP
01084A FD5C 5A           A        DECB              DECR COUNT
01085A FD5D 26   FB   FD5A        BNE    KCMD2A    LOOP TIL DONE
01086A FD5F 39                    RTS

01088                             * "G" GO TO USER PROGRAM COMMAND
01089                             * EXECUTE AT PC ADDRESS OR UPDATE
01090                             * PC IF EXPRESSION IN COMMAND LINE.

01092A FD60 8D   84   FCE6 GOCMD  BSR    CHKEOL    FIRST CHAR=RETURN?
01093A FD62 27   0A   FD6E        BEQ    GOCMD2    SKIP IF SO
01094A FD64 17   FC24 F98B        LBSR   EVAL      ELSE GET EXPR
01095A FD67 25   B9   FD22        BCS    BCMDER    EXIT IF ERROR
01096A FD69 109E 02      A        LDY    USERSP    GET SP ADDR
01097A FD6C ED   2A      A        STD    10,Y      MAKE PC=EXPR RESULT
01098                             * INSERT BREAKPOINTS
01099A FD6F 109E 0A      A GOCMD2 LDY    BKPTBL    GET BREAKPOINNT TBLE ADDR
01100A FD71 C6   0C      A        LDB    #12       MAX COUNT
01101A FD73 EB   A4      A GOCMD3 LDU    ,Y        GET ADDRESS
01102A FD75 27   08   FD7F        BEQ    GOCMD4    EXIT IF NOT ACTIVE
01103A FD77 A6   C4      A        LDA    ,U        GET INSTR OPCODE
01104A FD79 A7   22      A        STA    2,Y       SAVE IN TABLE
01105A FD7B 86   3F      A        LDA    #$3F      INSERT SWI
01106A FD7D A7   C4      A        STA    ,U        IN PGM
01107A FD7F 31   23      A GOCMD4 LEAY   3,Y       MOVE TO NEXT TBLE ENTRY
01108A FD81 5A           A        DECB              DONE YET?
01109A FD82 26   EF   FD73        BNE    GOCMD3    LOOP IF NOT
01110A FD84 10DE 02      A        LDS    USERSP    ELSE RESTORE USER SP
01111A FD87 3B                    RTI               AND GO...

01113                             * "M" COMMAND: DISPLAY FORMATTED HEX
```

```
01114                             * AND ASCII DUMP OF MEMORY

01116A FD88 8D   4C   FDD6 MEMCMD BSR    BEGEND    GET BEG, END ADDRESSES
01117A FD8A 25   96   FD22        BCS    BCMDER    EXIT IF ERROR
01118A FD8C CA   0F      A        ORB    #$0F      ROUND END ADDR MOD 16
01119A FD8E C3   0001    A        ADDD   #1
01120A FD91 1E   03      A        EXG    D,U
01121A FD93 C4   F0      A        ANDB   #$F0      ROUND BEG ADDR MOD 16
01122A FD95 34   46      A        PSHS   D,U       SAVE BOTH ON STACK
01123                             * TOP OF DISPLAY LOOP
01124A FD97 10AE E4      A MEMCM2 LDY    ,S        GET CURRENT ADDR
01125A FD9A 10AC 62      A MEM2A  CMPY   2,S       COMPARE TO END ADDR
01126A FD9D 26   02   FDA1        BNE    MEM2B     DONE YET?
01127A FD9F 35   C6      A        PULS   D,U,PC    YES; POP TEMPS + RTS
01128A FDA1 9E   06      A MEM2B  LDX    IOBUF     NO; SETUP NEXT DISPLAY LINE
01129A FDA3 1F   20      A        TFR    Y,D       COPY CURRENT ADDRESS
01130A FDA5 17   FA5B F800        LBSR   BIN4HS    CONVERT TO HEX
01131                             * OUTPUT 16 MEMORY BYTES IN HEX
01132A FDA8 C6   08      A        LDB    #8        SETUP WORD COUNT
01133A FDAA 34   04      A        PSHS   B         SAVE ON STACK
01134A FDAC 17   FF08 FCB7 MEMCM3 LBSR   C4HEX     OUTPUT WORD + SPACE
01135A FDAF 6A   E4      A        DEC    ,S        DECREMENT COUNTER
01136A FDB1 26   F9   FDAC        BNE    MEMCM3    LOOP TIL DONE
01137                             * OUTPUT 16 MEMORY BYTES IN ASCII
01138A FDB3 C6   10      A        LDB    #16       BYTE COUNT
01139A FDB5 E7   E4      A        STB    ,S        PUT ON STACK
01140A FDB7 10AE 61      A        LDY    1,S       RESTORE CURRENT ADDR
01141A FDBA A6   A0      A MEMCM4 LDA    ,Y+       GET CHAR
01142A FDBC 81   7E      A        CMPA   #$7E      IN ASCII RANGE?
01143A FDBE 22   04   FDC4        BHI    MEM4A     IF NOT SUBSTITUTE
01144A FDC0 81   20      A        CMPA   #$20      DISPLAYABLE CHAR?
01145A FDC2 24   02   FDC6        BHS    MEMCM5    IF YES, GO STORE IT
01146A FDC4 86   2E      A MEM4A  LDA    #'.       ELSE SUBSTITUTE PERIOD
01147A FDC6 A7   80      A MEMCM5 STA    ,X+       STORE CHAR IN BUFFER
01148A FDC8 6A   E4      A        DEC    ,S        DECREMENT COUNT
01149A FDCA 26   EE   FDBA        BNE    MEMCM4    LOOP IF NOT ZERO
01150A FDCC 32   61      A        LEAS   1,S       POP COUNTER
01151A FDCE 10AF E4      A        STY    ,S        SAVE NEW CURRENT ADDRESS
01152A FDD1 17   017A FF4E        LBSR   WRTLIN    DISPLAY THE LINE
01153A FDD4 20   C4   FD9A        BRA    MEM2A     END LINE LOOP

01155                             * SUBROUTINE BEGEND: GET BEGIN AND
01156                             * END ADDRESS EXPRESSIONS

01158A FDD6 17   FBB2 F98B BEGEND LBSR   EVAL      FIRST EXPRESSION
01159A FDD9 25   05   FDE0        BCS    BGND2     EXIT IF ERROR
01160A FDDB 1F   03      A        TFR    D,U       COPY RESULT TO U
01161A FDDD 17   FBAB F98B        LBSR   EVAL      SECOND EXPRESSION
01162A FDE0 39                    BGND2  RTS

01164                             * "C" COMMAND: CLEAR AND TEST MEMORY

01166A FDE1 8D   F3   FDD6 CCMD   BSR    BEGEND    GET ADDR PARAMETERS
01167A FDE3 1025 FDBB FBA2        LBCS   CMDERR    EXIT IF ERROR
01168A FDE7 34   06      A        PSHS   D         SAVE END ADDRESS ON STACK
01169A FDE9 11A3 E4      A CCMD2  CMPU   ,S        DONE YET?
01170A FDEC 23   02   FDF0        BLS    CCMD3     CONTINUE IF NOT
01171A FDEE 35   86      A        PULS   D,PC      ELSE POP TEMP + RTS
01172A FDF0 CC   8008    A CCMD3  LDD    #$8008    LOAD COUNT AND PATTERN
01173A FDF3 A7   C4      A        STA    ,U        STORE PATTERN
01174A FDF5 A1   C4      A CCMD4  CMPA   ,U        COMPARE PATTERN TO MEM
01175A FDF7 26   08   FE01        BNE    CCMD5     ERROR IF NO MATCH
01176A FDF9 44           A        LSRA              ROTATE PATTERN
01177A FDFA 64   C4      A        LSR    ,U        ROTATE MEMORY
01178A FDFC 5A           A        DECB              ALL BITS TESTED?
01179A FDFD 26   F6   FDF5        BNE    CCMD4     LOOP IF NOT
01180A FDFF 20   10   FE11        BRA    CCMD6     ELSE CONTINUE
01181A FE01 17   FE05 FC09 CCMD5  LBSR   INDENT    REPORT BAD LOCATION
01182A FE04 CC   2D20    A        LDD    #$2D20    OUTPUT  -
01183A FE07 ED   81      A        STD    ,X++
01184A FE09 1F   30      A        TFR    U,D       COPY CURRENT ADDR
01185A FE0B 17   FA00 F80F        LBSR   BIN4HX    CONVERT TO HEXT
01186A FE0E 17   013D FF4E        LBSR   WRTLIN    OUTPUT THIS LINE
01187A FE11 33   41      A CCMD6  LEAU   1,U       INCREMENT CURRENT ADDRESS
01188A FE13 20   D4   FDE9        BRA    CCMD2     BOTTOM OF LOOP

01190                             * BREAKPOINT SERVICE ROUTINE
01191                             *
01192                             * SAVE USER STACK, SWITCH TO
01193                             * DEBUG STACK, REMOVE BREAKPOINTS
01194                             * AND DUMP USER REGISTERS

01196A FE15 A6   8D 01CD  BKPT   LDA    DIRPAG,PCR GET DEBUG DP
01197A FE19 1F   8B      A        TFR    A,DP
01198A FE1B 10DF 02      A        STS    USERSP    SAVE USER STACK PTR
01199A FE1E EC   6A      A        LDD    10,S      GET USER'S PC
01200A FE20 83   0001    A        SUBD   #1        DECREMENT IT
01201A FE23 ED   6A      A        STD    10,S      REPLACE IT
01202A FE25 10DE 04      A        LDS    DBUGSP    LOAD DEBUG STACK
01203A FE28 17   FEFF FD2A        LBSR   FNDBKP    ADDR IN TABLE?
01204A FE2B 27   05   FE32        BEQ    BKPT2     SKIP IF SO
01205A FE2D C6   0D      A        LDB    #E$ILBK   ELSE REPORT ILLEGAL BREAKPOINT
01206A FE2F 17   FD70 FBA2        LBSR   CMDERR
01207                             * REMOVE BREAKPOINTS FROM PGM
01208A FE32 109E 0A      A BKPT2  LDY    BKPTBL
01209A FE35 C6   0C      A        LDB    #12       MAX COUNT
01210A FE37 A6   A4      A BKPT3  LDX    ,Y        GET ADDR OF BKPT
01211A FE39 27   04   FE3F        BEQ    BKPT4     SKIP IF NOT ACTIVE
01212A FE3B A6   22      A        LDA    2,Y       GET SAVED OPCODE
01213A FE3D A7   84      A        STA    ,X        RESTORE IT
01214A FE3F 31   23      A BKPT4  LEAY   3,Y       MOVE TO NEXT ENTRY
01215A FE41 5A           A        DECB              DONE YET?
01216A FE42 26   F3   FE37        BNE    BKPT3     LOOP IF NOT
01217A FE44 17   FE38 FC7F        LBSR   DSPREG    PRINT REGISTERS
01218A FE47 16   FD2C FB76        LBRA   COMAND    GOTO COMMAND LOOP

01220                             *
01221                             *       SINGLE STEP SERVICE ROUTINE
01222                             *
01223A FE4A 86   01      A SSPK   LDA    #1
01224A FE4C A7   9D 0197          STA    [TIMER,PCR] STOP TIMER
01225A FE50 A6   8D 0192          LDA    DIRPAG,PCR GET DEBUGGER DIRECT PAGE
01226A FE54 1F   8B      A        TFR    A,DP
01227A FE56 10DF 02      A        STS    USERSP    SAVE USER STACK
01228A FE59 10DE 04      A        LDS    DBUGSP    LOAD DEBUGGER STACK
01229A FE5C 17   FE20 FC7F        LBSR   DSPREG
01230A FE5F 16   FD14 FB76        LBRA   COMAND

01232                             *
01233                             *       SINGLE STEP COMMAND PROCESSOR
01234                             *
01235A FE62 10DE 02      A SSCMD  LDS    USERSP    LOAD USER STACK
01236A FE65 6F   9D 017E          CLR    [TIMER,PCR] START TIMER
01237A FE69 3B                    RTI               GO TO PROGRAM

01239                             *
01240                             *       RELOCATE COMMAND PROCESSOR
01241                             *
```

```
01242A FE6A 17  FF69 FDD6 RCMD   LBSR  BEGEND   PARSE REST OF CMD LINE
01243A FE6D 1025 FD31 FBA2       LBCS  CMDERR   BRANCH IF ERROR
01244A FE71 C3  0000    A        ADDD  #0       TEST D FOR 0
01245A FE74 26  01      FE77     BNE   RCMD1    BRANCH IF NOT 0
01246A FE76 39                   RTS            BRANCH IF 0
01248A FE77 109E 00     A RCMD1  LDY   DOT      SET PARMS FOR BLOCK MOVE
01249A FE7A 1F  31      A        TFR   U,X
01250A FE7C 16  FAF8 F977        LBRA  BLKMOV   DO BLOCK MOVE
01252                   *
01253                   *
01254                   *        SEARCH COMMAND PROCESSOR
01255A FE7F 17  FF54 FDD6 SCCMD  LBSR  BEGEND   PARSE REST OF COMMAND LINE
01256A FE82 1025 FD1C FBA2       LBCS  CMDERR   BRANCH IF ERROR
01257A FE86 34  40      A        PSHS  U        PUSH TO USE FOR COMPARE
01258A FE88 9E  00      A        LDX   DOT      GET STARTING ADDRESS
01259A FE8A 4D                   TSTA           CHECK IF VALUE IS 1 OR 2 BYTES
01260A FE8B 26  0A      FE97     BNE   SCCMD2   BRANCH IF 2 BYTE KEY
01262A FE8D E1  80      A SCCMD1 CMPB  ,X+      DO SINGLE BYTE COMPARE
01263A FE8F 27  11      FEA2     BEQ   SCCMD3   BRANCH IF KEY FOUND
01264A FE91 AC  E4      A        CMPX  ,S       AT END?
01265A FE93 26  F8      FE8D     BNE   SCCMD1   BRANCH IF NOT
01266A FE95 35  C0      A        PULS  U,PC     CLEAN STACK AND RETURN
01268A FE97 10A3 80     A SCCMD2 CMPD  ,X+      DO DOUBLE BYTE COMPARE
01269A FE9A 27  06      FEA2     BEQ   SCCMD3   BRANCH IF KEY FOUND
01270A FE9C AC  E4      A        CMPX  ,S       AT END?
01271A FE9E 26  F7      FE97     BNE   SCCMD2   BRANCH IF NOT
01272A FEA0 35  C0      A        PULS  U,PC     CLEAN STACK AND RETURN
01274A FEA2 30  1F      A SCCMD3 LEAX  -1,X     ADJUST X
01275A FEA4 1F  10      A        TFR   X,D      PUT X IN D
01276A FEA6 32  62      A        LEAS  2,S      CLEAN STACK
01277A FEA8 16  FD25 FBD0        LBRA  DSPDOT   DISPLAY AND SET NEW DOT
01279                   * TABLES, CONSTANTS AND STRINGS
01281                   * DEFAULT PC TARGET
01282A FEAB 3F          NOGO     SWI
01284                   * TITLE STRING
01285A FEAC     43      A TITLE  FCC   /CMS 9609 DEBUG/
01286A FEBA     00      A        FCB   0
01288                   * PROMPT STRING
01289A FEBB     2A      A PROMPT FCC   /* /
01290A FEBD     00      A        FCB   0
01292                   * REGISTER DUMP STRING
01293A FEBE     20      A REGSTR FCC   / SP      CC       A  B DP/
       A FED4   20      A        FCC   /  X     Y     U     PC  OP . ../
       A FEF1   00      A        FCB   0
01297                   * ERROR MESSAGE
01298A FEF2     07      A ERRMSG FCB   $07      BEEP
01299A FEF3     45      A        FCC   /ERR /
01300A FEF7     00      A        FCB   0
01303                   * COMMAND DISPATCH TABLE
01305                   * TABLE BUILDER MACRO
01306                   CMDENT MACR
01307                    FCB \0
01308                    FCB \1-CMDTBL
01309                   ENDM
01311
01312A FEF8     FEF8    A CMDTBL EQU   *
01313A FEFB                      CMDENT '.,DOTCMD
01314A FEFE                      CMDENT '=,CNGMEM
01315A FF01                      CMDENT $0D,DOTFWD
01316A FF04                      CMDENT $20,CNVCMD
01317A FF07                      CMDENT '-,DOTBAK
01318A FF0A                      CMDENT ':,REGCMD
01319A FF0D                      CMDENT 'X,XCMD
01320A FF10                      CMDENT 'M,MEMCMD
01321A FF13                      CMDENT 'C,CCMD
01322A FF16                      CMDENT 'B,BCMD
01323A FF19                      CMDENT 'G,GOCMD
01324A FF1C                      CMDENT 'N,SSCMD
01325A FF1F                      CMDENT 'R,RCMD
01326A FF22     00      A        CMDENT 'S,SCCMD
                                 FCB   0
01328
01330                   *        TTL   INPUT/OUTPUT ROUTINES
01331                   * I/O ROUTINES FOR INTERACTIVE DEBUGGER
01332                   *
0133?                   * STAND-ALONE VERSION
01335
01336                   * DIRECT PAGE DECLARATIONS
01337A 000C     FF23    A SAVORG SET   *
                                 ORG   DPORG
01339A 000C     0001    A NULCNT RMB   1        NULL COUNT
01340A 000D     0001    A ECHO   RMB   1        ECHO FLAG (0=OFF)
01341A 000E     0002    A ACVECT RMB   2        ACIA VECTOR
01342A 0010     0002    A SW3VCT RMB   2        SWI3 VECTOR
01343A 0012     0002    A SW2VCT RMB   2        SWI2 VECTOR
01344A 0014     0002    A SWIVCT RMB   2        SWI VECTOR
01345A 0016     0002    A IRQVCT RMB   2        IRQ VECTOR
01346A 0018     0002    A FRQVCT RMB   2        FIRQ VECTOR
01347A 001A     0002    A NMIVCT RMB   2        NMI VECTOR
01349
01350A FF23     001C    A DPORG  SET   *
                                 ORG   SAVORG
01353                   * I/O INITIALIZATION
01354A FF23 AE  8D 00C5 IOINIT   LDX   ACADDR,PCR GET ACIA ADDRESS
01355A FF27 9F  0E      A        STX   ACVECT   INIT VECTOR
01356A FF29 86  03      A        LDA   #$3      ACIA MASTER RESET
01357A FF2B 97  0D      A        STA   ECHO     TURN ECHO ON
01358A FF2D A7  84      A        STA   ,X       MASTER RESET ACIA
01359A FF2F A6  8D 00BB          LDA   ACINIT,PCR GET ACIA INIT VALUE
01360A FF33 A7  84      A        STA   ,X       INITIALIZE IT
01361A FF35 A6  8D 00B6          LDA   ACNULL,PCR GET INITIAL NULL COUNT
01362A FF39 97  0C      A        STA   NULCNT
01363A FF3B AE  8D 00A8          LDX   TIMER,PCR GET TIMER ADDRESS
01364A FF3F A6  8D 00A8          LDA   TMINIT,PCR GET INIT VALUE
01365A FF43 A7  01      A        STA   1,X      INIT TIMER
01366A FF45 10AE 8D 009F         LDY   TIMOUT,PCR GET TIMOUT VALUE
01367A FF4A 10AF 04     A        STY   4,X      SET VALUE INTO TIMER
01368A FF4D 39                   RTS            DONE
01370
01371                   * SUBROUTINE WRTLIN - OUTPUT LINE WITH
                        * CARRIAGE RETURN
01373A FF4E 86  0D      A WRTLIN LDA   #$D      PUT EOL CHAR
01374A FF50 A7  80      A        STA   ,X+      AT END OF BUFFER
01375A FF52 34  10      A        PSHS  X        SAVE BUFFER END ADDR
01376A FF54 9E  06      A        LDX   IOBUF    GET BUF BEG ADDR
01377A FF56 A6  80      A WRTLN2 LDA   ,X+      GET CHAR
01378A FF58 8D  46      FFA0     BSR   OUTCHR   OUTPUT IT
01379A FF5A AC  E4      A        CMPX  ,S       DONE YET?
01380A FF5C 25  F8      FF56     BLO   WRTLN2   LOOP IF NOT
01381A FF5E 32  62      A        LEAS  2,S      POP END
01382A FF60 9E  06      A        LDX   IOBUF    RESTORE BUFFER ADDR
01383A FF62 20  25      FF89     BRA   LFNULL   GO FINISH UP
01385
01386                   * SUBROUTINE RDLIN - READ INPUT LINE.
01387                   * PROCESS BACKSPACE AND LINE DELETE.
01388                   * RETURNS DATA IN BUFFER, OR C BIT
                        * SET IF LINE DELETE
01390A FF64 9E  06      A RDLIN  LDX   IOBUF    GET BUFFER ADDR
01391A FF66 8D  46      FFAE RDLN2 BSR INPCHR   GET A CHARACTER
01392A FF68 81  08      A        CMPA  #$08     BACKSPACE?
01393A FF6A 26  08      FF74     BNE   RDLN3    SKIP IF NOT
01394A FF6C 9C  06      A        CPX   IOBUF    DONT BACKSPACE...
01395A FF6E 23  F6      FF66     BLS   RDLN2    PAST BEG OF LINE
01396A FF70 30  1F      A        LEAX  -1,X     ELSE DECR BUFFER PTR
01397A FF72 20  F2      FF66     BRA   RDLN2
01398A FF74 81  18      A RDLN3  CMPA  #$18     LINE DELETE?
01399A FF76 26  04      FF7C     BNE   RDLN4    SKIP IF NOT
01400A FF78 1A  01      A        ORCC  #1       WITH C BIT SET
01401A FF7A 20  0A      FF86     BRA   RDLN5
01402A FF7C A7  80      A RDLN4  STA   ,X+      PUT CHAR IN BUFFER
01403A FF7E 81  0D      A        CMPA  #$D      WAS IT EOL?
01404A FF80 26  E4      FF66     BNE   RDLN2    IF NOT GET ANOTHER
01405A FF82 8D  05      FF89     BSR   LFNULL   ELSE ECHO LINE FEED
01406A FF84 1C  FE      A        ANDCC #$FE     CLEAR CARRY...
01407A FF86 9E  06      A RDLN5  LDX   IOBUF    RESTORE POINTER
01408A FF88 39                   RTS            ... AND RETURN
01410
01411                   * SUBROUTINE LFNULL
                        * OUTPUT LINE FEED AND NULLS
01412A FF89 86  0A      A LFNULL LDA   #$0A     LINE FEED
01413A FF8B 8D  13      FFA0     BSR   OUTCHR   OUTPUT IT
01414A FF8D 4F                   CLRA
01415A FF8E D6  0C      A        LDB   NULCNT   GET NULL COUNT
01416A FF90 27  05      FF97     BEQ   LFNUL3   EXIT IF ZERO
01417A FF92 8D  0C      FFA0 LFNUL2 BSR OUTCHR  OUTPUT THE NULL
01418A FF94 5A                   DECB           DECR COUNT
01419A FF95 26  FB      FF92     BNE   LFNUL2   LOOP TIL DONE
01420A FF97 39             LFNUL3 RTS           THAT'S ALL FOLKS
01422
01423                   * SUBROUTINE WRTSTR -
01424                   * OUTPUT A STRING POINTED TO BY
01425                   * Y UNTIL A ZERO BYTE IS FOUND.
                        * DO NOT PRINT CR, LF, ETC.
01426A FF98 A6  A0      A WRTSTR LDA   ,Y+      GET CHAR
01427A FF9A 27  EA      FF86     BEQ   RDLN5    EXIT IF ZERO BYTE
01428A FF9C 8D  02      FFA0     BSR   OUTCHR   OUTPUT IT
01429A FF9E 20  F8      FF98     BRA   WRTSTR   LOOP TIL FINISHED
01431                   * CHARACTER I/O ROUTINES
01433                   * OUTPUT CHARACTER TO ACIA
01434A FFA0 34  16      A OUTCHR PSHS  D,X
01435A FFA2 9E  0E      A        LDX   ACVECT   GET VECTOR ADDRESS
01436A FFA4 E6  84      A OUTCH2 LDB   ,X       READ ACIA STATUS
01437A FFA6 C4  02      A        ANDB  #$02     WAIT FOR READY
01438A FFA8 27  FA      FFA4     BEQ   OUTCH2
01439A FFAA A7  01      A        STA   1,X      PUT CHAR IN OUTPUT REG
01440A FFAC 35  96      A        PULS  D,X,PC
01442
01443                   * INPUT CHARACTER FROM ACIA
                        * CHARACTER RETURNED IN ACC A
01444A FFAE 34  10      A INPCHR PSHS  X        SAVE XREG
01445A FFB0 9E  0E      A        LDX   ACVECT   GET ACIA ADDRESS
01446A FFB2 A6  84      A INPCH2 LDA   ,X       READ ACIA STATUS REGISTER
01447A FFB4 84  01      A        ANDA  #$01     MASK READY BIT
01448A FFB6 27  FA      FFB2     BEQ   INPCH2   WAIT TIL READY
01449A FFB8 A6  01      A        LDAA  1,X      GET CHAR
01450A FFBA 84  7F      A        ANDA  #$7F     MASK OUT PARITY
01451A FFBC 35  10      A        PULS  X        RESTORE X REG
01452A FFBE 0D  0D      A        TST   ECHO
01453A FFC0 26  DE      FFA0     BNE   OUTCHR   OUTPUT IT IF ECHO ON
01454A FFC2 39                   RTS            ELSE RETURN
01456                   *
01457                   *
01458                   *
01459                   * INTERRUPT SERVICE ROUTINES
01460                   *
01461           00E7    A DPAG   EQU   $E7      EQUATE FOR DIRECT PAGE
01463                   * MACRO TO GENERATE INTERRUPT DISPATCH
01464                   * USING DIRECT PAGE ADDRESS WITHOUT
01465                   * DISTURBING REGISTERS
01467                   INTDSP MACR
01468                    JMP [(DPAG*$100)+\0]
01469                   ENDM
01471                   * INTERRUPT DISPATCH ROUTINES
01473A FFC3             SW3SVC INTDSP SW3VCT
01474A FFC7             SW2SVC INTDSP SW2VCT
01475A FFCB             SWISVC INTDSP SWIVCT
01476A FFCF             FRQSVC INTDSP FRQVCT
01477A FFD3             NMISVC INTDSP NMIVCT
01478A FFD7             IRQSVC INTDSP IRQVCT
01479
01481
01482                   * SUBROUTINE TO INITIALIZE SWI
                        * VECTOR FOR BREAKPOINTS
01484A FFDB 109F 14     A SETSWI STY   SWIVCT
01485A FFDE 39                   RTS
01487                   *
01488                   *
01489                   *        SUBROUTINE TO INITIALIZE NMI VECTOR
01490A FFDF 109F 1A     A SETNMI STY   NMIVCT
01491A FFE2 39                   RTS
```

```
01493                           *
01494                           *
01495                           * SYSTEM DEFINITION TABLE
01496                           *
01497                           *
01498A FFE6                             ORG     $FFE6

01500A FFE6     E7      A DIRPAG FCB    DPAG    DEBUGGER DIRECT PAGE
01501A FFE7     E3E0    A TIMER  FDB    $E3E0   PTM ADDRESS
01502A FFE9     000C    A TIMOUT FDB    12      CYCLES TO EXECUTE BEFORE NMI
01503A FFEB     43      A TMINIT FCB    $43     VALUE TO INIT TIMER WITH
01504A FFEC     E3C0    A ACADDR FDB    $E3C0   ACIA ADDRESS
01505A FFEE     16      A ACINIT FCB    $16     ITS INITIALIZATION CONSTANT
01506A FFEF     00      A ACNULL FCB    0       NULL COUNT

01508                           * HARDWARE INTERRUPT VECTORS

01510A FFF0     0000    A       FDB     0       RESERVED
01511A FFF2     FFC3    A       FDB     SW3SVC
01512A FFF4     FFC7    A       FDB     SW2SVC
01513A FFF6     FFCF    A       FDB     FRQSVC
01514A FFF8     FFD7    A       FDB     IRQSVC
01515A FFFA     FFCB    A       FDB     SWISVC
01516A FFFC     FFD3    A       FDB     NMISVC
01517A FFFE     FB1F    A       FDB     START


01519                           END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```