ACCESS NAMES TABLE

```
SOURCE ACCESS NAME=      PPC2.P359.V100880.SRC.PARSE
OBJECT ACCESS NAME=      PPC2.P359.V100880.OBJ.PARSE2
LISTING ACCESS NAME=     PPC2.P359.V100880.LST.PARSES
ERROR ACCESS NAME=       .XMAERR
OPTIONS=                 XREF
MACRO LIBRARY PATHNAME=
```

```
LINE      KEY   NAME

 0002      A    VERSION
                    =>PPC2.P359.V100880.SRC.P359
```

```
0034                    IDT  'PARSE'
0035                ********************************************************************
0036                *                                                                  *
0037                *       PPPPPP     AAAAA    RRRRRR      SSSSSS     EEEEEEE        *
0038                *       P     P   A     A   R     R    S      S    E              *
0039                *       P     P   A     A   R     R    S           E              *
0040                *       PPPPPP    AAAAAAA   RRRRRR      SSSSSS     EEEEE          *
0041                *       P         A     A   R  R              S    E              *
0042                *       P         A     A   R   R             S    E              *
0043                *       P         A     A   R    R     S      S    E              *
0044                *       P         A     A   R     R     SSSSSS     EEEEEEE        *
0045                *                                                                  *
0046                *                                                                  *
0047                *         PPPPPP       33333      555555     99999                 *
0048                *         P     P     3     3     5         9     9                *
0049                *         P     P           3     5         9     9                *
0050                *         PPPPPP       33333     55555      999999                *
0051                *         P                3          5          9                *
0052                *         P                3          5          9                *
0053                *         P          3     3     5     5    9     9                *
0054                *         P           33333       5555      99999                 *
0055                *                                                                  *
0056                ********************************************************************
0060                *       SELECT WHICH GROMs PARSES CAN BE DONE FROM
0061                        ASMIF VERS=DX10
0062                *       GROMs 0,1,2 and 3
0063                EGROM   EQU  0                  ASSEMBLED FOR SIMULATOR
0064                        ASMELS
0065                *       GROMs 4,5,6 and 7
0066        0001    EGROM   EQU  1                  ASSEMBLED FOR EGROM BOX
0067                        ASMEND
```

```
                *                    BASIC PARSE CODE
                *
                *       REGISTER USAGE
                *            RESERVED FOR GPL INTERPRETER
                *               R13, R14, R15
                *               R13 CONTAINS THE READ ADDRESS FOR GROM
                *               R14 IS USED IN BASSUP/10 FOR THE VDPRAM POINTE
                *            RESERVED IN BASIC SUPPORT
                *               R8  MSBy CURRENT CHARACTER (LIKE CHAT IN GPL)
                *               R8  LSBy ZERO
                *               R9  STACK POINTER PAD+@STKADD
                *               R10 READ DATA PORT ADDRESS FOR PROGRAM DATA.
                *       ALL EXITS TO GPL MUST GO THROUGH "NUDGO5"
                *
                *
                *
                       DEF  PARSEG,CONTG,EXECG,RTNG,EOSTMT,EOLINE
                       DEF  VPUSH,VPOP,PSHPRS,EOL,NUDEND,ERRMUV
                       DEF  PGMCHR,PGMSUB,ERR1
                       DEF  EXEC50,PARSE,CONT,CONTIN
                       DEF  ERRSYN,ERR,CALGPL,VPSH23,VPOP20
                       DEF  LTRUE,LTST90,WARN$$,ARGTST
                       DEF  CBH65,CBH67,C3,C4,C16,C24,ERRS0,GOTO90
0000
                       REF  GRMRA,GRMWA,HHUSC,R7LB,VDPRD,VDPWD,WRVDP
                       REF  CFI,NSTRCN,COMPCT
                       REF  SCOMPB,SADD,SSUB,SMULT,SDIV
                       REF  GROMA,STVDP3
                       REF  SYM,SMB,GETV,GETV1,MOVFAC
                       REF  RESET,ASSG
                       REF  ERRT
                       REF  R1LB,R6LB,R9LB,R11LB
                       REF  STREND,ERRCOD,STVSPT,RTNADD,NUDTAB,PGMPTR
                       REF  EXTRAM,STLN,ENLN,PRGFLG,FLAG
                       REF  STKEND,FAC,FAC2,FAC4,FAC6,FAC7,FAC10
                       REF  FAC12,ERRCOD,PGMPT1,ARG
                       REF  VRAM,PAD,VSPTR,CHAT,STKADD
                       REF  O.OR,O.AND,O.XOR,O.NOT,NLPR,NMINUS,NPLUS
                       REF  NABS,NATN,NCOS,NEXP,NINT,NLOG,NSGN,NSIN
                       REF  NSQR,NTAN,LEXP,NFOR,NNEXT
                       REF  SMTSRT,CSN01,GETCH,GETCGR,GET1
                       REF  GRAM,RAMFLG,GROM
                       REF  CALL,SUBXIT
                *------------------CONDITIONAL ASSEMBLY--------------------*
                       ASMIF VERS=DX10
                       DEF  SAVREG,SAVRE2,SETREG
                C3     DATA 3
                       ASMELS
                       REF  SAVREG,SAVRE2,SETREG
                       ASMEND
                *------------------CONDITIONAL ASSEMBLY--------------------*
                *
0081            ELSE$  EQU  >81                ELSE TOKEN
0082            SSEP$  EQU  >82                STMT SEPARATOR
0083            TREM$  EQU  >83                TAIL REMARK
0084            IF$    EQU  >84                IF
0085            GO$    EQU  >85                GO TOKEN
0086            GOTO$  EQU  >86                GOTO TOKEN
0087            GOSUB$ EQU  >87                GOSUB TOKEN
008E            BREAK$ EQU  >8E                BREAK TOKEN
```

```
0129                0096  NEXT$   EQU  >96          NEXT TOKEN
0130                00A1  SUB$    EQU  >A1          SUB TOKEN
0131                00A5  ERROR$  EQU  >A5          ERROR TOKEN
0132                00A6  WARN$   EQU  >A6          WARNING TOKEN
0133                00B0  THEN$   EQU  >B0          THEN TOKEN
0134                00B1  TO$     EQU  >B1          TO TOKEN
0135                00B3  COMMA$  EQU  >B3          COMMA TOKEN
0136                00BE  EQ$     EQU  >BE          EQUAL TOKEN
0137                00C0  GT$     EQU  >C0          GREATER THAN TOKEN
0138                00C2  MINUS$  EQU  >C2          MINUS TOKEN
0139                00C4  DIVI$   EQU  >C4          DIVIDE TOKEN
0140                00C7  STRIN$  EQU  >C7          STRING TOKEN
0141                00C9  LN$     EQU  >C9          LINE NUMBER TOKEN
0142                      *
0143 0000 0018      C24     DATA 24                 CONSTANT 24
0144 0002 0142'     EXRTNA  DATA EXRTN              RETURN FOR EXEC
0145                      *
0146 0004 0200      ERRSO   LI   R0,>0703           Issue STACK OVERFLOW message
     0006 0703
0147 0008 0460           B    @ERR
     000A 01EE'
```

```
0149                    *
0150                    *       GRAPHICS LANGUAGE ENTRY TO PARSE
0151                    *
0152 000C 06A0  PARSEG BL   @SETREG            Set up registers for BASIC
     000E 0000
0153                    *------------------CONDITIONAL ASSEMBLY----------------------*
0154                           ASMIF VERS=DX10
0155                           S    @GROMA,R15         Get relative GROM address
0156                           DRI  R15,>8000          Mark as GROM address
0157                           MOV  R15,R11            Move for common code
0158
0159                           ASMELS
0160 0010
0161 0010 D2ED                 MOVB @GRMRA(R13),R11    Get GROM address
     0012 0000
0162 0014 D82D                 MOVB @GRMRA(R13),@R11LB Both bytes
     0016 0012'
     0018 0000
0163 001A
0164                           ASMIF EGROM
0165 001A 0608                 DEC  R11
0166                           ASMELS
0167                           AI   R11,>7FFF
0168                           ASMEND
0169 001C
0170                           ASMEND
0171                    *-------------END OF CONDITIONAL ASSEMBLY-------------------*
0172                    *
0173                    *               9900 ENTRY TO PARSE
0174                    *
0175 001C 05C9  PARSE  INCT R9                  Get room for return address
0176 001E 0289         CI   R9,STKEND           Stack full ?
     0020 0000
0177 0022 13F0         JH   ERRSO               Yes - too many levels deep
0178 0024 C64B         MOV  R11,*R9             Save the return address
0179 0026 D1C8  P05    MOVB R8,R7               Test for token beginning
0180 0028 1102         JLT  P10                 If token-look it up
0181 002A 0460         B    @PSYM               If not token is a symbol
     002C 0420'
0182                    *
0183 002E 06A0  P10    BL   @PGMCHR             Get next character
     0030 081A'
0184 0032 0977         SRL  R7,7                Change last char to offset
0185 0034 0227         AI   R7,->B7*2           Check for legal NUD
     0036 FE92
0186 0038 0287         CI   R7,NTABLN           Within the legal NUD address?
     003A 0056
0187 003C 1B22         JH   CONT15              No-check for legal LED
0188 003E C1E7         MOV  @NTAB(R7),R7        Get NUD address
     0040 05A4'
0189 0042 1525         JGT  B9900               If 9900 code
0190      0044' P17    EQU  $                   R7 contains offset into nudtab
0191                    *                          plus >8000
0192 0044 0247         ANDI R7,>7FFF            If GPL code, get rid of MSBit
     0046 7FFF
0193 0048 A1E0         A    @NUDTAB,R7          Add in table address
     004A 0000
0194 004C       NUDGO5
0195 004C 06A0         BL   @SAVREG             Restore GPL pointers
     004E 0000
```

```
0196                *----------------------CONDITIONAL ASSEMBLY-------------------*
0197                        ASMIF VERS=DX10
0198                        AI    R7,GROM           Add in GROM base for return
0199                        MOV   R7,R15            Move for use by GPL interp
0200                        ASMELS
0201 0050 D847              MOVB  R7,@GRMWA(R13)    Write out new GROM address
     0052 0000
0202 0054 06C7              SWPB  R7                Bare the LSBytes
0203 0056 D847              MOVB  R7,@GRMWA(R13)    Put it out too
     0058 0052'
0204                        ASMEND
0205                *----------------END OF CONDITIONAL ASSEMBLY-----------------*
0206 005A 0460              B     @RESET            Go back to GPL interp
     005C 0000
0207 005E 10F2  P17L        JMP   P17
```

```
0210                    *
0211                    *           CONTINUE ROUTINE FOR PARSE
0212                    *
0213 0060 06A0  CONTG   BL    @SETREG           GPL entry-set BASIC registers
     0062 00CE'
0214 0064       CONT
0215 0064 C199          MOV   *R9,R6            Get last address from stack
0216 0066 15C6          JGT   CONT10            9900 code if not negative
0217                    *--------------------CONDITIONAL ASSEMBLY--------------------*
0218                          ASMIF VERS=DX10
0219                          ANDI R6,>7FFF     Get GROM address
0220                          AI    R6,GROM     Add in GROM offset
0221
0222                          ASMELS
0223 0068
0224                          ASMIF EGROM
0225                          ASMELS
0226                          ANDI R6,>7FFF
0227                          ASMEND
0228 0068
0229 0068 D846          MOVB  R6,@GRMWA(R13)    Write out new GROM address
     006A 0058'
0230 006C 06C6          SWPB  R6                Bare the second byte
0231 006E D846          MOVB  R6,@GRMWA(R13)    Put it out too
     0070 006A'
0232 0072 C18D          MOV   R13,R6            Set up to test precedence
0233                          ASMEND
0234                    *------------------END OF CONDITIONAL ASSEMBLY--------------*
0235 0074
0236                    *
0237 0074 9216  CONT10  CB    *R6,R8            Test precedence
0238 0076 1411          JHE   NUDNDL            Have parsed far enough->return
0239 0078 0978          SRL   R8,7              Make into table offset
0240 007A 0228          AI    R8,->88*2         Minimum token for a LED (*2)
     007C FE90
0241 007E 0288          CI    R8,LTBLEN         Maximum token for a LED (*2)
     0080 001C
0242 0082 1B09  CONT15  JH    NOLEDL            If outside legal LED range-err
0243 0084 C1E8          MOV   @LTAB(R8),R7      Pick up address of LED handler
     0086 05FA'
0244 0088 04C8          CLR   R8                Clear 'CHAT' for getting new
0245 008A 06A0          BL    @PGMCHR           Get next token
     008C 081A'
0246 008E 0457  B9900   B     *R7               Go to the LED handler
0247 0090
0248 0090 0649  NUDE10  DECT  R9                Back up subroutine stack
0249                          ASMIF EGROM
0250 0092 0587          INC   R7                Skip over precedence
0251                          ASMELS
0252                          AI    R7,>8001    Get rid of GROM flag and
0253                    *                           skip over precedence
0254                          ASMEND
0255 0094 10DB          JMP   NUDG05            Goto code to return to GPL
0256 0096 0460  NOLEDL  B     @NOLED
     0098 01EA'
0257 009A 1073  NUDNDL  JMP   NUDND1
```

```
00C0 00BC
0261                *          Execute one or more lines of BASIC
0262 00BC
0263       00BC' EXECG  EQU   $                    GPL entry point for execution
0264 00BC 06A0          BL    @SETREG             Set up registers
     00BE 00E2'
0265 00C0 04E0          CLR   @ERRCOD             Clear the return code
     00A2 0000
0266 00A4 D020          MOVB  @PRGFLG,R0          Imperative statement?
     00A6 0000
0267 00A8 131A          JEQ   EXEC15              Yes-handle it as such
0268 00AA
0269                *          Loop for each statement in the program
0270 00AA
0271       00AA' EXEC10 EQU   $
0272 00AA D020          MOVB  @FLAG,R0            Now test for trace mode
     00AC 0000
0273 00AE 0A30          SLA   R0,3                Check the trace bit in FLAG
0274 00B0 115F          JLT   TRACL               If set->display line number
0275 00B2 C820  EXEC11  MOV   @EXTRAM,@PGMPTR     Get text pointer
     00B4 0000
     00B6 0000
0276 00B8 0660          DECT  @PGMPTR             Back to the ln # to check bkpt
     00BA 00B6'
0277 00BC 06A0          BL    @PGMCHR             Get the first byte of ln #
     00BE 081A'
0278 00C0 02C0          STST  R0                  Save status for breakpt check
0279 00C2 05A0          INC   @PGMPTR             Get text pointer again
     00C4 00BA'
0280 00C6 06A0          BL    @PGMCHR             Go get the text pointer
     00C8 081A'
0281 00CA 06C8          SWPB  R8                  Save 1st byte of text pointer
0282 00CC 06A0          BL    @PGMCHR             Get 2nd byte of text pointer
     00CE 081A'
0283 00D0 06C8          SWPB  R8                  Put text pointer in order
0284 00D2 C808          MOV   R8,PGMPTR           Set new text pointer
     00D4 00C4'
0285 00D6 04C8          CLR   R8                  Clean up the mess
0286 00D8 0A20          SLA   R0,2                Check breakpoint status
0287 00DA 1101          JLT   EXEC15              If no breakpoint set - cont
0288 00DC 177A          JNC   BRKPNT              If breakpoint set-handle it
0289       00DE' EXEC15 EQU   $
0290                *----------------------CONDITIONAL ASSEMBLY-----------------*
0291                       ASMIF VERS=DX10
0292
0293                       ASMELS
0294 00DE
0295                       REF   C1000
0296       00E0' C3        EQU   $+2                 Constant data 3
0297       00E1' CB3       EQU   $+3                 Constant byte 3
0298 00DE 0300          LIMI  3                   Let interrupts loose
     00E0 0003
0299       00E4' C0        EQU   $+2                 Constant data 0
0300 00E2 0300          LIMI  0                   Shut down interrupts
     00E4 0000
0301 00E6 04E0          CLR   @>83D6              Reset VDP timeout
     00E8 83D6
0302                *      CB    @>83C2,@HHUSC       Check shift-C on HHU
0303                *      JEQ   BRKP1L              Got one-handle the break
0304 00EA 020C          LI    R12,>24             Load console KBD addr in CRU
```

```
      00EC 00C4
0305  00EE 30E0          LDCR @C0,3              Select keyboard section
      00F0 00E4'
0306  00F2 020C          LI   R12,6              Read address
      00F4 0006
0307  00F6 3600          STCR R0,8               SCAN the keyboard
0308  00F8 2420          CZC  @C1000,R0          Shift-key depressed?
      00FA 0000
0309  00FC 160A          JNE  EXEC16             No-execute the BASIC stmt
0310  00FE 020C          LI   R12,>24            Test column 3 of keyboard
      0100 0024
0311  0102 30E0          LDCR @CB3,3             Select keyboard section
      0104 00E1'
0312  0106 020C          LI   R12,6              Read address
      0108 0006
0313  010A 3600          STCR R0,8               SCAN the keyboard
0314  010C 2420          CZC  @C1000,R0          Shift-C depressed ?
      010E 00FA'
0315  0110 132E          JEQ  BRKP1L             Yes-so take BASIC breakpoint
0316                      ASMEND
0317                *----------------END OF CONDITIONAL ASSEMBLY-------------*
0318  0112 C820    EXEC16 MOV  @PGMPTR,@SMTSRT   Save start of statement
      0114 00D4'
      0116 0000
0319  0118 05C9          INCT R9                 Get subroutine stack space
0320  011A C660          MOV  @EXRTNA,*R9        Save the GPL return address
      011C 0002'
0321  011E 06A0          BL   @PGMCHR            Now get 1st character of stmt
      0120 081A'
0322  0122 1320          JEQ  EXRTN3             If EOL after EOS
0323  0124 1102    EXEC17 JLT  EXEC20             If top bit set->keyword
0324  0126 0460          B    @NLET              If not->fake a 'LET' stmt
      0128 04E4'
0325                *
0326  012A C1C8    EXEC20 MOV  R8,R7              Save 1st token so can get 2nd
0327  012C 05A0          INC  @PGMPTR            Increment the perm pointer
      012E 0114'
0328                *----------------CONDITIONAL ASSEMBLY---------------------*
0329                      ASMIF VERS=DX10
0330                      MOVB *R10+,R8           Read the character
0331                      ASMELS
0332  0130 D21A           MOVB *R10,R8            Read the character
0333                      ASMEND
0334                *----------------END OF CONDITIONAL ASSEMBLY---------------*
0335  0132 0977          SRL  R7,7               Convert 1st to table offset
0336  0134 0227          AI   R7,->AA*2          Check for legal stmt token
      0136 FEAC
0337  0138 1558          JGT  ERR1               Not in range -> error
0338  013A C1E7          MOV  @STMTTB(R7),R7     Get address of stmt handler
      013C 05A2'
0339  013E 118F          JLT  P17L               If top bit set -> GROM code
0340  0140 0457          B    *R7                If 9900 code - goto it!
0341                *
0342  0142    83    EXRTN  BYTE >83               Unused bytes for data constant
0343  0143    65    CBH65  BYTE >65                since NUDEND skips precedence
0344  0144-0288          CI   R8,SSEP$*256       EOS only?
      0146 8200
0345  0148 13CA          JEQ  EXEC15             Yes-continue on this line
0346  014A D020    EXRTN2 MOVB @PRGFLG,R0         Did we execute an imperative?
      014C 00A6'
```

```
0347 014E 1351          JEQ  EXEC50          Yes - so return to top-level
0348 0150 6820          S    @04,@EXTRAM     No - so goto the next line
     0152 0626'
     0154 0034'
0349 0156 8820          C    @EXTRAM,@STLN   Check to see if end of program
     0158 0154'
     015A 0000
0350 015C 144B          JHE  EXEC10          No - so loop for the next line
0351 015E 1049          JMP  EXEC50          Yes - so return to top-level
0352                *
0353                *       Stmt handler for ::
0354                *
0355 0160 D208  SMTSEP MOVB R8,R8            EOL?
0356 0162 16E0         JNE  EXEC17           NO-there is another stmt
0357 0164 0649  EXRTN3 DECT R9               YES
0358 0166 10F1         JMP EXRTN2            Jump back into it
0359 0168
0360                *       Continue after a breakpoint
0361 0168 06A0  CONTIN BL   @SETREG          Set up BASIC registers
     016A 009E'
0362 016C 1038  EXC1SL JMP  EXEC15           Continue execution
0363 016E
0364 016E 1038  BRKP1L JMP  BRKPN1
0365 0170 104E  TRACL  JMP  TRACE
```

```
                    *       Test for required end-of-statement
0396 0172 D808  EOL     MOVB R8,R8              EOL reached?
0397 0174 1306          JEQ  NUDND1             Yes
0398 0176-0288          CI   R8,TREM$*256       Higher than tail remark token?
     0178 8300
0399 017A 1327          JH   ERR1               Yes-its an error
0400 017C-0288          CI   R8,ELSE$*256       Tail, ssep or else?
     017E 8100
0401 0180 1A34          JL   ERR1               No-error
0402                *
0403                *           Return from call to PARSE
0404                *             (entered from CONT)
0405 0182
0406 0182 C1D9  NUDND1  MOV  *R9,R7             Get the return address
0407 0184 1185          JLT  NUDE10             If negative - return to GPL
0408 0186 0649          DECT R9                 Back up the subroutine stack
0409 0188 0467          B    @2(R7)             And return to caller
     018A 0002
0410                *                                  (Skip the precedence word)
0411 018C
0412 018C
0413 018C D208  NUDEND  MOVB R8,R8              Check for EOL
0414 018E 13F9          JEQ  NUDND1             If EOL
0415 0190-0288  NUDND2  CI R8,STRIN$*256        Lower than a string?
     0192 C700
0416 0194 1A08          JL   NUDND4             Yes
0417 0196-0288          CI   R8,LN$*256         Higher than a line # ?
     0198 C900
0418 019A 1315          JEQ  SKPLN              Skip line numbers
0419 019C 1A08          JL   SKPSTR             Skip string or numeric
0420 019E 06A0  NUDND3  BL   @PGMCHR            Read next character
     01A0 081A'
0421 01A2 13EF          JEQ  NUDND1             If EOL
0422 01A4 10F5          JMP  NUDND2             Continue scan of line
0423 01A6-0288  NUDND4  CI   R8,TREM$*256       Higher than a tail remark?
     01A8 8300
0424 01AA 1BF9          JH   NUDND3             Yes
0425 01AC-0288          CI   R8,SSEP$*256       Lower than stmt sep(else)?
     01AE 8200
0426 01B0 1AF6          JL   NUDND3             Yes
0427 01B2 10E7          JMP  NUDND1             TREM or SSEP
0428                *
0429 01B4 06A0  SKPSTR  BL   @PGMCHR
     01B6 081A'
0430 01B8 06C8          SWPB R8                 Prepare to add
0431 01BA A808          A    R8,@PGMPTR         Skip it
     01BC 012E'
0432 01BE 04C8          CLR  R8                 Clear lower byte
0433 01C0 06A0  SKPS01  BL   @PGMCHR            Get next token
     01C2 081A'
0434 01C4 10E3          JMP  NUDEND             Go on
0435                *
0436 01C6 05E0  SKPLN   INCT @PGMPTR            Skip line number
     01C8 01BC'
0437 01CA 10FA          JMP  SKPS01             Go on
0438                *
0439                *             Return from "CALL" to GPL
0440 01CC
0441 01CC 06A0  RTNG    BL   @SETREG            Set up registers again
     01CE 016A'
```

```
0415 C1C0 1008          JMP   NUDND1            And jump back into it!
0416                    ********************************************************************
```

```
01D2
                    *               Handle Breakpoints
01D2
01D2 D020  BRKPNT MOVB @FLAG,R0           Check flag bits.
     01D4 00A3'
01D6 0A10         SLA  R0,1               Check bit 6 for breakpoint.
01D9 1109         JLT  EXEC15L            If set then ignore breakpoint.
01DA              BRKPN2
01DA 0200         LI   R0,BRKFL           Breakpoint return vector.
     01DC 0001
01DE 1007         JMP  EXIT               Return to top-level
                  *
01E0              BRKPN1
01E0 D020         MOVB @FLAG,R0           Move flag bits.
     01E2 01D4'
01E4 0A10         SLA  R0,1               Check bit 6 for breakpoint.
01E6 1195         JLT  EXEC16             If set then ignore breakpoint.
01E8 10F8         JMP  BRKPN2             Bit not set.
                  *
                  *                   Error handling from 9900 code
                  *
          01EA'   ERRSYN EQU  $             These all issue same message
01EA              ERR1
01EA              NONUD
01EA              NOLED
01EA 0200         LI   R0,ERRSN           *SYNTAX ERROR return code
     01EC 0003
01EE              EXIT
01EE C800  ERR    MOV  R0,@ERRCOD         Load up return code for GPL
     01F0 00A2'
01F2
01F2
                  *           General return to GPL portion of BASIC
01F2
01F2 C1E0  EXEC50 MOV  @RTNADD,R7         Get return address
     01F4 0000
01F6 0460         B    @NUDGO5            Use common code to link back
     01F8 004C'
01FA
                  *           Handle STOP and END statements
01FA
01FA              STOP
01FA 0649  END    DECT R9                 Pop last call to PARSE
01FC 10FA         JMP  EXEC50             Jump to return to top-level
01FE
01FE
                  *                 Error codes for return to GPL
01FE
          0003    ERRSN  EQU  >0003           ERROR SYNTAX
          0103    ERROM  EQU  >0103           ERROR OUT OF MEMORY
          0203    ERRIOR EQU  >0203           ERROR INDEX OUT OF RANGE
          0303    ERRLNF EQU  >0303           ERROR LINE NOT FOUND
          0403    ERREX  EQU  >0403           ERROR - EXECUTION
                  *                       >0004 WARNING NUMERIC OVERFLOW
          0001    BRKFL  EQU  >0001           BREAKPOINT RETURN VECTOR
                         REF  C2          >0002 TRACE MODE RETURN VECTOR
          0005    ERROR  EQU  >0005           ON ERROR
          0006    UDF    EQU  >0006           FUNCTION REFERENCE
          0007    BREAK  EQU  >0007           ON BREAK
          0008    CONCAT EQU  >0008           CONCATENATE (&) STRINGS
```

```
0472        0009  WARN   EQU  >0009              ON WARNING
0473 01FE
0474 01FE
0475                *          Warning routine (only OVERFLOW)
0476                *
0477 01FE C820  WARN$$ MOV  @C4,@ERRCOD       Load warning code for GPL
     0200 00E6'
     0202 01F0'
0478 0204 0208        LI   R11,CONT-2        To optimize for return
     0206 0062'
0479 0208
0480 0208
0481                *          Return to GPL as a CALL
0482 0208
0483 0208 05C9  CALGPL INCT R9                Get space on subroutine stack
0484 020A C64B        MOV  R11,*R9            Save return address
0485 020C 10F2        JMP  EXEC50             And go to GPL
0486 020E
0487 020E
0488                *          Trace a line (Call GPL routine)
0489                *
0490 020E C820  TRACE  MOV  @C2,@ERRCOD       Load return vector
     0210 0000
     0212 0202'
0491 0214 0208        LI   R11,EXEC11-2       Set up for return to execute
     0216 00B0'
0492 0218 10F7        JMP  CALGPL             Call GPL to display line #
0493 021A
0494                *          Special code to handle concatenate (&)
0495 021A 0200  CONC   LI   R0,CONCAT         Go to GPL to handle it
     021C 0008
0496 021E 10E7        JMP  EXIT               Exit to GPL interp
```

```
0527                    *
0528                    *
0529                    *       ON ERROR, ON WARN and ON BREAK
0530                    *
0531 0260 0200  ONERR   LI    R0,ERROR          ON ERROR code
     0262 0005
0532 0264 1004          JMP   EXIT              Return to GPL code
0533                    *
0534 0266 0200  ONWARN  LI    R0,WARN           ON WARN code
     0268 0009
0535 026A 10C1          JMP   EXIT              Return to GPL code
0536                    *
0537 026C 0200  ONBRK   LI    R0,BREAK          ON BREAK code
     026E 0007
0538 0270 10BE          JMP   EXIT              Return to GPL code
0539                    *
0540                    *
0541 0272
0542                    *
0543                    *       NUD routine for "GO"
0544                    *
0545 0272 0403  GO      CLR   R3                Dummy "ON" index for common
0546 0274 1020          JMP   ON30              Merge into "ON" code
0547 0276
0548                    *
0549                    *
0550                    *       NUD ROUTINE FOR "ON"
0551                    *
0552 0276        ON
0553 0276-0288          CI    R8,WARN$*256      On warning?
     0278 A600
0554 027A 13F5          JEQ   ONWARN            Yes. goto ONWARN
0555 027C-0288          CI    R8,ERROR$*256     On error?
     027E A500
0556 0280 13EF          JEQ   ONERR             Yes. goto ONERR
0557 0282-0288          CI    R8,BREAK$*256     On break?
     0284 8E00
0558 0286 13F2          JEQ   ONBRK             Yes goto ONBRK
0559                    *
0560                    *       Normal "ON" statement
0561                    *
0562 0288 06A0          BL    @PARSE            PARSE the index value
     028A 001C'
0563 028C   B3          BYTE  COMMA$            Stop on a comma or less
0564 028D   66  CBH66   BYTE  >66               Unused byte for a constant
0565 028E 06A0          BL    @NUMCHK           Ensure index is a number
     0290 0738'
0566 0292 04E0          CLR   @FAC10            Assume no error in CFI
     0294 0258'
0567 0296 06A0          BL    @CFI              Convert Floating to Integer
     0298 0000
0568 029A D020          MOVB  @FAC10,R0         Test error code
     029C 0294'
0569 029E 1603          JNE   GOTO90            If overflow -> BAD VALUE
0570 02A0 C0E0          MOV   @FAC,R3           Get the index
     02A2 0000
0571 02A4 1503          JGT   ON20              Must be positive
0572 02A6 0200  GOTO90  LI    R0,ERRIOR         Negative -> BAD VALUE
     02A8 0203
0573 02AA 10A1  GOTO95  JMP   ERR               Jump to error handler
```

```
0577                    *
0578          02A0'     ON20    EQU   $                 Now check GO TO/SUB
0579 02A0-02B8          ON40    CI    R8,GO$*256        Bare "GO"  token?
     02A2 8500
0580 02A4 160E                  JNE   ON40              No - check other possibilities
0581 02A6 06A0                  BL    @PGMCHR           Yes - get next token
     02A8 081A'
0582 02AA-02B8          ON30    CI    R8,TO$*256        "GO TO" ?
     02AC 8100
0583 02AE 1065                  JEQ   GOTO50            Yes handle GO TO like GOTO
0584 02B0-02B8                  CI    R8,SUB$*256       "GO SUB" ?
     02B2 A100
0585 02B4 1005                  JMP   ON50              Merge to common code to test
0586                    *
0587 02B6-02B8          ON40    CI    R8,GOTO$*256      "GOTO" ?
     02B8 8600
0588 02BA 135F                  JEQ   GOTO50            Yes - go handle it
0589 02BC-02B8                  CI    R8,GOSUB$*256     "GOSUB" ?
     02BE 8700
058A 02C0 160E          ON50    JNE   ERR1              No - so is an error
058B 02C2 06A0                  BL    @PGMCHR           Get next token
     02C4 081A'
058C 02C6 1002                  JMP   GOSUB2            Goto gosub code
058D                    *
058E 02C8 10BA          ERR1B   JMP   ERR1              Issue error message
```

```
02D6
                    *           NUD routine for "GOSUB"
02D6
02D6 0400   GOSUB   CLR  R0                  Dummy index for "ON" code
                    *             Common GOSUB code
02D8
02D8'       GOSUB2  EQU  #                   Now build a FAC entry
02D8 0201           LI   R1,FAC              Optimize to save bytes
     02DA 02A2'
02DC C040           MOV  R0,*R1+             Save the "ON" index
                    *                        in case of garbage collection
02DE DC60           MOVB @CBH66,*R1+         Indicate GOSUB entry on stack
     02E0 028D'
02E2 0581           INC  R1                  Skip FAC3
02E4 C460           MOV  @PGMPTR,*R1         Save current ptr w/in line
     02E6 024E'
02E8 05F1           INCT *R1+                Skip line # to correct place
02EA C460           MOV  @EXTRAM,*R1         Save current line # pointer
     02EC 0158'
02EE 06A0           BL   @VPUSH              Save the stack entry
     02F0 0730'
02F2 C0E0           MOV  @FAC,R3             Restore the "ON" index
     02F4 02DA'
02F6 1C01           JMP  GOTO20              Jump to code to find the line
02F8
02F8
                    *             NUD routine for "GOTO"
02F8
02F8 04C3   GOTO    CLR  R3                  Dummy index for "ON" code
02FA
                    *     Common (ON) GOTO/GOSUB THEN/ELSE code to find line
02FA
02FA        GOTO20
                    *           Get line number from program
02FA
02FA-0288           CI   R8,LN$*256          Must have line number token
     02FC C900
02FE 16EA           JNE  ERR1B               Don't - so error
0300 06A0   GETL10  BL   @PGMCHR             Get MSByte of the line number
     0302 081A'
0304 D008           MOVB R8,R0               Save it
0306 06A0           BL   @PGMCHR             Read the character
     0308 081A'
030A 0603           DEC  R3                  Decrement the "ON" index
030C 1534           JGT  GOTO40              Loop if not there yet
030E
                    *             Find the program line
                    *
                    *--------------------CONDITIONAL ASSEMBLY--------------------
                          ASMIF VERS=DX10
                          MOV  @STLN,R1      Get end of line number table
                          MOVB @RAMFLG,R2    See if ERAM or VDP
                          JEQ  GOTO31        VDP - so search it
                          MOV  R1,R2         Copy the address
                          AI   R2,GRAM       Add in ERAM offset
                          JMP  GOTO32        And goto search code

                    GOTO31 MOV  R1,R2        Copy the address
                          AI   R2,VRAM       Add in RAM offset
```

```
0644
0645                GOTO32 C     R1,@ENLN          End of program?
0646                       JHE   GOTO34            Yes- line # is not in program
0647                       MOVB  *R2+,R3           Save in temporary place for
0648                *                              bkpt checking
0649                       ANDI  R3,>7FFF          Reset possible bkpt
0650                       CB    R3,R0             Compare 1st byte of # - Match?
0651                       JNE   GOTO35            Not a match - so move on
0652                       CB    *R2+,R8           2nd byte match?
0653                       JEQ   GOTO36            Yes!! The line is found
0654                GOTO33 MOVB  *R2+,R3           NO - skip 1st byte of line ptr
0655                       AI    R1,4              Advance to next # in table
0656                       MOVB  *R2+,R3           Skip 2nd byte of line pointer
0657                       JMP   GOTO32            Check next # in table
0658
0659                GOTO35 MOVB  *R2+,R3           Skip 2nd byte of line number
0660                       JMP   GOTO33
0661
0662                       ASMELS
0663 030E
0664 030E C0A0             MOV   @STLN,R1          Get into line # table
     0310 015A'
0665 0312 D0A0             MOVB  @RAMFLG,R2        Check ERAM flag to see where?
     0314 023A'
0666 0316 1310             JEQ   GOTO31            From VDP - go handle it
0667 0318 C081             MOV   R1,R2             Copy address
0668 031A 8801  GOT32      C     R1,@ENLN          Finished w/line # table?
     031C 0000
0669 031E 1422             JHE   GOTO34            Yes - so line doesn't exist
0670 0320 D0F2             MOVB  *R2+,R3           Save in temp to check brkpt
0671 0322 0243             ANDI  R3,>7FFF          Reset possible bkpt
     0324 7FFF
0672 0326 9003             CB    R3,R0             Compare 1st byte of # - Match?
0673 0328 1605             JNE   GOT35             Not a match -so move on
0674 032A 9232             CB    *R2+,R8           2nd byte match?
0675 032C 131E             JEQ   GOTO36            Yes - line is found!
0676 032E
0677 032E 05C2  GOT33      INCT  R2                Skip line pointer
0678 0330 C042             MOV   R2,R1             Advance to next line in table
0679 0332 10F3             JMP   GOT32             Go back for more
0680 0334
0681 0334 D0F2  GOT35      MOVB  *R2+,R3           Skip 2nd byte of line #
0682 0336 10FB             JMP   GOT33             And jump back in
0683 0338
0684 0338 D7E0  GOTO31     MOVB  @R1LB,*R15        Get the data from the VDP
     033A 0000
0685 033C 0202             LI    R2,VDPRD          Load up to read data
     033E 0000
0686 0340 D7C1             MOVB  R1,*R15           Write out MSByte of address
0687 0342
0688 0342 8801  GOTO32     C     R1,@ENLN          Finished w/line # table?
     0344 031C'
0689 0346 140E             JHE   GOTO34            Yes - so line doesn't exist
0690 0348 D0D2             MOVB  *R2,R3            Save in temporary place for
0691                *                              bkpt checking
0692 034A 0243             ANDI  R3,>7FFF          Reset possible bkpt
     034C 7FFF
0693 034E 9003             CB    R3,R0             Compare 1st byte of # - Match?
0694 0350 1607             JNE   GOTO35            Not a match -so move on
0695 0352 9212             CB    *R2,R8            2nd byte match?
```

```
0696 0354 130A          JEQ   GOTO36           Yes - line is found!
0697 0356
0698 0356 D0D2  GOTO33  MOVB  *R2,R3           Skip 1st byte of line pointe
0699 0358 0221          AI    R1,4             Advance to next line in table
     035A 0004
0700 035C D0D2          MOVB  *R2,R3           Skip 2nd byte of line pointer
0701 035E 10F1          JMP   GOTO32           Go back for more
0702 0360
0703 0360 D0D2  GOTO35  MOVB  *R2,R3           Skip 2nd byte of line #
0704 0362 10F9          JMP   GOTO33           And jump back in
0705                    ASMEND
0706                    *--------------------END OF CONDITONAL ASSEMBLY---------------*
0707 0364 0200  GOTO34  LI    R0,ERRLNF        LINE NOT FOUND error vector
     0366 0303
0708 0368 10A0          JMP   GOTO95           Jump for error exit
0709                    *
0710 036A 05C1  GOTO36  INCT  R1               Adjust to line pointer
0711 036C C801          MOV   R1,@EXTRAM       Save for execution of the line
     036E 02EC'
0712 0370 0649          DECT  R9               Pop saved link to goto
0713 0372 0460          B     @EXEC10          Reenter EXEC code directly
     0374 00AA'
0714 0376
0715 0376 06A0  GOTO40  BL    @PGMCHR          Get next token
     0378 081A'
0716 037A 06A0          BL    @EOSTMT          Premature end of statement?
     037C 03FE'
0717 037E 1393          JEQ   GOTO90           Yes => BAD VALUE for index
0718 0380-0288          CI    R8,COMMA$*256    Comma next ?
     0382 B300
0719 0384 1603          JNE   ERR1C            No - error
0720 0386 06A0  GOTO50  BL    @PGMCHR          Yes - get next character
     0388 081A'
0721 038A 10B7          JMP   GOTO20           And check this index value
0722                    *
0723 038C 10A3  ERR1C   JMP   ERR1B            Linking because long-distance
0724 038E
0725 038E 0200  ERR51   LI    R0,>0903         RETURN WITHOUT GOSUB
     0390 0903
0726 0392 108B          JMP   GOTO95           Exit to GPL
```

```
0729                    *           NUD entry for "RETURN"
0730 0394
0731 0394 8820  RETURN C    @VSPTR,@STVSPT    Check bottom of stack
     0396 0000
     0398 0000
0732 039A 12F9          JLE  ERR51            Error -> RETURN WITHOUT GOSUB
0733 039C 06A0          BL   @VPOP            Pop entry
     039E 07D0'
0734 03A0 9820          CB   @CBH66,@FAC2     Check ID for a GOSUB entry
     03A2 028D'
     03A4 0000
0735 03A6 160B          JNE  RETU30           Check for ERROR ENTRY.
0736                    *
0737                    *       Have a GOSUB entry
0738                    *
0739 03A8 06A0          BL   @EOSTMT          Must have EOS after return
     03AA 03FE'
0740 03AC 16F3          JNE  RETURN           Not EOS. Then error return?
0741 03AE C820          MOV  @FAC4,@PGMPTR    Get return ptr w/in line
     03B0 0000
     03B2 02E6'
0742 03B4 C820          MOV  @FAC6,@EXTRAM    Get return line pointer
     03B6 0000
     03B8 036E'
0743 03BA 0460          B    @SKPS01          Go adjust it and get back.
     03BC 01C0'
0744 03BE          RETU30
0745                    *   Check ERROR entry.
0746 03BE 9820          CB   @CBH69,@FAC2     ERROR ENTRY?
     03C0 0641'
     03C2 03A4'
0747 03C4 1307          JEQ  RETU40           Yes. Take care of error entry.
0748 03C6 9820          CB   @CBH6A,@FAC2     Subprogram entry?
     03C8 03FC'
     03CA 03C2'
0749 03CC 16E3          JNE  RETURN           No.  Look some more.
0750 03CE 06A0          BL   @VPUSH           Push it back. Keep information.
     03D0 0750'
0751 03D2 10DD          JMP  ERR51            RETURN WITHOUT GOSUB error.
0752                    *
0753                    *       Have an ERROR entry.
0754                    *       RETURN, RETURN line #, RETURN or RETURN NEXT follows.
0755                    *
0756 03D4          RETU40
0757 03D4 04C3          CLR  R3               In case of a line number.
0758 03D6-0288          CI   R8,LN$*256       Check for a line number.
     03D8 C900
0759 03DA 1392          JEQ  GETL10           Yes. Treat like GOTO.
0760                    *
0761 03DC C820          MOV  @FAC4,@PGMPTR    Get return ptr w/in line.
     03DE 03B0'
     03E0 03B2'
0762 03E2 C820          MOV  @FAC6,@EXTRAM    Get return line pointer.
     03E4 03B6'
     03E6 03B8'
0763 03E8 06A0          BL   @EOSTMT          EOL now?
     03EA 03FE'
0764 03EC 1305          JEQ  BEXC15           Yes. Treat like GOSUB rtn.
0765                    *
0766 03EE-0288          CI   R8,NEXT$*256     NEXT now?
```

```
      03F0 9600
07e7  03F2 16CC          JNE  ERR1C              No, so its an error.
07e8  03F4 0460          B    @SKPS01            Yes. So execute next statem
      03F6 01C0'
07e9                     *
0770  03F8 0460  BEXC15  B    @EXEC15            Execute next line.
      03FA 000E'
0771  03FC    6A CBH6A   BYTE >6A                Subprogram call stack ID.
0772  03FE               EVEN
0773                     ********************************************************
0774                     *       EOSTMT - Check for end-of-statement
0775                     *
0776                     *       Returns with condition '=' if EOS
0777                     *            else condition '<>' if not EOS
0778                     ********************************************************
0779  03FE D208  EOSTMT  MOVB R8,R8              EOL or non-token?
0780  0400 1305          JEQ  EOSTM1             EOL-return condition '='
0781  0402 1504          JGT  EOSTM1             Non-token return cond '<>'
0782  0404-0288          CI   R8,TREM$*256       In the EOS range (>81 to >83
      0406 8300
0783  0408 1601          JH   EOSTM1             No-return condition '<>'
0784  040A 820B          C    R8,R8              Yes-force condition to '='
0785  040C 045B  EOSTM1  RT
0786                     *
0787                     *
0788                     ********************************************************
0789                     *       EOLINE - Tests for End Of Line; either a >00 or a
0790                     *                ':'
0791                     *       Returns with condition '=' if EOL else condition
0792                     *                '<>' if not EOL
0793                     ********************************************************
0794  040E D208  EOLINE  MOVB R8,R8              EOL?
0795  0410 1302          JEQ  EOLNE1             Yes-return with '=' set
0796  0412-0288          CI   R8,TREM$*256       Set condition on a tail rema
      0414 8300
0797  0416 045B  EOLNE1  RT                      And return
```

```
0800 0418 0200  SYMB20 LI    R0,UDF          Long distance
     041A 0005
0801 041C 04E0         B     @GOTO95
     041E 02AA'
0802 0420
0803 0420
0804                *        NUD for a symbol (variable)
0805 0420
0806 0420 06AC  PSYM   BL    @SYM            Get symbol table entry
     0422 0000
0807 0424 06A0         BL    @GETV           Get 1st byte of entry
     0426 0000
0808 0428 02F4'        DATA  FAC             SYM left pointer in FAC
0809 042A 0A11         SLA   R1,1            UDF reference?
0810 042C 11F5         JLT   SYMB20          Yes - special code for it
0811 042E 06A0         BL    @SMB            No - Get value space pointer
     0430 0000
0812 0432 9820         CB    @FAC2,@CBH65    String reference?
     0434 030A'
     0436 0143'
0813 0438 1303         JEQ   SYMB10          Yes - Special code for it
0814 043A 06A0         BL    @MOVFAC         No - numeric ->copy into FAC
     043C 0000
0815 043E 0460  SYMB10 B     @CONT           And continue the PARSE
     0440 0064'
```

```
                        *           Statement entry for IF statement
0818
0819 0442
0820 0442 06A0  IF      BL    @PARSE          Evaluate the expression
     0444 001C'
0821 0446   23         BYTE  COMMA$          Stop on a comma
0822 0447   67  C8H67  BYTE  >67             Unused byte for a constant
0823 0448 06A0         BL    @NUMCHK         Ensure the value is a number
     044A 0738'
0824 044C 0403         CLR   R3              Create a dummy "ON" index
0825 044E-0288         CI    R8,THEN$*256    Have a "THEN" token
     0450 B000
0826 0452 169C         JNE   ERR1C           No - error
0827 0454 0520         NEG   @FAC            Test if condition true i.e<>0
     0456 0428'
0828 0458 1610         JNE   IF$10           True - branch to the special #
0829 045A 06A0         BL    @PGMCHR         Advance to line number token
     045C 081A'
0830 045E-0288         CI    R8,LN$*256      Have the line # token?
     0460 C900
0831 0462 1619         JNE   IF$20           No-must look harder for ELSE
0832 0464 05E0         INCT  @PGMPTR         Skip the line number
     0466 03EC'
0833 0468 06A0         BL    @PGMCHR         Get next token
     046A 081A'
0834 046C-0288 IF$5    CI    R8,ELSE$*256    Test if token is ELSE
     046E 8100
0835 0470 1304         JEQ   IF$10           We do! So branch to the line #
0836 0472 0460         B     @EOL            We don't - so better be EOL
     0474 0172'
0837 0476 0460 GETL1$  B     @GETL10
     0478 0300'
0838                   *
0839 047A 06A0 IF$10   BL @PGMCHR            Get 1st token of clause
     047C 081A'
0840 047E-0288         CI  R8,LN$*256        Line # token?
     0480 C900
0841 0482 13F9         JEQ   GETL1$          Yes - go there
0842 0484 06A0         BL    @EOSTMT         EOS?
     0486 03FE'
0843 0488 1381 JEQ1C   JEQ ERR1C             Yes-its an error
0844 048A-0208         LI R8,SSEP$*256       Cheat to do a continue
     048C 8200
0845 048E 0620         DEC   @PGMPTR         Back up to get 1st char
     0490 0466'
0846 0492 0460         B     @CONT           Continue on
     0494 0064'
0847                   *
0848                   *    LOOK FOR AN ELSE CLAUSE SINCE THE CONDITION WAS FALSE
0849                   *
0850 0496 0203 IF$20   LI R3,1               IF/ELSE pair counter
     0498 0001
0851 049A 06A0         BL    @EOLINE         Trap out EOS following THEN/
     049C 040E'                                                      ELSE
0852                   *
0853 049E 13F4         JEQ   JEQ1C           error
0854 04A0-0288 IF$25   CI    R8,ELSE$*256    ELSE?
     04A2 8100
0855 04A4 1603         JNE   IF$27           If not
0856 04A6 0603         DEC   R3              Matching ELSE?
0857 04A8 13E8         JEQ   IF$10           Yes-do it
```

```
0859 044A 100F          JMP   IF$35             No-go on
0859 044C-0258  IF$27   CI    R8,IF$*256        Check for it
     044E 840C
0860 0450 1602          JNE   IF$28             Not an IF
0861 0452 0581          INC   R1                Increment nesting level
0862 0454 100A          JMP   IF$35               And go on
0863 0456-0288  IF$28   CI    R8,STRIN$*256     Lower than a string?
     0458 C700
0864 045A 1A04          JL    IF$30             Yes
0865 045C-0288          CI    R8,LN$*256        Higher or = to a line #
     045E C900
0866 0460 1307          JEQ   IF$40             = line #
0867 0462 1A09          JL    IF$50             Skip strings and numerics
0868 0464 06A0  IF$30   BL    @EOLINE           EOL?
     0466 040E'
0869 0468 13D1          JEQ   IF$5              Yes - done scanning
0870 046A 06A0  IF$35   BL    @PGMCHR           Get next character
     046C 081A'
0871 046E 10E8          JMP   IF$25               And go on
0872                *
0873                *       SKIP LINE #'S
0874                *
0875 0470 05E0  IF$40   INCT  @PGMPTR           Skip the line #
     0472 0490'
0876 0474 10FA          JMP    IF$35            Go on
0877                *
0878                *       SKIP STRINGS AND NUMERICS
0879                *
0880 0476 06A0  IF$50   BL    @PGMCHR           Get # of bytes to skip
     0478 081A'
0881 047A 06C8          SWPB  R8                Swap for add
0882 047C A808          A     R8,@PGMPTR        Skip it
     047E 0472'
0883 0480 04C8          CLR   R8                Clear lsb of R8
0884 0482 10F3          JMP   IF$35
0885                *
0886                *
```

```
0889                ***************************************************************
0890                *                  'LET' statement handler                    *
0891                *       Assignments are done by putting an entry on the       *
0892                *       stack for the destination variable and getting        *
0893                *       the source value into the FAC.  Multiple assingments  *
0894                *       are handled by stacking the variable entrys and then  *
0895                *       looping for the assingments.  Numeric assingments     *
0896                *       pose no problems, strings are more complicated.       *
0897                *       String assignments are done by assigning the source   *
0898                *       string to the last variable specified in the list     *
0899                *       and changing the FAC entry so that the string         *
0900                *       assigned to the next-to-the-last variable comes       *
0901                *       from the permanent string belonging to the variable   *
0902                *       just assigned.                                        *
0903                *       e.g.    A$,B$,C$="HELLO"                              *
0904                *                                                             *
0905                *            C$-------->HELLO (source string)                 *
0906                *                       !                                     *
0907                *                       V                                     *
0908                *            B$-------->HELLO (copy from C$'s string          *
0909                *                       !                                     *
0910                *                       V                                     *
0911                *            A$-------->HELLO (copy from B$'s string          *
0912                ***************************************************************
0913 04E4          NLET
0914 04E4 04E0             CLR   @PAD                  Counter for multiple assign's
     04E6 0000
0915 04E8 06A0     NLETO5  BL    @SYM                  Get symbol table address
     04EA 0422'
0916 04EC 06A0             BL    @GETV                 Get first byte of entry
     04EE 0426'
0917 04F0 0456'            DATA  FAC                   SYM left pointer in FAC
0918 04F2 0A11             SLA   R1,1                  Test if a UDF
0919 04F4 1110             JLT   ERRMUV                Is a UDF - so error
0920 04F6 06A0             BL    @SMB                  Get value space pointer
     04F8 0430'
0921 04FA 06A0             BL    @VPUSH                Push s.t. pointer on stack
     04FC 0750'
0922 04FE 05A0             INC   @PAD                  Count the variable
     0500 04E6'
0923 0502-0288             CI    R8,EQ$*256            Is the token an '='?
     0504 BE00
0924 0506 130B             JEQ   NLET10                Yes - go into assignment loop
0925 0508-0288             CI    R8,COMMA$*256         Must have a comma now
     050A B300
0926 050C 161E             JNE   ERR1C$                Didn't - so error
0927 050E 06A0             BL    @PGMCHR               Get next token
     0510 081A'
0928 0512 15EA             JGT   NLETO5                If legal symbol character
0929 0514 101A             JMP   ERR1C$                If not - error
0930 0516
0931 0516 0200     ERRMUV  LI    R0,>0D03              MULTIPLY USED VARIABLE
     0518 0D03
0932 051A 0460             B     @ERR
     051C 01EE'
0933 051E
0934 051E 06A0     NLET10  BL    @PGMCHR               Get next token
     0520 081A'
0935 0522 06A0             BL    @PARSE                PARSE the value to assign
     0524 001C'
```

```
0936 0526   33          BYTE TREM$              Parse to the end of statement
0937 0527   65  STCODE  BYTE >65                 Use the wasted byte
0938                    *       Loop for assignments
0939 0528 06AC  NLET15  BL   @ASSG              Assign the value to the symbol
     052A 0000
0940 052C 05E1          DEC  @PAD               One less to assign - done?
     052E 0300'
0941 0530 130A          JEQ  LETCON             Yes - branch out
0942 0532 9BE0          CB   @FAC2,@STCODE      String or numeric?
     0534 0434'
     0536 0527'
0943 0538 16F7          JNE  NLET15             Numeric - just loop for more
0944 053A C806          MOV  R6,@FAC4           Get pointer to new string
     053C 03DE'
0945 053E C820          MOV  @ARG,@FAC          Get pointer to last s.t. entry
     0540 0000
     0542 04F0'
0946 0544 10F1          JMP  NLET15             Now loop to assign more
0947 0546
0948 0546 0460  LETCON  B    @EOL               Yes - continue the PARSE
     0548 0172'
0949                    *
0950 054A 0460  ERR1C$  B    ERR1C              For long distance jump
     054C 038C'
```

```
0953                    *                STATEMENT TABLE
0954 054E
0955 054E 01EA'         DATA NONUD       (SPARE)     (80)
0956 0550 01EA'         DATA NONUD       ELSE        (81)
0957 0552 01E0'         DATA SMTSEP      :           (82)
0958 0554 01E3'         DATA NUDND1      !           (83)
0959 0556 0442'         DATA IF          IF          (84)
0960 0558 0272'         DATA GO          GO          (85)
0961 055A 02F8'         DATA GOTO        GOTO        (86)
0962 055C 02D6'         DATA GOSUB       GOSUB       (87)
0963 055E 0394'         DATA RETURN      RETURN      (88)
0964 0560 018C'         DATA NUDEND      DEF         (89)
0965 0562 018C'         DATA NUDEND      DIM         (8A)
0966 0564 01FA'         DATA END         END         (8B)
0967 0566 0000          DATA NFOR        FOR         (8C)
0968 0568 04E4'         DATA NLET        LET         (8D)
0969 056A 8002          DATA >8000+>02   BREAK       (8E)
0970 056C 8004          DATA >8000+>04   UNBREAK     (8F)
0971 056E 8006          DATA >8000+>06   TRACE       (90)
0972 0570 8008          DATA >8000+>08   UNTRACE     (91)
0973 0572 8016          DATA >8000+>16   INPUT       (92)
0974 0574 0182'         DATA NUDND1      DATA        (93)
0975 0576 8012          DATA >8000+>12   RESTORE     (94)
0976 0578 8014          DATA >8000+>14   RANDOMIZE   (95)
0977 057A 0000          DATA NNEXT       NEXT        (96)
0978 057C 800A          DATA >8000+>0A   READ        (97)
0979 057E 01FA'         DATA STOP        STOP        (98)
0980 0580 8032          DATA >8000+>32   DELETE      (99)
0981 0582 0182'         DATA NUDND1      REM         (9A)
0982 0584 0276'         DATA ON          ON          (9B)
0983 0586 800C          DATA >8000+>0C   PRINT       (9C)
0984 0588 0000          DATA CALL        CALL        (9D)
0985 058A 018C'         DATA NUDEND      OPTION      (9E)
0986 058C 8018          DATA >8000+>18   OPEN        (9F)
0987 058E 801A          DATA >8000+>1A   CLOSE       (A0)
0988 0590 01FA'         DATA STOP        SUB         (A1)
0989 0592 8034          DATA >8000+>34   DISPLAY     (A2)
0990 0594 0182'         DATA NUDND1      IMAGE       (A3)
0991 0596 8024          DATA >8000+>24   ACCEPT      (A4)
0992 0598 01EA'         DATA NONUD       ERROR       (A5)
0993 059A 01EA'         DATA NONUD       WARNING     (A6)
0994 059C 0000          DATA SUBXIT      SUBEXIT     (A7)
0995 059E 059C'         DATA SUBXIT      SUBEND      (A8)
0996 05A0 800E          DATA >8000+>0E   RUN         (A9)
0997 05A2 8010  STMTTB  DATA >8000+>10   LINPUT      (AA)
```

```
1000                    *                   NUD TABLE
1001 05A4
1002 05A4 0000  NTAB    DATA NLPR           LEFT PAREN  (B7)
1003 05A6 01EA'         DATA NONUD          CONCATENATE (B8)
1004 05A8 01EA'         DATA NONUD          SPARE       (B9)
1005 05AA 01EA'         DATA NONUD          AND         (BA)
1006 05AC 01EA'         DATA NONUD          OR          (BB)
1007 05AE 01EA'         DATA NONUD          XOR         (BC)
1008 05B0 0000          DATA O.NOT          NOT         (BD)
1009 05B2 01EA'         DATA NONUD          =           (BE)
1010 05B4 01EA'         DATA NONUD          <           (BF)
1011 05B6 01EA'         DATA NONUD          >           (CO)
1012 05B8 0000          DATA NPLUS          +           (C1)
1013 05BA 0000          DATA NMINUS         -           (C2)
1014 05BC 01EA'         DATA NONUD          *           (C3)
1015 05BE 01EA'         DATA NONUD          /           (C4)
1016 05C0 01EA'         DATA NONUD          ^           (C5)
1017 05C2 01EA'         DATA NONUD          SPARE       (C6)
1018 05C4 0000          DATA NSTRCN         QUOTED STRING(C7)
1019 05C6 0220'         DATA NUMCON         UNQUOTED STRING/NUMERIC(C8)
1020 05C8 01EA'         DATA NONUD          LINE NUMBER (C9)
1021 05CA 8026          DATA >8000+>26      EOF         (CA)
1022 05CC 0000          DATA NABS           ABS         (CB)
1023 05CE 0000          DATA NATN           ATN         (CC)
1024 05D0 0000          DATA NCOS           COS         (CD)
1025 05D2 0000          DATA NEXP           EXP         (CE)
1026 05D4 0000          DATA NINT           INT         (CF)
1027 05D6 0000          DATA NLOG           LOG         (DO)
1028 05D8 0000          DATA NSGN           SGN         (D1)
1029 05DA 0000          DATA NSIN           SIN         (D2)
1030 05DC 0000          DATA NSQR           SQR         (D3)
1031 05DE 0000          DATA NTAN           TAN         (D4)
1032 05E0 8036          DATA >8000+>36      LEN         (D5)
1033 05E2 8038          DATA >8000+>38      CHR$        (D6)
1034 05E4 803A          DATA >8000+>3A      RND         (D7)
1035 05E6 8030          DATA >8000+>30      SEG$        (D8)
1036 05E8 802A          DATA >8000+>2A      POS         (D9)
1037 05EA 802C          DATA >8000+>2C      VAL         (DA)
1038 05EC 802E          DATA >8000+>2E      STR         (DB)
1039 05EE 8028          DATA >8000+>28      ASC         (DC)
1040 05F0 801C          DATA >8000+>1C      PI          (DD)
1041 05F2 8000          DATA >8000+>00      REC         (DE)
1042 05F4 801E          DATA >8000+>1E      MAX         (DF)
1043 05F6 8020          DATA >8000+>20      MIN         (EO)
1044 05F8 8022          DATA >8000+>22      RPT$        (E1)
1045      0056  NTABLN EQU  $-NTAB
```

LED TABLE

```
1048                    *
104F 05FA
1050 05FA               LTAB
1051 05FA 021A'                 DATA CONC               %       (B8)
1052 05FC 01EA'                 DATA NOLED              SPARE   (B9)
1053 05FE 0001                  DATA 0.OR               OR      (BA)
1054 0600 0000                  DATA 0.AND              AND     (BB)
1055 0602 0000                  DATA 0.XOR              XOR     (BC)
1056 0604 01EA'                 DATA NOLED              NOT     (BD)
1057 0606 0634'                 DATA EQUALS             =       (BE)
1058 0608 0616'                 DATA LESS               <       (BF)
1059 060A 0624'                 DATA GREATR             >       (C0)
1060 060C 06C4'                 DATA PLUS               +       (C1)
1061 060E 06F0'                 DATA MINUS              -       (C2)
1062 0610 06FC'                 DATA TIMES              *       (C3)
1063 0612 0708'                 DATA DIVIDE             /       (C4)
1064 0614 0000                  DATA LEXP               ^       (C5)
1065      001C          LTBLEN  EQU  $-LTAB
```

```
1068                ********************************************************
1069                *                  Relational operators                *
1070                *                                                      *
1071                *     Logical comparisons encode the type of comparison *
1072                *     and use common code to PARSE the expression and set *
1073                *     the status bits.                                 *
1074                *                                                      *
1075                *     The types of legal comparisons are:              *
1076                *        0  EQUAL                                      *
1077                *        1  NOT EQUAL                                  *
1078                *        2  LESS THAN                                  *
1079                *        3  LESS OR EQUAL                              *
1080                *        4  GREATER THAN                               *
1081                *        5  GREATER OR EQUAL                           *
1082                *                                                      *
1083                *     This code is saved on the subroutine stack       *
1084                ********************************************************
1085 0616            LESS
1086 0616 0202           LI   R2,2                 LESS-THAN code for common rtn
     0618 0002
1087 061A-0288           CI   R8,GT$*256           Test for '>' token
     061C C000
1088 061E 1604           JNE  LT10                 Jump if not
1089 0620 0642           DECT R2                   Therefore, NOT-EQUAL code
1090 0622 1005           JMP  LT15                 Jump to common
1091 0624            GREATR
1092      0626'      C4    EQU  $+2                  Constant 4
1093 0624 0202           LI   R2,4                 GREATER-THAN code for common
     0626 0004
1094 0628-0288       LT10  CI   R8,EQ$*256          Test for '=' token
     062A BE00
1095 062C 1605           JNE  LTSTO1               Jump if '>='
1096 062E 06A0       LT15  BL   @PGMCHR             Must be plain old '>' or '<'
     0630 081A'
1097 0632 1001           JMP  LEDLE                Jump to test
1098                *
1099 0634            EQUALS
1100 0634 07C2           SETO R2                   Equal bit for common routine
1101 0636 0582       LEDLE INC  R2                  Sets to zero
1102 0638            LTSTO1
1103 0638 05C9           INCT R9                   Get room on stack for code
1104 063A C642           MOV  R2,*R9               Save status matching code
1105 063C 06A0           BL   @PSHPRS              Push 1st arg and PARSE the 2nd
     063E 0742'
1106 0640   C0           BYTE GT$                  Parse to a '>'
1107 0641   69       CBH69 BYTE >69                Used in RETURN routine.
1108 0642 C119           MOV  *R9,R4               Get the type code from stack
1109 0644 0649           DECT R9                   Reset subroutine stack pointer
1110 0646 D324           MOVB @LTSTAB(R4),R12      Get address bias to branch to
     0648 0680'
1111 064A 088C           SRA  R12,8                Right justify
1112 064C 06A0           BL   @ARGTST              Test for matching arguments
     064E 0714'
1113 0650 131A           JEQ  LTST20               Handle strings specially
1114 0652 06A0           BL   @SCOMPB              Floating point comparison
     0654 0000
1115 0656 046C       LTST15 B    @LTSTXX(R12)      Interpret the status by code
     0658 065A'
1116 065A            LTSTXX
1117 065A 1504       LTSTGE JGT  LTRUE             Test if GREATER or EQUAL
```

```
1118 0650 1303  LTSTEQ JEQ  LTRUE          Test if EQUAL
1119 0652 0404  LFALSE CLR  R4             FALSE is a ZERO
1120 0654 1003         JMP  LTST90         Put it into FAC
1121 0656 13FD  LTSTNE JEQ  LFALSE         Test if NOT-EQUAL
1122 0658 0204  LTRUE  LI   R4,>BFFF       TRUE is a minus-one
     065A BFFF
1123 065C 0200  LTST90 LI   R3,FAC         Store result in FAC
     065E 0542'
1124 0660 CCC4         MOV  R4,*R3+        Exp & 1st byte of manitissa
1125 0662 04F3         CLR  *R3+           ZERO the remaining digits
1126 0664 04F3         CLR  *R3+           ZERO the remaining digits
1127 0666 04F3         CLR  *R3+           ZERO the remaining digits
1128 0668 1039         JMP  LEDEND         Jump to end of LED routine
1129 066A
1130 066A 13F6  LTSTLE JEQ  LTRUE          Test LESS-THAN or EQUAL
1131 066C 11F5  LTSTLT JLT  LTRUE          Test LESS-THAN
1132 066E 10F1         JMP  LFALSE         Jump to false
1133 0670 15F3  LTSTGT JGT  LTRUE          Test GREATER-THAN
1134 0672 10EF         JMP  LFALSE         Jump to false
1135 0674
1136                   *          Data table for offsets for types
1137                   *
1138 0674   02  LTSTAB BYTE LTSTEQ-LTSTXX  EQUAL            (0)
1139 0675   08         BYTE LTSTNE-LTSTXX  NOT EQUAL        (1)
1140 0676   1E         BYTE LTSTLT-LTSTXX  LESS THAN        (2)
1141 0677   1C         BYTE LTSTLE-LTSTXX  LESS OR EQUAL    (3)
1142 0678   22         BYTE LTSTGT-LTSTXX  GREATER THAN     (4)
1143 0679   00         BYTE LTSTGE-LTSTXX  GREATER OR EQUAL(5)
```

```
1145                    *               STRING COMPARISON
1146  0686
1147  0686              LTST20
1148  0686 C2A0                 MOV   @FAC4,R10       Pointer to string1
      0688 0830'
1149  068A D1E0                 MOVB  @FAC7,R7        R7 =  string2 length
      068C 0000
1150  068E 06A0                 BL    @VPOP           Get LH arg back
      0690 07D0'
1151  0692 C120                 MOV   @FAC4,R4        Pointer to string2
      0694 0838'
1152  0696 D1A0                 MOVB  @FAC7,R6        R6 =  string2 length
      0698 083C'
1153  069A D146                 MOVB  R6,R5           R5 will contain shorter length
1154  069C 91C6                 CB    R6,R7           Compare the 2 lengths
1155  069E 1101                 JLT   CSTR05          Jump if length2 < length1
1156  06A0 D147                 MOVB  R7,R5           Swap if length1 > length2
1157  06A2 0985   CSTR05  SRL   R5,8                 Shift for speed and test zero
1158  06A4 130D                 JEQ   CSTR20          If ZERO-set status with length
1159  06A6 C0CA   CSTR10  MOV   R10,R3                Current character location
1160  06A8 058A                 INC   R10             Increment pointer
1161  06AA 06A0                 BL    @GETV1          Get from VDP
      06AC 0000
1162  06AE D001                 MOVB  R1,R0           And save for comparison
1163  06B0 C0C4                 MOV   R4,R3           Current char location in ARG
1164  06B2 0584                 INC   R4              Increment pointer
1165  06B4 06A0                 BL    @GETV1          Get from VDP
      06B6 06AC'
1166  06B8 9001                 CB    R1,R0           Compare the characters
1167  06BA 16CD                 JNE   LTST15          Return with status if <>
1168  06BC 0605                 DEC   R5              Otherwise, decrement counter
1169  06BE 15F3                 JGT   CSTR10          And loop for each character
1170  06C0
1171  06C0 91C6   CSTR20  CB    R6,R7           Status set by length compare
1172  06C2 10C9                 JMP   LTST15          Return to do test of status
```

```
1175                    *                ARITHMETIC FUNCTIONS
1176 06C4
1177 06C4 06A0  PLUS    BL    @PSHPRS    Push left arg and PARSE righ
     06C6 0742'
1178 06C8   C2          BYTE  MINUS$,0   Stop on a minus!!!!!!!!!!!!
     06C9   00
1179 06CA 0202          LI    R2,SADD    Address of add routine
     06CC 0000
1180 06CE
1181 06CE 04E0  LEDEX   CLR   @FAC10     Clear error code
     06D0 029C'
1182 06D2 06A0          BL    @ARGTST    Make sure both numerics
     06D4 0714'
1183 06D6 132E          JEQ   ARGTO5     If strings - error
1184 06D8 06A0          BL    @SAVREG    Save registers
     06DA 004E'
1185 06DC 0692          BL    *R2        Do the operation
1186 06DE 06A0          BL    @SETREG    Restore registers
     06E0 0248'
1187 06E2 D0A0          MOVB  @FAC10,R2  Test for overflow
     06E4 06D0'
1188 06E6 1602          JNE   LEDERR     If overflow ->error
1189 06E8 0460  LEDEND  B     @CONT      Continue the PARSE
     06EA 0064'
1190                    *
1191 06EC 0460  LEDERR  B     @WARN$$    Overflow - issue warning
     06EE 01FE'
1192                    *
1193                    *
1194 06F0 06A0  MINUS   BL    @PSHPRS    Push left arg and PARSE righ
     06F2 0742'
1195 06F4   C2          BYTE  MINUS$,0   Parse to a minus
     06F5   00
1196 06F6 0202          LI    R2,SSUB    Address of subtract routine
     06F8 0000
1197 06FA 10E9          JMP   LEDEX      Common code for the operatio
1198                    *
1199                    *
1200 06FC 06A0  TIMES   BL    @PSHPRS    Push left arg and PARSE righ
     06FE 0742'
1201 0700   C4          BYTE  DIVI$,0    Parse to a divide!!!!!!!!!!!
     0701   00
1202 0702 0202          LI    R2,SMULT   Address of multiply routine
     0704 0000
1203 0706 10E3          JMP   LEDEX      Common code for the operatio
1204                    *
1205                    *
1206 0708 06A0  DIVIDE  BL    @PSHPRS    Push left arg and PARSE righ
     070A 0742'
1207 070C   C4          BYTE  DIVI$,0    Parse to a divide
     070D   00
1208 070E 0202          LI    R2,SDIV    Address of divide routine
     0710 0000
1209 0712 10DD          JMP   LEDEX      Common code for the operatio
```

```
1212                ********************************************************************
1213                *     Test arguments on both the stack and in the FAC              *
1214                *                  Both must be of same type                       *
1215                *       CALL:                                                      *
1216                *            BL   @ARGTST                                          *
1217                *            JEQ          If string                                *
1218                *            JNE          If numeric                               *
1219                ********************************************************************
1220 0714           ARGTST
1221 0714 C1A0             MOV  @VSPTR,R6          Get stack pointer
     0716 0396'
1222 0718 0586             INCT R6
1223                *--------------------CONDITIONAL ASSEMBLY-------------------*
1224                        ASMIF VERS=DX10
1225                        AI   R6,VRAM            Get character from stack
1226                *                               >65 => STRING (IN R8)
1227                *                               >65 => STRING (IN R4)
1228                        CB   *R6,@CBH65         String in operand 1?
1229
1230                        ASMELS
1231 071A
1232 071A D7E0             MOVB @R6LB,*R15         Load 2nd byte of stack address
     071C 0000
1233 071E 1000             NOP                     Kill some time
1234 0720 D7C6             MOVB R6,*R15            Load 1st byte of stack address
1235 0722 1000             NOP                     Kill some time
1236 0724 9820             CB   @VDPRD,@CBH65      String in operand 1?
     0726 033E'
     0728 0143'
1237                        ASMEND
1238                *---------------END OF CONDITIONAL ASSEMBLY----------------*
1239 072A 1606             JNE  ARGT10             No - numeric
1240 072C 9820             CB   @FAC2,@CBH65       Yes - is other the same?
     072E 0534'
     0730 0143'
1241 0732 1306             JEQ  ARGT20             Yes - do string comparison
1242 0734 0460  ARGT05 B    @ERRT                   Data types don't match
     0736 0000
1243 0738
1244 0738
1245 0738          NUMCHK
1246 0738 9820  ARGT10 CB   @FAC2,@CBH65       2nd operand can't be string
     073A 072E'
     073C 0143'
1247 073E 13FA             JEQ  ARGT05             If so - error
1248 0740 045B  ARGT20 RT                          OK so return with status
```

```
1250                *       Subroutine to set up registers on entry from GPL
1251 0742
1252.
1253                *--------------------CONDITIONAL ASSEMBLY----------------
1254                        ASMIF VERS=DX10
1255                SETREG  CLR   R8                LSByte must be cleared
1256                        MOVB  @CHAT,R2          Get current character
1257                        MOVB  @STKADD,R9        Get subroutine stack address
1258                        SRL   R9,8              Shift for use
1259                        AI    R9,PAD            Add in base offset
1260                        RT                      And return
1261
1262                        ASMELS
1263 0742
1264                *       USE CONSOLE ROUTINE FOR SPEED
1265 0742
1266                        ASMEND
1267                *---------------END OF CONDITIONAL ASSEMBLY----------------*
1268                *
1269                *
1270                *       Subroutine to restore GPL memory locations
1271                *         Also used to save R8 and R9 for calls
1272                *                 to floating point
1273 0742
1274                *--------------------CONDITIONAL ASSEMBLY--------------------*
1275                        ASMIF VERS=DX10
1276                SAVREG  MOVB  R8,@CHAT          Put current char in for GPL
1277                SAVRE2  S     @PADAD,R9         Calculate current stack addr
1278                        MOVB  @R9LB,@STKADD     Put in for GPL
1279                        RT                      And return
1280
1281                PADAD   DATA PAD                Data word for subtract
1282
1283                        ASMELS
1284 0742
1285                *       USE CONSOLE ROUTINE FOR SPEED
1286                *SAVREG  MOVB  R8,@CHAT          Put current char in for GPL
1287                *SAVRE2  AI    R9,NEGPAD        Calculate current stack addr
1288                *        MOVB  @R9LB,@STKADD     Save it for GPL
1289                *        RT                      And return
1290                        ASMEND
                    *---------------END OF CONDITIONAL ASSEMBLY----------------
```

```
1292                    *              VPUSH followed by a PARSE
1293 0742
1294 0742 0589  PSHPRS  INCT R9                  Get room on stack
1295 0744 0289          CI   R9,STKEND           Stack full?
     0746 0020'
1296 0748 1541          JH   VPSH27              Yes - error
1297 074A C648          MOV  R11,*R9             Save return on stack
1298 074C 020B          LI   R11,PC5             Optimize for the parse
     074E 0026'
1299 0750
1300                    *                 Stack VPUSH routine
1301 0750
1302 0750               VPUSH
1303 0750 0200          LI   R0,8                Pushing 8-byte entries
     0752 0008
1304 0754 A800          A    R0,@VSPTR           Update the pointer
     0756 0716'
1305 0758 C060          MOV  @VSPTR,R1           Now get the new pointer
     075A 0756'
1306                    *----------------------CONDITIONAL ASSEMBLY-------------------*
1307                    ASMIF VERS=DX10
1308                    MOV  R1,R14              Move it for the write
1309                    AI   R14,VRAM            Add in the VDP offset
1310                    LI   R1,FAC              Source is FAC
1311                    VPSH15 MOVB *R1+,*R14+   Move a byte
1312
1313                    ASMELS
1314 075C
1315 075C D7E0          MOVB @R1LB,*R15          Write new address to VDP chip
     075E 033A'
1316 0760 0261          ORI  R1,WRVDP            Enable the write
     0762 0000
1317 0764 D7C1          MOVB R1,*R15             Write 1st byte of address
1318 0766 0201          LI   R1,FAC              Source is FAC
     0768 066A'
1319 076A D831  VPSH15  MOVB *R1+,@VDPWD         Move a byte
     076C 0000
1320                    ASMEND
1321                    *-----------------END OF CONDITIONAL ASSEMBLY----------------*
1322 076E 0600          DEC  R0                  Decrement the count  - Done?
1323 0770 15FC          JGT  VPSH15              No - more to move
1324 0772 C00B          MOV  R11,R0              Save the return address
1325 0774 9820          CB   @FAC2,@CBH65        Pushing a string entry?
     0776 073A'
     0778 0143'
1326 077A 160E          JNE  VPSH20              No - so done
1327 077C C1A0          MOV  @VSPTR,R6           Entry on stack
     077E 075A'
1328 0780 0226          AI   R6,4                Pointer to the string is here
     0782 0004
1329 0784 C060          MOV  @FAC,R1             Get the string's owner
     0786 0768'
1330 0788 0281          CI   R1,>001C            Is it a temporary string?
     078A 001C
1331 078C 1605          JNE  VPSH20              No - so done
1332 078E C060  VPSH19  MOV  @FAC4,R1            Get the address of the string
     0790 0694'
1333 0792 1302          JEQ  VPSH20              If null string - nothing to do
1334 0794 06A0          BL   @STVDP3             Set the backpointer
     0796 0000
```

```
1335                    *
1336 079B C060  VPSH20  MOV   @VSPTR,R1       Check for buffer-zone
     079A 077E'
1337       079E' C16    EQU   $+2
1338 079C 0221          AI    R1,16           Correct by 16
     079E 0010
1339 07A0 8501          C     R1,@STREND      At least 16 bytes between
     07A2 0000
1340                    *                       stack and string space?
1341 07A4 1236          JLE   VPOP18          Yes - so OK
1342 07A6 05C9          INCT  R9              No - save return address
1343 07A8 C640          MOV   R0,*R9             on stack
1344 07AA 06A0          BL    @COMPCT         Do the garbage collection
     07AC 0000
1345 07AE C019          MOV   *R9,R0          Restore return address
1346 07B0 0649          DECT  R9              Fix subroutine stack pointer
1347 07B2 C060          MOV   @VSPTR,R1       Get value stack pointer
     07B4 079A'
1348 07B6 0221          AI    R1,16           Buffer zone
     07B8 0010
1349 07BA 8501          C     R1,@STREND      At least 16 bytes now?
     07BC 07A2'
1350 07BE 1229          JLE   VPOP18          Yes - so OK
1351 07C0 0200  VPSH23  LI    R0,ERROM        No - so MEMORY FULL error
     07C2 0103
1352 07C4 06A0  VPSH25  BL    @SETREG         In case of GPL call
     07C6 06E0'
1353 07C8 0460          B     @ERR
     07CA 01EE'
1354 07CC 0460  VPSH27  B     @ERRSO          STACK OVERFLOW
     07CE 0004'
```

```
1356                    *               Stack VPOP routine
1357 07D0
1358 07D0 0202  VPOP    LI    R2,FAC            Destination is FAC
     07D2 0786'
1359 07D4 C0E0          MOV   @VSPTR,R1         Get stack pointer
     07D6 0734'
1360 07D8 8801          C     R1,@STVSPT        Check for stack underflow
     07DA 0398'
1361 07DC 1218          JLE   VPOP20            Yes!! - error
1362                    *--------------------CONDITIONAL ASSEMBLY--------------------*
1363                    ASMIF VERS=DX10
1364                    LI    R0,8              Popping 8 bytes
1365                    S     R0,@VSPTR         Correct stack pointer
1366                    MOV   R1,R14            Pointer to source
1367                    AI    R14,VRAM          Add in the VDP offset
1368                    VPOP10 MOVB *R14+,*R2+  Move a byte
1369
1370                    ASMELS
1371 07DE
1372 07DE D7E1          MOVB  @R1LB,*R15        Write 2nd byte of address
     07E0 07DE'
1373 07E2 0200          LI    R0,8              Popping 8 bytes
     07E4 0008
1374 07E6 D7C1          MOVB  R1,*R15           Write 1st byte of address
1375 07E8 6800          S     R0,@VSPTR         Adjust stack pointer
     07EA 07D6'
1376 07EC DCA0  VPOP10  MOVB  @VDPRD,*R2+       Move a byte
     07EE 0726'
1377                    ASMEND
1378                    *------------------END OF CONDITIONAL ASSEMBLY---------------*
1379 07F0 0600          DEC   R0                Decrement the counter - Done?
1380 07F2 15FC          JGT   VPOP10            No - finish the work
1381 07F4 C00B          MOV   R11,R0            Save return address
1382 07F6 9820          CB    @FAC2,@CBH65      Pop a string?
     07F8 0776'
     07FA 0143'
1383 07FC 160A          JNE   VPOP18            No - so done
1384 07FE 0406          CLR   R6                For backpointer clear
1385 0800 C0E0          MOV   @FAC,R3           Get string owner
     0802 07D2'
1386 0804 0283          CI    R3,>001C          Pop a temporary?
     0806 001C
1387 0808 13C2          JEQ   VPSH19            Yes - must free it
1388 080A 06A0          BL    @GET1             No- get new pointer from s.t.
     080C 0000
1389 080E C801          MOV   R1,@FAC4          Set new pointer to string
     0810 0790'
1390 0812 0450  VPOP18  B     *R0               And return
1391 0814
1392 0814 0200  VPOP20  LI    R0,ERREX          * SYNTAX ERROR
     0816 0403
1393 0818 10D5          JMP   VPSH25
```

```
                        *       Get next character from BASIC program
                        *     The returned status reflects the character
                        *  RAMFLG = >00 : No ERAM or imperative statement
                        *           >FF : With ERAM and a program is being
                        *                 executed
                        *
1401 081A               PGMCHR
1402 081A D220                  MOVB @RAMFLG,R8         Test ERAM flag
     081C 0314'
1403 081E 160A                  JNE  PGMC10             ERAM and a program is being
1404                    *       Next label is for entry from SUBPROG.
1405 0820               PGMSUB                          executed
1406                    *
1407                    *------------------CONDITIONAL ASSEMBLY--------------------*
1408                            ASMIF VERS=DX10
1409                            MOV  @PGMPTR,R10        Use R10 for the read
1410                            AI   R10,VRAM           Add in the base
1411                    PGMNXT  INC  @PGMPTR            Increment the perm pointer
1412                            MOVB *R10+,R8           Get the byte from the VDP
1413                            RT                      And return
1414                    *
1415                    PGMC10  MOV  @PGMPTR,R10        Use R10 for the read
1416                            AI   R10,GRAM           Add in GRAM offset
1417                            JMP  PGMNXT             Get the next character
1418
1419                            ASMELS
1420 0820
1421 0820 D7E0                  MOVB @PGMPT1,*R15       Write 2nd byte of address
     0822 0000
1422 0824 020A                  LI   R10,VDPRD          Read data address
     0826 07EE'
1423 0828 D7E0                  MOVB @PGMPTR,*R15       Write 1st byte of address
     082A 04DE'
1424 082C 05A0                  INC  @PGMPTR            Increment the perm pointer
     082E 082A'
1425 0830 D21A                  MOVB *R10,R8            Read the character
1426 0832 045B                  RT                      And return
1427 0834
1428 0834               PGMC10
1429 0834 C2A0                  MOV  @PGMPTR,R10
     0836 082E'
1430 0838 05A0          PGMNXT  INC  @PGMPTR
     083A 0836'
1431 083C D23A                  MOVB *R10+,R8           Write 2nd byte of address
1432 083E 045B                  RT
1433 0840
1434                            ASMEND
1435                    *------------------END OF CONDTITIONAL ASSEMBLY-----------
1436 0840
1437                            END
NO ERRORS,    0034 WARNINGS
```

```
$          0840'          0190  0263  0271  0289  0296  0297  0299  0437  0575
                          0600  1045  1065  1092  1337
ARG     R  0540'  0104    0945
ARGTOS     0734'  1242    1183  1247
ARGT10     0738'  1246    1239
ARGT20     0740'  1248    1241
ARGTST  D  0714'  1220    0090  1112  1182
ASSG    R  052A'  0098    0939
B9900      008E'  0246    0189
BEXC15     03FB'  0770    0764
BREAK      0007   0470    0537
BREAK$     008E   0128    0557
BRKFL      0001   0466    0426
BRKP1L     016E'  0364    0315
BRKPN1     01E0'  0429    0364
BRKPN2     01DA'  0425    0433
BRKPNT     01D2'  0422    0288
CC         00E4'  0299    0305
C1000   R  010E'  0295    0308  0314
C16     D  079E'  1337    0091
C2      R  0210'  0467    0490
C24     D  0000'  0143    0091
C3      D  00E0'  0296    0091
C4      D  0626'  1092    0091  0348  0477
CALGPL  D  0208'  0483    0089  0492
CALL    R  0588'  0111    0984
CB3        00E1'  0297    0311
CBH65   D  0143'  0343    0091  0812  1236  1240  1246  1325  1382
CBH66      028D'  0564    0604  0734
CBH67   D  0447'  0822    0091
CBH69      0641'  1107    0746
CBH6A      03FC'  0771    0748
CFI     R  0298'  0094    0567
CHAT    R          0105
COMMA$     00B3   0135    0563  0718  0821  0925
COMPCT  R  07AC'  0094    1344
CONC       021A'  0495    1051
CONCAT     0008   0471    0495
CONT    D  0064'  0214    0088  0478  0524  0815  0846  1189
CONT10     0074'  0237    0216
CONT15     0082'  0242    0187
CONTG   D  0060'  0213    0085
CONTIN  D  0168'  0361    0088
CSN01   R  0244'  0109    0517
CSTR05     06A2'  1157    1155
CSTR10     06A6'  1159    1169
CSTR20     06C0'  1171    1158
DIVI$      00C4   0139    1201  1207
DIVIDE     0708'  1206    1063
DX10       0001   0003    0004  0061  0113  0154  0197  0218  0291  0329  0634
                          1224  1253  1274  1307  1363  1408
EGROM      0001   0066    0164  0224  0249
ELSE$      0081   0121    0373  0834  0854
END        01FA'  0454    0966
ENLN    R  0344'  0102    0668  0688
EOL     D  0172'  0369    0086  0836  0948
EOLINE  D  040E'  0794    0085  0851  0868
EOLNE1     0416'  0797    0795
EOSTM1     040C'  0785    0780  0781  0783
EOSTMT  D  03FE'  0779    0085  0716  0739  0763  0842
```

```
EQ#          00BE    0136  0923  1094
EQUALS       0634'   1099  1057
ERR      D   01EE'   0443  0089  0147  0573  0932  1353
ERR1     D   01EA'   0438  0087  0337  0372  0374  0520  0587  0591
ERR1B        02D4'   0591  0624  0723
ERR1C        0380'   0723  0719  0767  0826  0843  0950
ERR1C#       054A'   0950  0926  0929
ERRS1        03EE'   0725  0732  0751
ERRCOD   R   0212'   0101  0265  0443  0477  0490
                     0104
ERREX        0403    0464  1392
ERRIOR       0203    0462  0572
ERRLNF       0303    0463  0707
ERRMUV   D   0516'   0931  0086  0919
ERROM        0103    0461  1351
ERROR        0005    0468  0531
ERROR#       00A5    0131  0555
ERRSN        0003    0460  0441
ERRSO    D   0004'   0146  0091  0177  1354
ERRSYN   D   01EA'   0437  0089
ERRT     R   0736'   0099  1242
EXC15L       016C'   0362  0424
EXEC10       00AA'   0271  0350  0713
EXEC11       00B2'   0275  0491
EXEC15       00DE'   0289  0267  0287  0345  0362  0770
EXEC16       0112'   0318  0309  0432
EXEC17       0124'   0323  0356
EXEC20       012A'   0326  0323
EXEC50   D   01F2'   0448  0088  0347  0351  0455  0485
EXECG    D   0090'   0263  0085
EXIT         01EE'   0442  0427  0496  0532  0535  0538
EXRTN        0142'   0342  0144
EXRTN2       014A'   0346  0358
EXRTN3       0154'   0357  0322
EXRTNA       0002'   0144  0320
EXTRAM   R   03E6'   0102  0275  0348  0349  0608  0711  0742  0762
FAC      R   0802'   0103  0570  0601  0610  0808  0827  0917  0945  1123  13
                           1329  1358  1385
FAC10    R   06E4'   0103  0511  0522  0566  0568  1181  1187
FAC12    R   024C'   0104  0508  0519
FAC2     R   07F8'   0103  0734  0746  0748  0812  0942  1240  1246  1325  13
FAC4     R   0810'   0103  0741  0761  0944  1148  1151  1332  1389
FAC6     R   03E4'   0103  0742  0762
FAC7     R   0698'   0103  1149  1152
FLAG     R   01E2'   0102  0272  0422  0430
GET1     R   080C'   0109  1388
GETCGR   R   0240'   0109  0516
GETCH    R   0236'   0109  0513
GETL1#       0476'   0837  0841
GETL10       0300'   0625  0759  0837
GETV     R   04EE'   0097  0807  0916
GETV1    R   06B6'   0097  1161  1165
GO           0272'   0545  0960
GO$          0085    0125  0576
GOSUB        02D6'   0596  0962
GOSUB$       0087    0127  0586
GOSUB2       02D8'   0600  0589
GOT32        031A'   0668  0679
GOT33        032E'   0677  0682
GOT35        0334'   0681  0673
```

```
GOTO      02F8'   0616  0961
GOTO$     0086    0126  0584
GOTO20    02FA'   0620  0611  0721
GOTO31    0308'   0684  0666
GOTO32    0242'   0688  0701
GOTO33    0336'   0698  0704
GOTO34    0364'   0707  06a9  0689
GOTO35    0360'   0703  0694
GOTO3a    036A'   0710  0675  0696
GOTO40    0376'   0715  0629
GOTO50    0386'   0720  0580  0585
GOTO90 D  02A6'   0572  0091  0569  0717
GOTO95    02AA'   0573  0708  0726  0801
GRAM   R          0110
GREATR    0624'   1091  1059
GRMRA  R  0016'   0093  0161  0162
GRMWA  R  0070'   0093  0201  0203  0229  0231
GROM   R          0110
GROMA  R          0096
HT$       0000    0137  1087  1106
HHUSC  R          0093
IF        0442'   0820  0959
IF$       0084    0124  0859
IF$10     047A'   0839  0828  0835  0857
IF$20     0496'   0850  0831
IF$25     04A0'   0854  0871
IF$27     04AC'   0859  0855
IF$28     04B6'   0863  0860
IF$30     04C4'   0868  0864
IF$35     04CA'   0870  0858  0862  0876  0884
IF$40     04D0'   0875  0866
IF$5      046C'   0834  0869
IF$50     04D6'   0880  0867
JEQ1C     0488'   0843  0853
LEDEND    06EB'   1189  1128
LEDERR    06EC'   1191  1188
LEDEX     06CE'   1181  1197  1203  1209
LEDLE     0636'   1101  1097
LESS      0616'   1085  1058
LETCON    0546'   0948  0941
LEXP   R  0614'   0108  1064
LFALSE    065E'   1119  1121  1132  1134
LN$       00C9    0141  0390  0623  0758  0830  0840  0865
LT10      0628'   1094  1088
LT15      062E'   1096  1090
LTAB      05FA'   1050  0243  1065
LTBLEN.   0Q1C    1065  0241
LTRUE  D  0664'   1122  0090  1117  1118  1130  1131  1133
LTST01    0638'   1102  1095
LTST15    0656'   1115  1167  1172
LTST20    0686'   1147  1113
LTST90 D  0668'   1123  0090  1120
LTSTAB    0680'   1138  1110
LTSTEQ    065C'   1118  1138
LTSTGE    065A'   1117  1143
LTSTGT    067C'   1133  1142
LTSTLE    0676'   1130  1141
LTSTLT    0678'   1131  1140
LTSTNE    0662'   1121  1139
LTSTXX    065A'   1116  1115  1138  1139  1140  1141  1142  1143
```

```
MINUS       06F0'   1194  1061
MINUS#      00C2    0138  1178  1195
MOVFAC  R   042C'   0097  0814
NABS    R   05CC'   0107  1022
NATN    R   05CE'   0107  1023
NCOS    R   05D0'   0107  1024
NEXP    R   05D2'   0107  1025
NEXT#       0096    0129  0766
NFOR    R   0566'   0108  0967
NINT    R   05D4'   0107  1026
NLET        04E4'   0913  0324  0968
NLET05      04E8'   0915  0928
NLET10      051E'   0934  0924
NLET15      0528'   0939  0943  0946
NLOG    R   05D6'   0107  1027
NLPR    R   05A4'   0106  1002
NMINUS  R   05BA'   0106  1013
NNEXT   R   057A'   0108  0977
NOLED       01EA'   0440  0256  1052  1056
NOLEDL      0096'   0256  0242
NONUD       01EA'   0439  0955  0956  0992  0993  1003  1004  1005  1006  1007
                            1009  1010  1011  1014  1015  1016  1017  1020
NPLUS   R   05B8'   0106  1012
NSGN    R   05DB'   0107  1028
NSIN    R   05DA'   0107  1029
NSQR    R   05DC'   0108  1030
NSTRCN  R   05C4'   0094  1018
NTAB        05A4'   1002  0188  1045
NTABLN      0056    1045  0186
NTAN    R   05DE'   0108  1031
NUDE10      0090'   0248  0380
NUDEND  D   018C'   0386  0086  0407  0964  0965  0985
NUDGO5      004C'   0194  0255  0449
NUDND1      0182'   0379  0257  0370  0387  0394  0400  0415  0958  0974  0981
                            0990
NUDND2      0190'   0388  0395
NUDND3      019E'   0393  0397  0399
NUDND4      01A6'   0396  0389
NUDNDL      009A'   0257  0238
NUDTAB  R   004A'   0101  0193
NUMC49      0242'   0517  0515
NUMCHK      0738'   1245  0565  0823
NUMCON      0220'   0507  1019
O.AND   R   0600'   0106  1054
O.NOT   R   05B0'   0106  1008
O.OR    R   05FE'   0106  1053
O.XOR   R   0602'   0106  1055
ON          0276'   0552  0982
ON20        02AC'   0575  0571
ON30        02B6'   0579  0546
ON40        02C2'   0584  0577
ON50        02CC'   0587  0582
ONBRK       026C'   0537  0558
ONERR       0260'   0531  0556
ONWARN      0266'   0534  0554
P05         0026'   0179  1298
P10         002E'   0183  0180
P17         0044'   0190  0207
P17L        005E'   0207  0339
P359        0000    0003  0003
```

```
FAD       R  052E'  0105  0914  0922  0940
PARSE     D  0010'  0175  0088  0562  0820  0935
PARSE9    D  0000'  0152  0085
PGMCIO       0834'  1429  1403
PGMCMD    D  081A'  1401  0087  0183  0245  0277  0280  0282  0321  0393  0402
                           0406  0521  0578  0588  0625  0627  0715  0720  0829
                           0833  0839  0870  0880  0927  0934  1096
PGMPT1    R  0822'  0104  1421
PGMPTR    R  083A'  0101  0275  0276  0279  0284  0318  0327  0404  0409  0508
                           0510  0519  0606  0741  0761  0832  0845  0875  0882
                           1423  1424  1429  1430
PGMSUB    D  0820'  1405  0087
PLUS         06C4'  1177  1060
PRGFLG    R  014C'  0102  0266  0346
PSHPRS    D  0742'  1294  0086  1105  1177  1194  1200  1206
PSYM         0420'  0806  0181
R0           0000        0146  0266  0272  0273  0278  0286  0307  0308  0313
                           0314  0346  0422  0423  0426  0430  0431  0441  0443
                           0495  0522  0531  0534  0537  0568  0572  0626  0672
                           0693  0707  0725  0800  0931  1162  1166  1303  1304
                           1322  1324  1343  1345  1351  1373  1375  1379  1381
                           1390  1392
R1           0001        0601  0602  0604  0605  0606  0607  0608  0664  0667
                           0668  0678  0686  0688  0699  0710  0711  0809  0918
                           1162  1166  1305  1316  1317  1318  1319  1329  1330
                           1332  1336  1338  1339  1347  1348  1349  1359  1360
                           1374  1389
R10          000A        0332  1148  1159  1160  1422  1425  1429  1431
R11          000B        0161  0165  0178  0478  0484  0491  1297  1298  1324
                           1381
R11LB     R  0018'  0100  0162
R12          000C        0304  0306  0310  0312  1110  1111  1115
R13          000D        0161  0162  0201  0203  0229  0231  0232
R15          000F        0684  0686  1232  1234  1315  1317  1372  1374  1421
                           1423
R1LB      R  07E0'  0100  0684  1315  1372
R2           0002        0665  0667  0670  0674  0677  0678  0681  0685  0690
                           0695  0698  0700  0703  1086  1089  1093  1100  1101
                           1104  1179  1185  1187  1196  1202  1208  1358  1376
R3           0003        0513  0516  0545  0570  0596  0602  0610  0616  0628
                           0670  0671  0672  0681  0690  0692  0693  0698  0700
                           0703  0757  0824  0850  0856  0861  1123  1124  1125
                           1126  1127  1159  1163  1385  1386
R4           0004        0514  1108  1110  1119  1122  1124  1151  1163  1164
R5           0005        1153  1156  1157  1168
R6           0006        0215  0229  0230  0231  0232  0237  0944  1152  1153
                           1154  1171  1221  1222  1234  1327  1328  1384
R6LB      R  071C'  0100  1232
R7           0007        0179  0184  0185  0186  0188  0188  0192  0193  0201
                           0202  0203  0243  0246  0250  0326  0335  0336  0338
                           0338  0340  0379  0382  0448  1149  1154  1156  1171
R7LB      R        0093
R8           0008        0179  0237  0239  0240  0241  0243  0244  0281  0283
                           0284  0285  0326  0332  0344  0355  0355  0369  0369
                           0371  0373  0386  0386  0388  0390  0396  0398  0403
                           0404  0405  0509  0510  0553  0555  0557  0576  0579
                           0581  0584  0586  0623  0626  0674  0695  0718  0758
                           0766  0779  0779  0782  0784  0784  0794  0794  0796
                           0825  0830  0834  0840  0844  0854  0859  0863  0865
                           0881  0882  0883  0923  0925  1087  1094  1402  1425
```

```
                                1431
RP              0009            0175  0176  0178  0215  0248  0319  0320  0357  037
                                0381  0454  0483  0484  0712  1103  1104  1108  11(
                                1294  1295  1297  1342  1343  1345  1346
FFLG    R                 0100
RAMFLG  R       0810'     0110  0514  0665  1402
RESET   R       0050'     0098  0206
RETU3C          038E'     0744  0735
RETU40          03D4'     0756  0747
RETURN          0394'     0731  0740  0749  0963
RTNADD  R       01F4'     0101  0448
RTNG    D       01CC'     0414  0085
SADD    R       06CC'     0095  1179
SAVRE2  R       0232'     0117  0512
SAVREG  R       06DA'     0117  0195  1184
SCOMPB  R       0654'     0095  1114
SDIV    R       0710'     0095  1208
SETREG  R       07C6'     0117  0152  0213  0264  0361  0414  0518  1186  1352
SAPLN           01C6'     0409  0391
SAPS01          01C0'     0406  0410  0743  0768
SAPSTR          01B4'     0402  0392
SMB     R       04F8'     0097  0811  0920
SMTSEP          01E0'     0355  0957
SMTSRT  R       0116'     0109  0318
SMULT   R       0704'     0095  1202
SSEP$           0082      0122  0344  0398  0844
SSUB    R       06F8'     0095  1196
STCODE          0527'     0937  0942
STKADD  R                 0105
STKEND  R       0746'     0103  0176  1295
STLN    R       0310'     0102  0349  0664
STMTTB          05A2'     0997  0338
STOP            01FA'     0453  0979  0988
STREND  R       07BC'     0101  1339  1349
STRIN$          00C7      0140  0388  0863
STVDP3  R       0796'     0096  1334
STVSPT  R       07DA'     0101  0731  1360
SUB$            00A1      0130  0581
SUBXIT  R       059E'     0111  0994  0995
SYM     R       04EA'     0097  0806  0915
SYMB10          043E'     0815  0813
SYMB20          0418'     0800  0810
THEN$           00B0      0133  0825
TIMES           06FC'     1200  1062
TO$             00B1      0134  0579
TRACE           020E'     0490  0365
TRACL           0170'     0365  0274
TREM$           0083      0123  0371  0396  0782  0796  0936
UDF             0006      0469  0800
VDPRD   R       0826'     0093  0685  1236  1376  1422
VDPWD   R       076C'     0093  1319
VERMAC  M                 A0001 0003
VERS            0000      0003  0004  0061  0113  0154  0197  0218  0291  0329  0634
                                1224  1253  1274  1307  1363  1408
VPOP    D       07D0'     1358  0086  0733  1150
VPOP10          07EC'     1376  1380
VPOP18          0812'     1390  1341  1350  1383
VPOP20  D       0814'     1392  0089  1361
VPSH15          076A'     1319  1323
VPSH19          078E'     1332  1387
```

LABEL		VALUE	DEFN	REFERENCES								
VPSH20		0798'	1336	1326	1331	1333						
VPSH22	D	0700'	1351	0089								
VPSH25		0704'	1352	1393								
VPSH27		0700	1354	1296								
VPUSH	D	0790'	1302	0086	0609	0750	0921					
VRAM	R		0105									
VSPTR	R	07EA'	0105	0731	1221	1304	1305	1327	1336	1347	1359	1375
WARN		0009	0472	0534								
WARN$		00A6	0132	0553								
WARN$$	D	01FE'	0477	0090	0523	1191						
WRVDP	R	0762'	0093	1316								