ACCESS NAMES TABLE

```
SOURCE ACCESS NAME=      PPC2.P359.V081180.SRC.BASSUP
OBJECT ACCESS NAME=      PPC2.P359.V081180.OBJ.BASSUPS
LISTING ACCESS NAME=     PPC2.P359.V081180.LST.BASSUPS
ERROR ACCESS NAME=
OPTIONS=                 XREF
MACRO LIBRARY PATHNAME=


LINE      KEY   NAME

 0002      A    VERSION
                    =>PPC2.P359.V081180.SRC.P359
```

```
0031                    IDT   'BASSUP'
0032            ********************************************************************
0033            *                                                                  *
0034            *        BBBBB     AAAA     SSSS     SSSS   U     U  PPPP         *
0035            *        B    B   A    A   S    S   S    S  U     U  P   P        *
0036            *        B    B   A    A   S        S       U     U  P   P        *
0037            *        BBBBB    AAAAAA    SSSS     SSSS   U     U  PPPP         *
0038            *        B    B   A    A        S        S  U     U  P            *
0039            *        B    B   A    A   S    S   S    S  U     U  P            *
0040            *        BBBBB    A    A    SSSS     SSSS    UUUU   P            *
0041            *                                                                  *
0042            *             PPPP       3333      555555      9999              *
0043            *             P   P     3    3     5          9    9             *
0044            *             P   P          3     5          9    9             *
0045            *             PPPP        3333     55555       99999            *
0046            *             P              3         5          9              *
0047            *             P         3    3     5    5     9    9             *
0048            *             P          3333      5555       9999              *
0049            *                                                                  *
0050            ********************************************************************
```

```
0055                *                  General BASIC Support Routines
0056                *                      (NOT including PARSE)
0057 0000
0058                        DEF   FBSYMB,SYNCHK
0059                        DEF   SYMB,SMBB,SYM,SMB,MOVFAC
0060                        DEF   PUTV,PUTV1
0061                        DEF   ASSGNV,ASSG,VPUSHG,PGMCH
0062                        DEF   ERRT
0063                        DEF   COMPCG,GETSTG
0064                *
0065                        REF   VDPWD,VDPRD,WRVDP
0066                        REF   RESET,ERRMUV
0067                        REF   COMPCT,GETSTR
0068                        REF   VRAM
0069                        REF   SETREG,SAVREG,PGMCHR,ARGTST
0070                        REF   VPUSH,CFI,VPOP,BASE
0071                        REF   ERR,ERRSYN,PSHPRS
0072                        REF   PUTSTK,GETSTK,POPSTK
0073                        REF   R1LB,R4LB,R5LB,R6LB,R7LB
0074                        REF   BYTE,SREF,SYMTAB,CHAT
0075                        REF   STATUS,NEXT
0076                        REF   FAC,FAC2,FAC4,FAC5,FAC6,FAC10,FAC15
0077                        REF   ARG,ARG2,ARG4
0078                        REF   RAMTOP,GRAM,GET,GET1,GETG
0079                *-------------------CONDITIONAL ASSEMBLY--------------------*
0080                        ASMIF VERS=DX10
0081                        DEF   STVDP,STVDP3,FBS001,FBS,GETV,GETV1
0082                        ASMELS
0083                        REF   STVDP,STVDP3,FBS001,FBS,GETV,GETV1
0084                        DEF   SET
0085                        ASMEND
0086                *-----------------END OF CONDITIONAL ASSEMBLY---------------*
0087                *
0088        00B3    COMMA$  EQU   >B3
0089        00B6    RPAR$   EQU   >B6
0090        0087    LPAR$   EQU   >B7
0091        0503    ERRBS   EQU   >0503             BAD SUBSCRIPT ERROR CODE.
0092        0603    ERRTM   EQU   >0603             ERROR STRING/NUMBER MISMATCH
0093                *
0094 0000   6500    STCODE  DATA  >6500
0095 0002   0006    C6      DATA  >0006
0096                *
```

```
0099                *            Entry to find BASIC symbol table entry for
0100                *                        Graphics Language
0101  0004
0102  0004 06A0  FBSYMB BL   @FBS                Search the symbol table
      0006 0000
0103  0008 0000         DATA RESET               If not found - condition reset
0104  000A FE20  SET    SCCB @BIT2,@STATUS       Set GPL condition
      000C 0123'
      000E 0000
0105  0010 0460         B    @NEXT               If found - condition set
      0012 0000
0106  0014
0107                ****************************************************************
0108                *        Entry to find BASIC symbol table entry for            *
0109                *                    Assembly Language                         *
0110                *                                                              *
0111                *      BL   @FBS                                               *
0112                *      DATA NOTFOUND                                           *
0113                *      R4 - SYMBOL TABLE ENTRY ADDRESS RETURN                  *
0114                *      R8 - NOT DESTROYED                                      *
0115                *                                                              *
0116                ****************************************************************
0117                *-----------------CONDITIONAL ASSEMBLY---------------------*
0118                       ASMIF VERS=DX10
0119
0120                FBS    MOV  @SYMTAB,R4          Get symbol table pointer
0121                       JEQ  FBS006              Jump out if table is empty
0122                *      Next label for subprogram routine
0123                FBS001
0124                       MOVB @FAC15,R3           Length of desired symbol
0125                       CLR  R7                  Clear counter (MSByte)
0126                *-----------------CONDITIONAL ASSEMBLY---------------------*
0127                       ASMIF VERS=DX10
0128                FBS002 INC  R4                  Point to length byte in table
0129                       MOV  R4,R14              R14 is VDPRAM pointer
0130                       AI   R14,VRAM            Add in VDP offset
0131                       CB   *R14+,R3            Compare lengths of names
0132                       JEQ  FBS010              If same - compare the names
0133                       MOVB *R14+,R6            Not the same - so get link
0134                       MOVB *R14+,@R6LB           to next entry in table
0135
0136                       ASMELS
0137
0138                FBS002 INC  R4                  Point to length byte of entry
0139                       MOVB @R4LB,*R15          Load 2nd byte of address
0140                       NOP                      Kill some time
0141                       MOVB R4,*R15             Load 1st byte of address
0142                       LI   R10,VDPRD           VDP read-data address
0143                       CB   *R10,R3             Compare length of names
0144                       JEQ  FBS010              If same - compare names
0145                       MOVB *R10,R6             Not the same - so get link
0146                       NOP                      Kill some time
0147                       MOVB *R10,@R6LB          Get 2nd byte too
0148                       ASMEND
0149                *--------------END OF CONDITIONAL ASSEMBLY-----------------*
0150                FBS004 MOV  R6,R4               Transfer link and test for end
0151                       JNE  FBS002              Loop in not end of table
0152                FBS006 MOV  *R11,R11            End of table - Get rtn vector
0153                       RT                       And return
0154
```

```
0155                *       Length matches - Compare names
0156                *
0157                FBS010
0158                *--------------------CONDITIONAL ASSEMBLY--------------------
0159                        ASMIF VERS=DX10
0160                        MOVB  *R14+,R6          Get link in case of NO match
0161                        MOVB  *R14+,@R6LB       Both bytes
0162                        MOVB  *R14+,R5          Get pointer to name
0163                        MOVB  R3,@R7LB          Counter for compare
0164                        MOVB  *R14+,@R5LB       2nd byte of pointer to name
0165                        MOV   R5,R14            Load VDP RAM address
0166                        AI    R14,VRAM          Add in VDP offset
0167                        LI    R2,FAC            Source for compare is in FAC
0168                FBS014  CB    *R14+,*R2+        Compare a byte
0169
0170                        ASMELS
0171
0172                        MOVB  *R10,R6           Get link in case of NO match
0173                        NOP                     Kill some time
0174                        MOVB  *R10,@R6LB        Both bytes of link
0175                        NOP                     Kill some more time
0176                        MOVB  *R10,R5           Get pointer to name
0177                        MOVB  R3,@R7LB          Get length to count
0178                        MOVB  *R10,R2           2nd byte of pointer to name
0179                        MOVB  R2,*R15           Write out 2nd byte of pointer
0180                        NOP                     Kill some time
0181                        MOVB  R5,*R15           Write out 1st byte of pointer
0182                        LI    R2,FAC            Source is in FAC
0183                FBS014  CB    *R10,*R2+         Compare a character
0184                        ASMEND
0185                *--------------------END OF CONDITIONAL ASSEMBLY--------------
0186                        JNE   FBS004            Not equal - try next entry
0187                        DEC   R7                Count characters compared
0188                        JGT   FBS014            More left - so loop to compare
0189                        DEC   R4                Adjust return table pointer
0190                        MOV   R4,@FAC           Save for GPL
0191                        B     @2(R11)           Take the "found" return vector
0192
0193                        ASMELS
0194 0014
0195                *       USES CONSOLE ROUTINE FOR SPEED
0196 0014
0197                        ASMEND
0198                *------------------END OF CONDITIONAL ASSEMBLY--------------
```

```
0201                    *       Graphics language entry for COMPCT to take
0202                    *            advantage of common code
0203 0014
0204 0014 0206  COMPCG  LI    R6,COMPCT         Address of COMPCT
     0016 0000
0205 0018 100E          JMP   SMBB10            Jump to set up
0206 001A
0207                    *       Graphics language entry for GETSTR to take
0208                    *            advantage of common code
0209 001A
0210 001A 0206  GETSTG  LI    R6,GETSTR         Address of MEMCHK
     001C 0000
0211 001E 1008          JMP   SMBB10            Jump to set up
0212 0020
0213                    *       Graphics language entry for SMB to take
0214                    *            advantage of common code
0215 0020
0216 0020 0206  SMBB    LI    R6,SMB            Address of SMB routine
     0022 0054'
0217 0024 1008          JMP   SMBB10            Jump to set up
0218 0026
0219                    *       Graphics language entry for ASSGNV to take
0220                    *            advantage of common code
0221 0026
0222 0026 0206  ASSGNV  LI    R6,ASSG           Address of ASSGNV routine
     0028 01A8'
0223 002A 1005          JMP   SMBB10            Jump to set up
0224 002C
0225                    *       Graphics language entry for SYM to take
0226                    *            advantage of common code
0227 002C
0228 002C 0206  SYMB    LI    R6,SYM            Address of SYM routine
     002E 018A'
0229 0030 1002          JMP   SMBB10            Jump to set up
0230 0032
0231                    *       Graphics language entry for SMB to take
0232                    *            advantage of common code
0233 0032
0234 0032 0206  VPUSHG  LI    R6,VPUSH          Address of VPUSH routine
     0034 0000
0235 0036
0236 0036 C1CB  SMBB10 MOV   R11,R7             Save return address
0237 0038 06A0          BL    @PUTSTK           Save current GROM address
     003A 0000
0238 003C 06A0          BL    @SETREG           Set up BASIC registers
     003E 0000
0239 0040 05C9          INCT  R9                Get space on subroutine stack
0240 0042 C647          MOV   R7,*R9            Save the return address
0241 0044 0696          BL    *R6               Branch and link to the routine
0242 0046 C1D9          MOV   *R9,R7            Get return address
0243 0048 0649          DECT  R9                Restore subroutine stack
0244 004A 06A0          BL    @SAVREG           Save registers for GPL
     004C 0000
0245 004E 06A0          BL    @GETSTK           Restore GROM address
     0050 0000
0246 0052 0457          B     *R7               Return to GPL
```

```
0249                    ***************************************************************
0250                    *       Subroutine to find the pointer to variable space
0251                    *       of each element of symbol table entry.  Decides
0252                    *       whether symbol table entry pointed to by FAC,FAC+1
0253                    *       is a simple variable, string variable or array
0254                    *       variable, and returns proper 8-byte block in FAC
0255                    *       through FAC7
0256                    ***************************************************************
0257 0054
0258 0054 05C9  SMB     INCT R9                         Get space on subroutine stack
0259 0056 C64B          MOV  R11,*R9                    Save return address
0260 0058 C820          MOV  @FAC,@FAC4                 Copy pointer to table entry
     005A 0000
     005C 0000
0261 005E A820          A    @C6,@FAC4                  Add 6 so point a value space
     0060 0002'
     0062 005C'
0262 0064 06A0          BL   @GETV                      Get 1st byte of table entry
     0066 0000
0263 0068 005A'         DATA FAC                        Pointer is in FAC
0264                    *
0265 006A C101          MOV  R1,R4                      Copy for later use.
0266 006C C081          MOV  R1,R2                      Copy for later use.
0267 006E 0A21          SLA  R1,2                       Check for UDF entry
0268 0070 1821          JOC  BERMUV                     If UDF - then error
0269                    *
0270 0072 C104          MOV  R4,R4                      Check for string.
0271 0074 1102          JLT  SMB02                      Skip if it is a srting.
0272 0076 04E0          CLR  @FAC2                      Clear for numeric case.
     0078 0000
0273                    *
0274                    *       In case of subprogram call check if parameter is
0275                    *       shared by its calling program.
0276                    *
0277 007A 0A11  SMB02   SLA  R1,1                       Check for the shared bit.
0278 007C 1705          JNC  SMB04                      If it is not shared skip.
0279 007E 06A0          BL   @GET                       Get the value space pointer
     0080 0000
0280 0082 0062'         DATA FAC4                           in the symbol table.
0281 0084 C801          MOV  R1,@FAC4                   Store the value space address.
     0086 0082'
0282                    *
0283                    *       Branches to take care of string and array cases.
0284                    *       Only the numeric variable case stays on.
0285                    *
0286 0088 D104  SMB04   MOVB R4,R4                      R4 has header byte information.
0287 008A 1116          JLT  SMB050                     Take care of string.
0288 008C       SMB05
0289 008C 0A54          SLA  R4,5                       Get only the dimension number.
0290 008E 09D4          SRL  R4,13
0291 0090 162A          JNE  SMB020                     go to array case.
0292                    *
0293                    *       Numeric ERAM cases are special.
0294                    *       If it is shared get the actual v.s. address from ERAM.
0295                    *       Otherwise get it from VDP RAM.
0296                    *
0297 0092 D120          MOVB @RAMTOP,R4                 Check for ERAM.
     0094 0000
0298 0096 1308          JEQ  SMB010                     ERAM not exist. Normal return.
0299                    *                               Yes ERAM case.
```

```
0300 0098 0A32          SLA  R2,3              R2 has a header byte.
0301 009A 1704          JNC  SMB06             Shared bit is not ON.
0302                    *                      Yes, it is shared....
0303 009C 06A0          BL   @GETG             Get v.s. pointer from ERAM
     009E 0000
0304 00A0 00B6'         DATA FAC4
0305 00A2 1003          JMP  SMB08
0306                    *
0307 00A4 06A0  SMB06   BL   @GET              Not shared.
     00A6 0080'
0308 00A8 00A0'         DATA FAC4              Get v.s. address from VDP RAM.
0309                    *
0310 00AA C801  SMB08   MOV  R1,@FAC4          Store it in FAC4 area.
     00AC 00A8'
0311 00AE
0312                    *
0313                    *      Return from the SMB routine.
0314                    *
0315 00AE C2D9  SMB010  MOV  *R9,R11           Restore return address
0316 00B0 0649          DECT R9                Restore stack
0317 00B2 0458          RT                     And return
0318 00B4
0319 00B4 0460  BERMUV  B    @ERRMUV           * INCORRECT NAME USAGE
     00B6 0000
0320 00B8
0321                    *
0322                    *      Start looking for the real address of the symbol.
0323                    *
0324 00B8       SMB050
0325 00B8-0288          CI   R8,LPAR$*256      String - now string array?
     00BA B700
0326 00BC 13E7          JEQ  SMB05             Yes - process as an array
0327 00BE C820  SMB51   MOV  @STCODE,@FAC2     String ID code in FAC2
     00C0 0000'
     00C2 0078'
0328 00C4 C820          MOV  @FAC4,@FAC        Get string pointer address
     00C6 00AC'
     00C8 0068'
0329 00CA 06A0          BL   @GET              Get exact pointer to string
     00CC 00A6'
0330 00CE 00C8'         DATA FAC
0331                    *
0332 00D0 C801          MOV  R1,@FAC4          Save pointer to string
     00D2 00C6'
0333 00D4 C0C1          MOV  R1,R3             Was it a null?
0334 00D6 1304          JEQ  SMB57             Length is 0 - so is null
0335 00D8 0603          DEC  R3                Otherwise point at length byte
0336 00DA 06A0          BL   @GETV1            Get the string length
     00DC 0000
0337 00DE 0981          SRL  R1,8              Shift for use as double
0338 00E0 C801  SMB57   MOV  R1,@FAC6          Put into FAC entry
     00E2 0000
0339 00E4 10E4          JMP  SMB010            And return
0340                    *
0341                    *      Array cases are taken care of here.
0342                    *
0343 00E6       SMB020
0344 00E6 C804          MOV  R4,@FAC2          Now have a dimension counter
     00E8 00C2'
0345                    *                      that is initialized to max
```

```
0345                    *       *FAC+4,FAC+5 already points to 1st dimension maxima
0347                    *                       in symbol table
0348 00EA 04C2                  CLR   R2                    Clear index accumulator
0349 00EC               SMB025
0350 00EC C802                  MOV   R2,@FAC6              Save accumulator in FAC
     00EE 00E2'
0351 00F0 06A0                  BL    @PGMCHR               Get next character
     00F2 0000
0352 00F4 06A0                  BL    @PSHPRS               PUSH and PARSE subscript
     00F6 0000
0353 00F8   B7                  BYTE  LPAR$,0               Up to a left par or less
     00F9   00
0354 00FA 9820                  CB    @FAC2,@STCODE         Dimension can't be a string
     00FC 00E8'
     00FE 0000'
0355 0100 1441                  JHE   ERRT                  It is - so error
0356 0102
0357                    *       Now do float to integer conversion of dimension
0358 0102 04E0                  CLR   @FAC10                Assume no error
     0104 0000
0359 0106 06A0                  BL    @CFI                  Gets 2 byte integer in FAC,FAC+
     0108 0000
0360 010A D120                  MOVB  @FAC10,R4             Error on conversion?
     010C 0104'
0361 010E 1636                  JNE   ERR3                  Yes - error BAD SUBSCRIPT
0362 0110 C160                  MOV   @FAC,R5               Save index just read
     0112 00CE'
0363 0114 06A0                  BL    @VPOP                 Restore FAC block
     0116 0000
0364 0118 06A0                  BL    @GET                  Get next dimension maximum
     011A 00CC'
0365 011C 00D2'                 DATA  FAC4                  FAC4 points into symbol table
0366                    *
0367 011E 8045                  C     R5,R1                 Subscript less-than maximum?
0368 0120 1B2D                  JH    ERR3                  No - index out of bounds
0369      0123' BIT2    EQU   $+1                           Constant >20 (instr is >D120)
0370 0122 D120                  MOVB  @BASE,R4              Fetch option base to check low
     0124 0000
0371 0126 1303                  JEQ   SMB040                If BASE=0, INDEX=0 is OK
0372 0128 0605                  DEC   R5                    Adjust BASE 1 index
0373 012A 1128                  JLT   ERR3                  If subscript was = 0 ->error
0374 012C 1001                  JMP   SMB041                Accumulate the subscripts
0375 012E
0376 012E 0581          SMB040  INC   R1                    Adjust size if BASE=0
0377 0130 3860          SMB041  MPY   @FAC6,R1              R1,R2 has ACCUM*MAX dim
     0132 00EE'
0378 0134 A085                  A     R5,R2                 Add latest to accumulator
0379 0136 05E0                  INCT  @FAC4                 Increment dim maxima pointer
     0138 011C'
0380 013A 0620                  DEC   @FAC2                 Decrement remaining-dim count
     013C 00FC'
0381 013E 1305                  JEQ   SMB070                All dimensions handled -> done
0382 0140-0288                  CI    R8,COMMA$*256         Otherwise - must be at a comma
     0142 B300
0383 0144 13D3                  JEQ   SMB025                We are - so loop for more
0384 0146 0460  ERR1    B     @ERRSYN                       Not a comma - so syntax er
     0148 0000
0385 014A
0386                    *       At this point the required number of dimensions have
0387                    *                       been scanned.
```

```
0388                    *       R2 Contains the index
0389                    *       R4 Points to the first array element
0390                    *          or points to the address in ERAM where the first
0391                    *          array element is
0392 014A
0393 014A-0288  SMB070  CI    R8,RPAR$*256      Make sure at a right paren
     014C B800
0394 014E 16FB          JNE   ERR1              Not - so error
0395 0150 06A0          BL    @PGMCHR           Get next token
     0152 00F2'
0396 0154 06A0          BL    @GETV             Now check string or numeric
     0156 0066'
0397 0158 0112'         DATA  FAC                  array by checking s.t.
0398 015A 110C          JLT   SMB71             If MSBit set is a string array
0399 015C 0A32          SLA   R2,3              Numeric - multiply by 8
0400 015E D0E0          MOVB  @RAMTOP,R3        Does ERAM exist ?
     0160 0094'
0401 0162 1305          JEQ   SMB071            No
0402 0164 06A0          BL    @GET              Yes-get the content of value
     0166 011A'
0403 0168 0138'         DATA  FAC4                 ptr
0404 016A C801          MOV   R1,@FAC4          Put it in FAC4
     016C 0168'
0405 016E A802  SMB071  A     R2,@FAC4          Add into values pointer
     0170 016C'
0406 0172 109D          JMP   SMB010            And return in the normal way
0407 0174
0408 0174 0A12  SMB71   SLA   R2,1              String - multiply by 2
0409 0176 A802          A     R2,@FAC4          Add into values pointer
     0178 0170'
0410 017A 10A1          JMP   SMB51             And build the string FAC entry
0411 017C
0412 017C
0413 017C 0200  ERR3    LI    R0,ERRBS          Bad subscript return vector
     017E 0503
0414 0180 0460  ERRX    B     @ERR              Exit to GPL
     0182 0000
0415 0184
0416 0184
0417 0184 0200  ERRT    LI    R0,ERRTM          String-number mismatch vector
     0186 0603
0418 0188 10FB          JMP   ERRX              Use the long branch
```

```
0421                    ***************************************************************
0422                    *       Subroutine to put symbol name into FAC and to         *
0423                    *       call FBS to find the symbol table entry for the       *
0424                    *       symbol                                                *
0425                    ***************************************************************
0426 018A
0427 018A 04E0  SYM     CLR   @FAC15          Clear the character counter
     018C 0000
0428 018E 0202          LI    R2,FAC          Copying string into FAC
     0190 0158'
0429 0192 C04B          MOV   R11,R1          Save return address
0430 0194 DC88  SYM1    MOVB  R8,*R2+         Save the character
0431 0196 05A0          INC   @FAC15          Count it
     0198 018C'
0432 019A 06A0          BL    @PGMCHR         Get next character
     019C 0152'
0433 019E 15FA          JGT   SYM1            Still chars in the name
0434 01A0 06A0          BL    @FBS            Got name - now find s.t. entry
     01A2 000E'
0435 01A4 014E'         DATA  ERR1            Return vector if not found
0436 01A6 0451          B     *R1             Return to caller if found
```

```
0439                ***************************************************************
0440                *       ASSGNV, callable from GPL or 9900 code, to              *
0441                *       assign a value to a symbol (strings and numerics).      *
0442                *       If numeric, the 8-byte value is in the FAC.  If         *
0443                *       string, the 8-byte descriptor is in the FAC.  The       *
0444                *       descriptor block (8-bytes) for the destination          *
0445                *       variable is on the stack.  There are two types of       *
0446                *       descriptor entries which are created by SMB in          *
0447                *       preparation for  ASSGNV, one for numerics and one       *
0448                *       for strings.                                            *
0449                *                              NUMERIC                          *
0450                *       +----------+----+-----+----------+----------+           *
0451                *       |S.T. ptr  | 00 |     | Value ptr |          |           *
0452                *       +----------+----+-----+----------+----------+           *
0453                *                              STRING                           *
0454                *       +----------+----+-----+----------+----------+           *
0455                *       |Value ptr | 65 |     | String ptr|String len |           *
0456                *       +----------+----+-----+----------+----------+           *
0457                *                                                               *
0458                *       CRITICAL NOTE: Because of the BL @POPSTK below, if      *
0459                *       a string entry is popped and a garbage collection       *
0460                *       has taken place while the entry was pushed on the       *
0461                *       stack, and the entry was a permanent string the         *
0462                *       pointer in FAC+4,5 will be messed up.  A BL @VPOP       *
0463                *       would have taken care of the problem but would have     *
0464                *       taken a lot of extra code.  Therefore, at               *
0465                *       ASSG50-ASSG54 it is assumed that the previous value     *
0466                *       assigned to the destination variable has been moved     *
0467                *       and the pointer must be reset by going back to the      *
0468                *       symbol table and getting the correct value pointer.     *
0469                ***************************************************************
0470 01A8
0471 01A8 C28B  ASSG    MOV  R11,R10           Save the return address
0472 01AA 06A0          BL   @ARGTST           Check arg and variable type
     01AC 0000
0473 01AE 02CC          STST R12               Save status of type
0474 01B0 06A0          BL   @POPSTK           Pop destination descriptor
     01B2 0000
0475                *                          into ARG
0476 01B4 0A3C          SLA  R12,3             Variable type numeric?
0477 01B6 1742          JNC  ASSG70            Yes - handle it as such
0478 01B8
0479                *       Assign a string to a string variable
0480 01B8 C060          MOV  @ARG4,R1          Get destination pointer
     01BA 0000
0481                *                          Dest have non-null value?
0482 01BC 1308          JEQ  ASSG54            No - null->never assigned
0483                *       Previously assigned -  Must first free the old value
0484 01BE 06A0          BL   @GET              Correct for POPSTK above
     01C0 0166'
0485 01C2 0000          DATA ARG               Pointer is in ARG
0486 01C4 C801          MOV  R1,@ARG4          Correct ARG+4,5 too
     01C6 01BA'
0487 01C8 04C6          CLR  R6                Clear for zeroing backpointer
0488 01CA 06A0          BL   @STVDP3           Free the string
     01CC 0000
0489 01CE
0490 01CE C120  ASSG54  MOV  @FAC6,R4          Is source string a null?
     01D0 0132'
0491 01D2 130C          JEQ  ASSG57            Yes - handle specially
```

```
0492 01D4 C0E0          MOV  @FAC,R3            Get address of source pointer
     01D6 0190'
0493 01D8 0283          CI   R3,>001C           Got a temporary string?
     01DA 001C
0494 01DC 16CC          JNE  ASSG56             No - more complicated
0495 01DE C120          MOV  @FAC4,R4           Pick up direct ptr to string
     01E0 0178'
0496 01E2
0497                *       Common string code to set forward and back pointers
0498 01E2
0499 01E2 C1A0  ASSG55 MOV   @ARG,R6           Ptr to symbol table pointer
     01E4 01C2'
0500 01E6 C044          MOV  R4,R1              Pointer to source string
0501 01E8 06A0          BL   @STVDP3            Set the backpointer
     01EA 01CC'
0502 01EC C060  ASSG57 MOV   @ARG,R1           Address of symbol table ptr
     01EE 01E4'
0503 01F0 C184          MOV  R4,R6              Pointer to string
0504 01F2 06A0          BL   @STVDP             Set the forward pointer
     01F4 0000
0505 01F6 045A          B    *R10               Done - return
0506 01F8
0507                *       Symbol-to-symbol assignment of strings
0508                *          Must create copy of string
0509 01F8
0510 01F8 C820  ASSG56 MOV @FAC6,@BYTE          Fetch length for GETSTR
     01FA 01D0'
     01FC 0000
0511 01FE
0512                *       NOTE:FAC     through    FAC+7 cannot be destroyed
0513                *          addr^of string  length^of string
0514 01FE
0515 01FE 06A0          BL   @VPUSH             So save it on the stack
     0200 0034'
0516 0202 C80A          MOV  R10,@FAC           Save return link in FAC since
     0204 01D6'
0517                *                                  GETSTR does not destroy FAC
0518 0206 06A0          BL   @GETSTR            Call GPL to do the GETSTR
     0208 001C'
0519 020A C2A0          MOV  @FAC,R10           Restore return link
     020C 0204'
0520 020E 06A0          BL   @VPOP              Pop the source info back
     0210 0116'
0521 0212
0522                *       Set up to copy the source string into the destination
0523 0212
0524 0212 C0E0          MOV  @FAC4,R3           R3 is now copy-from
     0214 01E0'
0525 0216 C160          MOV  @SREF,R5           R5 is now copy-to
     0218 0000
0526 021A C105          MOV  R5,R4              Save for pointer setting
0527 021C
0528                *       Registers to be used in the copy
0529                *       R1 - Used for a buffer
0530                *       R3 - Copy-from address
0531                *       R2 - # of bytes to be moved
0532                *       R5 - Copy-to address
0533 021C
0534 021C C0A0          MOV  @FAC6,R2           Fetch the length of the string
     021E 01FA'
```

```
0525                *------------------CONDITIONAL ASSEMBLY--------------------*
0526                        ASMIF VERS=DX10
0527                ASSG59  BL    @GETV1              Get a character
0528                        MOV   R5,R14              Load out destination address
0529                        AI    R14,VRAM            Add VDP offset
0540                        INC   R5                  Increment for next character
0541                        INC   R3                  Increment the copy-from
0542                        MOVB  R1,*R14             Put the character out
0543
0544                        ASMELS
0545 0220
0546 0220 0265              ORI   R5,WRVDP            Enable the VDP write
     0222 0000
0547 0224 06A0      ASSG59  BL    @GETV1              Get the character
     0226 00DC'
0548 0228 D7E0              MOVB  @R5LB,*R15          Load out destination address
     022A 0000
0549 022C 0583              INC   R3                  Increment the copy-from
0550 022E D7C5              MOVB  R5,*R15             1st byte of address too
0551 0230 0585              INC   R5                  Increment for next character
0552 0232 D801              MOVB  R1,@VDPWD           Put the character out
     0234 0000
0553                        ASMEND
0554                *---------------END OF CONDITIONAL ASSEMBLY----------------*
0555 0236 0602              DEC   R2                  Decrement count - finished?
0556 0238 15F5              JGT   ASSG59              No- loop for more
0557 023A 10D3              JMP   ASSG55              Yes - now set pointers
0558 023C
0559                *       Code to copy a numeric value into the symbol table
0560 023C
0561 023C 0202      ASSG70  LI    R2,8                Need to assign 8 bytes
     023E 0008
0562 0240 C160              MOV   @ARG4,R5            Destination pointer(R5),
     0242 01C6'
0563                *                                   from buffer(R4), (R2)bytes
0564 0244 C0E0              MOV   @RAMTOP,R3          Does ERAM exist ?
     0246 0160'
0565 0248 160C              JNE   ASSG77              Yes -write to ERAM
0566                *                                 No - write to VDP
0567                *--------------------CONDITIONAL ASSEMBLY--------------------*
0568                        ASMIF VERS=DX10
0569                        AI    R5,VRAM             Add in VDP offset
0570                        LI    R4,FAC              Source is FAC
0571                ASSG75  MOVB  *R4+,*R5+           Move a byte
0572
0573                        ASMELS
0574 024A
0575 024A D7E0              MOVB  @R5LB,*R15          Load out 2nd byte of address
     024C 022A'
0576 024E 0265              ORI   R5,WRVDP            Enable the write to the VDP
     0250 0222'
0577 0252 D7C5              MOVB  R5,*R15             Load out 1st byte of address
0578 0254 0204              LI    R4,FAC              Soruce is FAC
     0256 020C'
0579 0258 D834      ASSG75  MOVB  *R4+,@VDPWD         Move a byte
     025A 0234'
0580                        ASMEND
0581                *---------------END OF CONDITIONAL ASSEMBLY----------------*
0582 025C 0602              DEC   R2                  Decrement the counter - done?
0583 025E 15FC              JGT   ASSG75              No - loop for more
```

```
0584 026C 045A          B     *R10              Yes - return to the caller
0585 026E 0204  ASSG77  LI    R4,FAC            Source is in FAC
     0270 0256'
0586                   *-----------------CONDITIONAL ASSEMBLY----------------------*
0587                    ASMIF VERS=DX10
0588                    AI    R5,GRAM           Add in ERAM offset
0589                    ASMEND
0590                   *-----------------END OF CONDITIONAL ASSEMBLY----------------*
0591 0272 DD74  ASSG79  MOVB  *R4+,*R5+         Move a byte
0592 0274 0602          DEC   R2                Decrement the counter - done?
0593 0276 15FD          JGT   ASSG79            No - loop for more
0594 0278 045A          B     *R10              Yes - return to the caller
```

```
0597                    *       Check for required token
0598 026E               SYNCHK
0599                    *----------------------CONDITIONAL ASSEMBLY-----------------*
0600                            ASMIF VERS=DX10
0601                            MOVB *R15+,R0           Read required token
0602                            ASMELS
0603 026E D01D                  MOVB *R13,R0            Read required token
0604                            ASMEND
0605                    *-----------------END OF CONDITIONAL ASSEMBLY--------------*
0606 0270
0607 0270 9800                  CB   R0,@CHAT           Have the required token?
     0272 0000
0608 0274 1304                  JEQ  PGMCH              Yes-read next character
0609 0276 06A0                  BL   @SETREG            Error return requires R8/9 set
     0278 003E'
0610 027A 0460                  B    @ERRSYN            * SYNTAX ERROR
     027C 0148'
0611 027E
0612                    *       PGMCH - GPL entry point for PGMCHR to set up
0613                    *               registers
0614 027E
0615 027E C30B          PGMCH   MOV  R11,R12            Save return address
0616 0280 06A0                  BL   @PGMCHR            Get the next token
     0282 019C'
0617 0284 D808                  MOVB R8,@CHAT           Put it in for GPL
     0286 0272'
0618 0288 045C                  B    *R12               Return to GPL
0619 028A
0620 028A
0621 028A
0622                    *       GETV - Routine used to read from the VDP
0623 028A
0624                    *----------------------CONDITIONAL ASSEMBLY-----------------*
0625                            ASMIF VERS=DX10
0626                    GETV    MOV  *R11+,R3           Get address to read from
0627                            MOV  *R3,R3             Indirect through it
0628                    GETV1   MOV  R3,R14             Use R14 for read
0629                            AI   R14,VRAM           Add in VDP offset
0630                            MOVB *R14+,R1           Read the byte
0631
0632                            ASMELS
0633 028A
0634                    *       USES CONSOLE ROUTINE FOR SPEED
0635                    *GETV    MOV  *R11+,R3           Get address to read from
0636                    *        MOV  *R3,R3             Indirect through it
0637                    *GETV1   MOVB @R3LB,*R15         Write 2nd byte of address
0638                    *        NOP                     Kill some time
0639                    *        MOVB R3,*R15            Write 1st byte of address
0640                    *        NOP                     Kill some more time
0641                    *        MOVB @VDPRD,R1          Read the byte
0642                            ASMEND
0643                    *-----------------END OF CONDITIONAL ASSEMBLY--------------*
0644 028A 045B                  RT                      And return to the caller
0645 028C
0646 028C
0647 028C
0648 028C
0649 028C C13B          PUTV    MOV  *R11+,R4
0650 028E C114                  MOV  *R4,R4
0651                    *----------------CONDITIONAL ASSEMBLY----------------------*
```

```
0652                        ASMIF VERS=DX10
0653                PUTV1   MOV  R4,R14
0654                        AI   R14,VRAM
0655                        MOVB R1,*R14
0656
0657                        ASMELS
0658 0290
0659 0290 D7E0      PUTV1   MOVB @R4LB,*R15
     0292 0000
0660 0294 0264              ORI  R4,WRVDP
     0296 0250'
0661 0298 D7C4              MOVB R4,*R15
0662 029A 1000              NOP
0663 029C D801              MOVB R1,@VDPWD
     029E 025A'
0664                        ASMEND
0665                *-----------------END OF CONDITIONAL ASSEMBLY----------------*
0666 02A0 045B              RT
0667 02A2
0668 02A2
0669 02A2
0670                *       MOVFAC - copies 8 bytes from VDP(@FAC4)or ERAM(@FAC4)
0671                *                                              TO FAC
0672 02A2 C060      MOVFAC  MOV  @FAC4,R1           Get pointer to source
     02A4 0214'
0673 02A6 0202              LI   R2,8               8-byte values
     02A8 0008
0674 02AA 0203              LI   R3,FAC             Destination is FAC
     02AC 0264'
0675 02AE C020              MOV  @RAMTOP,R0         Does ERAM exist?
     02B0 0246'
0676 02B2 160A              JNE  MOVFA2             Yes - from ERAM
0677                *                               No - from VDP RAM
0678                *-----------------------CONDITIONAL ASSEMBLY-----------------*
0679                        ASMIF VERS=DX10
0680                        AI   R1,VRAM            Add in VDP offset
0681                MOVF1   MOVB *R1+,*R3+          Move a byte
0682                        DEC  R2                 Decrement counter - done?
0683                        JGT  MOVF1              No - loop for more
0684                        RT                      Yes - return to caller
0685
0686                MOVFA2  AI   R1,GRAM            Add in ERAM offset
0687                        JMP  MOVF1
0688
0689                        ASMELS
0690 02B4
0691 02B4 06C1              SWPB R1
0692 02B6 D7C1              MOVB R1,*R15            Load 2nd byte of address
0693 02B8 06C1              SWPB R1
0694 02BA D7C1              MOVB R1,*R15            Load 1st byte of address
0695 02BC 0205              LI   R5,VDPRD
     02BE 0000
0696 02C0 DCD5      MOVF1   MOVB *R5,*R3+           Move a byte
0697 02C2 0602              DEC  R2                 Decrement counter - done?
0698 02C4 15FD              JGT  MOVF1              No - loop for more
0699 02C6 045B              RT                      Yes - return to caller
0700 02C8
0701 02C8 DCF1      MOVFA2  MOVB *R1+,*R3+
0702 02CA 0602              DEC  R2
0703 02CC 16FD              JNE  MOVFA2
```

```
0704 020E 0438          RT
0707                    ASMEND
070B          *------------------END OF CONDITIONAL ASSEMBLY--------------*
```

```
0708                *       STVDP3 - sets backpointers for strings
0709 02E0
0710                *--------------------CONDITIONAL ASSEMBLY----------------------
0711                        ASMIF VERS=0X10
0712                STVDP3  AI    R1,-3             Point at the backpointer
0713                STVDP   MOV   R1,R14            User R14 for the write
0714                        AI    R14,VRAM          Add in VDP offset
0715                        MOVB  R6,*R14+          Move the 1st byte of pointer
0716                        MOVB  @R6LB,*R14        Move the 2nd byte of pointer
0717
0718                        ASMELS
0719 02D0
0720                *       USES CONSOLE ROUTINE FOR SPEED
0721                *STVDP3 AI    R1,-3              Point at the backpointer
0722                *STVDP   MOVB  @R1LB,*R15        Load 2nd byte of address
0723                *        ORI   R1,WRVDP          Enable the VDP write
0724                *        MOVB  R1,*R15           Load 1st byte of address
0725                *        NOP                     Kill some time
0726                *        MOVB  R6,@VDPWD         Move the 1st byte of pointer
0727                *        MOVB  @R6LB,@VDPWD      Move the 2nd byte of pointer
0728                        ASMEND
0729                *----------------END OF CONDITIONAL ASSEMBLY-----------------*
0730 02D0 045B              RT                      And return to the caller
0731                        END
NO ERRORS,    0003 WARNINGS
```

```
$        02D2'        0369
ARG    R 01EE    0077 0485 0499 0502
ARG2   R          0077
ARG4   R 0242'    0077 0480 0486 0562
ARGTST R 01AC'    0069 0472
ASSG   D 01AB'    0471 0061 0222
ASSG54   01CE'    0490 0482
ASSG55   01E2'    0499 0557
ASSG56   01F8'    0510 0494
ASSG57   01EC'    0502 0491
ASSG59   0224'    0547 0556
ASSG70   023C'    0561 0477
ASSG75   0258'    0579 0583
ASSG77   0262'    0585 0565
ASSG79   0266'    0591 0593
ASSGNV D 0026'    0222 0061
BASE   R 0124'    0070 0370
BERMUV   00B4'    0319 0268
BIT2     0133'    0369 0104
BYTE   R 01FC'    0074 0510
C4       0002'    0095 0361
CFT    R 0108'    0070 0359
CHAT   R 0256'    0074 0607 0617
COMMA$   00B3     0088 0382
COMPCG D 0014'    0204 0063
COMPCT R 0016'    0067 0204
DK1Q     0001     0003 0004 0080 0118 0536 0568 0587 0600 0625 0652
                       0679 0711
ERR    R 0182'    0071 0414
ERR1     0146'    0384 0394 0435
ERR3     017C'    0413 0361 0368 0373
ERRBS    0503     0091 0413
ERRMUV R 00B6'    0066 0319
ERRSYN R 027C'    0071 0384 0610
ERRT   D 0184'    0417 0062 0355
ERRTM    0603     0092 0417
ERRX     0180'    0414 0418
FAC    R 02AC'    0076 0260 0263 0328 0330 0362 0397 0428 0492 0516
                       0519 0578 0585 0674
FAC10  R 010C'    0076 0358 0360
FAC15  R 0198'    0076 0427 0431
FAC2   R 013C'    0076 0272 0327 0344 0354 0380
FAC4   R 02A4'    0076 0260 0261 0280 0281 0304 0308 0310 0328 0332
                       0365 0379 0403 0404 0405 0409 0495 0524 0672
FAC5   R          0076
FAC6   R 021E'    0076 0338 0350 0377 0490 0510 0534
FBS    R 01A2'    0083 0102 0434
FBS001 R          0083
FBSYMB D 0004'    0102 0058
GET    R 01C0'    0078 0279 0307 0329 0364 0402 0484
GET1   R          0078
GETG   R 009E'    0078 0303
GETSTG D 001A'    0210 0063
GETSTK R 0050'    0072 0245
GETSTR R 0208'    0067 0210 0518
GETV   R 0156'    0083 0262 0396
GETV1  R 0226'    0083 0336 0547
GRAM   R          0078
LPAR$    00B7     0090 0325 0353
MOVF1    02C0'    0696 0698
```

```
MOVFA2      0209'   0701  0676  0703
MOVFAC   D  02A3'   0672  0059
NEXT     R  0012'   0075  0105
P359        0000    0003  0003
PGMCH    D  03E7'   0615  0061  0608
PGMCHR   R  0333'   0069  0351  0395  0432  0616
PGPSTK   R  0102'   0072  0474
PSHFRS   R  00F6'   0071  0352
PUTSTK   R  003A'   0072  0237
PUTV     D  02BC'   0649  0060
PUTV1    D  0290'   0659  0060
R0          0000           0413  0417  0603  0607  0675
R1          0001           0265  0266  0267  0277  0281  0310  0332  0333  0337
                           0338  0367  0376  0377  0404  0429  0436  0480  0486
                           0500  0502  0552  0663  0672  0691  0692  0693  0694
                           0701
R10         000A           0471  0505  0516  0519  0584  0594
R11         000B           0236  0259  0315  0429  0471  0615  0649
R12         000C           0473  0476  0615  0618
R13         000D           0603
R14         000E           0548  0550  0575  0577  0659  0661  0692  0694
R14LB    R          0073
R2          0002           0266  0300  0348  0350  0378  0399  0405  0408  0409
                           0428  0430  0534  0555  0561  0582  0592  0673  0697
                           0702
R3          0003           0333  0335  0400  0492  0493  0524  0549  0564  0674
                           0696  0701
R4          0004           0265  0270  0270  0286  0286  0289  0290  0297  0344
                           0360  0370  0490  0495  0500  0503  0526  0578  05
                           0585  0591  0649  0650  0650  0660  0661
R4LB     R  0292'   0073   0659
R5          0005           0362  0367  0372  0378  0525  0526  0546  0550  0551
                           0562  0576  0577  0591  0695  0696
R5LB     R  0240'   0073   0548  0575
R6          0006           0204  0210  0216  0222  0228  0234  0241  0487  0499
                           0503
R6LB     R          0073
R7          0007           0236  0240  0242  0246
R7LB     R          0073
R8          0008           0325  0382  0393  0430  0617
R9          0009           0239  0240  0242  0243  0258  0259  0315  0316
RAMTOP   R  02B0'   0078   0297  0400  0564  0675
RESET    R  0008'   0066   0103
RPAR$       00B6    0089   0393
SAVREG   R  004C'   0069   0244
SET      D  000A'   0104   0084
SETREG   R  0278'   0069   0238  0609
SMB      D  0054'   0258   0059  0216
SMB010      00AE'   0315   0298  0339  0406
SMB02       007A'   0277   0271
SMB020      00E6'   0343   0291
SMB025      00EC'   0349   0383
SMB04       0088'   0286   0278
SMB040      012E'   0376   0371
SMB041      0130'   0377   0374
SMB05       008C'   0288   0326
SMB050      00B8'   0324   0287
SMB06       00A4'   0307   0301
SMB070      014A'   0393   0381
SMB071      016E'   0405   0401
```

```
SMBOB      004A'   0310  0305
SMB51      003E'   0327  0410
SMB57      00E0'   0338  0334
SMB71      0174'   0408  0398
SMBB    D  0020'   0216  0059
SMBB1C     003B'   0235  0205  0211  0217  0223  0229
SREF    R  0218    0074  0525
STATUS  R  000E'   0075  0104
STCODE     0000'   0094  0327  0354
STVDP   R  01F4'   0083  0504
STVDP3  R  01EA'   0083  0488  0501
SYM     D  018A'   0427  0059  0228
SYM1       0194'   0430  0433
SYMB    D  0020'   0228  0059
SYMTAB  R          0074
SYNCHK  D  026E'   0598  0058
VDPRD   R  028E'   0065  0695
VDPWD   R  029E'   0065  0552  0579  0663
VERMAC  M          A0001  0003
VER        0000    0003  0004  0080  0118  0536  0568  0587  0600  0625  0652
                         0679  0711
VPOP    R  0210'   0070  0363  0520
VPUSH   R  0200'   0070  0234  0515
VPUSHG  D  0032'   0234  0061
VRAM    R          0068
WRVDP   R  0296'   0065  0546  0576  0660
```