```
SOURCE ACCESS NAME=      PPC2.P359.SRC.STRING
OBJECT ACCESS NAME=      PPC2.P359.OBJ.STRINGS
LISTING ACCESS NAME=     PPC2.P359.LST.STRINGS
ERROR ACCESS NAME=       .XMAERR
OPTIONS=                 XREF
MACRO LIBRARY PATHNAME=


LINE      KEY   NAME

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

```
0031                    IDT  'STRING'
0032                ********************************************************************
0033                *
0034                *        SSS   TTTTT  RRRR   IIIII  N      N   GGG              *
0035                *       S   S    T    R   R    I    NN     N  G   G             *
0036                *       S        T    R   R    I    N N    N  G                 *
0037                *        SSS     T    RRRR     I    N  N   N  G  GG             *
0038                *           S    T    R R      I    N   N  N  G   G             *
0039                *       S   S    T    R  R     I    N    NN  G   G             *
0040                *        SSS     T    R   R  IIIII  N      N   GGG              *
0041                *                                                               *
0042                *             PPPPP      333    55555    999                   *
0043                *             P    P    3   3   5       9   9                  *
0044                *             P    P        3   5       9   9                  *
0045                *             PPPPP      333    555     9999                   *
0046                *             P             3       5      9                    *
0047                *             P          3   3   5   5     9                    *
0048                *             P           333     555     9                     *
0049                *                                                               *
0050                ********************************************************************
```

```
0055                    DEF   MEMCHG, MEMCHK, GETSTR, COMPCT, NSTRCN
0056 0000
0057                    REF   VDPWD, WRVDP
0058                    REF   R0LB, R3LB, R4LB, R6LB
0059                    REF   FAC, FAC6
0060                    REF   CBH65, GRAM, VRAM
0061                    REF   STRSP, STREND, BYTE, FREPTR, VSPTR, ERRCOD, PGMPTR
0062                    REF   RAMFLG, SREF
0063                    REF   GETV1, PUTV, PUTV1, VPSH23, STVDP, PGMCHR, CONT
0064                    REF   GET1, SET, RESET
```

```
0067                    ************************************************************
0068                    *
0069                    *                    MEMORY CHECK ROUTINE
0070                    *                                                            *
0071                    *       IT CHECKS TO SEE IF THERE IS ENOUGH ROOM TO           *
0072                    *       INSERT A SYMBOL TABLE ENTRY OR A P.A.B. INTO         *
0073                    *       THE VDP BETWEEN THE STATIC SYMBOL TABLE/PAB          *
0074                    *       AREA AND THE DYNAMIC STRING AREA.  IF THERE          *
0075                    *       IS ENOUGH SPACE IT SIMPLY RETURNS.  IF THERE         *
0076                    *       IS NOT IT ATTEMPTS TO MOVE THE STRING SPACE          *
0077                    *       DOWN(TO LOWER ADDRESS) AND THEN INSERT THE           *
0078                    *       NEEDED AREA BETWEEN THE TWO.  NOTE: IT MAY           *
0079                    *       INVOKE COMPCT TO DO A GARBAGE COLLECTION.  IF        *
0080                    *       THERE IS NOT ENOUGH SPACE AFTER COMPCT THEN          *
0081                    *       ISSUES  *MEMORY FULL   MESSAGE.                      *
0082                    *                                                            *
0083                    *       INPUT:  # OF BYTES NEEDED IN  FAC,FAC+1              *
0084                    *       USES:   R0,R12 AS TEMPORARIES AS WELL AS R0-R6       *
0085                    *               WHEN INVOKING COMPCT                         *
0086                    ************************************************************
0087 0000 06A0  MEMCHG  BL    @MEMCHK           GPL entry point
     0002 000A'
0088 0004 0000          DATA  SET               If NOT enough memory
0089 0006 0460          B     @RESET            If enough memory
     0008 0000
0090 000A
0091 000A C30B  MEMCHK  MOV   R11,R12           Save return address
0092 000C C020          MOV   @FREPTR,R0        GET BEGINNING OF S.T. FREE S
     000E 0000
0093 0010 6020          S     @STRSP,R0         CALCULATE SIZE OF GAP
     0012 0000
0094 0014 8020          C     @FAC,R0           ENOUGH SPACE ALREADY?
     0016 0000
0095 0018 1A3C          JL    MEMC08            YES - DONE - RTN
0096 001A 06A0          BL    @COMPCT           NO-COMPACTIFY STRING SPACE
     001C 010A'
0097 001E C020          MOV   @STREND,R0        GET STRING FREE SPACE
     0020 0000
0098 0022 6020          S     @VSPTR,R0         CALCULATE SIZE OF GAP
     0024 0000
0099 0026 0220          AI    R0,-64            VSPTR OFFSET TOO
     0028 FFC0
0100 002A C2A0          MOV   @FAC,R10          GET TOTAL # NEEDED BACK
     002C 0016'
0101 002E 8280          C     R0,R10            ENOUGH ROOM NOW?
0102 0030 1A32          JL    MEMERR            NO-   *MEMORY FULL
0103                    *
0104                    *******NOW MOVE THE DYNAMIC STRING AREA DOWN IN MEMORY
0105                    *
0106 0032 C020          MOV   @STRSP,R0         CALCULATE # OF BYTES
     0034 0012'
0107 0036 C0A0          MOV   @STREND,R2        Beginning of move address
     0038 0020'
0108 003A 6002          S     R2,R0                in the total string spac
0109 003C 680A          S     R10,@STREND       SET FREE PTR(COPY-TO ADDRESS,
     003E 0038'
0110 0040 C000          MOV   R0,R0             NO BYTES TO MOVE?
0111 0042 130D          JEQ   MEMC04            RIGHT
0112 0044 C0C2          MOV   R2,R3             ADDR FOR GETV
0113 0046 0583          INC   R3
```

```
0114 0048 C120          MOV   @STREND,R4        ADDR FOR PUTV
     004A 003E'
0115 004C 0584          INC   R4
0116 004E 06A0  MEMC03  BL    @GETV1            GET THE BYTE
     0050 0000
0117 0052 06A0          BL    @PUTV1            PUT THE BYTE
     0054 0000
0118 0056 0583          INC   R3                INC THE FROM
0119 0058 0584          INC   R4                INC THE TO
0120 005A 0600          DEC   R0                DEC THE COUNT
0121 005C 15F8          JGT   MEMC03            IF NOT DONE
0122              *                             MOVE IT
0123 005E 680A  MEMC04  S     R10,@STRSP        SET NEW STRING SPACE PTR
     0060 0034'
0124              *
0125              *******NOW FIX UP STRING PTRS
0126              *
0127 0062 C020          MOV   @STRSP,R0         GET BEGINNING OF STRING SPACE
     0064 0060'
0128 0066 8020  MEMC05  C     @STREND,R0        FINISHED?
     0068 004A'
0129 006A 1413          JHE   MEMC08            YES
0130 006C 04C1          CLR   R1                CLEAR LOWER BYTE
0131 006E C0C0          MOV   R0,R3             FOR GETV
0132 0070 06A0          BL    @GETV1            GET LENGTH BYTE
     0072 0050'
0133 0074 06C1          SWPB  R1                SWAP FOR ADD
0134 0076 6001          S     R1,R0             POINT AT BEGINNING OF STRING
0135 0078 C0C0          MOV   R0,R3             FOR THE GETV1 BELOW
0136 007A 0223          AI    R3,-3             POINT AT THE BACKPOINTER
     007C FFFD
0137 007E 06A0          BL    @GET1             GET THE BACK POINTER
     0080 0000
0138              *                             BOTH BYTES
0139 0082 C041          MOV   R1,R1             FREE STRING?
0140 0084 1303          JEQ   MEMC06            YES
0141 0086 C180          MOV   R0,R6             PTR TO STRING FOR STVDP
0142 0088 06A0          BL    @STVDP            SET FORWARD PTR
     008A 0000
0143 008C 0220  MEMC06  AI    R0,-4             NOW POINT AT NEXT LENGTH
     008E FFFC
0144 0090 10EA          JMP   MEMC05            CONTINUE ON
0145 0092
0146 0092 046C  MEMC08  B     @2(R12)           Return with space allocated
     0094 0002
0147 0096
0148 0096 C31C  MEMERR  MOV   *R12,R12          Pick up error return address
0149 0098 045C          B     *R12              * MEMORY FULL(prescan time)
0150 009A
0151 009A 0460  ERRMEM  B     @VPSH23           * MEMORY FULL(Execution time)
     009C 0000
```

```
0154                ***************************************************************
0155                *                                                             *
0156                *       GETSTR - checks to see if there is enough             *
0157                *            space in the string area to allocate             *
0158                *            a string.  it there is it allocates it.          *
0159                *            if there is not it does a garbage                *
0160                *            collection and once again checks to see          *
0161                *            if there is enough room.  If so it               *
0162                *            allocates it,  if not it issues a                *
0163                *            * MEMORY FULL   message.                         *
0164                *                                                             *
0165                *       INPUT:  #OF BYTES NEEDED IN @BYTE                     *
0166                *       OUTPUT:  PTR TO NEW STRING IN @SREF                   *
0167                *                 BOTH LENGTH BYTES IN PLACE & ZEROED BKPTR   *
0168                *                 @STREND POINTS 1ST FREE BYTE(NEW)           *
0169                *       USES:   R0-R6 TEMPORARIES                             *
0170                *                                                             *
0171                *       Note:   COMPCT allows a buffer zone of 8 stack        *
0172                *               entries above what is there when COMPCT is    *
0173                *               called. This should allow enough space to     *
0174                *               avoid a collision between the string space    *
0175                *               and the stack.  If garbage begins to appear   *
0176                *               in the string space that can't be accounted   *
0177                *               for, the buffer zone will need to be          *
0178                *               increased.                                    *
0179                ***************************************************************
0180 009E C020  GETSTR MOV  @BYTE,R0          GET # OF BYTES NEEDED
     00A0 0000
0181 00A2 C30B        MOV  R11,R12           SAVE RTN ADDR
0182 00A4 8C30        C    *R0+,*R0+         ADJUST FOR BACKPTR & 2 LENGTHS
0183                *                           (INCREMENT BY 4)
0184 00A6 C060        MOV  @STREND,R1        CHECK IF ENOUGH ROOM
     00A8 0068'
0185 00AA 6040        S    R0,R1             BY ADVANCING THE FREE PTR
0186 00AC C0A0        MOV  @VSPTR,R2         GET VALUE STACK PTR
     00AE 0024'
0187 00B0 0222        AI   R2,64             ALLOW BUFFER ZONE
     00B2 0040
0188 00B4 8081        C    R1,R2             ENOUGH SPACE?
0189 00B6 1B0E        JH   GETS10            YES-ALL IS WELL
0190 00B8 06A0        BL   @COMPCT           NO-COMPACTIFY
     00BA 010A'
0191 00BC C0A0        MOV  @VSPTR,R2         GET VALUE STACK POINTER
     00BE 00AE'
0192 00C0 0222        AI   R2,64             ALLOW BUFFER ZONE
     00C2 0040
0193 00C4 C020        MOV  @BYTE,R0          GET # OF BYTES BACK
     00C6 00A0'
0194 00C8 8C30        C    *R0+,*R0+         INCREMENT BY 4
0195 00CA C060        MOV  @STREND,R1        GET NEW END OF STRING SPACE
     00CC 00A8'
0196 00CE 6040        S    R0,R1             ADVANCE IT
0197 00D0 8081        C    R1,R2             ENOUGH SPACE NOW?
0198 00D2 12E3        JLE  ERRMEM            NO *MEMORY FULL
0199                *
0200 00D4 0220  GETS10 AI   R0,-4             GET EXACT LENGTH BACK
     00D6 FFFC
0201 00D8 D060        MOVB @ROLB,R1          STORE ENTRY LENGTH
     00DA 0000
0202 00DC 06A0        BL   @PUTV             PUT THE ENDING LENGTH
```

```
      00DE 0000
0203  00E0 00CC'        DATA STREND              BYTE IN THE STRING
0204  00E2 6800         S    R0,@STREND          PT AT FIRST BYTE OF STRING
      00E4 00E0'
0205  00E6 C820         MOV  @STREND,@SREF       POINT SREF AT THE STRING
      00E8 00E4'
      00EA 0000
0206  00EC 0620         DEC  @STREND             POINT AT LEADING LENGTH BYTE
      00EE 00E8'
0207  00F0 06A0         BL   @PUTV               PUT THE LEADING LENGTH
      00F2 00DE'
0208  00F4 00EE'        DATA STREND              BYTE IN THE STRING
0209  00F6 0660         DECT @STREND             POINT AT BACKPOINTER
      00F8 00F4'
0210  00FA 04C6         CLR  R6                  ZERO FOR THE BACKPOINTER
0211  00FC C060         MOV  @STREND,R1          ADDR OF THE BACKPOINTER
      00FE 00F8'
0212  0100 06A0         BL   @STVDP              CLEAR THE BACKPOINTER
      0102 008A'
0213  0104 0620         DEC  @STREND             POINT AT 1ST FREE BYTE
      0106 00FE'
0214  0108 045C         B    *R12                ALL DONE
```

```
0217                    ***************************************************************
0218                    *                                                             *
0219                    *       COMPCT - Is the string garbage collection routine.    *
0220                    *                It can be invoked by GETSTR or MEMCHK.  It   *
0221                    *                Copies all used strings to the top of the    *
0222                    *                string space suppressing out all of the      *
0223                    *                unused strings.                              *
0224                    *                                                             *
0225                    *       INPUT:  NONE                                          *
0226                    *       OUTPUT: UPDATED @STRSP AND @STREND                    *
0227                    *       USES:   R0-R6 AS TEMPORARIES                          *
0228                    ***************************************************************
0229 010A C1CB  COMPCT  MOV   R11,R7            Save rtn address
0230 010C C020          MOV   @FREPTR,R0        Get pointer to free space
     010E 000E'
0231 0110 C160          MOV   @STRSP,R5         Get pointer to string space
     0112 0064'
0232 0114 C800          MOV   R0,@STRSP         Set new string space pointer
     0116 0112'
0233 0118 0585          INC   R5                Compensate for decrement
0234 011A 0605  COMP03  DEC   R5                Point at length of string
0235 011C 8160          C     @STREND,R5        At end of string space?
     011E 0106'
0236 0120 1A03          JL    COMP05            No-check this string for copy
0237 0122 C800          MOV   R0,@STREND        Yes-set end of free space
     0124 011E'
0238 0126 0457          B     *R7               Return to caller
0239 0128
0240 0128 C085  COMP05  MOV   R5,R2             Copy ptr to end in case move
0241 012A C0C5          MOV   R5,R3             Copy ptr to end to read length
0242 012C 06A0          BL    @GETV1            Read the length byte
     012E 0072'
0243 0130 D181          MOVB  R1,R6             Put it in R6 for adds
0244 0132 0986          SRL   R6,8              Need in LSByte for word
0245 0134 6146          S     R6,R5             Point at the string start
0246 0136 0225          AI    R5,-3             Point at the back pointer
     0138 FFFD
0247 013A C0C5          MOV   R5,R3             Set up for GETV
0248 013C 06A0          BL    @GET1             Get the backpointer
     013E 0080'
0249 0140 C041          MOV   R1,R1             Is this string garbage?
0250 0142 13EB          JEQ   COMP03            Yes-just ignore it
0251 0144
0252                    *       PERTINENT REGISTERS AT THIS POINT:
0253                    *            R0- is where the string will end
0254                    *            R6- # of bytes to be moved(does not
0255                    *                include lengths and backpointer)
0256                    *            R2- points at trailing length byte of string
0257                    *                to be moved
0258                    *       IN GENERAL : MOVE (R6) BYTES FROM VDP(R2-R6) TO
0259                    *                    VDP(R0-R6)   moving backwards i.e.
0260                    *                    the last byte of the entry is moved
0261                    *                    first, then the next to the last byte...
0262 0144 8DB6          C     *R6+,*R6+         INCR by 4 to include overhe
0263 0146 C0C2          MOV   R2,R3             Restore ptr to end of string
0264 0148 C100          MOV   R0,R4             Get ptr to end of string space
0265 014A 06A0  COMP10  BL    @GETV1            Read a byte
     014C 012E'
0266 014E 06A0          BL    @PUTV1            Write a byte
     0150 0054'
```

```
0267 0152 0603          DEC   R3                Decrement source pointer
0268 0154 0604          DEC   R4                Decrement destination pointer
0269 0156 0606          DEC   R6                Decrement the counter
0270 0158 15F8          JGT   COMP10            Loop if not finished
0271 015A 0244          ANDI  R4,>3FFF          Delete VDP write-enable & reg
     015C 3FFF
0272 015E C004          MOV   R4,R0             Set new free space pointer
0273 0160 0584          INC   R4                Point at backptr just moved
0274 0162 C0C4          MOV   R4,R3             Copy pointer to read it
0275 0164 06A0          BL    @GET1             Get the backpointer
     0166 013E'
0276                *   R1 now contains the address of the forward pointer
0277 0168 C183          MOV   R3,R6             Address of the string entry
0278 016A 0226          AI    R6,3              Point at the string itself
     016C 0003
0279                *   R6 now contains the address of the string
0280 016E 06A0          BL    @STVDP            Reset the forward pointer
     0170 0102'
0281 0172 10D3          JMP   COMP03            Loop for next string
```

```
0284                ***************************************************************
0285                *       NSTRCN - Nud for string constants                     *
0286                *       Copies the string into the string space and sets      *
0287                *       up the FAC with a string entry of the following       *
0288                *       form:                                                 *
0289                *                                                             *
0290                *       +----------+-----+----+-----------+-----------+       *
0291                *       |  >001C   | >65 | XX |  Pointer  |   Length  |       *
0292                *       |          |     |    | to string | of string |       *
0293                *       +----------+-----+----+-----------+-----------+       *
0294                *       FAC       +2    +3    +4          +6                  *
0295                ***************************************************************
0296 0174          NSTRCN
0297 0174 06C8             SWPB R8
0298 0176 C808             MOV  R8,@FAC6          Save length
     0178 0000
0299 017A C808             MOV  R8,@BYTE          For GETSTR
     017C 00C6'
0300 017E 06C8             SWPB R8
0301 0180 06A0             BL   @GETSTR           Get result string
     0182 009E'
0302 0184 0200             LI   R0,>001C          Get address of SREF
     0186 001C
0303 0188 0201             LI   R1,FAC            Optimize to save bytes
     018A 002C'
0304 018C CC40             MOV  R0,*R1+           Indicate temporary string
0305 018E DC60             MOVB @CBH65,*R1+       Indicate a string
     0190 0000
0306 0192 DC40             MOVB R0,*R1+           Byte is not used
0307 0194 C460             MOV  @SREF,*R1         Save pointer to string
     0196 00EA'
0308 0198 C0A0             MOV  @BYTE,R2          Get number of bytes to copy in
     019A 017C'
0309 019C 1318             JEQ  NSTR20            If none to copy
0310 019E C111             MOV  *R1,R4            Get pointer to destination
0311 01A0 C0E0             MOV  @PGMPTR,R3        Get pointer to source
     01A2 0000
0312 01A4 D020             MOVB @RAMFLG,R0        ERAM or VDP?
     01A6 0000
0313 01A8 1609             JNE  NSTR10            ERAM
0314 01AA
0315                *      Get the string from VDP
0316 01AA 06A0     NSTR05  BL   @GETV1            Get a byte
     01AC 014C'
0317 01AE 06A0             BL   @PUTV1            Put a byte
     01B0 0150'
0318 01B2 0583             INC  R3                Next in source
0319 01B4 0584             INC  R4                Next in destination
0320 01B6 0602             DEC  R2                1 less to move
0321 01B8 16F8             JNE  NSTR05            If more to move-do it
0322 01BA 1009             JMP  NSTR20            Else if done- exit
0323 01BC
0324                *      Get the string from ERAM
0325                *--------------------CONDITIONAL ASSEMBLY----------------
0326                       ASMIF VERS=DX10
0327                NSTR10 AI   R3,GRAM
0328                       AI   R4,VRAM
0329                NSTR15 MOVB *R3+,*R4+          COPY THE BYTE
0330
0331                       ASMELS
```

```
 0332 01BC
 0333 01BC D7E0  NSTR10 MOVB @R4LB,*R15        Write 2nd byte of VDP addr
      01BE 0000
 0334 01C0 0264         ORI  R4,WRVDP          Enable VDP write
      01C2 0000
 0335 01C4 D7C4         MOVB R4,*R15           Write 1st byte of VDP addr
 0336 01C6 D833  NSTR15 MOVB *R3+,@VDPWD       Move byte from ERAM to VDP
      01C8 0000
 0337                   ASMEND
 0338                  *-----------------END OF CONDITIONAL ASSEMBLY--------------*
 0339 01CA 0602         DEC  R2                1 less to move
 0340 01CC 16FC         JNE  NSTR15            If not done - loop for more
 0341 01CE       NSTR20
 0342 01CE A820         A    @FAC6,@PGMPTR     Skip the string
      01D0 0178'
      01D2 01A2'
 0343 01D4 06A0         BL   @PGMCHR           Get character following string
      01D6 0000
 0344 01D8 0460         B    @CONT             And continue on
      01DA 0000
 0345                   END
NO ERRORS,      NO WARNINGS
```

```
BYTE    R  019A'   0061  0180  0193  0299  0308
CBH65   R  0190'   0060  0305
COMP03     011A'   0234  0250  0281
COMP05     0128'   0240  0236
COMP10     014A'   0265  0270
COMPCT  D  010A'   0229  0055  0096  0190
CONT    R  01DA'   0063  0344
DX10       0001    0003  0004  0326
ERRCOD  R          0061
ERRMEM     009A'   0151  0198
FAC     R  018A'   0059  0094  0100  0303
FAC6    R  01D0'   0059  0298  0342
FREPTR  R  010E'   0061  0092  0230
GET1    R  0166'   0064  0137  0248  0275
GETS10     00D4'   0200  0189
GETSTR  D  009E'   0180  0055  0301
GETV1   R  01AC'   0063  0116  0132  0242  0265  0316
GRAM    R          0060
MEMC03     004E'   0116  0121
MEMC04     005E'   0123  0111
MEMC05     0066'   0128  0144
MEMC06     008C'   0143  0140
MEMC08     0092'   0146  0095  0129
MEMCHG  D  0000'   0087  0055
MEMCHK  D  000A'   0091  0055  0087
MEMERR     0096'   0148  0102
NSTR05     01AA'   0316  0321
NSTR10     01BC'   0333  0313
NSTR15     01C6'   0336  0340
NSTR20     01CE'   0341  0309  0322
NSTRCN  D  0174'   0296  0055
P359       0000    0003  0003
PGMCHR  R  01D6'   0063  0343
PGMPTR  R  01D2'   0061  0311  0342
PUTV    R  00F2'   0063  0202  0207
PUTV1   R  01B0'   0063  0117  0266  0317
R0         0000          0092  0093  0094  0097  0098  0099  0101  0106  0108
                         0110  0110  0120  0127  0128  0131  0134  0135  0141
                         0143  0180  0182  0182  0185  0193  0194  0194  0196
                         0200  0204  0230  0232  0237  0264  0272  0302  0304
                         0306  0312
R0LB    R  00DA'   0058  0201
R1         0001          0130  0133  0134  0139  0139  0184  0185  0188  0195
                         0196  0197  0201  0211  0243  0249  0249  0303  0304
                         0305  0306  0307  0310
R10        000A          0100  0101  0109  0123
R11        000B          0091  0181  0229
R12        000C          0091  0146  0148  0148  0149  0181  0214
R15        000F          0333  0335
R2         0002          0107  0108  0112  0186  0187  0188  0191  0192  0197
                         0240  0263  0308  0320  0339
R3         0003          0112  0113  0118  0131  0135  0136  0241  0247  0263
                         0267  0274  0277  0311  0318  0336
R3LB    R          0058
R4         0004          0114  0115  0119  0264  0268  0271  0272  0273  027
                         0310  0319  0334  0335
R4LB    R  01BE'   0058  0333
R5         0005          0231  0233  0234  0235  0240  0241  0245  0246  0247
R6         0006          0141  0210  0243  0244  0245  0262  0262  0269  0277
                         0278
```

```
R6LB    R           0058
R7          0007            0229    0238
R8          0008            0297    0298    0299    0300
RAMFLG  R   01A6'   0062    0312
RESET   R   0008'   0064    0089
SET     R   0004'   0064    0088
SREF    R   0196'   0062    0205    0307
STREND  R   0124'   0061    0097    0107    0109    0114    0128    0184    0195    0203    0204
                            0205    0206    0208    0209    0211    0213    0235    0237
STRSP   R   0116'   0061    0093    0106    0123    0127    0231    0232
STVDP   R   0170'   0063    0142    0212    0280
VDPWD   R   01C8'   0057    0336
VERMAC   M          A0001   0003
VERS        0000    0003    0004    0326
VPSH23  R   009C'   0063    0151
VRAM    R           0060
VSPTR   R   00BE'   0061    0098    0186    0191
WRVDP   R   01C2'   0057    0334
```