```
SOURCE ACCESS NAME=      PPC2.P359.SRC.SCROLL
OBJECT ACCESS NAME=      PPC2.P359.OBJ.SCROLLS
LISTING ACCESS NAME=     PPC2.P359.LST.SCROLLS
ERROR ACCESS NAME=
OPTIONS=                 XREF
MACRO LIBRARY PATHNAME=


LINE      KEY   NAME

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

```
0032                    IDT  'SCROLL'
0033            ********************************************************************
0034            *
0035            *        SSSSS     CCCCC    RRRRRR      OOOOO    L         L        *
0036            *       S     S   C     C   R     R    O     O   L         L        *
0037            *       S         C         R     R    O     O   L         L        *
0038            *        SSSSS    C         RRRRRR     O     O   L         L        *
0039            *             S   C         R   R      O     O   L         L        *
0040            *       S     S   C     C   R    R     O     O   L         L        *
0041            *        SSSSS     CCCCC    R     R     OOOOO    LLLLLLL   LLLLLLL  *
0042            *                                                                    *
0043            *         PPPPPP       33333      555555     99999                  *
0044            *         P     P     3     3     5         9     9                 *
0045            *         P     P           3     5         9     9                 *
0046            *         PPPPPP       33333      55555      999999                 *
0047            *         P                 3          5          9                 *
0048            *         P                 3          5          9                 *
0049            *         P           3     3     5     5    9     9                 *
0050            *         P            33333       5555      99999                  *
0051            *                                                                    *
0052            ********************************************************************
```

```
0057                    DEF   IO,SCROLL
0058 0000
0059                    REF   BDISP,UPDSCR,GROM,GRAM
0060                    REF   R0LB,R1LB,R2LB,R3LB,R9LB,R10LB,WRVDP,VDPRD,VDPWD
0061                    REF   GRMWA,FAC,FAC4,FAC8,CCPADR,DSRFLG,CCPPTR
0062                    REF   PABPTR,GETV1,PUTV1,MNUM,MNUM1,CHAT
0063                    REF   PUTSTK,GETSTK,PAD
0064 0000
0065      0005  FLG     EQU   5
0066 0000
0067 0000
0068                *-------------------CONDITIONAL ASSEMBLY-------------------*
0069                    ASMIF VERS=DX10
0070
0071                SCROLL
0072                    LI    R0,BDISP          Get display address
0073                    LI    R1,32             Second line on display
0074                    AI    R1,BDISP          Get address of second line
0075                    LI    R2,736            Going to move 736 bytes
0076                SCROL1 MOVB *R1+,*R0+       Move a byte
0077                    DEC   R2                One less to move
0078                    JNE   SCROL1            Do more if not done
0079                *       Now, clear the bottom line of the screen
0080                    LI    R1,>7F80          Edge char and space char
0081                    LI    R2,28             28 chars on a line
0082                    LI    R4,>2E0           New line address
0083                    AI    R4,BDISP          Bias to screen buffer
0084                    MOVB  R1,*R4+           Put in 1st edge character
0085                    MOVB  R1,*R4+           Put in 2nd edge character
0086                    SWPB  R1                Bring the space char to bare
0087                SCROL2 MOVB R1,*R4+         Copy space onto screen
0088                    DEC   R2                One less to move
0089                    JNE   SCROL2            Loop if more to do
0090                    SWPB  R1                Bare the edge character again
0091                    MOVB  R1,*R4+           Copy edge character
0092                    MOVB  R1,*R4+           Copy last edge character
0093                    B     @UPDSCR           Update the screen
0094                *                               and return
0095
0096                    ASMELS
0097 0000
0098                *       R12  total number of bytes to move
0099                *       R10  move from
0100                *       R9   move to
0101                *       R8   minor counter (buffer counter)
0102                *       R7   buffer pointer
0103 0000
0104 0000 020C  SCROLL LI   R12,736           Going to move 736 bytes
     0002 02E0
0105 0004 020A          LI   R10,32           Address to move from
     0006 0020
0106 0008 04C9          CLR  R9               Address to move to
0107 000A C18B          MOV  R11,R6           Save return address
0108 000C 06A0          BL @SCR01             Scroll the screen
     000E 0036'
0109 0010 0205          LI   R5,VDPWD         Optimize for speed later
     0012 0000
0110 0014 0204          LI   R4,>02E0         Addr of bottom line of screen
     0016 02E0
0111 0018 0201          LI   R1,>7F80         Edge character and space char
```

```
      001A 7F80
0112  001C 0202          LI    R2,28               28 characters on bottom line
      001E 001C
0113  0020 06A0          BL    @PUTV1              Init VDP & put out 1st edge ch
      0022 0000
0114  0024 D541          MOVB  R1,*R5              Put out 2nd edge char
0115  0026 06C1          SWPB  R1                  Bare the space character
0116  0028 D541  SCRBOT  MOVB  R1,*R5              Write out space char
0117  002A 0602          DEC   R2                  One less to move
0118  002C 16FD          JNE   SCRBOT              Loop if more
0119  002E 06C1          SWPB  R1                  Bare the edge char again
0120  0030 D541          MOVB  R1,*R5              Output edge char
0121  0032 D541          MOVB  R1,*R5              Output edge char
0122  0034 0456          B     *R6                 And return to GPL
0123  0036
0124                *     Generalized move routine
0125  0036
0126  0036 04C8  SCR01   CLR   R8                  Clear minor counter
0127  0038 D7E0          MOVB  @R10LB,*R15         Write LSB the read-address
      003A 0000
0128  003C 02A7          STWP  R7                  Get the WS pointer
0129  003E D7CA          MOVB  R10,*R15            Write out MSB of read-address
0130  0040 DDE0  SCR02   MOVB  @VDPRD,*R7+         Read a byte
      0042 0000
0131  0044 058A          INC   R10                 Inc read-from address
0132  0046 0588          INC   R8                  Inc minor counter
0133  0048 060C          DEC   R12                 Dec total counter
0134  004A 1303          JEQ   SCR04               If all bytes read-write ther
0135  004C 0288          CI    R8,12               Filled WS buffer area?
      004E 000C
0136  0050 11F7          JLT   SCR02               No - read more
0137  0052 D7E0  SCR04   MOVB  @R9LB,*R15          Write LSB of write-address
      0054 0000
0138  0056 0269          ORI   R9,WRVDP            Enable the VDP write
      0058 0000
0139  005A D7C9          MOVB  R9,*R15             Write MSB of write-address
0140  005C 02A7          STWP  R7                  Get WS buffer pointer
0141  005E D837  SCR06   MOVB  *R7+,@VDPWD         Write a byte
      0060 0012'
0142  0062 0589          INC   R9                  Increment write-address
0143  0064 0608          DEC   R8                  Decrement counter
0144  0066 16FB          JNE   SCR06               Move more if not done
0145  0068 C30C          MOV   R12,R12             More on major counter?
0146  006A 16E5          JNE   SCR01               No - go do another read
0147  006C 045B          RT                        Yes - done
0148  006E
0149                     ASMEND
0150                *--------------END OF CONDITIONAL ASSEMBLY----------------*
```

```
0153                ****************************************************************
0154                *       Decode which I/O utility is being called               *
0155                *                                                              *
0156                *       Tag field following the XML IO has the following       *
0157                *       meaning:                                               *
0158                *            0 - Line list - utility to search keyword         *
0159                *                table to restore keyword from token           *
0160                *            1 - Fill space - utility to fill record with      *
0161                *                spaces when outputting incomplete records     *
0162                *            2 - Copy string - utility to copy a string,       *
0163                *                adding the screen offset to each character    *
0164                *                for display purposes                          *
0165                *            3 - Clear ERAM  - utility to clear ERAM at the    *
0166                *                address specified by the data word            *
0167                *                following the IO tag and the # of bytes       *
0168                *                specified by the lenght following the         *
0169                *                address word.  Note that each data word is    *
0170                *                the address of a CPU memory location.         *
0171                ****************************************************************
0172                *------------------CONDITIONAL ASSEMBLY----------------------*
0173                        ASMIF VERS=DX10
0174
0175                IO      MOVB *R15+,R0           read selector from GROM
0176
0177                        ASMELS
0178 006E
0179 006E D01D      IO      MOVB *R13,R0            Read selector from GROM
0180                        ASMEND
0181                *-----------------END OF CONDITIONAL ASSEMBLY---------------*
0182 0070 0980              SRL  R0,8               Shift for decoding
0183 0072 1358              JEQ  LLIST              0 is tag for Line list
0184 0074 0600              DEC  R0
0185 0076 132C              JEQ  FILSPC             1 is tag for Fill space
0186 0078 0600              DEC  R0
0187 007A 130E              JEQ  CSTRIN             2 is tag for Copy string
0188                *                               3 is tag for CLRGRM string
0189                *                                  fall into it
```

```
0191                    *       CLRGRM
0192 007C
0193                    *       R1 - address of clearing start
0194                    *       R2 - number of bytes to clear
0195 007C
0196 007C               CLRGRM
0197                    *--------------------CONDITIONAL ASSEMBLY--------------------
0198                            ASMIF VERS=DX10
0199                            MOVB *R15+,R1           Read address of ERAM pointer
0200                            MOVB *R15+,R2           Read address of byte count
0201                            SRL  R1,8
0202                            SRL  R2,8
0203                            AI   R1,PAD
0204                            AI   R2,PAD
0205                            MOV  *R1,R1             Read the ERAM pointer
0206                            MOV  *R2,R2             Read the byte count
0207                            AI   R1,GRAM            Add in ERAM offset
0208
0209                            ASMELS
0210 007C
0211 007C 0201                  LI   R1,PAD             Get CPU RAM offset
     007E 0000
0212 0080 C081                  MOV  R1,R2              Need for next read too
0213 0082 B81D                  AB   *R13,@R1LB         Add address of ERAM pointer
     0084 0000
0214 0086 C051                  MOV  *R1,R1             Read the ERAM address
0215 0088 B81D                  AB   *R13,@R2LB         Read address of byte count
     008A 0000
0216 008C C092                  MOV  *R2,R2             Read the byte count
0217 008E
0218                            ASMEND
0219                    *---------------END OF CONDTIONAL ASSEMBLY-----------------
0220 008E 04C0                  CLR  R0                 Clear for clearing ERAM
0221 0090 DC40          CLRGR1  MOVB R0,*R1+            Clear a byte
0222 0092 0602                  DEC  R2                 One less to clear-done?
0223 0094 16FD                  JNE  CLRGR1             No - loop for rest
0224 0096 045B                  RT                      Yes-return
```

```
0226                    *       CSTRIN
0227 0098
0228                    *       R0 - MNUM
0229                    *       R1 - GETV/PUTV buffer
0230                    *       R3 - FAC4/GETV address
0231                    *       R4 - CCPADR/PUTV address
0232                    *       R5 - return adress
0233 0098
0234 0098 C14B  CSTRIN  MOV   R11,R5            Save return address
0235 009A D020          MOVB  @MNUM,R0          Get Mnum
     009C 0000
0236 009E 1317          JEQ   CSTR10            If no bytes to copy
0237 00A0 0980          SRL   R0,8              Shift to use as counter
0238 00A2 C120          MOV   @CCPADR,R4        Get copy-to address
     00A4 0000
0239 00A6 C0E0          MOV   @FAC4,R3          Get copy-from address
     00A8 0000
0240 00AA 06A0  CSTR05  BL    @GETV1            Get byte
     00AC 0000
0241 00AE B060          AB    @DSRFLG,R1        Add screen offset
     00B0 0000
0242 00B2 06A0          BL    @PUTV1            Put the offset byte out
     00B4 0022'
0243 00B6 0583          INC   R3                Increment from address
0244 00B8 0584          INC   R4                Increment to address
0245 00BA 0600          DEC   R0                One less to move
0246 00BC 16F6          JNE   CSTR05            Loop if not done
0247 00BE C803          MOV   R3,@FAC4          Restore for GPL
     00C0 00A8'
0248                    *-------------------CONDITIONAL ASSEMBLY-------------------*
0249                            ASMIF VERS=DX10
0250
0251                            MOVB R0,@MNUM           Clear for GPL
0252                            ANDI R4,>BFFF           Throw away VDP write enable
0253                            MOV  R4,@CCPADR         Restore for GPL
0254                            MOV  R5,R11             Restore return to R11
0255                            B    @UPDSCR            Update screen
0256
0257                            ASMEND
0258                    *-----------------END OF CONDITIONAL ASSEMBLY--------------*
0259 00C2
0260 00C2 D800  CSTR07  MOVB  R0,@MNUM          Clear for GPL
     00C4 009C'
0261      00C9' CBHFF   EQU   $+3
0262 00C6 0244          ANDI  R4,>BFFF          Throw away VDP write enable
     00C8 BFFF
0263 00CA C804          MOV   R4,@CCPADR        Restore for GPL
     00CC 00A4'
0264 00CE       FILS$6
0265 00CE 0455  CSTR10  B     *R5               Return
```

```
0267                    *       FILSPC
0268 00D0
0269                    *       R0 - MNUM
0270                    *       R1 - Buffer for GETV/PUTV
0271                    *       R2 - MNUM1
0272                    *       R3 - pointer for GETV
0273                    *       R4 - CCPADR, pointer for PUTV
0274                    *       R5 - return address
0275 00D0
0276 00D0 C14B  FILSPC  MOV   R11,R5            Save return address
0277 00D2 D0A0          MOVB  @MNUM1,R2         Get pointer to end of record
     00D4 0000
0278 00D6 1604          JNE   FILS$1            If spaces to fill for sure
0279 00D8 9802          CB    R2,@CCPPTR        Any filling to do?
     00DA 0000
0280 00DC 1604          JNE   FILS$2            Yes-do it normally
0281 00DE 0455          B     *R5               No-255 record already full
0282 00E0 9802  FILS$1  CB    R2,@CCPPTR        Any filling to do?
     00E2 00DA'
0283 00E4 12F4          JLE   FILS$6            No - record is complete
0284 00E6 70A0  FILS$2  SB    @CCPPTR,R2        Compute # of bytes to change
     00E8 00E2'
0285 00EA B802          AB    R2,@CCPPTR        Point to end
     00EC 00E8'
0286 00EE D020          MOVB  @DSRFLG,R0        Filling with zeroes?
     00F0 00B0'
0287 00F2 160A          JNE   FILS$3            No - fill with spaces
0288 00F4 C0E0          MOV   @PABPTR,R3        Check if internal files
     00F6 0000
0289 00F8 0223          AI    R3,FLG            5-byte offset into PAB
     00FA 0005
0290 00FC 04C1          CLR   R1                Initialize to test below
0291 00FE 06A0          BL    @GETV1            Get byte from PAB
     0100 00AC'
0292 0102 0241          ANDI  R1,>0800          Internal?
     0104 0800
0293 0106 1602          JNE   FILS$4            Yes - zero fill
0294 0108 0220  FILS$3  AI    R0,>2000          Add space character for filler
     010A 2000
0295 010C
0296 010C 0982  FILS$4  SRL   R2,8              Shift count for looping
0297 010E C120          MOV   @CCPADR,R4        Get start address to fill
     0110 00CC'
0298 0112 D040          MOVB  R0,R1             Put filler in place for PUTV
0299 0114 06A0  FILS$5  BL    @PUTV1            Put out a filler
     0116 00B4'
0300 0118 0584          INC   R4                Increment filler pointer
0301 011A 0602          DEC   R2                One less to fill
0302 011C 16FB          JNE   FILS$5            Loop if more
0303 011E D802          MOVB  R2,@MNUM1         Restore for GPL
     0120 00D4'
0304 0122 10CF          JMP   CSTR07
```

```
0306                    *       LLIST
0307 0124
0308                    *       R0 - FAC - address of keytab in GROM
0309                    *       R1 - keyword length
0310 0124
0311 0124 C30B  LLIST   MOV   R11,R12           Save return address
0312 0126 06A0          BL    @PUTSTK           Save GROM address
     0128 0000
0313 012A C020          MOV   @FAC,R0           Get address of keytab
     012C 0000
0314 012E D220          MOVB  @CHAT,R8          Get token to search for
     0130 0000
0315 0132 0201          LI    R1,1              Assume one character keyword
     0134 0001
0316                    *---------------------CONDITIONAL ASSEMBLY------------------*
0317                            ASMIF VERS=DX10
0318
0319                    LLIS$4  MOV   R0,R14            Load GROM address
0320                            AI    R14,GROM          Add offset
0321                            MOVB  *R14+,R3          Read address of table
0322                            MOVB  *R14+,@R3LB       Both bytes
0323
0324                            ASMELS
0325 0136
0326 0136 DB40  LLIS$4  MOVB  R0,@GRMWA(R13)    Load GROM address of table
     0138 0000
0327 013A DB60          MOVB  @R0LB,@GRMWA(R13) Both bytes
     013C 0000
     013E 0138'
0328 0140 D0DD          MOVB  *R13,R3           Read address of x-char table
0329 0142 D81D          MOVB  *R13,@R3LB        Both bytes
     0144 0000
0330 0146
0331                            ASMEND
0332                    *--------------------END OF CONDITIONAL ASSEMBLY------------*
0333 0146 A0C1  LLIS$5  A     R1,R3             Add length of keyword to point
0334                    *                          at token
0335                    *---------------------CONDITIONAL ASSEMBLY------------------*
0336                            ASMIF VERS=DX10
0337
0338                            MOV   R3,R14            Write out new GROM addr
0339                            AI    R14,GROM          Add GROM offset
0340                            MOVB  *R14+,R4          Read token value
0341                            MOVB  *R14+,R5          Read possible end of x-char
0342                    *                               table
0343                            ASMELS
0344 0148
0345 0148 DB43          MOVB  R3,@GRMWA(R13)    Write out new GROM addr
     014A 013E'
0346 014C DB60          MOVB  @R3LB,@GRMWA(R13)    which points to token
     014E 0144'
     0150 014A'
0347 0152 D11D          MOVB  *R13,R4           Read token value
0348 0154 D15D          MOVB  *R13,R5           Read possible end of x-char
0349                    *                               table
0350                            ASMEND
0351                    *---------------------END OF CONDITIONAL ASSEMBLY-----------*
0352 0156 9204          CB    R4,R8             Token value match?
0353 0158 1307          JEQ   LLIS$6            Yes !!! found the keyword
0354 015A 0583          INC   R3                NO-so skip token value
```

```
0355 015C 9805          CB    R5,@CBHFF         Reach end of x-char table?
     015E 00C9'
0356 0160 16F2          JNE   LLIS$5            No-so check more in this tab
0357 0162 05C0          INCT  R0                Point into x+1 char table
0358 0164 0581          INC   R1                Try x+1 char table
0359 0166 10E7          JMP   LLIS$4            Loop to check it
0360 0168
0361              *     Come here when found keyword
0362 0168
0363 0168       LLIS$6
0364 0168 60C1          S     R1,R3             Subtract length to pt at k.w.
0365 016A C803          MOV   R3,@FAC8          Save ptr to keyword for GPL
     016C 0000
0366 016E C801          MOV   R1,@FAC4          Save keyword length for GPL
     0170 00C0'
0367 0172 D808          MOVB  R8,@FAC           Save CHAT for GPL
     0174 012C'
0368 0176 06A0          BL    @GETSTK           Restore GROM address
     0178 0000
0369 017A 045C          B     *R12              And return to GPL
0370 017C
0371                    END
NO ERRORS,      NO WARNINGS
```

```
LABEL     VALUE   DEFN  REFERENCES
$         017C'         0261
BDISP   R           0059
CBHFF     00C9'   0261  0355
CCPADR  R 0110'   0061  0238  0263  0297
CCPPTR  R 00EC'   0061  0279  0282  0284  0285
CHAT    R 0130'   0062  0314
CLRGR1    0090'   0221  0223
CLRGRM    007C'   0196
CSTR05    00AA'   0240  0246
CSTR07    00C2'   0260  0304
CSTR10    00CE'   0265  0236
CSTRIN    0098'   0234  0187
DSRFLG  R 00F0'   0061  0241  0286
DX10      0001    0003  0004  0069  0173  0198  0249  0317  0336
FAC     R 0174'   0061  0313  0367
FAC4    R 0170'   0061  0239  0247  0366
FAC8    R 016C'   0061  0365
FILS$1    00E0'   0282  0278
FILS$2    00E6'   0284  0280
FILS$3    0108'   0294  0287
FILS$4    010C'   0296  0293
FILS$5    0114'   0299  0302
FILS$6    00CE'   0264  0283
FILSPC    00D0'   0276  0185
FLG       0005    0065  0289
GETSTK  R 0178'   0063  0368
GETV1   R 0100'   0062  0240  0291
GRAM    R         0059
GRMWA   R 0150'   0061  0326  0327  0345  0346
GROM    R         0059
IO      D 006E'   0179  0057
LLIS$4    0136'   0326  0359
LLIS$5    0146'   0333  0356
LLIS$6    0168'   0363  0353
LLIST     0124'   0311  0183
MNUM    R 00C4'   0062  0235  0260
MNUM1   R 0120'   0062  0277  0303
P359      0000    0003  0003
PABPTR  R 00F6'   0062  0288
PAD     R 007E'   0063  0211
PUTSTK  R 0128'   0063  0312
PUTV1   R 0116'   0062  0113  0242  0299
R0        0000          0179  0182  0184  0186  0220  0221  0235  0237  0245
                        0260  0286  0294  0298  0313  0326  0357
R0LB    R 013C'   0060  0327
R1        0001          0111  0114  0115  0116  0119  0120  0121  0211  0212
                        0214  0214  0221  0241  0290  0292  0298  0315  0333
                        0358  0364  0366
R10       000A          0105  0129  0131
R10LB   R 003A'   0060  0127
R11       000B          0107  0234  0276  0311
R12       000C          0104  0133  0145  0145  0311  0369
R13       000D          0179  0213  0215  0326  0327  0328  0329  0345  0346
                        0347  0348
R15       000F          0127  0129  0137  0139
R1LB    R 0084'   0060  0213
R2        0002          0112  0117  0212  0216  0216  0222  0277  0279  0282
                        0284  0285  0296  0301  0303
R2LB    R 008A'   0060  0215
R3        0003          0239  0243  0247  0288  0289  0328  0333  0345  0354
```

				0364	0365							
R3LB	R	014E'	0060	0329	0346							
R4		0004		0110	0238	0244	0262	0263	0297	0300	0347	035
R5		0005		0109	0114	0116	0120	0121	0234	0265	0276	028
				0348	0355							
R6		0006		0107	0122							
R7		0007		0128	0130	0140	0141					
R8		0008		0126	0132	0135	0143	0314	0352	0367		
R9		0009		0106	0138	0139	0142					
R9LB	R	0054'	0060	0137								
SCRBOT		0028'	0116	0118								
SCR01		0036'	0126	0108	0146							
SCR02		0040'	0130	0136								
SCR04		0052'	0137	0134								
SCR06		005E'	0141	0144								
SCROLL	D	0000'	0104	0057								
UPDSCR	R		0059									
VDPRD	R	0042'	0060	0130								
VDPWD	R	0060'	0060	0109	0141							
VERMAC	M		A0001	0003								
VERS		0000	0003	0004	0069	0173	0198	0249	0317	0336		
WRVDP	R	0058'	0060	0138								