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

```
0001                    IDT  'SUPPORTS'
0002                   *
0003                   *       VERSION FOR GROM
0004                   *
0005                   *       THIS IS THE SUPPORT FOR 9900 ASSEMBLY LANGUAGE WITH
0006                   *            EXTENDED BASIC ON THE TI 99/4.  THE CALL INIT
0007                   *            SUBPROGRAM WILL LOAD THIS ROUTINE INTO THE
0008                   *            EXPANSION RAM FROM GROM.  TO LOAD THE OBJECT
0009                   *            INTO GROM THE OBJECT MUST BE TEXT EDITED.
0010                   *            IN THE FIRST RECORD CHANGE THE DATA AFTER THE
0011                   *            "9" TAG FROM "E000" OR "2000" TO THE ADDRESS
0012                   *            DESIRED AND CHANGE THE "7" TAG TO A "8" TAG.
0013                   *       NOTE:   BSS INSTRUCTIONS CANNOT BE USED IN THIS
0014                   *            PROGRAM IN ORDER TO LOAD PROPERLY INTO GROM.
0015                   *
0016  0001  ON         EQU  1
0017  0000  OFF        EQU  0
0018                   *
0019  0000  DEBUG      EQU  OFF              FOR DEBUGGER ADDRESSING
0020                   *
0021                        ASMIF DEBUG=ON
0022                   ILFA   EQU  >FFF8
0023                   SET    EQU  >00EC
0024                        ASMELS
0025  4000  ILFA       EQU  >4000
0026  00CE  SET        EQU  >00CE
0027                        ASMEND
0028                   *
0029                   *       SYSTEM EQUATES
0030                   *
0031  000E  SCNKEY     EQU  >000E
0032  6010  EXBASX     EQU  >6010            EXTENDED BASIC XML TABLES (BASE)
0033  830C  BYTE       EQU  >830C
0034  8310  OLDS       EQU  >8310            OLD VALUE PTR BEFORE EVALUATE SUB
0035                   *                     ARGUMENT
0036  8312  COUNT      EQU  >8312            NO. OF ARGUMENT
0037  831C  SREF       EQU  >831C
0038  8322  ERRCOD     EQU  >8322            PLACE TO RETURN ERROR CODE
0039  8343  BASE       EQU  >8343
0040  834A  FAC        EQU  >834A
0041  8354  SCTEMP     EQU  >8354
0042  8355  SCLEN      EQU  >8355
0043  8356  SCNAME     EQU  >8356
0044  8350  FAC6       EQU  FAC+6
0045  836E  VSPTR      EQU  >836E            VALUE STACK PTR
0046  8370  MAXVDP     EQU  >8370
0047  83CC  SNDADD     EQU  >83CC
0048  83CE  STFLGS     EQU  >83CE
0049  83D0  CRULST     EQU  >83D0
0050  83E0  GPLWS      EQU  >83E0            GPL/EXTENDED BASIC WORKSPACE
0051                   *
0052  4000  WRVDP      EQU  >4000            Write enable for VDP
0053  8800  VDPRD      EQU  >8800            VDP read data address
0054  8C00  VDPWD      EQU  >8C00            VDP write data address
0055  8C02  VDPWA      EQU  >8C02            VDP write address address
0056                   *
0057                   *       ERROR EQUATES
0058                   *
0059  0200  ERRNO      EQU  >0200              2   Numeric Overflow
0060  0300  ERRSYN     EQU  >0300              3   Syntax Error
```

```
0061  0400  ERRIBS EQU  >0400    4   Illegal after subprogram
0062  0500  ERRNQS EQU  >0500    5   Unmatched quotes
0063  0600  ERRNTL EQU  >0600    6   Name too long
0064  0700  ERRSNM EQU  >0700    7   String-num mismatch
0065  0800  ERROBE EQU  >0800    8   Option base error
0066  0900  ERRMUV EQU  >0900    9   Improperly used name
0067  0A00  ERRIM  EQU  >0A00   10   Image error
0068  0B00  ERRMEM EQU  >0B00   11   Memory full
0069  0C00  ERRSO  EQU  >0C00   12   Stack overflow
0070  0D00  ERRNWF EQU  >0D00   13   Next without for
0071  0E00  ERRFNN EQU  >0E00   14   For next nesting
0072  0F00  ERRSNS EQU  >0F00   15   Must be in subprogram
0073  1000  ERRRSC EQU  >1000   16   Recursive subprogram call
0074  1100  ERRMS  EQU  >1100   17   Missing subend
0075  1200  ERRRWG EQU  >1200   18   Return without gosub
0076  1300  ERRST  EQU  >1300   19   String truncated
0077  1400  ERRBS  EQU  >1400   20   Bad subscript
0078  1500  ERRSSL EQU  >1500   21   Speech string too long
0079  1600  ERRLNF EQU  >1600   22   Line not found
0080  1700  ERRBLN EQU  >1700   23   Bad line number
0081  1800  ERRLTL EQU  >1800   24   Line too long
0082  1900  ERRCC  EQU  >1900   25   Can't continue
0083  1A00  ERRCIP EQU  >1A00   26   Command illegal in program
0084  1B00  ERROLP EQU  >1B00   27   Only legal in a program
0085  1C00  ERRBA  EQU  >1C00   28   Bad argument
0086  1D00  ERRNPP EQU  >1D00   29   No program present
0087  1E00  ERRBV  EQU  >1E00   30   Bad value
0088  1F00  ERRIAL EQU  >1F00   31   Incorrect argument list
0089  2000  ERRINP EQU  >2000   32   Input error
0090  2100  ERRDAT EQU  >2100   33   Data error
0091  2200  ERRFE  EQU  >2200   34   File error
0092  2400  ERRIO  EQU  >2400   36   I/O error
0093  2500  ERRSNF EQU  >2500   37   Subprogram not found
0094  2700  ERRPV  EQU  >2700   39   Protection violation
0095  2800  ERRIVN EQU  >2800   40   Unrecognized character
0096  2900  WRNNO  EQU  >2900   41   Numeric overflow
0097  2A00  WRNST  EQU  >2A00   42   String truncated
0098  2B00  WRNNPP EQU  >2B00   43   No program present
0099  2C00  WRNINP EQU  >2C00   44   Input error
0100  2D00  WRNIO  EQU  >2D00   45   I/O error
0101  2E00  WRNLNF EQU  >2E00   46   Line not found
0102        *
```

```
0114                    ASMIF DEBUG=ON
0115                      AORG >E000
0116                    ASMELS
0117 2000                 AORG >2000
0118                    ASMEND
0119                *
0110 2000 205A            DATA NAMLNK          XML LINK TO NAME LINK ROUTINE
0111                *
0112 2002 24F4  FFA       DATA IFFA            FIRST FREE ADDRESS (FFA) POINTER
0113 2004 4000  LFA       DATA ILFA            LAST  FREE ADDRESS (LFA) POINTER
0114                *
0115 2006 AA55  HAA       DATA >AA55           CONSTANT TO SIGNAL INITIALIZED MEM.
0116                *
0117                *       UTILITY BLWP VECTORS
0118                *
0119 2008 2038            DATA UTILWS,NUMASG     NUMERIC ASSIGNMENT
     200A 2096
0120 200C 2038            DATA UTILWS,NUMREF     NUMERIC REFERENCE
     200E 217E
0121 2010 2038            DATA UTILWS,STRASG     STRING ASSIGNMENT
     2012 21E2
0122 2014 2038            DATA UTILWS,STRREF     STRING REFERENCE
     2016 2340
0123 2018 2038            DATA UTILWS,XMLLNK     LINK TO SYSTEM UTILITIES
     201A 2432
0124 201C 2038            DATA UTILWS,KSCAN      KEYBOARD SCAN
     201E 246E
0125 2020 2038            DATA UTILWS,VSBW       VDP single byte write
     2022 2484
0126 2024 2038            DATA UTILWS,VMBW       VDP multiple byte write
     2026 2490
0127 2028 2038            DATA UTILWS,VSBR       VDP single byte read
     202A 249E
0128 202C 2038            DATA UTILWS,VMBR       VDP multiple byte read
     202E 24AA
0129 2030 2038            DATA UTILWS,VWTR       VDP write to register
     2032 24B8
0130 2034 2038            DATA UTILWS,ERR        Return error code to Basic
     2036 2090
0131                *
0132                *       UTILTIY WORKSPACE
0133 2038 0000  UTILWS DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
     203A 0000
     203C 0000
     203E 0000
     2040 0000
     2042 0000
     2044 0000
     2046 0000
     2048 0000
     204A 0000
     204C 0000
     204E 0000
     2050 0000
     2052 0000
     2054 0000
     2056 0000
0134       203D  R2LB    EQU   UTILWS+5
0135       2041  R4LB    EQU   UTILWS+9
0136                *
```

```
0137                    ;* D A T A   C O N S T A N T S
0138                    ;
0139  2058    6F        C3H65   BYTE >65
0140  2059    20        EQBIT   BYTE >20
0141                    ;
```

```
0143                    *
0144                    *       NAME LINK ROUTINE
0145                    *
0146                    *            ALC ROUTINE NAME IN FAC
0147                    *
0148 205A 0060  NAMLNK  MOV   @LFA,R1       ADDRESS POINTER
     205C 2004
0149 205E 0281  NAML10  CI    R1,ILFA
     2060 4000
0150 2062 130E          JEQ   NAMERR        NAME NOT FOUND
0151                    *                   TEST NEXT NAME
0152 2064 C001          MOV   R1,R0             R0=SOURCE
0153 2066 0202          LI    R2,FAC            R2=DEST
     2068 834A
0154 206A 8CB0          C     *R0+,*R2+     TEST 2 BYTES
0155 206C 1606          JNE   NAML20
0156 206E 8CB0          C     *R0+,*R2+     TEST 2 BYTES
0157 2070 1604          JNE   NAML20
0158 2072 8CB0          C     *R0+,*R2+     TEST LAST 2 BYTES
0159 2074 1602          JNE   NAML20
0160 2076 C030          MOV   *R0+,R0       NAME MATCHES, GET CODE ADDRESS
0161 2078 0450          B     *R0           AND BRANCH TO IT
0162                    *
0163 207A 0221  NAML20  AI    R1,8          NAME DOESN'T MATCH, GO TO NEXT ONE
     207C 0008
0164 207E 10EF          JMP   NAML10
0165                    *
0166 2080 0200  NAMERR  LI    R0,ERRSNF     "NOT FOUND" ERROR
     2082 2500
0167                    *
0168                    *       GENERAL ERROR RETURN
0169                    *
0170 2084 C800  ERROR   MOV   R0,@ERRCOD
     2086 8322
0171 2088 02E0  ERR2    LWPI  GPLWS
     208A 83E0
0172 208C 0460          B     @SET
     208E 00CE
0173                    *
0174 2090 C81D  ERR     MOV   *R13,@ERRCOD
     2092 8322
0175 2094 10F9          JMP   ERR2
```

```
0177                *
0178                *       NUMERIC ASSIGNMENT UTILITY
0179                *
0180                *
0181                *
0182                *
0183                *         UTILITY ROUTINE NUMASG
0184                *          A utility to allow a numeric value to be assigned
0185                *          to a numeric variable or a numeric array passed
0186                *          as an argument to sub.
0187                *          The floating point variable to be assigned is in
0188                *          FAC.
0189                *          The stack has info. for each argument.
0190                *          For a simple numeric variale, the stack entry loo
0191                *          like this :
0192                *                    ----------------------------------------
0193                *                    !Ptr to      !>00!  !Ptr to       !      !
0194                *                    !S.T. entry  !    !  !Value space  !      !
0195                *                    ----------------------------------------
0196                *           For a array argument, the stack entry look
0197                *           like this :
0198                *                    ----------------------------------------
0199                *                    ! Ptr to     !   !  !Ptr to       !      !
0200                *                    ! S.T. entry !   !  !Dim. info.   !      !
0201                *                    ----------------------------------------
0202                *          REGISTER USAGE
0203                *           R0 : Array element no. or zero
0204                *           R1 : Argument number
0205                *           R0,R1,R2,R3,R4,R5,R6,R7,R8 are used in this
0206                *           routine
0207                *
0208                *           5 types of arguments :
0209                *  Identifier 0 : Numeric Expression
0210                *  Identifier 1 : String Expression
0211                *  Identifier 2 : Numeric Variable
0212                *  Identifier 3 : String Variable
0213                *  Identifier 4 : Numeric Array
0214                *  Identifier 5 : String Array
0215                *
0216                *
0217 2096 C01D  NUMASG      MOV   *R13,R0     For the change of the work spac
0218 2098 C06D              MOV   @2(R13),R1  For the change of the work spac
     209A 0002
0219 209C 06A0              BL    @CKARG      Check valid argument no. and
     209E 20DC
0220                ***************  set up stack entry ptr in R5  *************
0221                *******  CHECK THE ARGUMENT IDENTIFIER IN CPU RAM *********
0222                *******  A SIMPLE NUMERIC VAR. OR AN ARRAY REF.    *********
0223 20A0 C0C1              MOV   R1,R3       Get the argument no.
0224 20A2 0603              DEC   R3          Offset 1
0225 20A4 0223              AI    R3,>8300    For CPU address
     20A6 8300
0226 20A8 D0D3              MOVB  *R3,R3      Get the one byte identifier in
0227                *                         CPU
0228 20AA 1361              JEQ   NUERR2      Can't do the assignment to a
0229                *                         numeric expression argument
0230 20AC 0983              SRL   R3,8        Make it double
0231 20AE 0643              DECT  R3          Flag 2 is for NUMERIC VARIABLE
0232 20B0 1612              JNE   NUM04       No - go to the code for array
0233                ******  FOR ASSIGMENT TO A SIMPLE NUMERIC VAR.      ********
```

```
                    MOV   R0,R0       In this case, R0 must contain 0
                    JNE   NUERR2      ERROR : BAD ARGUMENT
                    MOV   R5,R3       Get the ptr to the stack entry
                    INCT  R3          Point to the ID byte of the
                *                     stack entry
                    BL    @GETV1      Get 1 byte (ID) from VDP
                *                     Numeric ID byte must be 0
                    JNE   NUERR1      ERROR:STRING-NO. MISMATCH
                    INCT  R3          Point to the value space ptr
                    BL    @GET1       Get the value space ptr in R1
NUM00               LI    R4,FAC      Source is in FAC
NUM01               LI    R2,8        8 bytes to copy
NUM02               MOVB  *R4+,*R1+   Move ONE byte  (MAY NOT START ON
                    DEC   R2          Decrement the counter - done?
                    JGT   NUM02       No - loop for more
                    RTWP              Yes - return to the caller
                ***** FOR ASSIGNMENT TO ONE ELEMENT OF AN ARRAY **********
                ***** R0 : ARRAY ELEMENT NO. R5: PTR TO STACK ENTRY ******
NUM04               BL    @ARYBND     Check array element  w/in bound
                *                                 and find the ptr to element in ERAM
                    JMP   NUM00       Go to the general code to move
                *                     from FAC to ERAM
                **********************************************************
                *******   CHECK VALID ARGUMENT NO. AND SET UP STACK ENTRY **
                *******   PTR IN R5                                    *****
CKARG               MOV   R1,R1       Argument no. can't be 0
                    JEQ   NUERR2      ERROR : BAD ARGUMENT
                    SLA   R1,8        Make one byte
                    CB    @COUNT,R1   Valid argument?
                    JLT   NUERR2      ERROR:BAD ARGUMENT
                    SRL   R1,8        Make it two byte again
                    MOV   R1,R5       Get the argument no.
                    SLA   R5,3        8 bytes per stack entry
                    AI    R5,8        Skip the 8 bytes file name
                *                     on top of stack
                    A     @OLDS,R5    Add to the old stack ptr
                *                     R5 : Ptr to the stack entry
                    RT
                **********************************************************
                ********* CHECK THE ARRAY ELEMENT NO. IS WITHIN BOUND  *****
ARYBND              MOV   R11,R9      Save rtn addr.
                    DECT  R3          Flag 4 is for ARRAY
                    JNE   NUERR1      NO-Assume string arg.
                *                     Error:STRING-NO MISMATCH
                    MOV   R5,R3       Get the stack entry ptr
                    BL    @GET1       Get the S.T. entry ptr
                    MOV   R1,R3       R1 : Data reg. for GET1
                *                     R3 : Addr. reg. for GET1
                    BL    @GETV1      Get the 1st byte of S.T. entry
                    JLT   NUERR1      String bit is set - ERROR
```

```
0284                    *                               STRING-NO MISMATCH
0285 2100 06A0                  BL      @ARY1           Go through this general routi
     210E 211C
0286 2110 06A0                  BL      @GET1           Get the ptr to ERAM in R1
     2112 23CA
0287 2114 6004                  S       R4,R0           Check the BASE : 0 : O.K.
0288                    *                                       BASE : 1 : offset 1
0289 2116 0A30                  SLA     R0,3            8 bytes per element
0290 2118 A040                  A       R0,R1           Add to the ptr in ERAM
0291 211A 0459                  B       *R9
0292                    ********************************************************
0293 211C C28B  ARY1            MOV     R11,R10         Save RTN addr.
0294 211E 0A51                  SLA     R1,5            R1 still contains 1st byte from
0295                    *
0296                    *                               func bits
0297 2120 09D1                  SRL     R1,13           Shift back to use as a double
0298                    ******  R1:NO. OF DIMENSION, R5: STACK ENTRY PTR   ********
0299 2122 C201                  MOV     R1,R8           Save no. of dimension for later
0300 2124 D120                  MOVB    @BASE,R4        Check the BASE
     2126 8342
0301 2128 0984                  SRL     R4,8            Make it double
0302 212A 1303                  JEQ     NUM08           If BASE=0, INDEX=0 -- it's o.k.
0303 212C 0600                  DEC     R0              Decrement the INDEX
0304 212E 1123                  JLT     NUERR3          BASE=1, INDEX=0---ERROR
0305                    *                               BAD SUBSCRIPT
0306 2130 0580                  INC     R0              Restore INDEX
0307 2132 0206  NUM08           LI      R6,1            Initial value for accumulator
     2134 0001
0308 2136 C0C5                  MOV     R5,R3           Put the stack entry ptr in R3
0309 2138 0223                  AI      R3,4            Try to get the dim. info. ptr
     213A 0004
0310 213C 06A0                  BL      @GET1           Go to get the dim. info. ptr
     213E 23CA
0311 2140 C0C1                  MOV     R1,R3           Put it in R3
0312 2142 0643                  DECT    R3              For the following INCT instruct.
0313 2144 05C3  NUM10           INCT    R3              Point to dimension infor. entry
0314                    *                               2-byte maxima per dimension
0315 2146 06A0                  BL      @GET1           Get the dimension maxima in R1
     2148 23CA
0316 214A 0581                  INC     R1              BASE=0, add 1 offset
0317 214C 6044                  S       R4,R1           BASE=1, O.K.
0318 214E 3981                  MPY     R1,R6           Get the max. array index
0319                    *                               R6 : accumulator
0320 2150 C186                  MOV     R6,R6           First 2 bytes must be 0 here
0321 2152 1611                  JNE     NUERR3          else BAD SUBSCRIPT
0322 2154 C187                  MOV     R7,R6           Get last two bytes result
0323 2156 0608                  DEC     R8              Decrement the counter
0324                    *                               R8 : no. of dimension
0325 2158 15F5                  JGT     NUM10           Loop for more
0326 215A 0606                  DEC     R6              BASE=0:element no. must range
0327                    *                               from 0 to (R6-1)
0328 215C A184                  A       R4,R6           BASE=1:element no. range 1 -- R6
0329 215E 8180                  C       R0,R6           User's INDEX must <= max. INDEX
0330 2160 150A                  JGT     NUERR3          NO - ERROR:BAD SUBSCRIPT
0331 2162 05C3                  INCT    R3              Points to value space which
0332                    *                               contains ptr to the beginning
0333                    *                               of array element value in ERAM
0334 2164 045A                  B       *R10            RTN
0335                    ********************************************************
0336 2166 0200  NUERR1          LI      R0,ERRSNM       STRING-NO. MISMATCH
```

```
                2168 0700
0337 216A 04C0                  B       @ERROR      Error out
                216C 2084
0338 216E 0300  NUERR2          LI      R0,ERRBA    BAD ARGUMENT
                2170 1000
0339 2172 04C0                  B       @ERROR      Error out
                2174 2084
0340 2176 0300  NUERR3          LI      R0,ERRBS    BAD SUBSCRIPT
                2178 1400
0341 217A 0460                  B       @ERROR
                217C 2084
```

```
0343                    *
0344                    *       NUMERIC REFERENCE UTILITY
0345                    *
0346                    *
0347                    *
0348                    *
0349                    *         UTILITY ROUTINE NUMREF
0350                    *          A utility to allow reference to a numeric exp.
0351                    *          , numeric variable or a numeric array passed
0352                    *          as an argument to sub.
0353                    *          The floating point variable assigned will be
0354                    *          in FAC after calling NUMREF.
0355                    *          The stack has info. for each argument.
0356                    *          For a simple numeric variale, the stack entry loo
0357                    *          like this :
0358                    *               ----------------------------------------
0359                    *               |Ptr to      |>00|  |Ptr to      |      |
0360                    *               |S.T. entry  |   |  |Value space |      |
0361                    *               ----------------------------------------
0362                    *           For a array reference, the stack entry look
0363                    *           like this :
0364                    *              ----------------------------------------
0365                    *              | Ptr to     |   |  |Ptr to      |      |
0366                    *              | S.T. entry |   |  |Dim. info.  |      |
0367                    *              ----------------------------------------
0368                    *           For a numeric exp., the stack entry has the
0369                    *           8 bytes floating points value come back from
0370                    *           a parse.
0371                    *          REGISTER USAGE
0372                    *           R0 : Array element no. or 0
0373                    *           R1 : Argument number
0374                    *           R0,R1,R2,R3,R4,R5,R6,R7,R8 are used in this
0375                    *           routine
0376                    *
0377                    *
0378 217E C01D  NUMREF      MOV   *R13,R0     For the change of the work spac
0379 2180 C06D              MOV   @2(R13),R1  For the change of the work spac
     2182 0002
0380 2184 06A0              BL    @CKARG      Check valid argument no.
     2186 20DC
0381                    *                     and set up stack ptr.
0382                    *******   R5 : Ptr. to the stack entry, R1 : Argument no.
0383                    *******  CHECK THE ARGUMENT IDENTIFIER IN CPU RAM *********
0384                    ***** A NUMERIC EXP., SIMPLE NUMERIC VAR. OR AN ARRAY *****
0385 2188 C0C1              MOV   R1,R3       Get the argument no.
0386 218A 0603              DEC   R3          Offset 1
0387 218C 0223              AI    R3,>8300    For CPU address
     218E 8300
0388 2190 D0D3              MOVB  *R3,R3      Get the one byte identifier in
0389                    *                     CPU
0390 2192 0983              SRL   R3,8        Make it double
0391 2194 160E              JNE   REF00       Not a numeric expression
0392                    ******  A NUMERIC EXP., MOVE THE 8 BYTES STACK ENTRY ******
0393                    ******  FROM VDP TO FAC.                            ******
0394 2196 C000              MOV   R0,R0       R0 in this case must contain
0395 2198 1622              JNE   RFERR2      ERROR : BAD ARGUMENT
0396 219A 0202              LI    R2,8        8 bytes to move
     219C 0008
0397 219E 0204              LI    R4,FAC      Destination addr.
     21A0 834A
```

```
0399 21A2 C0C5                MOV   R5,R3       Source addr.: stack entry ptr
0399 21A4 06A0  REF03         BL    @GET1       Get 2 bytes from VDP.
     21A6 230A
0400 21A8 CCC1                MOV   R1,*R4+     Put 2 bytes in FAC
0401 21AA 0583                INCT  R3          Update source addr.
0402 21AC 0642                DECT  R2          Update the counter
0403 21AE 15FA                JGT   REF03       More to move
0404 21B0 0380                RTWP              Return to caller
0405                ********  A SIMPLE NUMERIC VAR. OR AN ARRAY ***************
0406 21B2 0643  REF00         DECT  R3          Flag 2 is for NUMERIC VARIABLE
0407 21B4 160F                JNE   REF04       NO - go to the code for array
0408                ******  FOR REFERENCE TO A SIMPLE NUMERIC VAR.      *********
0409 21B6 C000                MOV   R0,R0       In this case, R0 must contain 0
0410 21B8 1612                JNE   RFERR2      ERROR : BAD ARGUMENT
0411 21BA C0C5                MOV   R5,R3       Get the ptr to the stack entry
0412 21BC 0583                INCT  R3          Point to the ID byte of the
0413                *                           stack entry
0414 21BE 06A0                BL    @GETV1      Get 1 byte (ID) from VDP
     21C0 2406
0415                *                           Numeric ID byte must be 0
0416 21C2 160B                JNE   RFERR1      ERROR:STRING-NO. MISMATCH
0417 21C4 0583                INCT  R3          Point to the value space ptr
0418 21C6 06A0                BL    @GET1       Get the value space ptr in R1
     21C8 230A
0419 21CA C101  REF01         MOV   R1,R4       Put the source addr. in R4
0420                *                           for the "general move" code
0421 21CC 0201                LI    R1,FAC      Destination addr. is in FAC
     21CE 834A
0422 21D0 0460                B     @NUM01      Goto the "general move" code an
     21D2 20CA
0423                *                           return to caller
0424                *****  FOR ASSIGNMENT TO ONE ELEMENT OF AN ARRAY ***********
0425                *****  R0 : ARRAY ELEMENT NO. R5: PTR TO STACK ENTRY ******
0426 21D4 06A0  REF04         BL    @ARYBND     Check array elem. is w/in bound
     21D6 20F8
0427                *             and find the ptr to the array element in ERAM *
0428 21D8 10F8                JMP   REF01       Go to the general code to move
0429                *                           from ERAM to FAC
0430                ************************************************************
0431 21DA 0460  RFERR1        B     @NUERR1     STRING-NO. MISMATCH
     21DC 2166
0432 21DE 0460  RFERR2        B     @NUERR2     BAD ARGUMENT
     21E0 216E
```

```
0434                    *
0435                    *
0436                    *       STRING ASSIGNMENT UTILITY
0437                    *
0438                    *
0439                    *
0440                    *         UTILITY ROUTINE STRASG
0441                    *          A utility to allow a string to be assigned to a
0442                    *          string variable or one element of an string
0443                    *          array passed as an argument to subpro.
0444                    *          The stack has info. for each argument.
0445                    *          For a string variale, the stack entry look likes
0446                    *          this ----------------------------------------
0447                    *               !Ptr to     !>65!  !Pointer  ! String  !
0448                    *               !Value space!   !  !to String! Length  !
0449                    *               ----------------------------------------
0450                    *          For a string array, the stack entry look likes
0451                    *          this ----------------------------------------
0452                    *               !Ptr to     !   !  !Ptr to   !          !
0453                    *               !S. T. entry!   !  !Dim info.!          !
0454                    *               ----------------------------------------
0455                    *          USER'S REGISTER USAGE
0456                    *           R0 : Array element no. or zero
0457                    *           R1 : Argument number
0458                    *           R2 : String address in ERAM, (the first byte
0459                    *                of the string is the length of the string.)
0460                    ************************************************************
0461                    *           THIS ROUTINE REGISTER USAGE
0462                    *           R0 : Always is the string addr. in ERAM
0463                    *                except when routine ARY1 is called,
0464                    *                then has the array element no.
0465                    *           R1 : Argument no.
0466                    *           R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10 are used in
0467                    *           this routine
0468                    *          UTILWS work space usage
0469                    *           UTILWS : Array element provided by user
0470                    *           UTILWS+2 : RTN addr. for VGASSN.
0471                    *           UTILWS+4 : Temporary place for R5 when VPUSH,
0472                    *                      GETSTR, ASSGNV are called.
0473                    *           UTILWS+6 : Temporary place for @BYTE when
0474                    *                      GETSTR is called.
0475                    *           UTILWS+8 : RTN addr. for GPL
0476                    *           UTILWS+12 : String addr. in ERAM provided by user
0477                    *
0478                    *
0479 21E2 C81D  STRASG  MOV   *R13,@UTILWS    Array element from user
     21E4 2038
0480 21E6 C82D          MOV   @2(R13),@GPLWS+2 For change of work space
     21E8 0002
     21EA 83E2
0481                    *                    Argument no. provided by user is in R1
0482 21EC C82D          MOV   @4(R13),@UTILWS+12 String addr. in ERAM
     21EE 0004
     21F0 2044
0483 21F2 02E0          LWPI  GPLWS          Use GPL workspace in order to
     21F4 83E0
0484                    *                    use existing Extended BASIC ROM
0485                    *                    routines which are in XML table
0486 21F6 C80B          MOV   R11,@UTILWS+8  Save RTN addr.
     21F8 2040
```

```
0487 21FA 0020          MOV   @UTILWS+12,R0 ERAM string addr.is in R0 now
     21FC 2044
0488 21FE 06A0          BL    @CXARG      Check valid argument no. and
     2200 2DDC
0489                 *                    set up ptr to stack entry in R5
0490 2202 C301          MOV   R1,R3       Get argument no.
0491 2204 0603          DEC   R3          Offset 1
0492 2206 0223          AI    R3,>8300    For CPU address
     2208 8300
0493 220A D0D3          MOVB  *R3,R3      Get the one byte identifier in
0494                 *                    CPU
0495 220C 0983          SRL   R3,8        Make it double
0496 220E 0603          DEC   R3          Check if string expression
0497 2210 1332          JEQ   STERR2      Can't do the assignment to a
0498                 *                    string expression argument
0499 2212 0643          DECT  R3          Flag 3 is for string variable
0500 2214 164A          JNE   STARY       NO : Is STRING ARRAY ?
0501                 ******  ASSIGNMENT TO STRING VARIABLE    *******
0502 2216 C2A0          MOV   @UTILWS,R10 In this case : array element
     2218 2038
0503                 *                    has to be 0
0504 221A 162D          JNE   STERR2      ERROR : BAD ARGUMENT
0505 221C C005          MOV   R5,R3       Addr.reg. for GETV1
0506 221E 0503          INCT  R3          Point to the ID byte of the
0507                 *                    stack entry
0508 2220 06A0          BL    @GETV1      Get 1 byte (ID) from VDP
     2222 2406
0509 2224 9801          CB    R1,@CBH65   Is this a string variable ?
     2226 2058
0510 2228 1620          JNE   STERR1      No - STRING NO. MISMATCH ERROR
0511                 ************************************************************
0512                 ******  PUSH THIS STACK ENTRY ON TOP OF THE STACK FOR ******
0513                 ******  ROUTINE "ASSGNV" TO USE                       ******
0514                 ************************************************************
0515                 ****    MOVE THIS STRING VARIABLE ENTRY TO CPU FAC AREA  ***
0516                 ****    IN ORDER TO DO A VPUSH                           ***
0517 222A 0206          LI    R6,8        # of bytes to move
     222C 0008
0518 222E 0204          LI    R4,FAC      Destination addr. in CPU
     2230 834A
0519 2232 C0C5          MOV   R5,R3       Restore stack entry ptr
0520                 *                    R3 : addr. reg. for GET1
0521 2234 06A0  STR02   BL    @GET1       Get 2 bytes from VDP
     2236 23CA
0522 2238 CD01          MOV   R1,*R4+     Put to FAC area
0523 223A 05C3          INCT  R3          Update source addr.
0524 223C 0646          DECT  R6          Update the counter
0525 223E 15FA          JGT   STR02       More to move
0526                 ******  DO THE VPUSH, GETSTR, COPY STRING TO STRING SPACE
0527                 ******  AND ASSIGNMENT                  ******************
0528 2240 06A0          BL    @VGASSN
     2242 22DA
0529                 **********    NOW UPDATES THE STACK ENTRY OF THIS    *******
0530                 **********    STRING VARIABLE                        *******
0531                 **********    R5 STILL POINTS TO THE OLD STACK ENTRY *******
0532                 **********    FOR THIS STRING VARIABLE               *******
0533 2244 0225          AI    R5,4        "POINTER TO STRING" entry
     2246 0004
0534 2248 C105          MOV   R5,R4       Addr. reg. for PUT1
0535 224A C046          MOV   R6,R1       Put the ptr. to string (come
```

```
                        *               back from ASSGNV) into data  reg. for PUT1
0537 2240 C6A0          BL      @PUT1       Put 2 bytes of data to VDP
     2242 23E6
0538 2244 0584          INCT    R4          "LENGTH OF STRING" entry
0539 2246 D090          MOVB    *R0,R1      Put the length into data reg.
0540 2248 0981          SRL     R1,8        Make it two bytes
0541 224A 06A0          BL      @PUT1       Put 2 bytes of data to VDP
     224C 23E6
0542 224E C2E0  STRTN   MOV     @UTILWS+8,R11 Restore RTN addr.
     2250 2040
0543 2252
0544 2252 C820          MOV     @UTILWS+6,@BYTE   Restrore @BYTE
     2254 203E
     2256 830C
0545 2258
0546 2258 02E0          LWPI    UTILWS      Go back to old workspace
     225A 2039
0547 225C 0380          RTWP                Return
0548                    ********************************************************************
0549 225E 0200  STERR1  LI      R0,ERRSNM   STRING-NO. MISMATCH
     2260 0700
0550 2262 C2E0  ERRTN   MOV     @UTILWS+8,R11  Restore RTN addr.
     2264 2040
0551 2266 0460          B       @ERROR
     2268 2084
0552 226A 0200  STERR2  LI      R0,ERRBA    BAD ARGUMENT
     226C 1C00
0553 226E 0460          B       @ERRTN      Error return
     2270 2262
0554                    ********************************************************************
0555                    ************ Common routine for STRASG & STRREF to check
0556                    ************ valid array element
0557 2272 C0EB  STARYC  MOV     R11,R2      Save RTN addr.
0558 2274 0643          DECT    R3          Flag 5 is for string array
0559 2276 16F3          JNE     STERR1      ERROR : STRING NO. MISMATCH
0560 2278 C0C5          MOV     R5,R3       Get the stack entry ptr
0561 227A 06A0          BL      @GET1       Get the S.T. entry ptr
     227C 23CA
0562 227E C0C1          MOV     R1,R3       R1 : Data reg. for GET1
0563                    *                   R3 : Addr. reg. for GET1
0564 2280 06A0          BL      @GETV1      Get the 1st byte of S.T. entry
     2282 2406
0565 2284 1102          JLT     STARY1      O.K.
0566 2286 0460          B       @STERR1     ERROR : STRING-NO MISMATCH
     2288 226A
0567 228A C020  STARY1  MOV     @UTILWS,R0  Put array element in R0 now
     228C 2038
0568                    *                   for the general routine ARY1
0569 228E 06A0          BL      @ARY1       Go through general routine to
     2290 211C
0570                    *               check array element is out of bound
0571                    *  R4:BASE, R0:ARRAY ELEMENT, R3:PTR TO BEGINNING OF VALUE
0572                    *                                                  SPACE
0573 2292 6004          S       R4,R0       Check the BASE : 0 : O.K.
0574                    *                                   BASE : 1 : Offset 1
0575 2294 0A10          SLA     R0,1        2 bytes per element
0576 2296 A0C0          A       R0,R3       Ptr to the exect element we want
0577 2298 06A0          BL      @GET1       Get the string ptr
     229A 23CA
0578 229C 0452          B       *R2         RTN
```

```
0579                    ********************************************************
0580                    ********************************************************
0581 22AA 06A0  STARY   BL    @STARYC     Go through general routine
     22AC 227E
0582                    *                                 shared by STRREF
0583                    *    NOW  R1 : PTR TO THE OLD STRING
0584                    *         R3 : PTR TO THE VALUE SPACE
0585                    *** SET UP FAC ENTRY AND PUSH IT ON STACK FOR ASSIGN ROUT. *
0586 22AE 0206          LI    R6,FAC      Destination addr.
     22B0 834A
0587 22B2 CDB3          MOV   R3,*R6+     Put ptr to value space in
0588 22B4 DDA0          MOVB  @CBH65,*R6+ Put >65 ID in
     22B6 2058
0589 22B8 DDB4          MOVB  R4,*R6+     Clear the unuse byte
0590 22BA CDB1          MOV   R1,*R6+     Put the ptr to string in
0591 22BC C0C1          MOV   R1,R3       Set up addr. reg. for GETV1
0592 22BE 1602          JNE   STARY4      Not null ptr : o.k.
0593 22C0 04D6          CLR   *R6         Clear the length bytes
0594 22C2 1009          JMP   STARY6      Go on
0595 22C4 0603  STARY4  DEC   R3         Point to string length
0596 22C6 0640          BL    @GETV1      Get the length from VDP
     22C8 2406
0597 22CA 0981          SRL   R1,8        Make it two bytes
0598 22CC C581          MOV   R1,*R6      Put the length in FAC area
0599                    ******  FAC ENTRY HAS BEEN SET UP NOW  **********************
0600 22CE C020  STARY6  MOV   @UTILWS+12,R0 Restore the ERAM string addr.
     22D0 2044
0601                    *                                    to R0
0602 22D2 06A0          BL    @VGASSN     Do the VPUSH, GETSTR AND ASSIGN
     22D4 22DA
0603                    *****   ASSIGNMENT WILL POP THE STACK, OLD STACK ENTRY ****
0604                    *****   DOES NOT NEED TO BE UPDATED IN THIS CASE      ******
0605 22D6 0460          B     @STRTN      RTN
     22D8 225A
0606                    ********************************************************
0607                    ********************************************************
0608 22DA C80B  VGASSN  MOV   R11,@UTILWS+2  Save rtn addr.
     22DC 203A
0609                    *********  VPUSH changed R0,R5  values    ******************
0610 22DE C805          MOV   R5,@UTILWS+4
     22E0 203C
0611                    ********************************************************
0612 22E2 C2E0          MOV   @>601E,R11  XML table addr. for VPUSH
     22E4 601E
0613 22E6 069B          BL    *R11        Do the VPUSH
0614                    ************  RESTORE R0,R5   *******************************
0615 22E8 C020          MOV   @UTILWS+12,R0
     22EA 2044
0616 22EC C160          MOV   @UTILWS+4,R5
     22EE 203C
0617                    ********************************************************
0618                    *****  GETS THE STRING SPACE, COPIES THE STRING FROM ERAM **
0619                    *****  INTO VDP STRING SPACE, AND SETS UP THE FAC WITH A  **
0620                    *****  STRING ENTRY OF THE FOLLOWING FORM:                **
0621                    *****  ---------------------------------------           **
0622                    *****  | >001C | >65 | XX |  POINTER  |  LENGTH   |      **
0623                    *****  |       |     |    | TO STRING | OF STRING |      **
0624                    *****  ---------------------------------------           **
0625                    ***** FAC     +2    +3   +4          +6                  **
0626                    ********************************************************
```

```
0627 22F0 D190          MOVB  *R0,R6      R0:Addr. of string (length byte
0628                    *                 & string) supplies by ALC
0629 22F2 0926          SRL   R6,8        Set up the length byte
0630 22F4
0631 22F4 C820          MOV   @BYTE,@UTILWS+6   Save @BYTE
     22F6 8300
     22F8 203E
0632 22FA
0633 22FA C806          MOV   R6,@BYTE    For GETSTR routine
     22FC 8300
0634 22FE C806          MOV   R6,@FAC6    Set up length byte entry in FAC
     2300 8350
0635                    ********** GETSTR changes R0,R5 value ******************
0636                    ************************************************************
0637 2302 C2E0          MOV   @>6012,R11  XML table addr. for GETSTR
     2304 6012
0638 2306 069B          BL    *R11        Call GETSTR routine
0639 2308 C020          MOV   @UTILWS+12,R0   Restore R0
     230A 2044
0640                    *************     SET UP FAC ENTRY          ******************
0641 230C 0206          LI    R6,FAC      Optimize to save bytes
     230E 834A
0642 2310 0204          LI    R4,>001C    Get addr. of SREF
     2312 001C
0643 2314 CD84          MOV   R4,*R6+     Indicate a temporary string
0644 2316 DDA0          MOVB  @CBH65,*R6+ Indicate a string
     2318 2058
0645 231A DD84          MOVB  R4,*R6+     Byte is not used
0646 231C C5A0          MOV   @SREF,*R6   Save ptr to the string
     231E 831C
0647                    *************     COPY THE STRING TO STRING SPACE  **********
0648 2320 C0A0          MOV   @BYTE,R2    Get # of bytes to copy
     2322 8300
0649 2324 1309          JEQ   STR04       If none to copy
0650 2326 C116          MOV   *R6,R4      Get pointer to destination
0651                    *                 R4:addr. reg. for PUTV1 below
0652 2328 C0C0          MOV   R0,R3       R0:Ptr to string (length byte)
0653 232A 0583          INC   R3          Skip the len byte-ptr to string
0654 232C D073  STR06   MOVB  *R3+,R1     Get one byte from ERAM
0655 232E 06A0          BL    @PUTV1      Put 1 byte in VDP
     2330 241A
0656 2332 0584          INC   R4          Update the destination addr.
0657 2334 0602          DEC   R2          1 less to move
0658 2336 15FA          JGT   STR06       If not done-loop for more
0659                    *                 ASSGNV destroys all the reg.
0660 2338 C2E0  STR04   MOV   @>6028,R11  XML table addr. for ASSGNV
     233A 6028
0661 233C 069B          BL    *R11        Call ASSGNV
0662                    *****     RESTORE R0,R5   ************
0663 233E C020          MOV   @UTILWS+12,R0
     2340 2044
0664 2342 C160          MOV   @UTILWS+4,R5
     2344 203C
0665 2346 C2E0          MOV   @UTILWS+2,R11   Restore RTN addr.
     2348 203A
0666 234A 045B          RT
```

```
0668                *
0669                *
0670                *       STRING REFERENCE UTILITY
0671                *
0672                *
0673                *
0674                *         UTILITY ROUTINE STRREF
0675                *          A utility to allow a ref. to a string expression,
0676                *          a string variable or a string array passed as an
0677                *          argument to subpro.
0678                *
0679                *          The stack has info. for each argument.
0680                *           For a string expression, the stack entry looks
0681                *           likes this
0682                *                ----------------------------------------
0683                *                | >001C     |>65|  |Pointer   | String |
0684                *                |or Ptr to  |   |  |to String | Length |
0685                *                |Value space|   |  |          |        |
0686                *                ----------------------------------------
0687                *                   >001C : For a temporary string
0688                *                   Ptr to Value Space : for a permanent string
0689                *          For a string variale, the stack entry looks like
0690                *          this ----------------------------------------
0691                *                |Ptr to     |>65|  |Pointer   | String  |
0692                *                |Value space|   |  |to String| Length  |
0693                *                ----------------------------------------
0694                *           For a string array, the stack entry looks likes
0695                *           this
0696                *                ----------------------------------------
0697                *                |Ptr to     |  |  |Ptr to    |         |
0698                *                |S.T. entry |  |  |Dim info. |         |
0699                *                ----------------------------------------
0700                *
0701                *          USER'S REGISTER USAGE
0702                *           RO : Array element or zero
0703                *           R1 : Argument number
0704                *           RO : String address in ERAM, (the first byte
0705                *                of the string is the length of the string.)
0706                *                Before calling STRREF : The string len has
0707                *                the max. length which indicates how many
0708                *                space followed the lengh byte has been
0709                *                allocated for the string.
0710                *                After calling STRREF : The length byte has
0711                *                the actual length of the string which
0712                *                locates right after the length byte.
0713                *
0714                *
0715                ****************************************************************
0716 234C C01D  STRREF      MOV    *R13,RO     For the change of the work space
0717                *                          Array element now is in RO
0718 234E C06D              MOV    @2(R13),R1  For the change of the work space
     2350 0002
0719                *                          Argument no. is in R1
0720 2352 06A0              BL     @CKARG      Check valid argument no. and
     2354 20DC
0721                *                          set up ptr to stack entry in R5
0722 2356 C0C1              MOV    R1,R3       Get the argument no.
0723 2358 0603              DEC    R3          Offset 1
0724 235A 0223              AI     R3,>8300    For CPU address
     235C 8300
```

```
0725 235E D0D3              MOVB  *R3,R3      Get one byte identifier in CPU
0726 2360 0983              SRL   R3,8        Make it double
0727 2362 0603              DEC   R3          Flag 1 is for string expressic
0728 2364 1302              JEQ   SRF01       String expression is o.k.
0729 2366 0643              DECT  R3          Flag 3 is for string variable
0730 2368 1623              JNE   STARYF      Not string exp. or var., must
0731                  *                       string array in this case.
0732                  ************  REF. TO STRING EXP. OR STRING VAR.  *******
0733 236A C000  SRF01       MOV   R0,R0       R0 must be 0 in this case
0734 236C 1628              JNE   SRFER2      ERROR : BAD ARGUMENT
0735 236E C02D              MOV   @4(R13),R0  Now move string ptr in ERAM
     2370 0004
0736                  *                       into R0
0737 2372 C0C5              MOV   R5,R3       Addr.reg. for GETV1
0738 2374 05C3              INCT  R3          Point to the ID byte of the
0739                  *                       stack entry
0740 2376 06A0              BL    @GETV1      Get 1 byte (ID) from VDP
     2378 2406
0741 237A 9801              CB    R1,@CBH65   Is this a string variable ?
     237C 2058
0742 237E 161D              JNE   SRFER1      No - STRING NO. MISMATCH ERROR
0743 2380 05C3              INCT  R3          Points to "ptr to string"
0744 2382 06A0              BL    @GET1       Get the "ptr to string" from
     2384 23CA
0745                  *                       stack entry in VDP
0746 2386 C041  SRF03       MOV   R1,R1       Is it null ptr?
0747 2388 1307              JEQ   SRF05       Yes - put 0 in length byte
0748 238A C181              MOV   R1,R6       R6 now contains "ptr to string
0749 238C 0601              DEC   R1          Points to the length byte of
0750                  *                       this string in string space
0751 238E C0C1              MOV   R1,R3       Set up addr. reg. for GETV1
0752 2390 06A0              BL    @GETV1      Get the 1 byte actual length
     2392 2406
0753                  *                       of this string in R1
0754 2394 9050              CB    *R0,R1      Compare the max. len. user
0755                  *                       provided with the actual len.
0756 2396 1A15              JL    SRFER3      Not enough space - ERROR
0757                  *                       STRING TRUNCATED ERROR
0758 2398 DC01  SRF05       MOVB  R1,*R0+     Enough space - o.k.
0759                  *                       Put the actual len. into the
0760                  *                       length byte
0761 239A 1309              JEQ   SRF08       0 byte length - don't move
0762                  ****** R0 : Points to the allocated string space in ERAM *
0763                  *                       Destination addr. for the
0764                  *                       following move
0765 239C C0C6              MOV   R6,R3       Set up addr. reg. for GETV1
0766                  *                       R3 now contains "ptr to string
0767                  *                       Source addr. for following mov
0768 239E 0981              SRL   R1,8        Make actual len. byte a double
0769 23A0 C141              MOV   R1,R5       R5 : # of bytes to move
0770 23A2 06A0  SRF07       BL    @GETV1      Get 1 byte from VDP
     23A4 2406
0771 23A6 DC01              MOVB  R1,*R0+     Put 1 byte into ERAM
0772 23A8 0583              INC   R3          Update the source addr.
0773 23AA 0605              DEC   R5          Update the counter
0774 23AC 15FA              JGT   SRF07       More to move - loop back
0775 23AE 0380  SRF08       RTWP              Return the caller
0776                  ********************************************************
0777                  ***********   REF. TO STRING ARRAY       ***************
0778 23B0 06A0  STARYF      BL    @STARYC     Check array element is within
```

```
                2382 227E
0774                    *       bound and find the string element ptr in value space
0775 2384 C02D                  MOV   @4(R13),R0 Put string addr. in ERAM provided
     2386 0004
0777                    *  by the user into R0 now. (for the change of work space)
0778 2388 10E1                  JMP   SRFO3      Go to the general code to move
0779                    *                        the string from VDP to ERAM
0780                    ********************************************************
0781 238A 0460  2166   SRFER1   B     @NUERR1    STRING-NO. MISMATCH
0782 238E 0460  216E   SRFER2   B     @NUERR2    BAD ARGUMENT
0783 2392 0200  1300   SRFER3   LI    R0,ERRST   STRING TRUNCATED
0784 2396 0460  2094            B     @ERROR     Error out
```

```
0790                ********************************************************
0791                ***      UTILITY ROUTINES TO ACESS VDP
0792                ********************************************************
0793                ****** GET1 : Get two bytes of data from VDP
0794                ******       R3 : Address in VDP
0795                ******       R1 : Where the two bytes data stored
0796 23DA 06C3      GET1        SWPB  R3
0797 23DC D803                  MOVB  R3,@VDPWA
     23DE 8C02
0798 23D0 06C3                  SWPB  R3
0799 23D2 D803                  MOVB  R3,@VDPWA
     23D4 8C02
0800 23D6 1000                  NOP
0801 23D8 D060                  MOVB  @VDPRD,R1
     23DA 8800
0802 23DC 06C1                  SWPB  R1
0803 23DE D060                  MOVB  @VDPRD,R1
     23E0 8800
0804 23E2 06C1                  SWPB  R1
0805 23E4 045B                  RT
0806                ****** PUT1 : Put two bytes of data into VDP
0807                ******       R4 : Address on VDP
0808                ******       R1 : Data
0809 23E6 06C4      PUT1        SWPB  R4
0810 23E8 D804                  MOVB  R4,@VDPWA
     23EA 8C02
0811 23EC 06C4                  SWPB  R4
0812 23EE 0264                  ORI   R4,WRVDP
     23F0 4000
0813 23F2 D804                  MOVB  R4,@VDPWA
     23F4 8C02
0814 23F6 1000                  NOP
0815 23F8 D801                  MOVB  R1,@VDPWD
     23FA 8C00
0816 23FC 06C1                  SWPB  R1
0817 23FE D801                  MOVB  R1,@VDPWD
     2400 8C00
0818 2402 06C1                  SWPB  R1
0819 2404 045B                  RT
0820                ****** GETV1 : Get 1 byte of data from VDP
0821                ******       R3 : Address in VDP
0822                ******       R1 : Where the 1 byte data stored
0823 2406 06C3      GETV1       SWPB  R3
0824 2408 D803                  MOVB  R3,@VDPWA
     240A 8C02
0825 240C 06C3                  SWPB  R3
0826 240E D803                  MOVB  R3,@VDPWA
     2410 8C02
0827 2412 1000                  NOP
0828 2414 D060                  MOVB  @VDPRD,R1
     2416 8800
0829 2418 045B                  RT
0830                ****** PUTV1 : Put 1 bytes of data into VDP
0831                ******       R4 : Address on VDP
0832                *****        R1 : Data
0833 241A 06C4      PUTV1       SWPB  R4
0834 241C D804                  MOVB  R4,@VDPWA
     241E 8C02
0835 2420 06C4                  SWPB  R4
0836 2422 0264                  ORI   R4,WRVDP
```

```
                2424 4000
0E07  2426 D814             MOVB  R4,@VDPWA
      2428 8002
0808  242A 1010             NOP
0809  242C D801             MOVB  R1,@VDPWD
      242E 8002
0840  2430 0458             RT
```

```
0842                    *
0842                    *       LINK TO SYSTEM XML UTILITIES
0844                    *
0845 2432 C83E  XMLLNK  MOV   *R14+,@GPLWS+2   GET ARGUMENT
     2434 83E2
0846 2436 0220          LWPI  GPLWS
     2438 83E0
0847 243A C802          MOV   R11,@UTILWS+22  Save GPL return address
     243C 204E
0848 243E C0C1          MOV   R1,R2
0849 2440 0281          CI    R1,>40
     2442 0040
0850 2444 180A          JH    XMLL30          IF SYS. XML WE HAVE ADDRESS
0851 2446 C0A1          MOV   @EXBASX(R1),R2  Get address from XML table
     2448 6010
0852 244A 0281          CI    R1,4            Is it MEMCHK?
     244C 0004
0853 244E 1605          JNE   XMLL30          No
0854 2450 C0A2          MOV   @2(R2),R2
     2452 0002
0855 2454 C0B2          BL    *R2             Special case for MEMCHK
0856 2456 2464          DATA  MEMERR          Error return from MEMCHK
0857 2458 1001          JMP   XMLL40
0858 245A 0692  XMLL30  BL    *R2             GO TO ROUTINE
0859 245C 02E0  XMLL40  LWPI  UTILWS          GET BACK TO RIGHT WS
     245E 2038
0860 2460 C80B          MOV   R11,@GPLWS+22   Restore GPL return address
     2462 83F6
0861 2464 0380          RTWP
0862 2466 0200  MEMERR  LI    R0,ERRMEM
     2468 0800
0863 246A 0460          B     @ERROR
     246C 2084
```

```
0865                    *
0866                    *       KEYBOARD SCAN
0867                    *
0868 24CE 02E0  KSCAN   LWPI GPLWS
     2470 83E0
0869 2472 C80B          MOV  R11,@UTILWS+22   Save GPL return address
     2474 2C4E
0870 2476 06A0          BL   @SCNKEY
     2478 000E
0871 247A 02E0          LWPI UTILWS
     247C 2038
0872 247E C80B          MOV  R11,@GPLWS+22    Restore GPL return address
     2480 83F6
0873 2482 0380          RTWP
0874                    *
0875                    ** VDP single byte write
0876                    *
0877 2484 06A0  VSBW    BL   @WVDPWA          Write out address
     2486 24CA
0878 2488 D82D          MOVB @2(R13),@VDPWD   Write data
     248A 0002
     248C 8C00
0879 248E 0380          RTWP                  Return to calling program
0880                    *
0881                    ** VDP multiple byte write
0882                    *
0883 2490 06A0  VMBW    BL   @WVDPWA          Write out address
     2492 24CA
0884 2494 D821  VWTMOR  MOVB *R1+,@VDPWD      Write a byte
     2496 8C00
0885 2498 0602          DEC  R2               Decrement byte count
0886 249A 16FC          JNE  VWTMOR           More to write?
0887 249C 0380          RTWP                  Return to calling Program
0888                    *
0889                    ** VDP single byte read
0890                    *
0891 249E 06A0  VSBR    BL   @WVDPRA          Write out address
     24A0 24D0
0892 24A2 D860          MOVB @VDPRD,@2(R13)   Read data
     24A4 8800
     24A6 0002
0893 24A8 0380          RTWP                  Return to calling program
0894                    *
0895                    ** VDP multiple byte read
0896                    *
0897 24AA 06A0  VMBR    BL   @WVDPRA          Write out address
     24AC 24D0
0898 24AE DC60  VRDMOR  MOVB @VDPRD,*R1+      Read a byte
     24B0 8800
0899 24B2 0602          DEC  R2               Decrement byte count
0900 24B4 16FC          JNE  VRDMOR           More to read?
0901 24B6 0380          RTWP                  Return to calling program
0902                    *
0903                    ** VDP write to register
0904                    *
0905 24B8 CC5D  VWTR    MOV  *R13,R1          Get register number and value
0906 24BA D82D          MOVB @1(R13),@VDPWA   Write out value
     24BC 0001
     24BE 8C02
0907 24C0 0261          ORI  R1,>8000         Set for register write
```

```
                24C2 8000
0908 24C4 D5C1          MOVB R1,@VDPWA          Write out register number
                24C6 8C02
0909 24C8 0380          RTWP                    Return to calling program
0910                    *
0911                    ** Set up to write to VDP
0912                    *
0913 24CA 0201  WVDPWA  LI   R1,>4000
                24CC 4000
0914 24CE 1001          JMP  WVDPAD
0915                    *
0916                    ** Set up to read VDP
0917                    *
0918 24D0 04C1  WVDPRA  CLR  R1
0919                    *
0920                    ** Write VDP address
0921                    *
0922 24D2 C09D  WVDPAD  MOV  *R13,R2            Get VDP address
0923 24D4 D820          MOVB @R2LB,@VDPWA       Write low byte of address
                24D6 8C02
                24D8 8C02
0924 24DA E0E1          SOC  R1,R2              Properly adjust VDP write bit
0925 24DC D5C2          MOVB R2,@VDPWA          Write high byte of address
                24DE 8C02
0926 24E0 C06D          MOV  @2(R13),R1         Get CPU RAM address
                24E2 0002
0927 24E4 C0AD          MOV  @4(R13),R2         Get byte count
                24E6 0004
0928 24E8 045B          RT                      Return to calling routine
0929                    *
0930       24F4 IFFA    EQU  $+10          INITIAL VALUE FOR FFA
0931                    END
NO ERRORS,      NO WARNINGS
```

```
$       24EA            0930
ARY1    2110    0292    0285    0569
ARYSNC  20F8    0274    0252    0426
BASE    8043    0039    0300
BYTE    8000    0033    0544    0631    0633    0648
CHKFC   2088    0129    0509    0583    0644    0741
CHKPC   2000    0259    0219    0320    0488    0720
COUNT   8012    0036    0262
CRULST  8000    0049
DEBUG   0000    0019    0021    0104
EQBIT   2059    0140
ERR     2090    0174    0130
ERR2    2088    0171    0175
ERRBA   1C00    0085    0338    0552
ERRBLN  1700    0080
ERRBS   1400    0077    0340
ERRBV   1E00    0087
ERRCC   1900    0082
ERRCIP  1A00    0093
ERRCOD  8002    0038    0170    0174
ERRCNT  2100    0090
ERRFE   2200    0091
ERRFNM  0E00    0071
ERRIAL  1F00    0088
ERRIBS  0400    0061
ERRIM   0A00    0067
ERRINP  2000    0089
ERRIO   2400    0092
ERRIVN  2800    0095
ERRLNF  1600    0079
ERRLTL  1800    0081
ERRMEM  0B00    0068    0862
ERRMS   1100    0074
ERRMUV  0900    0066
ERRNO   0200    0059
ERRNPP  1D00    0086
ERRNQS  0500    0062
ERRNTL  0600    0063
ERRNWF  0D00    0070
ERROBE  0800    0065
ERROLP  1B00    0084
ERROR   2084    0170    0337    0339    0341    0551    0788    0863
ERRPV   2700    0094
ERRRSC  1000    0073
ERRRWG  1200    0075
ERRSNF  2500    0093    0166
ERRSNM  0700    0064    0336    0549
ERRSNS  0F00    0072
ERRSO   0C00    0069
ERRSSL  1500    0078
ERRST   1300    0076    0787
ERRSYN  0300    0060
ERRTN   226E    0550    0553
EXBASX  6010    0032    0851
FAC     834A    0040    0044    0153    0244    0397    0421    0518    0586    0641
FAC6    8350    0044    0634
FFA     2002    0112
GET1    23CA    0796    0243    0279    0286    0310    0315    0399    0418    0521    0561
                        0577    0744
GETV1   2406    0823    0239    0282    0414    0508    0564    0596    0740    0752    0770
```

LABEL	VALUE	DEFN	REFERENCES
CPLNS	83E0	0050	0171 0480 0483 0845 0846 0860 0868 0872
HAA	2006	0115	
IFF4	24F4	0930	0112
ILFA	4000	0025	0113 0149
MESAN	244E	08e8	0124
LFA	2004	0113	0148
MAXVOF	83F0	0046	
MEMERR	24e6	0862	0856
NAMERR	20E0	0166	0150
NAML10	205E	0149	0164
NAML20	207A	0163	0155 0157 0159
NAMLNK	205A	0148	0110
NUERR1	2166	0336	0241 0276 0283 0431 0785
NUERR2	216E	0338	0228 0235 0260 0263 0432 0786
NUERR3	2176	0340	0304 0321 0330
NUMO0	2006	0244	0254
NUMO1	200A	0245	0422
NUMO2	200E	0246	0248
NUMO4	2006	0252	0232
NUMO8	2132	0307	0302
NUMIO	2144	0313	0325
NUM+32	2026	0217	0119
NUMREF	217E	0378	0120
OFF	0000	0017	0019
OLDB	8310	0034	0269
ON	0001	0016	0021 0104
PUT1	23E6	0809	0537 0541
PUTV1	241A	0833	0655
R0	0000		0152 0154 0156 0158 0160 0160 0161 0166 01 0217 0234 0234 0287 0289 0290 0303 0306 032 0336 0338 0340 0378 0394 0394 0409 0409 0487 0539 0549 0552 0567 0573 0575 0576 0600 0615 0627 0639 0652 0663 0716 0733 0733 0735 0754 0758 0771 0780 0787 0862
R1	0001		0148 0149 0152 0163 0218 0223 0246 0259 0259 0261 0262 0264 0265 0280 0290 0294 0297 0299 0311 0316 0317 0318 0379 0385 0400 0419 0421 0490 0509 0522 0535 0539 0540 0562 0590 0591 0597 0598 0654 0718 0722 0741 0746 0746 0748 0749 0751 0754 0758 0768 0769 0771 0801 0802 0803 0804 0815 0816 0817 0818 0828 0839 0848 0849 0851 0852 0884 0898 0905 0907 0908 0913 0918 0924 0926
R10	000A		0293 0334 0502
R11	000B		0274 0293 0486 0542 0550 0557 0608 0612 0613 0637 0638 0660 0661 0665 0847 0860 0869 0872
R13	000D		0174 0217 0218 0378 0379 0479 0480 0482 0716 0718 0735 0780 0878 0892 0905 0906 0922 0926 0927
R14	000E		0845
R2	0002		0153 0154 0156 0158 0245 0247 0396 0402 0557 0578 0648 0657 0848 0851 0854 0854 0855 0858 0885 0899 0922 0924 0925 0927
R2LB	203D	0134	0923
R3	0003		0223 0224 0225 0226 0226 0230 0231 0236 02 0242 0275 0278 0280 0308 0309 0311 0312 03 0331 0385 0386 0387 0388 0388 0390 0398 0401 0406 0411 0412 0417 0490 0491 0492 0493 0493 0495 0496 0499 0505 0506 0519 0523 0558 0560 0562 0576 0587 0591 0595 0652 0653 0654 0722

			0723	0724	0725	0725	0726	0727	0729	0737	0738
			0743	0751	0765	0772	0796	0797	0798	0799	0823
			0824	0825	0826						
R4	0004		0244	0246	0287	0300	0301	0317	0328	0397	0400
			0419	0518	0522	0534	0538	0573	0589	0642	0643
			0645	0650	0656	0809	0810	0811	0812	0813	0833
			0834	0835	0836	0837					
R4LB	2041	0135									
R5	0005		0236	0265	0266	0267	0269	0278	0308	0398	0411
			0505	0519	0533	0534	0560	0610	0616	0664	0737
			0769	0773							
R6	0006		0307	0318	0320	0320	0322	0326	0328	0329	0517
			0524	0535	0586	0587	0588	0589	0590	0593	0598
			0627	0629	0633	0634	0641	0643	0644	0645	0646
			0650	0748	0765						
R7	0007		0322								
R8	0008		0299	0323							
R9	0009		0274	0291							
REF00	2182	0406	0391								
REF01	21CA	0419	0428								
REF03	21A4	0399	0403								
REF04	21D4	0426	0407								
RFERR1	21DA	0431	0416								
RFERR2	21DE	0432	0395	0410							
SCLEN	8355	0042									
SCNAME	8356	0043									
SCNKEY	000E	0031	0870								
SCTEMP	8354	0041									
SET	000E	0026	0172								
SNDADD	83CC	0047									
SREF	831C	0037	0646								
SRF01	236A	0733	0728								
SRF03	2386	0746	0782								
SRF05	2398	0758	0747								
SRF07	23A2	0770	0774								
SRF08	23AE	0775	0761								
SRFER1	23BA	0785	0742								
SRFER2	23BE	0786	0734								
SRFER3	23C2	0787	0756								
STARY	22AA	0581	0500								
STARY1	2296	0567	0565								
STARY4	22C4	0595	0592								
STARY6	22CE	0600	0594								
STARYC	227E	0557	0581	0778							
STARYF	23B0	0778	0730								
STERR1	226A	0549	0510	0559	0566						
STERR2	2276	0552	0497	0504							
STFLGS	83CE	0048									
STR02	2234	0521	0525								
STR04	233B	0660	0649								
STR06	232C	0654	0658								
STRASG	21E2	0479	0121								
STRREF	234C	0716	0122								
STRTN	225A	0542	0605								
UTILWS	203B	0133	0119	0120	0121	0122	0123	0124	0125	0126	0127
			0128	0129	0130	0134	0135	0479	0482	0486	0487
			0502	0542	0544	0546	0550	0567	0600	0608	0610
			0615	0616	0631	0639	0663	0664	0665	0847	0859
			0869	0871							
VDPRD	8800	0053	0801	0803	0828	0892	0898				

LABEL	VALUE	DEFN	REFERENCES								
[illegible]	8002	0055	0797	0799	0810	0813	0824	0826	0834	0837	0906
			0908	0923	0925						
[illegible]	8010	0054	0815	0817	0839	0878	0864				
[illegible]	2254	0608	0528	0602							
[illegible]	2444	0897	0128								
[illegible]	2443	0883	0126								
[illegible]	244E	0898	0900								
[illegible]	2496	0891	0127								
VSBW	2484	0877	0125								
VSPTR	836E	0045									
VWTMOR	2494	0884	0886								
VWTR	2488	0905	0129								
WRNINP	2C00	0099									
WRNID	2D00	0100									
WRNLNF	2E00	0101									
WRNNO	2900	0096									
WRNNPP	2B00	0098									
WRNST	2A00	0097									
WRUCP	4000	0052	0812	0836							
[illegible]	2482	0922	0914								
[illegible]	2480	0918	0891	0897							
WZCPNA	240A	0913	0877	0883							
XMLL3C	245A	0858	0850	0853							
XMLL4C	2450	0859	0857								
XMLLNK	2432	0845	0123								