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


LINE      KEY   NAME

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

```
0031                    IDT   'SUBPROG'
0032            ********************************************************************
0033            *                                                                  *
0034            *     SSS    U   U  BBBB   PPPP   RRRR    OOO     GGGG          *
0035            *    S   S   U   U  B   B  P   P  R   R  O   O   G    G         *
0036            *    S       U   U  B   B  P   P  R   R  O   O   G              *
0037            *     SSS    U   U  BBBB   PPPP   RRRR   O   O   G  GG          *
0038            *        S   U   U  B   B  P      R R    O   O   G   G          *
0039            *    S   S   U   U  B   B  P      R  R   O   O   G   G          *
0040            *     SSS     UUU   BBBB   P      R   R   OOO     GGGG          *
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   CALL,SUBXIT,RESOLV
0056                *
0057                    REF   R0LB,R1LB,R2LB,R3LB,R4LB,R5LB,R6LB,R8LB
0058                    REF   P10LB,ASSG,C8,C16,EOL,FBS001
0059                    REF   EOSTMT,ERR,ERR1,GETV,GETV1,PUTV1
0060                    REF   GETG2,GET1,PUT1,PUTG2,VPUSH,VPOP
0061                    REF   MEMCHK,RESET,NUDEND,PGMCHR,PGMSUB,PSHPRS
0062                    REF   SMB,SYM,SAVREG
0063                    REF   ERRCOD,PGMPTR,RAMTOP
0064                    REF   FREPTR,PRGFLG,RAMFLG,SUBTAB,LSUBP,CALIST
0065                    REF   SYMTAB,SYMTA1,CRNBUF,SYMBOL
0066                    REF   FLAG,VDPWD,WRVDP,VSPTR,POPSTK
0067                    REF   GROM,GRMWD,GRAM
0068                    REF   PAD,FAC,FAC2,FAC4,FAC6,FAC8
0069                    REF   FAC15,ARG,ARG2,ARG3,ARG4
0070                    REF   VRAM,GRMWA,VDPRD
0071                *
0072                *
0073                *-----------------CONDITIONAL ASSEMBLY----------------------*
0074                    ASMIF VERS=DX10
0075                CONTAD DATA >6000             ADDRESS OF A CONTINUE STMT
0076                GPLIST EQU  >6026             GPL SUBPROGRAM LINKED LIST
0077                    ASMELS
0078 0000 A000     CONTAD DATA >A000             ADDRESS OF A CONTINUE STMT
0079      A026     GPLIST EQU  >A026             GPL SUBPROGRAM LINKED LIST
0080                    ASMEND
0081                *----------------END OF CONDITIONAL ASSEMBLY---------------*
0082                *
0083                *
0084      00B3      COMMA$ EQU  >B3               Comma token
0085      00B6      RPAR$  EQU  >B6               Right parenthesis token
0086      00B7      LPAR$  EQU  >B7               Left parenthesis token
0087      00C8      UNQST$ EQU  >C8               Unquoted string token
0088                *
0089                *
0090 0002 8000     INUSE  DATA >8000             In-use flag
0091 0004 4000     FNCFLG DATA >4000             User-Defined Function flag
0092 0006 2000     SHRFLG DATA >2000             Shared-value flag
0093 0008               EVEN
0094                *
0095                *      Error codes
0096                *
0097      1203      ERRSND EQU  >1203             * SUBEND NOT IN SUBPROGRAM
0098      0F03      ERRREC EQU  >0F03             * RECURSIVE SUBPROGRAM CALL
0099      0E03      ERRIAL EQU  >0E03             * INCORRECT ARGUMENT LIST
0100      1103      ERROLP EQU  >1103             * ONLY LEGAL IN A PROGRAM
```

```
0102                    ********************************************************************
0103                    *                                                                  *
0104                    *       CALL-STATEMENT EXECUTION                                   *
0105                    *                                                                  *
0106                    *       Finds the subprogram specified in the subprogram           *
0107                    *       table, evaluates and assigns any arguments to the          *
0108                    *       the formal parameters, builds the stack block, and         *
0109                    *       transfers control into the subprogram.                     *
0110                    *                                                                  *
0111                    *       General register usage:                                    *
0112                    *            R0-R6 -- Temporaries                                  *
0113                    *            R7 -- Pointer into formals in subprogram              *
0114                    *                  name entry                                      *
0115                    *            R8 -- Character returned by PGMCHR                    *
0116                    *            R9 -- Subroutine stack                                *
0117                    *            R10-- Temporary                                       *
0118                    *            R11-- Return link                                     *
0119                    *            R12-- Temporary                                       *
0120                    *            R13-- GROM read-data address                          *
0121                    *            R14-- Interpreter flags                               *
0122                    *            R15-- VDP write-address address                       *
0123                    ********************************************************************
0124 0008 C0A0  CALL    BL   @PGMCHR                Skip UNQST$ & get name length
     000A 0000          move *R14+,R8
0125 000C D808          MOVB R8,@FAC15              Save length for FBS
     000E 0000
0126 0010 D108          MOV  R8,R4                  For the copies to be made
0127 0012 0984          SRL  R4,8   swpb R4            below
0128 0014 C020          MOV  @PGMPTR,RO             Get pointer to name
     0016 0000                R14
0129 0018 D060          MOVB @RAMFLG,R1             ERAM or VDP?
     001A 0000
0130 001C 130D          JEQ  CALL04                 VDP
0131                    *       ERAM, must copy into VDP
0132 001E C140          MOV  RO,R5                  Pointer to string in ERAM
0133 0020 0200          LI   RO,CRNBUF              Destination in VDP
     0022 0000
0134 0024 CCC4          MOV  R4,R3                  Length for this move
0135                    *-----------------CONDITIONAL ASSEMBLY---------------------*
0136                            ASMIF VERS=DX10
0137
0138                            MOV  RO,R14         Move for VDP
0139                            AI   R14,VRAM       VDP offset
0140                            AI   R5,GRAM        ERAM offset
0141                    CALL02  MOVB *R5+,*R14+     Move a byte
0142
0143                            ASMELS
0144 0026
0145 0026 D7E0                  MOVB @ROLB,*R15     Load out the VDP write address
     0028 0000
0146 002A 0260                  ORI  RO,WRVDP       Enable the VDP write
     002C 0000
0147 002E D7C0                  MOVB RO,*R15        Second byte of VDP address
0148 0030 D835          CALL02  MOVB *R5+,@VDPWD    Move a byte
     0032 0000
0149 0034
0150                            ASMEND
0151                    *----------------END OF CONDITIONAL ASSEMBLY---------------*
0152 0034 0603                  DEC  R3             One less byte to move
0153 0036 16FC                  JNE  CALL02         Loop if not done
```

```
0154 003B A804  CALLC4 A     R4, @PGMPTR          Skip over the name
     003A 0016'                R14
0155 003C 0201         LI    R1,FAC               Destination in CPU
     003E 0000
0156                 *----------------CONDITIONAL ASSEMBLY----------------------*
0157                         ASMIF VERS=DX10
0158
0159                         MOV  R0,R14          Move for VDP
0160                         AI   R14,VRAM        Add VDP offset
0161                 CALL06  MOVB *R14+,*R1+      Move a byte
0162
0163                         ASMELS
0164 0040
0165 0040 D7E0               MOVB @ROLB,*R15      Load out VDP read address
     0042 0028'
0166 0044 0240               ANDI R0,>3FFF        Kill VDP write-enable
     0046 3FFF
0167 0048 D7C0               MOVB R0,*R15         Both bytes
0168 004A 1000               NOP                  Don't go too fast for it
0169 004C D060       CALL05  MOVB @VDPRD,*R1+     Move a byte
     004E 0000                    *R01
0170 0050
0171                         ASMEND
0172                 *----------------END OF CONDITIONAL ASSEMBLY--------------*
0173 0050 0604               DEC  R4              One less byte to move
0174 0052 16FC               JNE  CALL06          Loop if not done
0175 0054 C120               MOV  @SUBTAB,R4      Get beginning of subprog table
     0056 0000
0176 0058 133C               JEQ  SCAL89          If table empty, Search in GP
0177 005A 06A0               BL   @FBS001         Search subprogram table
     005C 0000
0178 005E 00D2'              DATA SCAL89          If not found, search in GPL
0179                 *       Pointer to table entry returned in both R4 and FAC
0180 0060 06A0               BL   @PGMCHR         Get next token
     0062 000A'                mov6 *R14+,R6
0181 0064 C0C4               MOV  R4,R3           Duplicate pointer for GETV
0182 0066 06A0               BL   @GETV1          Get flag byte
     0068 0000
0183 006A 1130               JLT  SCAL90          If attempted recursive call
0184 006C 0A11               SLA  R1,1            Check for BASIC/GPL program
0185 006E 1106               JLT  GPLSU           GPL subprogram
0186 0070 D2E0               MOVB @PRGFLG,R11     Imperative call to BASIC sub?
     0072 0000
0187 0074 1614               JNE  SCAL01          No--OK--Handle BASIC subprog
0188                 *
0189 0076 0200               LI   R0,ERROLP       Can't call a BASIC sub
     0078 1103
0190 007A 102D               JMP  SCAL91              imperatively
```

```
0192                    *
0193                    *       Handle a GPL subprogram
0194                    *
0195 007C 0609  GPLSU   INCT R9
0196 007E C3E0          MOV   @CONTAD,*R9+      Put addr of a cont on stack
     0080 0000'
0197 0082 C74D          MOV   R13,*R9           Save addr for real BASIC.
0198 0084 0223          AI    R3,6              Now set up new environment
     0086 0006
0199 0088 06A0          BL    @GET1             Get access addr of GPL subprog
     008A 0000
0200                    *--------------------CONDITIONAL ASSEMBLY--------------------*
0201                            ASMIF VERS=DX10
0202
0203                            MOV   R1,R15            Set up address of GPL program
0204                            AI    R15,GROM          GROM offset
0205
0206                            ASMELS
0207 008C
0208 008C D841                  MOVB R1,@GRMWA(R13)     Load out the address into GROM
     008E 0000
0209 0090 0601                  SWPB R1                 Need to kill time here.
0210 0092 D841                  MOVB R1,@GRMWA(R13)     Next byte also.
     0094 008E'
0211 0096
0212                            ASMEND
0213                    *----------------END OF CONDITIONAL ASSEMBLY----------------*
0214 0096 06A0          BL    @SAVREG           Restore registers to GPL
     0098 0000
0215 009A 0460          B     @RESET            And enter the routine
     009C 0000
```

```
                    *
                    *       Execute BASIC subprogram
                    *
009E                SCAL01
009E 0911                   SRL   R1,1              Restore mode to original form
00A0 F0E0                   SOCB  INUSE,R1          Set the in-use flag bit
     00A2 0002'
00A4 06A0                   BL    @PUTV1            Put the byte back
     00A6 0000
00A8 C303                   MOV   R3,R12            Save subtable address.
00AA 04E0                   CLR   @FAC2             Indicate non-special entry
     00AC 0000
00AE 06A0                   BL    @VPUSH            Push subprogram entry on stac
     00B0 0000
00B2 C10C                   MOV   R12,R4            Restore sub table  address.
00B4 C1C4                   MOV   R4,R7
00B6 0227                   AI    R7,6              Point to 1st argument in list
     00B8 0006
00BA C0C7                   MOV   R7,R3             Formals' pointer
00BC 06A0                   BL    @GET1             Check to see if any
     00BE 008A'
00C0 C0C1                   MOV   R1,R1             Any args?
00C2 133F                   JEQ   SCAL32            None--jump forward
00C4-0288                   CI    R8,LPAR$*256      Must see a left paren
     00C6 B700
00C8 1640                   JNE   SCAL34            If not, error
00CA 1013                   JMP   SCAL08            Jump into argument loop
00CC 0200           SCAL90  LI    R0,ERRREC         * RECURSIVE SUBPROGRAM CALL
     00CE 0F03
00D0 1002                   JMP   SCAL91
00D2 0200           SCAL89  LI    R0,>000A          GPL check for DSR subprogram
     00D4 000A
00D6 0460           SCAL91  B     @ERR
     00D8 0000
00DA 1031           SCAL93  JMP   SCAL12            Going down!
                    *
00DC                SCAL05
00DC 06A0                   BL    @POPSTK           Short stack pop routine
     00DE 0000
00E0 C1E0                   MOV   @ARG4,R7          To quickly restore R7
     00E2 0000
00E4 05C7                   INCT  R7                To account for SCAL80
00E6-0288           SCAL06  CI    R8,RPAR$*256      Actual list ended?
     00E8 B600
00EA 132D                   JEQ   SCAL30            Actuals all scanned
00EC-0288                   CI    R8,COMMA$*256     Must see a comma then
     00EE B300
00F0 1626                   JNE   SCAL12            Didn't, so error
                    *       Scan next actual.  Check if it is just a name
00F2 C820           SCAL08  MOV   @PGMPTR,@ERRCOD   Save text ptr in case of exp
     00F4 003A'                   R14
     00F6 0000
00F8 06A0                   BL    @PGMCHR           Get next character
     00FA 0062'             MOVB  *R14+,R8
00FC 1179                   JLT   SCAL40            No, so must be an expressio
00FE C307                   MOV   R7,R12            Save formals pointer
0100 06A0                   BL    @SYM              Read name & see if recognize
     0102 0000
0104 06A0                   BL    @GETV             Check function flag
     0106 0000
```

```
0258 0108 003E'          DATA FAC
0259 010A C1CC           MOV  R12,R7          Restore formals pointer first
0260 010C 2460           CZC  FNCFLG,R1       User-defined function?
     010E 0004'
0261 0110 166F           JNE  SCAL40          Yes--Pass by value
0262 0112-0288           CI   R8,LPAR$*256    Complex type?
     0114 B700
0263 0116 1620           JNE  SCAL15          No
0264 0118 06A0           BL   @PGMCHR         Check if formal array
     011A 00FA'
0265 011C-0288           CI   R8,RPAR$*256    FOO() ?
     011E B600
0266 0120 1319           JEQ  SCAL14          Yes, handle it as such
0267 0122-0288           CI   R8,COMMA$*256   or FOO(,...) ?
     0124 B300
0268 0126 1613           JNE  SCAL35          No--An array element FOO(I...
0269 0128 06A0  SCAL10   BL   @PGMCHR         Formal array--scan to end
     012A 011A'
0270 012C 06A0           BL   @EOSTMT         Check if end-of-statment
     012E 0000
0271 0130 1306           JEQ  SCAL12          Premature end of statment
0272 0132-0288           CI   R8,COMMA$*256   Another comma?
     0134 B300
0273 0136 13F8           JEQ  SCAL10          Yes--continue on to end
0274 0138-0288           CI   R8,RPAR$*256    End yet?
     013A B600
0275 013C 1308           JEQ  SCAL14          Yes--merge in below
0276 013E 0460  SCAL12   B    @ERR1           * SYNTAX ERROR
     0140 0000
0277                     *
0278 0142 0460  SCAL32   B    @SCAL62          Going down!
     0144 02B6'
0279 0146 0460  SCAL30   B    @SCAL60
     0148 02B2'
0280 014A 0460  SCAL34   B    @SCAL88
     014C 035E'
0281 014E 0460  SCAL35   B    @SCAL50
     0150 0242'
0282 0152 10C9  SCAL37   JMP  SCAL06
0283                     *
0284                     *
0285                     *    Here For Scalars/Arrays By Reference
0286                     *
0287 0154 06A0  SCAL14   BL   @PGMCHR         Pass the right parenthesis
     0156 012A'
0288 0158-0288  SCAL15   CI   R8,COMMA$*256   Just a name?
     015A B300
0289 015C 1303           JEQ  SCAL16          Yes
0290 015E-0288           CI   R8,RPAR$*256    Start an expression?
     0160 B600
0291 0162 1646           JNE  SCAL40          Yes--name starts an expression
0292 0164 06A0  SCAL16   BL   @GETV           Get mode of name
     0166 0106'
0293 0168 0108'          DATA FAC             Ptr to s.t. entry left by SYM
0294 016A D081           MOVB R1,R2           Save for check below
0295 016C 06A0           BL   @SCAL80         And fetch next formal info
     016E 0366'
0296 0170 D042           MOVB R2,R1           Copy for this check
0297 0172 0241           ANDI R1,>C700        Compare modes
     0174 C700
```

```
0295 0176 C006          MOV  R6,R0              Use a temporary register
0297 0178 0240          ANDI R0,>C700              for the comparison
     017A C700
0300 017C 8C01          C    R1,R0              Must be exact match
0301 017E 16E3          JNE  SCAL34             Else can't pass by reference
0302 0180 E140          SOC  SHRFLG,R6          Set the shared symbol flag
     0182 0006'
0303 0184 D046          MOVB R6,R1              Load up for PUTV
0304 0186 C105          MOV  R5,R4              Addr to put the flag
0305 0188 06A0          BL   @PUTV1             Set the flag in the s.t. entry
     018A 00A6'
0306 018C 0244          ANDI R4,>3FFF           Kill VDP write-enable bit
     018E 3FFF
0307                    *
0308                    *     The following section finds actual's value space addres
0309                    *       and puts it in R1.
0310                    *       FAC contains the symbol table's address.
0311                    *     If actual is NOT shared.....
0312                    *       Symbol table's address + 6 will point to the value sp
0313                    *       except for numeric ERAM case.  In a numeric ERAM case
0314                    *       GET1 to get pointer to the ERAM value space.
0315                    *     If actual is SHARED.....
0316                    *       GET1 to get the pointer in symbol table's address +
0317                    *       In a numeric ERAM case, GETG to get the indirect poi
0318                    *       to the actual's value space pointer after GET1 is ca
0319                    *
0320                    *
0321 0190 C060          MOV  @FAC,R1            Ptr to actual s.t. entry
     0192 0168'
0322 0194 0221          AI   R1,6               Ptr to actuals value space
     0196 0006
0323                    *
0324 0198 0246          ANDI R6,>8700           Keep info on string or array
     019A 8700
0325 019C 0242          ANDI R2,>2000           Is actual shared?
     019E 2000
0326 01A0 130C          JEQ  SCAL23             No--use it
0327                    *
0328 01A2 C0C1          MOV  R1,R3              Else look further
0329 01A4 06A0          BL   @GET1              Get the true pointer
     01A6 00BE'
0330                    *
0331 01A8 D186          MOVB  R6,R6             Array or string?
0332 01AA 160F          JNE  SCAL24             Yes - both are special cases
0333 01AC D0A0          MOVB @RAMTOP,R2         ERAM present?
     01AE 0000
0334 01B0 130C          JEQ  SCAL24             No ERAM - so skip
0335                    *
0336                    *     Numeric variable, shared, ERAM.
0337 01B2 C0C1          MOV  R1,R3              Get ptr to original from ERAM
0338 01B4 06A0          BL   @GETG2             Get indirect pointer.
     01B6 0000
0339 01B8 1008          JMP  SCAL24
0340                    *
0341                    *     Shared bit is NOT on.
0342                    *
0343 01BA D186  SCAL23  MOVB R6,R6              Check for array or string.
0344 01BC 1606          JNE  SCAL24             Yes - take what's in there.
0345 01BE D0A0          MOVB @RAMTOP,R2         ERAM exists?
     01C0 01AE'
```

```
0346 01D2 1303          JEQ   SCAL24           No.
0347 01D4 C0C1          MOV   R1,R3            Numeric and ERAM case.
0348 01D6 06A0          BL    @GET1            Get ERAM value space address.
     01D8 01F6'
0349                *
0350 01DA           SCAL24
0351                *                          R4 pointing to value space of
0352 01DA 0224          AI    R4,6                 subprogram's symbol table
     01DC 0006
0353 01DE D186          MOVB  R6,R6            Array or string case?
0354 01E0 160C          JNE   SCAL26           Yes - so just put ptr in VDP
0355                *
0356                *       Here check for ERAM program and if ERAM then
0357                *       copy the address of shared value space into
0358                *       corresponding value space in ERAM
0359                *
0360 01E2 D1A0          MOVB  @RAMTOP,R6       Get the ERAM flag
     01E4 01CC'
0361 01E6 1309          JEQ   SCAL26           If not ERAM - simple case
0362 01E8 C181          MOV   R1,R6            Keep shared value space addr
0363 01EA C0C4          MOV   R4,R3            Put ptr in value space in ERAM
0364 01EC 06A0          BL    @GET1            Get value space addr in ERAM
     01EE 01CB'
0365 01F0 C101          MOV   R1,R4            Copy addr into R4 for PUTG2
0366                *
0367                *
0368 01F2 C046          MOV   R6,R1            Get the value to put in ERAM.
0369 01F4 06A0          BL    @PUTG2           Write it into ERAM
     01F6 0000
0370 01F8 10B4          JMP   SCAL37           Loop for next argument
0371 01FA           SCAL26
0372 01FA 06A0          BL    @PUT1            Set symbol indirect link
     01FC 0000
0373 01FE 10B1          JMP   SCAL37           And loop for next arg
```

```
0375                    *
0376                    *       Here to pass an expression by value
0377                    *
0378 01F0 C820  SCAL40  MOV   @ERRCOD,@PGMPTR   Restore text pointer
     01F2 00F4'                      R14
     01F4 00F4'
0379 01F6 C807          MOV   R7,@FAC4         Save formals pointer
     01F8 0000
0380 01FA 04E0          CLR   @FAC2            Don't let VPUSH mess up
     01FC 00AC'
0381 01FE 06A0  SCAL42  BL    @PGMCHR          Set up for the parse
     0200 0156'         ANS   R7,R8
0382                    *                      Save formals ptr & SUBTAB pt
0382 0202 06A0          BL    @PSHPRS             and evaluate the expressi
     0204 0000
0384 0206   B6          BYTE  RPAR$            Stop on an rpar or comma
0385 0207   6A  CBH6A   BYTE  >6A              Use the wasted byte
0386 0208 06A0          BL    @POPSTK          Restore formals pointer
     020A 00DE'
0387 020C A820          A     @C16,@VSPTR      But keep it on stack
     020E 0010
     0210 0000
0388 0212 06A0          BL    @VPUSH           Save parse result
     0214 00B0'
0389 0216 C1E0          MOV   @ARG4,R7         Restore formals pointer.
     0218 00E2'
0390 021A 06A0          BL    @SCAL80          And fetch next formal's info
     021C 0366'
0391 021E C805          MOV   R5,@FAC          Set up for assignment
     0220 0192'
0392 0222 06A0          BL    @SMB             Get value space
     0224 0000
0393 0226 6820          S     @C16,@VSPTR      Get to s.t. info
     0228 020E'
     022A 0210'
0394 022C 06A0          BL    @VPUSH           Set up for ASSG
     022E 0214'
0395 0230 A820          A     @C8,@VSPTR       Get back to parse result
     0232 0000
     0234 022A'
0396 0236 06A0          BL    @VPOP            Get parse result back
     0238 0000
0397 023A 06A0          BL    @ASSG            Assign the value to the form
     023C 0000
0398 023E 0460          B     @SCAL05          And go back for more
     0240 00DC'
```

```
0400                    *
0401                    *       Here for array elements
0402                    *
0403 0242 0580  SCAL50  DEC   @PGMPTR R14       Restore text pointer to lpar
     0244 01F4'
0404 0246 0209          LI    R11,FAC2          Optimize to save
     0248 01FC'
0405 024A 04F9          CLR   *R11+             Don't let VPUSH mess up (FAC2)
0406 024C CEC7          MOV   R7,*R11+          Save formals pointer    (FAC4)
0407 024E C6E0          MOV   @ERRCOD,*R11      For save on stack       (FAC6)
     0250 01F2'
0408 0252 06A0          BL    @VPUSH            Save the info
     0254 022E'
0409 0256-0208          LI    R8,LPAR$*256      Load up R8 with the lpar again
     0258 B700
0410 025A C820          MOV   @FAC,@PAD         Save ptr to s.t. entry
     025C 0220'
     025E 0000
0411 0260 06A0          BL    @SMB              Check if name or expression
     0262 0224'
0412 0264-0288          CI    R8,COMMA$*256
     0266 B300
0413 0268 1309          JEQ   SCAL54            Name if ended on a comma
0414 026A-0288          CI    R8,RPAR$*256
     026C B600
0415 026E 1306          JEQ   SCAL54               or rpar
0416 0270 06A0          BL    @VPOP             Get saved info back
     0272 0238'
0417 0274 C820          MOV   @FAC6,@PGMPTR     Else expr-Restore text pointer
     0276 0000                        R14
     0278 0244'
0418 027A 10C1          JMP   SCAL42            And handle like an expression
0419                    *
0420                    *       Passing array elements by reference
0421                    *
0422 027C 06A0  SCAL54  BL    @POPSTK           Restore symbol ptr
     027E 020A'
0423 0280 C1E0          MOV   @ARG4,R7
     0282 0218'
0424 0284 06A0          BL    @SCAL80           Get next formal's info
     0286 0366'
0425 0288 06A0          BL    @GETV             Check actual's mode
     028A 0166'
0426 028C 025E'         DATA  PAD               Get back header information.
0427 028E 0241          ANDI  R1,>C000          Throw away all but string&func
     0290 C000
0428 0292 9046          CB    R6,R1             Check mode match(string/num)
0429 0294 1612          JNE   JNE88             Don't--so error
0430                    *       Can  set bit in R1 since MSB(R1)=MSB(R6)
0431 0296 F060          SOCB  SHRFLG,R1         Set the share flag
     0298 0006'
0432 029A C105          MOV   R5,R4             Address for PUTV
0433 029C 06A0          BL    @PUTV1            Put it in the s.t. entry
     029E 018A'
0434 02A0 0244          ANDI  R4,>3FFF          Kill VDP write-enable bit.
     02A2 3FFF
0435 02A4 C060          MOV   @FAC,R1           Assuming string,ref link=@FAC
     02A6 025C'
0436 02A8 D186          MOVB  R6,R6             Check if it is a string
0437 02AA 118F          JLT   SCAL24            If so, go set ref. link
```

```
0438 03AC C060          MOV   @FAC4,R1          Numeric, ref. link=@FAC4(v.s.
     03AE 01FB'
0439 0200 1090          JMP   SCAL24            Now set the link and go on
```

```
0441                    *
0442                    *       Here when done parsing actuals
0443                    *
0444 0232 06A0  SCAL60  BL    @PGMCHR           Pass the right parenthesis
     0234 0200'
0445 0236 06A0  SCAL62  BL    @EOSTMT           Must be at end of statement
     0238 012E'
0446 023A 1551  JNE88   JNE   SCAL88           If not--error
0447 023C C0C7          MOV   R7,R3             Formals must also have ended
0448 023E 05C7          INCT  R7
0449 0240 C807          MOV   R7,@FAC           Keep R7. POPSTK destroys R7.
     0242 02A6'
0450 0244 06A0          BL    @GET1             Get the last arg address.
     0246 01DE'
0451 0248 C041          MOV   R1,R1             Formals end?
0452 024A 1649          JNE   SCAL88            Didn't--so error
0453                    *
0454                    *       Now set up the stack entry
0455                    *
0456 024C 06A0          BL    @VPUSH            Check if enough room for push.
     024E 0254'
0457 0250 6E20          S     @C8,@VSPTR        Get back right pointer.
     0252 0232'
     0254 0234'
0458 0256 06A0          BL    @POPSTK           Retrieve ptr to subprog s.t.
     0258 027E'
0459 025A 020C          LI    R12,FAC           For code optimization
     025C 02C2'
0460 025E C04C          MOV   R12,R1            Store following data in FAC.
0461 0260 C81C          MOV   *R12,@ARG2        Save new environment pointer
     0262 0000
0462                    *
0463                    *       First push entry. PGMCHR, EXTRAM, SYMTAB and RAM(SYNBO
0464                    *
0465 0264 0200          LI    R0,PGMPTR         Optimize
     0266 0278'
0466 0268 CC70          MOV   *R0+,*R1+         Text pointer      PGMPTR
0467 026A CC70          MOV   *R0+,*R1+         Line table ptr    EXTRAM
0468 026C CC60          MOV   @SYMTAB,*R1+      Symbol table ptr
     026E 0000
0469 0270 0203          LI    R3,SYMBOL         Put address of SYMBOL
     0272 0000
0470 0274 06A0          BL    @GET1             Get RAM(SYMBOL) in REG1
     0276 02C6'
0471 0278 C801          MOV   R1,@FAC6          Move to FAC area.
     027A 0276'
0472 027C 06A0          BL    @VPUSH            Save first entry
     027E 02CE'
0473                    *
0474                    *       Push second entry. Subprogram table pointer, >6A,
0475                    *       on warning bits and @LSUBP in the second stack.
0476                    *
0477 0300 C10C          MOV   R12,R4            Going to build entry in FAC
0478 0302 CD20          MOV   @ARG,*R4+         Subprogram table entry pointer
     0304 0000
0479 0306 DD20          MOVB  @CBH6A,*R4+       >6A = Stack ID
     0308 0207'
0480 030A D0A0          MOVB  @FLAG,R2          Warning/break bits
     030C 0000
0481 030E 0242          ANDI  R2,>0600          Mask off other bits
```

```
                0310 0600
0482 0312 D802          MOVB R2,*R4+            Put bits in stack entry
0483                    *                       Nothing in @FAC4
0484 0314 C820          MOV  @LSUBP,@FAC6       Last subprog block on stack
     0316 0000
     0318 02F4'
0485 031A 06A0          BL   @VPUSH             Push final entry
     031C 02FE'
0486 031E C820          MOV  @VSPTR,@LSUBP      Set bottom of stack for the s
     0320 02D4'
     0322 0316'
0487                    *
0488                    *       Now build the new emvironment by modifying PGMCHR,
0489                    *       EXTRAM and pointer to sub's symbol table.
0490                    *
0491 0324 0200          LI   RO,PGMPTR          Optimization
     0326 02E6'
0492                    *------------------CONDITIONAL ASSEMBLY----------------------
0493                    ASMIF VERS=DX10
0494
0495                    MOV  @ARG2,R14          Point after arg list in the
0496                    AI   R14,VRAM           Subprogram table.
0497                    MOVB *R14+,*RO+         new PGMPTR
0498                    MOVB *R14+,*RO+
0499                    MOVB *R14+,*RO+         new EXTRAM
0500                    MOVB *R14+,*RO+
0501                    MOVB *R14+,@SYMTAB      new SYMTAB
0502                    MOVB *R14+,@SYMTA1
0503                    LI   R4,SYMBOL
0504
0505                    ASMELS
0506 0328
0507 0328 D7E0          MOVB @ARG3,*R15         2nd byte of address
     032A 0000
0508 032C 0201          LI   R1,VDPRD           Optimize to save bytes
     032E 004E'
0509 0330 D7E0          MOVB @ARG2,*R15         1st byte of address
     0332 02E2'
0510 0334 0204          LI   R4,4               Need 4 bytes
     0336 0004
0511 0338 DC11  SCAL70  MOVB *R1,*RO+           Read EXTRAM and PGMPTR
0512 033A 0604          DEC  R4
0513 033C 16FD          JNE  SCAL70             mov @PGMPTR,R14
0514 033E D811          MOVB *R1,@SYMTAB        New SYMTAB
     0340 02EE'
0515 0342 0204          LI   R4,SYMBOL          To set RAM(SYMBOL)
     0344 02F2'
0516 0346 D811          MOVB *R1,@SYMTA1
     0348 0000
0517 034A
0518                    ASMEND
0519                    *------------------END OF CONDITIONAL ASSEMBLY----------------*
0520 034A C060          MOV  @SYMTAB,R1
     034C 0340'
0521 034E 06A0          BL   @PUT1              New RAM(SYMBOL)
     0350 01EC'
0522                    *==================  END  ================================*
0523 0352 04E0          CLR  @ERRCOD            Clean up our mess
     0354 0250'
0524 0356 06A0          BL   @PGMCHR            Get the next token into RB.
                        movb *R14+,R8
```

```
      0358 0EB4'
0525  035A 0460          B     @NUDEND            Enter the subprogram
      035C 0000
0526  035E 0200  SCAL88 LI    R0,ERRIAL          * INCORRECT ARGUMENT LIST
      0360 0E03
0527  0362 0460          B     @ERR
      0364 0008'
```

```
0529                    ********************************************************
0530                    *
0531                    *       Fetch next formal and prep for assignment
0532                    *
0533                    *       Register modification:
0534                    *            R5 Addr of s.t. entry(formal's entry)
0535                    *            R6 Header byte of formal's entry
0536                    *            R7 Updated formal's pointer
0537                    *
0538                    *       Destroys: R1,R2,R3,R4,R11,R12
0539                    *
0540                    ********************************************************
0541 0366 C30B  SCAL80  MOV   R11,R12           Save return address
0542 0368 CCC7          MOV   R7,R3             Fetch symbol pointer
0543 036A 05C7          INCT  R7                Point to next formal
0544 036C 06A0          BL    @GET1             Fetch s.t. pointer
     036E 02F6'
0545 0370 C0C1          MOV   R1,R3             Set condition & put in place
0546 0372 13F5          JEQ   SCAL88            If too many actuals
0547 0374 C101          MOV   R1,R4             Save for below
0548 0376 C141          MOV   R1,R5             Save for return
0549 0378 06A0          BL    @GET1             Get header bytes
     037A 036E'
0550 037C 2060          COC   @SHRFLG,R1        Shared?
     037E 0006'
0551 0380 1313          JEQ   SCAL82            Yes--reset flag and old valu
0552 0382 C181          MOV   R1,R6             Save for return & test strin
0553 0384 1101          JLT   SCAL81            If it is a string, then SCAL
0554 0386 045C          B     *R12              Return.
0555 0388 0223  SCAL81  AI    R3,6              Is string--point at value p
     038A 0006
0556 038C 06A0          BL    @GET1             Get the value ptr
     038E 037A'
0557 0390 C101          MOV   R1,R4             Null value?
0558 0392 1312          JEQ   SCAL86            Yes
0559 0394 04C1          CLR   R1                No--must free current string
0560 0396 0224          AI    R4,-3             Point at the backpointer
     0398 FFFD
0561 039A 06A0          BL    @PUT1             Clear the backpointer
     039C 0350'
0562 039E C103          MOV   R3,R4
0563 03A0 04C1  SCAL84  CLR   R1                Needed for entry from below
0564 03A2 06A0          BL    @PUT1             Clear the forward pointer
     03A4 039C'
0565 03A6 045C          B     *R12              Just return
0566 03A8 0241  SCAL82  ANDI  R1,>DFFF          Reset the share flag
     03AA DFFF
0567 03AC 06A0          BL    @PUTV1            Put it there
     03AE 029E'
0568 03B0 0224          AI    R4,6              Point at ref pointer
     03B2 0006
0569 03B4 C1B1          MOV   R1,R6             Set for return
0570 03B6 11F4          JLT   SCAL84            If string clear ref pointer
0571 03B8 045C  SCAL86  B     *R12              Return
0572                    *
0573                    *
```

```
0575                    ***************************************************************
0577                    *                                                             *
0578                    *       Execute a SUBEXIT or SUBEND                           *
0579                    *                                                             *
0580                    ***************************************************************
0581 03DA               SUBXIT
0582 03DA C160                  MOV   @LSUBP,R5         Check for subprogram on stack
     03DC 0322'
0583 03DE 1333                  JEQ   SCAL98            Not one--so error
0584 03E0 8805                  C     R5,@VSPTR         Extra check on stack pointer
     03E2 0320'
0585 03E4 1830                  JH    SCAL98            Pointers are messed up--error
0586 03E6 06A0  SBXT05          BL    @VPOP             Get stack entry
     03E8 0272'
0587 03EA 9820                  CB    @FAC2,@CBH6A      Reached the subprogram entry?
     03EC 0248'
     03EE 0207'
0588 03F0 16FA                  JNE   SBXT05            Not yet
0589                    *
0590                    *       Reached the subprogram stack entry.  Get information.
0591                    *       FAC  area has subprogram's table poiner, >6a,
0592                    *       on warning bits and LSUBP.
0593                    *
0594 03F2 020C                  LI    R12,FAC           Optimize for the copies.
     03F4 02DC'
0595 03F6 C00C                  MOV   R12,R0            For this copy
0596 03F8 C0F0                  MOV   *R0+,R3           Subprogram pointer
0597 03FA 06A0                  BL    @GETV1            Get header byte in subprogram
     03FC 0068'
0598 03FE 5060                  SZCB  INUSE,R1          Reset the in-use bit
     0400 0002'
0599 0402 C103                  MOV   R3,R4
0600 0404 06A0                  BL    @PUTV1            Put it back
     0406 03AE'
0601 0408 C070                  MOV   *R0+,R1           On warning bits
0602 040A D120                  MOVB  @FLAG,R4          Get the current flag
     040C 030C'
0603 040E 0244                  ANDI  R4,>F900          Trash current warning bits
     0410 F900
0604 0412 F120                  SOCB  @R1LB,R4          OR the old ones back in
     0414 0000
0605 0416 D804                  MOVB  R4,@FLAG          And put flag back
     0418 03EC'
0606 041A 05C0                  INCT  R0                There is one word empty.
0607 041C C830                  MOV   *R0+,@LSUBP       Last subprogram block on stack
     041E 03BC'
0608                    *
0609                    *       Second subprogram stack entry. Restore pointers.
0610                    *       FAC area has PGMPTR, EXTRAM, SYMTAB, RAM(SYMBOL).
0611                    *
0612 0420 06A0                  BL    @VPOP             Get second entry
     0422 03C8'
0613 0424 C00C                  MOV   R12,R0            Put FAC in R0. (optimization)
0614 0426 0201                  ~~LI  R1,PGMPTR~~       For optimization
     0428 0326'
0615 042A C470                  MOV   *R0+,*R14         Restore text pointer    PGMPTR
0616 042C 0631                  DEC   ~~*R1+~~R14       Save code to decrement it
0617 042E CC70                  MOV   *R0+,~~*R1+~~EXTRAM  Line table pointer EXTRAM
0618 0430 C830                  MOV   *R0+,@SYMTAB      Restore symbol table pointer
     0432 034C'
```

```
0619 0414 C070          MOV   *R0+,R1          Restore permanent s.t. pointer
0620 0416 0204          LI    R4,SYMBOL        Place in VDP
     0418 0344'
0621 041A 06A0          BL    @FUT1            Put it out there
     041C 03A4'
0622                    *
0623 041E 06A0          BL    @PGMCHR          Load R8 with EOS/EOL & go on
     0420 0358'
0624 0422 04C0          B     @EOL
     0424 0000
0625 0426 0200  SCAL98  LI    R0,ERRSND        * SUBEND NOT IN SUBPROGRAM
     0428 1203
0626 042A 0460          B     @ERR
     042C 0364'
```

```
0629                    ****************************************************************
0630                    *                                                              *
0631                    *       RESOLV - Attempts to resolve all subprograms           *
0632                    *     referenced in call statements by first searching         *
0633                    *     the internal subprogram table (SUBTAB), then by          *
0634                    *     searching GROMs for GPL subprograms.  In RESGPL,         *
0635                    *     GROM link list is searched and if name found then        *
0636                    *     it builds a subprogram table.                            *
0637                    *     If, after searching all of the subprogram areas,         *
0638                    *     there are any subprograms whose location cannot          *
0639                    *     be determined, an error occurs.                          *
0640                    *                                                              *
0641                    ****************************************************************
0642 042E 05C9  RESOLV  INCT R9                  Save return address
0643 0430 C64B          MOV  R11,*R9
0644 0432 C160          MOV  @CALIST,R5         Pick up call list pointer
     0434 0000
0645 0436 1337          JEQ  RES50              If no subprogram references
0646 0438 C1A0  RES03   MOV  @SUBTAB,R6         Pick up subprogram table ptr
     043A 0056'
0647 043C 1327  RES05   JEQ  RES15              Try to resolve by checking
0648                    *                                                             *
0649                    *     Compares two names for a match when                     *
0650                    *     trying to resolve all references to subprograms.        *
0651                    *                                                             *
0652                    *     Register usage is generally as follows:                 *
0653                    *          R5 - Pointer to CALIST entry to be compared        *
0654                    *          R7 - Pointer to entry to be compared to SUBTAB     *
0655                    *               Returns as pointer to name if found or        *
0656                    *               zero if not found                             *
0657                    *          R10- Returned as length of name                    *
0658                    *                                                             *
0659                    *                                                             *
0660 043E C0C6          MOV  R6,R3              Put in place for GETV
0661 0440 0583          INC  R3                 Point at the name length
0662 0442 06A0          BL   @GETV1             Get the name length
     0444 03DC'
0663 0446 0981          SRL  R1,8               Put in Lsb and clear Msb
0664 0448 C101          MOV  R1,R4              Save it for the move
0665 044A 0223          AI   R3,3               Point at name pointer
     044C 0003
0666 044E 06A0          BL   @GET1              Get the name pointer
     0450 038E'
0667 0452 C1C1          MOV  R1,R7              Save in permanent
0668 0454 C801          MOV  R1,@PGMPTR         Save for compare
     0456 0408'
0669 0458 C0C5          MOV  R5,R3              To get the CALIST entry
0670 045A 0583          INC  R3                 Point at the name length
0671 045C 06A0          BL   @GETV1             Get the name length
     045E 0444'
0672 0460 9801          CB   R1,@R4LB           Name length match?
     0462 0000
0673 0464 161A          JNE  RES20              No--no match possible
0674 0466 C004          MOV  R4,R0              Save name length for compare
0675 0468 0223          AI   R3,3               Point at the name pointer
     046A 0003
0676 046C 06A0          BL   @GET1              Get the pointer to the name
     046E 0450'
0677 0470 C0C1          MOV  R1,R3              Set up to get the name
0678 0472 06A0  COMP10  BL   @GETV1             Get a char of CALIST name
```

```
      0474 048E'
0679                  *       Next PGMSUB call is the same as PGMCHR except in
0680                  *       skipping ERAM check
0681 0476 06A0                BL    @PGMSUB           Get a char of found name
     0478 0000
0682 047A 9281                CB    R1,R8             Chars match?
0683 047C 16DE                JNE   RES30             No--not same name
0684 047E 0583                INC   R3                Next character
0685 0480 0600                DEC   R0                Done with compare?
0686 0482 15F7                JNE   COMP10            No--check the rest
0687                  *
0688                  *       Found the subprogram in GROM and built the table.
0689                  *       Set resolved flag and get back.
0690                  *
0691 0484 C105                MOV   R5,R4             Set resolved flag now
0692 0486 0701                SETO  R1                Set up a resolved flag
0693 0488 06A0                BL    @PUTV1            And put the byte in
     048A 03E6'
0694                  *
0695 048C C0C5        RES15   MOV   R5,R3             Get call list pointer
0696 048E 0583                INCT  R3                Point at link
0697 0490 06A0                BL    @GET1             Get the new link
     0492 046E'
0698 0494 C141                MOV   R1,R5             Save and set condition
0699 0496 130E                JEQ   RESGPL            End of call list? Yes.
0700 0498 16CF                JNE   RES03             No. Go check the next in list.
0701 049A             RES20
0702 049A C0C6                MOV   R6,R3             Get next entry in subp table
0703 049C 0583                INCT  R3                Point at the link
0704 049E 06A0                BL    @GET1             Get the link
     04A0 0492'
0705 04A2 C181                MOV   R1,R6             Update subprogram table pointer
0706 04A4 10CB                JMP   RES05             And try next entry
0707                  *
0708 04A6 0403        RES50   CLR   R3                Indicate no error return
0709 04A8 C2D9        RES51   MOV   *R9,R11           Restore return address.
0710 04AA 0649                DECT  R9                Restore stack.
0711 04AC 045B                RT                      All resolved and O.K.
0712 04AE 0203        RES52   LI    R3,>001C
     04B0 001C
0713 04B2 10FA                JMP   RES51
```

```
0715              ***************************************************************
0716              *                    RESGPL routine                           *
0717              *                                                             *
0718              *      Resolves as a GPL subprogram by comparing names in     *
0719              *      CALL list and GROM link list in EXEC.  If name         *
0720              *      found in GROM then turn the resolved flag on and if    *
0721              *      not found an error occurs.  Fetch subprogram           *
0722              *      access address from the link list and builds           *
0723              *      builds a subprogram table for that call.               *
0724              *                                                             *
0725              ***************************************************************
0726 04B4        RESGPL
0727 04B4 C160           MOV  @CALIST,R5  Get the call list pointer.
     04B6 0434'
0728              *      Get the next subprogram in the call list that has
0729              *      not been resolved.
0730 04B8 CCC5   GETO1   MOV  R5,R3               Get pointer in call list
0731 04BA 13F5           JEQ  RES50               If end of list
0732 04BC 06A0           BL   @GETV1              Get the resolved flag
     04BE 0474'
0733 04C0 1306           JEQ  GPL00               If not resolved
0734 04C2 05C3   GETO3   INCT R3                  Point at link
0735 04C4 06A0           BL   @GET1               Get the link
     04C6 04A0'
0736 04C8 C141           MOV  R1,R5               Save it and set condition
0737 04CA 16F6           JNE  GETO1               If not end of list--go on
0738 04CC 10EC           JMP  RES50               Return.
0739              *
0740              *      Start looking at GROM subprogram link list.
0741              *
0742 04CE        GPL00
0743 04CE 0207           LI   R7,GPLIST           Load address of link list.
     04D0 A026
0744 04D2 CCC5           MOV  R5,R3               Copy CALIST address.
0745 04D4 0583           INC  R3                  Point to name length.
0746 04D6 06A0           BL   @GETV1              Get the name length.
     04D8 048E'
0747 04DA 0981           SRL  R1,8                Adjust to the right byte.
0748 04DC C001           MOV  R1,R0               Copy for later use.
0749 04DE 04CA           CLR  R10                 Clear for name length.
0750 04E0 0223           AI   R3,3                Pnt to name ptr in call list
     04E2 0003
0751 04E4        GPL10
0752              *---------- CONDITIONAL ASSEMBLY ----------------------------*
0753                     ASMIF VERS=DX10
0754
0755                     MOV  R7,R15              Get the address of linked list
0756                     AI   R15,GROM            Add offset for GROM
0757                     MOVB *R15+,R8            Get the next link in reg 8
0758                     MOVB *R15+,@R8LB
0759                     INCT R7                  Points to name length in GROM
0760                     MOVB *R15+,@R10LB        R15 points to 1st character
0761
0762                     ASMELS
0763 04E4
0764 04E4 DB47           MOVB R7,@GRMWA(R13)      Specify address in link list
     04E6 0094'
0765 04E8 06C7           SWPB R7                  Need to kill time here.
0766 04EA DB47           MOVB R7,@GRMWA(R13)      Move next byte.
     04EC 04E6'
```

```
                        SWPB R7                 Get R7 in right order.
                        MOVB *R13,R8            Read next link address from
                        MOVB *R13,@R8LB             linked list

                        INCT R7                 Point to name length in GROM
                        MOVB R7,@GRMWA(R13)     Specify name length address

                        SWPB R7                 Need to kill time here.
                        MOVB R7,@GRMWA(R13)     Move next byte.

                        SWPB R7                 Get R7 in right order.
                        MOVB *R13,@R10LB        Get the name length in GROM

                        ASMEND
                *-------------- END OF CONDITIONAL ASSEMBLY ----------------

                        C    R0,R10             Compare name length.
                        JEQ  GPL25              If matches, compare names
                GPLNXT
                        MOV  R8,R7              Didn't match-get link to next
                        JNE  GPL10              Loop if not end of list
                        MOV  R5,R3              If end of GPL list-ignore thi
                        JMP  GET03                  entry in calist
                *
                *       Start comparing the names.
                *
                GPL25
                        BL   @GET1              Get name pntr from call list

                *                               R1 contains address of name

                *---------------- CONDITIONAL ASSEMBLY ------------------
                        ASMIF       VERS=DX10
                        MOV  R1,R14             Get one character from VDP
                        AI   R14,VRAM           Add VDP offset
                GPL30   CB   *R14+,*R15+        Compare with the one in GROM

                        ASMELS

                        MOVB @R1LB,*R15         Get one character from VDP.

                        NOP
                        MOVB R1,*R15            Then compare with the one in
                GPL30   CB   *R13,@VDPRD            GROM - R13 points to GROM

                        ASMEND
                *---------------- END OF CONDITIONAL ASSEMBLY ----------------

                        JNE  GPLNXT             If no match get next in GROM
                        DEC  R10                All matched?
                        JNE  GPL30              No-loop for next characters
                *
                *       Found the GPL subprogram. Now start building GPL's
                *       subprogram table..
                *       First put all information in FAC scince they might
                *       get destroyed in MEMCHK.
                *       @FAC2 = Set program bit and name length.
                *       @FAC4 = Subprogram table link address
                *       @FAC6 = Pointer to name..
                *       @FAC8 = Access address in GROM.
```

```
0820                *       @FAC10 = Current call list address.
0821                *
0822 052A 0200              LI    R12,FAC2          Optimize for speed and space
     052C 0300'
0823 052E C700              MOV   R0,*R12           Keep length in FAC2
0824 0530 EF20              SOC   @FNCFLG,*R12+     Set program bit .
     0532 0004'
0825 0534 CF20              MOV   @SUBTAB,*R12+     Set up subtable link address.
     0536 043A'
0826 0538 06A0              BL    @GET1             Get pointer to name.
     053A 0516'
0827 053C CF01              MOV   R1,*R12+          Move it to FAC6.
0828                *-------------  CONDITIONAL ASSEMBLY  ----------------------*
0829                        ASMIF       VERS=DX10
0830
0831                        MOVB  *R15+,*R12+       Put access address in FAC8.
0832                        MOVB  *R15,*R12+
0833
0834                        ASMELS
0835 053E
0836 053E DF1D              MOVB  *R13,*R12+        Get access address from GROM
0837 0540 1000              NOP
0838 0542 DF1D              MOVB  *R13,*R12+        and put it in FAC8.
0839                        ASMEND
0840                *-------------  END OF CONDITIONAL ASSEMBLY  -------------*
0841                *
0842 0544 C705              MOV   R5,*R12           Save current call list addr
0843 0546
0844                *       Check if ERAM exists or imperative statement.
0845                *       If so then copy name into appropriate VDP area.
0846 0546 D1A0              MOVB  @RAMFLG,R6        ERAM present?
     0548 001A'
0847 054A 1603              JNE   GPL40             Yes - then save name in table
0848 054C D1A0              MOVB  @PRGFLG,R6        Imperative call?
     054E 0072'
0849 0550 1619              JNE   GPL60             No--handle normally
0850                *       Copy name into table area
0851 0552 C800  GPL40       MOV   R0,@FAC           Copy name length.
     0554 03D4'
0852 0556 06A0              BL    @MEMCHK           Get the space. FAC = name lengt
     0558 0000
0853 055A 04AE'             DATA  RES52             Error return address
0854 055C C0E0              MOV   @FAC6,R3          Get pointer to name
     055E 0318'
0855 0560 6820              S     @FAC,@FREPTR      New free pointer
     0562 0554'
     0564 0000
0856 0566 C120              MOV   @FREPTR,R4        New place of name
     0568 0564'
0857 056A 0584              INC   R4
0858 056C C804              MOV   R4,@FAC6          New pointer to name
     056E 055E'
0859 0570 CCA0              MOV   @FAC,R2           Counter for the move
     0572 0562'
0860                *       Now copy the name - character by character
0861 0574 06A0  GPL50       BL    @GETV1            Get a byte
     0576 04D8'
0862 0578 06A0              BL    @PUTV1            Put a byte
     057A 048A'
0863 057C 0583              INC   R3
```

```
----REF RESOLUTION----

0864 057E 0984          INC  R4
0865 0580 0602          DEC  R2                 Done?
0866 0582 16F8          JNE  GPL50              No--move the rest
0867 0584
0868                *   Restore all the information from FAC area and
0869                *   build subprogram's symbol table
0870 0584           GPL60
0871 0584 0820          MOV  @C8,@FAC           Need 8 bytes
     0586 02D2'
     0588 0572'
0872 058A 06A0          BL   @MEMCHK            Get the bytes. Check the space
     058C 0558'
0873 058E 04AE'         DATA RES52              Error return address
0874 0590 6820          S    @C8,@FREPTR        Update the free pointer
     0592 0586'
     0594 0568'
0875 0596 C020          MOV  @FREPTR,R0         Get location to move to
     0598 0594'
0876 059A 0580          INC  R0                 True pointer
0877 059C C800          MOV  R0,@SUBTAB         Update subprogram table ptr
     059E 053A'
0878 05A0 0201          LI   R1,FAC2            Subprogram' info starts FAC2.
     05A2 052C'
0879                *
0880                *-------------------CONDITIONAL ASSEMBLY----------------
0881                    ASMIF VERS=DX10
0882
0883                    AI   R0,VRAM            VDP offset
0884                    MOVB *R1+,*R0+          Mode byte
0885                    MOVB *R1+,*R0+          Name length
0886                    MOVB *R1+,*R0+          Link
0887                    MOVB *R1+,*R0+
0888                    MOVB *R1+,*R0+          Pointer to name
0889                    MOVB *R1+,*R0+
0890                    MOVB *R1+,*R0+          Pointer to sub
0891                    MOVB *R1+,*R0
0892
0893                    ASMELS
0894 05A4
0895 05A4 D7E0          MOVB @ROLB,*R15         Load out address
     05A6 0042'
0896 05A8 0260          ORI  R0,WRVDP           Enable VDP write
     05AA 002C'
0897 05AC D7C0          MOVB R0,*R15
0898 05AE 0200          LI   R0,VDPWD           Optimize to save bytes
     05B0 0032'
0899 05B2 0203          LI   R3,8               Going to move 8 bytes
     05B4 0008
0900 05B6 D431      GPL70 MOVB *R1+,*R0         Copy mode, name length, link
0901 05B8 0603            DEC  R3                   ptr to name, ptr to subpr
0902 05BA 16FD            JNE  GPL70
0903 052C
0904                    ASMEND
0905                *-----------------END OF CONDITIONAL ASSEMBLY----------
0906 05BC
0907 05BC C0D1          MOV  *R1,R3             Restore ptr into call list
0908 05BE 0460          B    @GET03             Check next entry in call li
     05C0 04C2'
0909                    END
NO ERRORS,    0013 WARNINGS
```

```
ARG      R  0304'   0069   0473
ARG2     R  0322'   0069   0461   0509
ARG3     R  032A'   0069   0507
ARG4     R  0282'   0069   0245   0389   0423
ASEG     R  0230'   0058   0097
C16      R  0226'   0058   0387   0393
CB       R  0592'   0058   0395   0457   0871   0874
CALIST   R  04B6'   0064   0644   0727
CALL     D  0008'   0124   0055
CALL02      0030'   0148   0153
CALL04      0038'   0154   0130
CALL06      004C'   0169   0174
CBH6A       0207'   0385   0479   0587
COMMA$      00B3    0084   0249   0267   0272   0288   0412
COMP10      0472'   0678   0686
CONTAD      0000'   0078   0196
CRNBUF   R  0022'   0065   0133
DX10        0001    0003   0004   0074   0136   0157   0201   0493   0753   0794   0829
                           0881
EOL      R  0424'   0058   0624
EOSTMT   R  02B8'   0059   0270   0445
ERR      R  042C'   0059   0240   0527   0626
ERR1     R  0140'   0059   0276
ERRCOD   R  0354'   0063   0252   0378   0407   0523
ERRIAL      0E03    0099   0526
ERROLP      1103    0100   0189
ERRREC      0F03    0098   0237
ERRSND      1203    0097   0625
FAC      R  0588'   0068   0155   0258   0293   0321   0391   0410   0435   0449   0459
                           0594   0851   0855   0859   0871
FAC15    R  000E'   0069   0125
FAC2     R  05A2'   0068   0225   0380   0404   0587   0822   0878
FAC4     R  02AE'   0068   0379   0438
FAC6     R  056E'   0068   0417   0471   0484   0854   0858
FAC8     R          0068
FBS001   R  005C'   0058   0177
FLAG     R  03FB'   0066   0480   0602   0605
FNCFLG      0004'   0091   0260   0824
FREPTR   R  0598'   0064   0855   0856   0874   0875
GET01       04B8'   0730   0737
GET03       04C2'   0734   0785   0908
GET1     R  053A'   0060   0199   0231   0329   0348   0364   0450   0470   0544   0549
                           0556   0666   0676   0697   0704   0735   0790   0826
GETG2    R  01B6'   0060   0338
GETV     R  028A'   0059   0257   0292   0425
GETV1    R  0576'   0059   0182   0597   0662   0671   0678   0732   0746   0861
GPL00       04CE'   0742   0733
GPL10       04E4'   0751   0783
GPL25       0514'   0789   0780
GPL30       0520'   0804   0810
GPL40       0552'   0851   0847
GPL50       0574'   0861   0866
GPL60       0584'   0870   0849
GPL70       05B6'   0900   0902
GPLIST      A026    0079   0743
GPLNXT      050C'   0781   0808
GPLSU       007C'   0195   0185
GRAM     R          0067
GRMWA    R  0500'   0070   0208   0210   0764   0766   0771   0773
GRMWD    R          0067
```

LABEL		VALUE	DEFN	REFERENCES								
GROM	R		0067									
INUSE		0003'	0090	0222	0598							
UNEEB		020A'	0446	0429								
UPARM		0007	0086	0234	0262	0409						
UBUSP	R	00FE'	0064	0484	0486	0582	0607					
VEMCHR	R	0580'	0061	0252	0872							
WUSEMB	R	0350'	0061	0525								
P359		0000	0003	0003								
PAD	R	028C'	0068	0410	0426							
PGMCHR	R	0420'	0061	0124	0180	0253	0264	0269	0287	0381	0444	0524
				0623								
PGMPTR	R	0456'	0063	0128	0154	0252	0378	0403	0417	0465	0491	0614
				0668								
PGMSUB	R	0478'	0061	0681								
POPSTK	R	02D8'	0066	0244	0386	0422	0458					
PRGFLG	R	054E'	0064	0186	0848							
PSHPRS	R	0204'	0061	0383								
PUT1	R	041C'	0060	0372	0521	0561	0564	0621				
PUTG2	R	01E6'	0060	0369								
PUTL1	R	057A'	0059	0223	0305	0433	0567	0600	0693	0862		
R0		0000		0126	0132	0133	0146	0147	0166	0167	0189	0237
				0239	0298	0299	0300	0465	0466	0467	0491	0511
				0526	0595	0596	0601	0606	0607	0613	0615	0617
				0618	0619	0625	0674	0685	0748	0779	0823	0851
				0875	0876	0877	0896	0897	0898	0900		
R0LB	R	05A6'	0057	0145	0165	0895						
R1		0001		0129	0155	0169	0184	0208	0209	0210	0221	0222
				0232	0232	0260	0294	0296	0297	0300	0303	0321
				0322	0328	0337	0347	0362	0365	0368	0427	04(
				0431	0435	0438	0451	0451	0460	0466	0467	04[illegible]
				0471	0508	0511	0514	0516	0520	0545	0547	0548
				0550	0552	0557	0559	0563	0566	0569	0598	0601
				0614	0615	0616	0617	0619	0663	0664	0667	0668
				0672	0677	0682	0692	0698	0705	0736	0747	0748
				0803	0827	0878	0900	0907				
R10		000A		0749	0779	0809						
R10LB	R	0506'	0058	0775								
R11		000B		0186	0404	0405	0406	0407	0541	0643	0709	
R12		000C		0224	0227	0255	0259	0459	0460	0461	0477	0541
				0554	0565	0571	0594	0595	0613	0822	0823	0824
				0825	0827	0836	0838	0842				
R13		000D		0197	0208	0210	0764	0766	0768	0769	0771	0773
				0775	0804	0836	0838					
R15		000F		0145	0147	0165	0167	0507	0509	0801	0803	0895
				0897								
R1LB	R	051A'	0057	0604	0801							
R2		0002		0294	0296	0325	0333	0345	0480	0481	0482	0859
				0865								
R2LB	R		0057									
R3		0003		0134	0152	0181	0198	0224	0230	0328	0337	0347
				0363	0447	0469	0542	0545	0555	0562	0596	0599
				0660	0661	0665	0669	0670	0675	0677	0684	0695
				0696	0702	0703	0708	0712	0730	0734	0744	0745
				0750	0784	0854	0863	0899	0901	0907		
R3LB	R		0057									
R4		0004		0126	0127	0134	0154	0173	0175	0181	0227	0[illegible]
				0304	0306	0352	0363	0365	0432	0434	0477	0478
				0479	0482	0510	0512	0515	0547	0557	0560	0562
				0568	0599	0602	0603	0604	0605	0620	0664	0674
				0691	0856	0857	0858	0864				

```
R4LB    R   04xx'   0057   0672
R5          00C5           0132  0148  0304  0391  0432  0548  0582  0584  0644
                           0669  0691  0695  0698  0727  0730  0736  0744  0784
                           0842
R5LB    R           0057
R6          00C6           0298  0302  0303  0324  0331  0331  0343  0343  0353
                           0353  0360  0362  0368  0428  0436  0436  0552  0569
                           0646  0660  0702  0705  0846  0848
R6LB    R           0057
R7          0007           0228  0229  0230  0245  0246  0255  0259  0379  0389
                           0406  0423  0447  0448  0449  0542  0543  0667  0743
                           0764  0765  0766  0767  0770  0771  0772  0773  0774
                           0782
R8          0008           0125  0126  0234  0247  0249  0262  0265  0267  0272
                           0274  0288  0290  0409  0412  0414  0682  0768  0782
R8LB    R   04F4'   0057   0769
R9          0009           0195  0196  0197  0642  0643  0709  0710
RAMFLG  R   0548'   0064   0129  0846
RAMTOP  R   01D4'   0063   0333  0345  0360
RES03       043F'   0646   0700
RES05       0430'   0647   0706
RES15       048C'   0695   0647
RES20       049A'   0701   0673  0683
RES50       04A6'   0708   0645  0731  0738
RES51       04A8'   0709   0713
RES52       04AE'   0712   0853  0873
RESET   R   009C'   0061   0215
RESGPL      04B4'   0726   0699
RESDLV  D   042E'   0642   0055
RPAR$       00B6    0085   0247  0265  0274  0290  0384  0414
SAVREG  R   0098'   0062   0214
SBXTOS      03C6'   0586   0588
SCAL01      009E'   0220   0187
SCAL05      00DC'   0243   0398
SCAL06      00E6'   0247   0282
SCAL08      00F2'   0252   0236
SCAL10      0128'   0269   0273
SCAL12      013E'   0276   0241  0250  0271
SCAL14      0154'   0287   0266  0275
SCAL15      0158'   0288   0263
SCAL16      0164'   0292   0289
SCAL23      01BA'   0343   0326
SCAL24      01CA'   0350   0332  0334  0339  0344  0346  0437  0439
SCAL26      01EA'   0371   0354  0361
SCAL30      0146'   0279   0248
SCAL32      0142'   0278   0233
SCAL34      014A'   0280   0235  0301
SCAL35      014E'   0281   0268
SCAL37      0152'   0282   0370  0373
SCAL40      01F0'   0378   0254  0261  0291
SCAL42      01FE'   0381   0418
SCAL50      0242'   0403   0281
SCAL54      027C'   0422   0413  0415
SCAL60      02B2'   0444   0279
SCAL62      02B6'   0445   0278
SCAL70      0338'   0511   0513
SCAL80      0366'   0541   0295  0390  0424
SCAL81      0388'   0555   0553
SCAL82      03A8'   0566   0551
SCAL84      03A0'   0563   0570
```

```
SCALSE      033E'   0571  0558
SCALSS      033E'   0526  0280  0446  0452  0546
SCALSF      0332    0539  0176  0178
SCALPO      0000'   0287  0193
SCALPL      0106'   0240  0190  0238
SCALPB      00DA'   0241
SCALPS      0416'   0623  0569  0585
S-RFLG      0016'   0092  0302  0431  0550
SMB     R   025E'   0062  0392  0411
SUBTAB  R   059E'   0064  0175  0646  0825  0877
SUBXIT  D   038A'   0581  0055
SYM     R   0102'   0062  0256
SYMBOL  R   0418'   0065  0469  0515  0620
SYMTA1  R   0348'   0065  0516
SYMTAB  R   0412'   0065  0468  0514  0520  0618
UNOST$      0008    0087
VDPRD   R   0522'   0070  0169  0508  0804
VDPWD   R   0580'   0066  0148  0898
VERMAC  M           A0001 0003
VERS        0000    0003  0004  0074  0136  0157  0201  0493  0753  0794  0829
                          0881
VPOP    R   0402'   0060  0396  0416  0586  0612
VPUSH   R   031C'   0060  0226  0388  0394  0408  0456  0472  0485
VRAM    R           0070
VSPTR   R   0302'   0066  0387  0393  0395  0457  0486  0584
VRVOP   R   05AA'   0066  0146  0896
```