```
                       TITLE SUBS
                      ************************************************************
                      *
                      *          SSSSS   U      U    BBBBBB      SSSSS
                      *        S     S  U      U    B     B    S     S
                      *        S        U      U    B     B    S
                      *         SSSSS   U      U    BBBBBB      SSSSS
                      *              S  U      U    B     B          S
                      *        S     S  U      U    B     B    S     S
                      *         SSSSS    UUUUU     BBBBBB      SSSSS
                      *
                      *
                      *           33333      555555      99999
                      *          3     3     5          9     9
                      *                3     5555      9     9
                      *             3333          5     999999
                      *                3          5          9
                      *          3     3     5    5         9
                      *           33333       5555         9
                      *
                      *
                      ************************************************************
                      *
                      *
                      *      NOTE: MOST OF ERROR CALLS ARE DIRECTLY REFERENCED
                      *            TO THOSE IN EXECS FILE.  ANY CHANGE IN EXECS
                      *            FILE WILL REQUIRE CHNAGES OF THESE EQUATES.
                      *
                      ************************************************************
6A70                  M$PSCN EQU  >6A70          Module PSCAN branch table ad
A000                  M$EXEC EQU  >A000          Module EXEC branch table addr
6010                  M$EDIT EQU  >6010          Module EDIT branch table addr
0010                  M$MON  EQU  >0010          Module MONITOR branch table add

0012                  RPL    EQU  M$MON+>02        Return Program Link
001E                  CHAR2$ EQU  M$MON+>08        Load small character set

6AB4                  ERR$$  EQU  M$PSCN+>14        Error routine
6A78                  CHKEND EQU  M$PSCN+>08        Check end of statement

6016                  SPRINT EQU  M$EDIT+>06        Initialize sprites

A01E                  SPCOL  EQU  M$EXEC+>1E        Changes color of sprite
A01C                  RETURN EQU  M$EXEC+>1C        Return to basic program
AD53                  ERRSYN EQU  >AD53             Direct reference.......
AD57                  ERRSNM EQU  >AD57             Direct reference.......
AD87                  ERRBV  EQU  >AD87             Direct reference.......
AD8B                  ERRIAL EQU  >AD8B             Direct reference.......
                      *                    XML  Equates
0012                  FLTINT EQU  >12               Convert floating to integer
0071                  GETSTR EQU  >71               System Getstring routine
0074                  PARSE  EQU  >74               Parse a value
0077                  VPUSH  EQU  >77               Push the FAC onto the stack
0078                  VPOP   EQU  >78               Value stack pop
0079                  PGMCHR EQU  >79               Get next program character
```

```
007A      56  SYM     EQU  >7A                Find symbol table entry
007B      57  SMB     EQU  >7B                Find symbol's value space
007C      58  ASSGNV  EQU  >7C                Assign a value
007E      59  SPEED   EQU  >7E                Subprogram speed up
0000      60  SYNCHK  EQU  0                   SYNtax CHecK XML selector
0002      61  RANGE   EQU  2                   RANGE XML selector
0080      62  CIF     EQU  >80                Convert integer to floating no
          63  ********************************************************
0072      64  STACK   EQU  >72                STACK FOR DATA
0073      65  SUBSTK  EQU  >73                SUBROUTINE STACK
0074      66  KEYBD   EQU  >74                KEYBOARD SELECTION
0075      67  RKEY    EQU  >75                KEY CODE
0076      68  JOYY    EQU  >76                JOYSTICK Y POSITION
0077      69  JOYX    EQU  >77                JOYSTICK X POSITION
0078      70  RANDOM  EQU  >78                RANDOM NUMBER GENERATOR
0079      71  TIMER   EQU  >79                TIMING REGISTER
007A      72  MOTION  EQU  >7A                NUMBER OF SPRINTES IN MOTION
007B      73  VDPSTT  EQU  >7B                VDP STATUS BYTE
          74  ********************************************************
          75  *               Temporary Workspace in CPU RAM
          76  *       The following used for sprites
0000      77  SP00    EQU  >00
0002      78  SP02    EQU  >02
0004      79  SP04    EQU  >04
0006      80  SP06    EQU  >06
0008      81  SPSAL   EQU  >08          Location of sprite attribute list.
000A      82  SPTMP   EQU  >0A          Temporary variable
          83  *       The following used for speech
0000      84  PTFBSL  EQU  >00                Ptr to 1st byte in Speak List
0002      85  PTLBSL  EQU  >02                Ptr to last byte in Speak List
0004      86  PTEBSL  EQU  >04                Ptr to end byte in Speak List
          87  * Note that PTEBSL points to the end of the temp speak list,
          88  * whereas PTLBSL points to the last byte actually used.
          89  * ie.  PTFBSL <= PTLBSL <= PTEBSL
0000      90  PHLEN   EQU  >00                PHrom data length
0001      91  PHRADD  EQU  >01                PHRom ADDress
          92
0006      93  PTFCIS  EQU  >06                Ptr to 1st char in string
0008      94  PTCCIS  EQU  >08                Ptr to current char in string
000A      95  PTLCIS  EQU  >0A                Ptr to last char in string
          96
000C      97  BYTE    EQU  >0C                # bytes to allocate for string
000C      98  PTFCIP  EQU  >0C                Ptr to 1st char in phrase
000E      99  PTCCIP  EQU  >0E                Ptr to current char in phrase
0010     100  PTLCIP  EQU  >10                Ptr to last char in phrase
         101
0012     102  PTFBPH  EQU  >12                Ptr to 1st byte in PHrom
0014     103  PTCCPH  EQU  >14                Ptr to current byte in PHrom
0016     104  PTLCPH  EQU  >16                Ptr to last byte in PHrom
         105  ********************************************************
         106  *               Permanent Workspace in CPU RAM
001B     107  STRSP   EQU  >1B                Start of string space
001A     108  STREND  EQU  >1A                End of string space
001C     109  SREF    EQU  >1C                Temproary string pointer
001E     110  SMTSRT  EQU  >1E                Start of current statement
```

```
0020    111  VARW   EQU  >20              Starting screen location
0022    112  ERRCOD EQU  >22              Return vector from ALC
0024    113  STVSPT EQU  >24              Base of value stack
0026    114  RTNG   EQU  >26              Return address for ALC
0028    115  NUDTAB EQU  >28              NUD table address for ALC
002A    116  VARA   EQU  >2A              Ending screen address
002C    117  PGMPTR EQU  >2C              Program text pointer
002E    118  EXTRAM EQU  >2E              Line # table pointer
0030    119  STLN   EQU  >30              Start of line number table
0032    120  ENLN   EQU  >32              End of line number table
0034    121  DATA   EQU  >34              DATA pointer for READ
0036    122  LNBUF  EQU  >36              Line table pointer for READ
        123  *      EQU  >38
003A    124  SUBTAB EQU  >3A              Subprogram symbol table
003C    125  IOSTRT EQU  >3C              PAB pointer
003E    126  SYMTAB EQU  >3E              Symbol table pointer
0040    127  FREPTR EQU  >40              Free space pointer for s.t.
0042    128  CHAT   EQU  >42              Current character/token
0043    129  BASE   EQU  >43              OPTION BASE value
0044    130  PRGFLG EQU  >44              Program/imperative flag
0045    131  FLAG   EQU  >45              General flag byte
0046    132  BUFLEV EQU  >46              Crunch-buffer destruction lev
0048    133  LSUBP  EQU  >48              Last subprogram block on stac
004A    134  FAC    EQU  >4A              Floating-point accumulator
004B    135  FAC1   EQU  FAC+1
004C    136  FAC2   EQU  FAC+2
004E    137  FAC4   EQU  FAC+4
0050    138  FAC6   EQU  FAC+6
0054    139  FAC10  EQU  FAC+10
        140  *      The following used by speech
004A    141  CCHAR  EQU  >4A              Current char
004B    142  SPLFLG EQU  >4B              SPeLL out phrase FLaG
        143  * Note that DATAD must follow immediately after TOTTIM.  Th
        144  * routine STDATA is counting on this fact!
004C    145  TOTTIM EQU  >4C              TOTal wait TIMe
004D    146  DATAAD EQU  >4D              Speech DATA ADdr
004F    147  PTLCIL EQU  >4F              Pointer To Last Char In List
0051    148  TIMLEN EQU  >51              TIMe LENght of timing char
0052    149  PHADDR EQU  >52              PHrom ADDRess
        150
0054    151  TEMP1  EQU  >54              TEMPorary CPU location 1
0056    152  TEMP2  EQU  >56              TEMPorary CPU location 2
        153
0058    154  READ   EQU  >58              Addr of speech peripheral
        155  *                            READ byte interface
005A    156  WRITE  EQU  >5A              Addr of speech peripheral
        157  *                            WRITE byte interface
005D    158  PHDATA EQU  >5D              PHrom DATA
005E    159  PTCBED EQU  >5E              Ptr To Current Byte Ext Data
0060    160  LENCST EQU  >60              LEN of Current ext data STrin
0062    161  LENWST EQU  >62              LEN of Whole ext data STring
0064    162  STRLEN EQU  >64              STRing LENght
        163  * Note that BYTE1, BYTE2, & BYTE3 must be in consecutive me
        164  * locations, and in the following order for SPGET to work!
0066    165  BYTE1  EQU  >66              BYTE 1
```

```
0067      166  BYTE2  EQU  >67        BYTE 2
0068      167  BYTE3  EQU  >68        BYTE 3
0069      168  SPKSTS EQU  >69        SPeaK StaTuS
          169  *
006C      170  FPERAD EQU  >6C
006E      171  VSPTR  EQU  >6E        Value stack pointer
          172  *
0089      173  GRAMFL EQU  >89        GRAM / VDP flag
```

```
                175  * ASCII symbol definitions
                176
0020            177  SPACE  EQU  : :
0023            178  NUMBER EQU  :#:
003B            179  SEMICO EQU  :;:
003A            180  COLON  EQU  ::::
002E            181  PERIOD EQU  :.:
002B            182  PLUS   EQU  :+:
002C            183  COMMAT EQU  :,:
002D            184  HYPEN  EQU  :-:
0030            185  ZERO   EQU  :0:
0039            186  NINE   EQU  :9:
                187
                188  *
                189  *              B A S I C   T O K E N   T A B L E
                190  *
00B3            191  COMMA$ EQU  >B3          ","
00B6            192  RPAR$  EQU  >B6          ")"
00B7            193  LPAR$  EQU  >B7          "("
00EC            194  ALL$   EQU  >EC          "ALL"
00FD            195  NUMBE$ EQU  >FD          "#"
```

```
                    197           GROM 5
                    198           ORG  >0D98
                    199
                    200           BASE 0,0,>800,>300,>400,>780,0
                    201   *                 -
                    202   *
                    203   *       SPRITE SUBROUTINES BRANCH TABLE
                    204   *
                    205   *
                    206
AD98 4FD4           207   CHAR1   BR   SPNUM3              Called in CHARLY.   EXEC
AD9A 4FCD           208   CHAR2   BR   SPNUM2              Called in CHARLY.   EXEC
AD9C 4D9C           209           BR   $                   Called in CHARLY.   EXEC
                    210   *
                    211   *
                    212   *       SUBROUTINE LINK LIST
                    213   *
AD9E ADA906         214   LINKS1 DATA #LINKS2,6,:SPRITE:,#SPRTE
ADA1 535052
ADA4 495445
ADA7 AE27
ADA9 ADB709         215   LINKS2 DATA #LINKS3,9,:DELSPRITE:,#SPRDEL
ADAC 44454C
ADAF 535052
ADB2 495445
ADB5 AE58
ADB7 ADC408         216   LINKS3 DATA #LINKS4,8,:POSITION:,#SPRPOS
ADBA 504F53
ADBD 495449
ADC0 4F4EAE
ADC3 83
ADC4 ADCE05         217   LINKS4 DATA #LINKS5,5,:COINC:,#SPRCOI
ADC7 434F49
ADCA 4E43AE
ADCD C4
ADCE ADDA07         218   LINKS5 DATA #LINKS6,7,:MAGNIFY:,#SPRMAG
ADD1 4D4147
ADD4 4E4946
ADD7 59AEF9
ADDA ADE506         219   LINKS6 DATA #LINKS7,6,:MOTION:,#SPRMOV
ADDD 4D4F54
ADE0 494F4E
ADE3 AF11
ADE5 ADF006         220   LINKS7 DATA #LINKS8,6,:LOCATE:,#SPRLOC
ADE8 4C4F43
ADEB 415445
ADEE AF21
ADF0 ADFC07         221   LINKS8 DATA #LINKS9,7,:PATTERN:,#SPRPAT
ADF3 504154
ADF6 544552
ADF9 4EAF35
ADFC AE0902         222   LINKS9 DATA #LINKSA,8,:DISTANCE:,#DIST
ADFF 444953
AE02 54414E
AE05 4345AF
```

```
AE08 45
AE09 AE1103    223  LINKSA DATA #LINKSB,3,:SAY:,#SAY
AE0C 534159
AE0F B083
AE11 AE1B05    224  LINKSB DATA #LINKSC,5,:SPGET:,#SPGET
AE14 535047
AE17 4554B1
AE1A DC
AE1B C01007    225  LINKSC DATA #ALC,7,:CHARSET:,#CHRSET
AE1E 434841
AE21 525345
AE24 54AFAD
     C010      226  ALC    EQU  >C010
```

```
                    228  ********************************************************************
                    229  *                                                                  *
                    230  *    CALL SPRITE(#SPRITE,CHAR,COLOR,Y,X,(YSPEED,XSPEED),...)       *
                    231  *                                                                  *
                    232  ********************************************************************
                    233  SPRTE
AE27 06AFC7         234          CALL SPNUM1            Check sprite mode and skip "("
AE2A 06AFCD         235          CALL SPNUM2            Get sprite number
                    236  SPRT3
AE2D 06B00A         237          CALL SPCHR             Put character number for sprite
AE30 0F7E           238          XML  SPEED
AE32 00             239          DATA SYNCHK
AE33 B3             240          DATA COMMA$            Check for comma and skip it
AE34 06A01E         241          CALL SPCOL             Put sprite color in SAL
AE37 0F7E           242          XML  SPEED             Insure at a comma
AE39 00             243          DATA SYNCHK
AE3A B3             244          DATA COMMA$
AE3B 06AFED         245          CALL SPLOC             Put location of sprite in SAL
AE3E BDB008         246          DST  @SP04+1,RAM(@SPSAL) Put in location of sprite
AE41 05
                    247  *
                    248  *       Finish defining SAL.  Check if velocity is specified
                    249  *
                    250  SPRT4
AE42 D642B3         251          $IF @CHAT .NE. COMMA$ GOTO LNKRTN  Finished!!!
AE45 5068
AE47 0F79           252          XML  PGMCHR            Skip ","                          1
AE49 D642FD         253          $IF @CHAT .EQ. NUMBE$ THEN  Next sprite specified        1
AE4C 4E53
AE4E 06AFD4         254            CALL SPNUM3          Get the next sprite number        1
AE51 4E2D           255            BR   SPRT3           And go!                           1
                    256          $END IF
AE53 06B01C         257          CALL SPMOVE            Get the velocity first
AE56 4E42           258          BR   SPRT4             And check for more again
```

```
                    260  ********************************************************************
                    261  *
                    262  *       CALL DELSPRITE(#SPR,.....) or (ALL)
                    263  *
                    264  ********************************************************************
                    265  SPRDEL
AE58 06AFC7         266          CALL SPNUM1                Insure at '('
                    267  SPDEL1
AE5B 0F79           268          XML  PGMCHR                Skip "(" or ","
AE5D D642FD         269          $IF @CHAT .EQ. NUMBE$ THEN  If sprite number
AE60 4E7A
AE62 0F79           270            XML   PGMCHR             Skip "#"
AE64 0F74           271            XML   PARSE              Parse the sprite number
AE66 B6             272            DATA  RPAR$
AE67 06AFD9         273            CALL SPNUM4              Check and convert number
AE6A B7E480         274            DCLR RAM(>480(SPSAL))    Stop motion if moving
AE6D 08
AE6E BFB008         275            DST  >C000,RAM(@SPSAL) Hide the sprite off screen
AE71 C000
AE73 D642B3         276            $IF @CHAT .EQ. COMMA$ GOTO SPDEL1 If more sprites
AE76 6E5B
AE78 4E61           277          $SELSE
AE7A 0F7E           278            XML  SPEED               Must have 'ALL' else error
AE7C 00             279            DATA SYNCHK
AE7D EC             280            DATA ALL$
AE7E 066016         281            CALL SPRINT              Reinitialize all sprites
                    282          $END IF
AE81 5068           283          BR   LNKRTN                Return to caller
```

```
                    285  ********************************************************************
                    286  *
                    287  *          CALL POSITION(#SPR,Y,X,.....)
                    288  *
                    289  ********************************************************************
                    290  SPRPOS
AEB3 06AFC7         291          CALL SPNUM1                Check for sprites and skip"("
                    292  SPRPO2
AEB6 06AFCD         293          CALL SPNUM2                Check sprite number
AEB9 06B06B         294          CALL PREPN                 Prepare Y-pos return variable
AEBC 0F7E           295          XML  SPEED                 Insure at a comma
AEBE 00             296          DATA SYNCHK
AEBF B3             297          DATA COMMA$
AEC0 BD00B0         298          DST  RAM(@SPSAL),@SP00 Read X,Y position
AEC3 08
AEC4 BC4B00         299          ST   @SP00,@FAC1          Get Y position
AEC7 D64BFE         300          $IF @FAC1 .EQ. >FE THEN
AECA 4EA0
AECC 954A           301            DINCT @FAC               Get 256 as an output
AECE 4EA2           302          $SELSE
AED0 944B           303            INCT @FAC1               Regular adjustment for user
                    304          $END IF
AED2 06AEB7         305          CALL SPRP03                Check, convert & assign value
AED5 06B06B         306          CALL PREPN                 Prepare X-pos return variable
AED8 BC4B01         307          ST   @SP00+1,@FAC1         Get X position
AEDB 914A           308          DINC  @FAC                 Adjust for the user
AEDD 06AEB7         309          CALL SPRP03                Check, convert & assign value
AEE0 D642B3         310          $IF @CHAT .EQ. COMMA$ GOTO SPRP02 If not finished...
AEE3 6E86
AEE5 5068           311          BR   LNKRTN                Return
                    312
                    313
AEE7 0F80           314  SPRP03 XML  CIF                   Convert integer to float
AEE9 D700C0         315          $IF @SP00 .DEQ. >C000 THEN  If hidden sprite
AEEC 004EC1
AEEF 874A           316            DCLR  @FAC               Return value zero
                    317          $END IF
AEF1 0F7C           318          XML  ASSGNV                Assign to variable
AEF3 00             319          RTN
```

```
                    321  *
                    322  ***************************************************************
                    323  *
                    324  *       CALL COINC(#SPR,#SPR,TORELANCE,CODE)
                    325  *                 or(#SPR,YLOC,XLOC,TORELANCE,CODE)
                    326  *                 or(ALL)
                    327  *
                    328  ***************************************************************
                    329  SPRCOI
AEC4 06AFC7         330           CALL SPNUM1
AEC7 0F79           331           XML  PGMCHR                  Skip "("
AEC9 D642EC         332           $IF @CHAT .EQ. ALL$ THEN     Check coinc of all sprites
AECC 4EDA
AECE 0F79           333             XML  PGMCHR                Skip 'ALL'
AED0 06B05B         334             CALL COMMA2                Check and skip ","
AED3 DA7B20         335             $IF .BIT5 @VDPSTT .EQ. 0 GOTO NULRTN Check VDP sta
AED6 7063
AED8 4EF0           336           $SELSE
AEDA 06AF6D         337             CALL CODIST                Get distance of 2 sprites
AEDD 06B05B         338             CALL COMMA                 Get tolerance level
AEE0 0F7E           339             XML  SPEED
AEE2 02             340             DATA RANGE                 Check against range
AEE3 0000FF         341             DATA 0,#255                FAC has torelance level
AEE6 C5004A         342             $IF @SP00 .DH. @FAC GOTO NULRTN .Y-loc out of rang.
AEE9 7063
AEEB C5044A         343             $IF @SP04 .DH. @FAC GOTO NULRTN  X-loc out of rang
AEEE 7063
                    344  *              If no coincidenc just return zero.
                    345           $END IF
AEF0 06B06B         346           CALL PREPN                 Prepare for numeric outp
AEF3 BF4ABF         347           DST  >BFFF,@FAC            Store -1 in FAC
AEF6 FF
AEF7 5066           348           BR   ASSRTN
```

```
                    350  ************************************************************
                    351  *                                                          *
                    352  *       CALL MAGNIFY(magnification factor=1 - 4)           *
                    353  *                                                          *
                    354  ************************************************************
                    355  SPRMAG
AEF9 06AFC7         356          CALL SPNUM1              Insure at '('
AEFC 0F79           357          XML  PGMCHR              Skip the '('
AEFE 0F74           358          XML  PARSE               Parse the magnification factor
AF00 B6             359          DATA RPAR$
AF01 0F7E           360          XML  SPEED
AF03 02             361          DATA RANGE
AF04 010004         362          DATA 1,#4                User input 1 through 4
                    363  *       Next statement adding >DF to subtract 1 from FAC
AF07 A248DF         364          ADD >DF,@FAC+1           Turn on screen and interrupt
AF0A 3D0001         365          MOVE 1 FROM @FAC+1 TO VDP(1) Store it to VDP reg
AF0D 0148
AF0F 5068           366          BR   LNKRTN
```

```
                    368  ********************************************************************
                    369  *
                    370  *          CALL MOTION(#SPR,YSPEED,XSPEED,...)
                    371  *
                    372  ********************************************************************
                    373  SPRMOV
AF11 06AFC7         374          CALL SPNUM1              Insure at '('
                    375  SPRMV2
AF14 06AFCD         376          CALL SPNUM2              Get sprite number
AF17 06B01C         377          CALL SPMOVE              Store velocity
AF1A D642B3         378          $IF @CHAT .EQ. COMMA$ GOTO SPRMV2 Loop if more
AF1D 6F14
AF1F 506B           379          BR   LNKRTN
```

```
                    381  ********************************************************************
                    382  *                                                                  *
                    383  *       CALL LOCATE(#SPR,YLOC,XLOC,....)                           *
                    384  *                                                                  *
                    385  ********************************************************************
                    386  SPRLOC
AF21 06AFC7         387          CALL SPNUM1                Insure at '('
                    388  SPRLC2
AF24 06AFCD         389          CALL SPNUM2                Check sprite number
AF27 06AFED         390          CALL SPLOC                 Read location
AF2A BDB008         391          DST  @SP04+1,RAM(@SPSAL)   Put in sprite location
AF2D 05
AF2E D642B3         392          $IF @CHAT .EQ. COMMA$ GOTO SPRLC2 Loop if more
AF31 6F24
AF33 5068           393          BR   LNKRTN
```

```
                395  ********************************************************************
                396  *
                397  *       CALL PATTERN(#SPR,CHAR,.....)
                398  *
                399  ********************************************************************
                400  SPRPAT
AF35 06AFC7     401          CALL SPNUM1              Insure at '('
                402  SPRPT2
AF38 06AFCD     403          CALL SPNUM2              Get sprite number
AF3B 06B00A     404          CALL SPCHR               Set the sprite character
AF3E D642B3     405          $IF @CHAT .EQ. COMMA$ GOTO SPRPT2 Loop if more
AF41 6F38
AF43 5068       406          BR   LNKRTN
```

```
                    408  ********************************************************************
                    409  *                                                                  *
                    410  *         CALL DISTANCE(#1,#2,DISTANCE)                            *
                    411  *                      or (#1,Y,X,DISTANCE)                        *
                    412  *                                                                  *
                    413  ********************************************************************
                    414  DIST
AF45 064FC7         415          CALL SPNUM1              Insure at '('
AF48 0F79           416          XML  PGMCHR              Skip '('
AF4A 06AF6D         417          CALL CODIST              Get distance in Y and X
AF4D 06B06B         418          CALL PREPN               Prepare return variable
AF50 A90000         419          DMUL @SP00,@SP00         X=X*X
AF53 A90404         420          DMUL @SP04,@SP04         Y=Y*Y
AF56 A10206         421          DADD @SP06,@SP02         @SP02=X*X+Y*Y
AF59 0D             422          OVF                      Checking overflow
AF5A 6F65           423          BS   OVER                If overflow-indicate max
AF5C BD4A02         424          DST  @SP02,@FAC          Put distance squared in FAC
AF5F C7027F         425          $IF @SP02 .DH. >7FFF THEN  If bigger than 128
AF62 FF4F69
AF65 BF4A7F         426  OVER      DST  >7FFF,@FAC        Put maximum value
AF68 FF
                    427          $END IF
AF69 0F80           428          XML  CIF                 Convert to floating format
AF6B 5066           429          BR   ASSRTN              Assign value and return
```

```
                    431  ********************************************************
                    432  *   CODIST routine gets locations of two sprites or one  *
                    433  *      sprite and y and x position specified by a user   *
                    434  *      and calculates absolute value of Y adn X distance *
                    435  ********************************************************
                    436  CODIST
AF6D B600           437          CLR  @SP00
AF6F 350007         438          MOVE 7 FROM @SP00 TO @SP00+1 Clear up first 8 bytes
AF72 C100
AF74 D642FD         439          $IF @CHAT .NE. NUMBE$ GOTO ERRSYN   Check for #
AF77 4D53
AF79 06AFD4         440          CALL SPNUM3              Get the first sprite
AF7C BD01B0         441          DST  RAM(@SPSAL),@SP00+1 Location of first sprite
AF7F 0B
AF80 9001           442          INC  @SP00+1             Increment to make range 1-256
AF82 BC0302         443          ST   @SP02,@SP02+1       Put X in SP02+1
AF85 B602           444          CLR  @SP02               Y in SP00+1
AF87 D642FD         445          $IF @CHAT .EQ. NUMBE$ THEN  Get 2nd sprite
AF8A 4F95
AF8C 06AFD4         446            CALL SPNUM3            Get the next sprite
AF8F BD05B0         447            DST  RAM(@SPSAL),@SP04+1 Location of second sprite
AF92 0B
AF93 4F9B           448          $SELSE
AF95 06AFED         449            CALL SPLOC             Get Y and X location
AF98 06B05B         450            CALL COMMA2            Check for comma and skip
                    451          $END IF
AF9B 9005           452          INC  @SP04+1             Increment to make range 1-256
AF9D A50004         453          DSUB @SP04,@SP00         Difference in Y at SP00
AFA0 B100           454          DABS @SP00               Get absolute value
AFA2 B605           455          CLR  @SP04+1             Clear byte before X
AFA4 A50502         456          DSUB @SP02,@SP04+1       Difference in Y at SP04
AFA7 B105           457          DABS @SP04+1             Get the absolute value
AFA9 BC0506         458          ST   @SP06,@SP04+1       Put in the right place
AFAC 00             459          RTN
```

```
                    461  ********************************************************
                    462  *       CHRSET restores the standard character set and the
                    463  *       standard colors for the standard character set
                    464  *       (black on transparent)
                    465  ********************************************************
AF8E 06A478         466  CHRSET CALL CHKEND             Must be at EOS now
AF91 4059           467         BR   ERRSYN             Else its an error
AF93 BF4A04         468         DST  >400,@FAC          Want to load chars at >400
AF96 00
AF97 06A018         469         CALL CHAR2$             Call monitor routine to load
AF9A BEA80F         470         ST   >10,RAM(>80F)      Set 1st set to black on trans
AF9D 10
AF9E 350010         471         MOVE 16 FROM RAM(>80F) TO RAM(>810) Ripple for rest
AFA1 A810A8
AFA4 0F
AFA5 06C012         472         CALL RPL                Return to the caller
```

```
                    474  *       ***************
                    475  *****   SPNUM1 ROUTINE  *****
                    476  *       ***************
                    477  SPNUM1 -
AFC7 D64287         478          $IF @CHAT .NE. LPAR$ GOTO ERRSYN  Should be "(".
AFCA 4D53
AFCC 00             479          RTN
                    480
                    481
                    482  *        ***************
                    483  ******   SPNUM2 ROUTINE  *****
                    484  *        ***************
                    485  SPNUM2
AFCD 0F79           486          XML  PGMCHR              Get the next character.
AFCF D642FD         487          $IF @CHAT .NE. NUMBE$ GOTO ERRSYN  Must be "#".
AFD2 4D53
                    488  SPNUM3
AFD4 0F79           489          XML  PGMCHR              Get next character.
AFD6 063058         490          CALL COMMA               Parse up to comma and skip it.
                    491  *
AFD9 0F7E           492  SPNUM4  XML  SPEED
AFDB 02             493          DATA RANGE               Verify the value is in range.
AFDC 010010         494          DATA 1,#28               Sprite number 1 - 28.
AFDF 9E48           495          DEC  @FAC+1              Adjust for internal use.
AFE1 E24A00         496          DSLL @FAC,2              Get loc of SAL.
AFE4 02
AFE5 A24A03         497          DADD >300,@FAC           sprite # * 4 + >300
AFE8 00
AFE9 BD084A         498          DST  @FAC,@SPSAL         Save SAL location.
AFEC 00             499          RTN
                    500
                    501
                    502
                    503  *       *************
                    504  *****   SPLOC ROUTINE  *****
                    505  *       *************
                    506  SPLOC
AFED 063058         507          CALL  COMMA       Parse up to comma and skip it.
AFF0 0F7E           508          XML  SPEED
AFF2 02             509          DATA RANGE               Range of y: 1-256
AFF3 010100         510          DATA 1,#256
AFF6 9E48           511          DECT @FAC+1              Adjust for internal use: FF -
AFF8 BD044A         512          DST  @FAC,@SP04          Store in SP04 area.
AFFB 0F74           513          XML  PARSE
AFFD B6             514          DATA RPAR$       Parse to ")" or less.
AFFE 0F7E           515          XML  SPEED
B000 02             516          DATA RANGE       Get x value. Range: 1 - 256.
B001 010100         517          DATA 1,#256
B004 9248           518          DEC  @FAC+1      Adjust for internal use: 0 - 255
B006 BC064B         519          ST   @FAC+1,@SP06     SP04+1=y-loc and SP06=x-loc.
B009 00             520          RTN
                    521
                    522
                    523  *       ***************
                    524  *****   SPCHR  ROUTINE  *****
```

```
                    525  *       ***************
                    526  SPCHR
B00A 0F74           527          XML   PARSE
B00C 86             528          DATA  RPAR$
B00D 0F7E           529          XML   SPEED
B00F 02             530          DATA  RANGE         Check upper range.
B010 2000BF         531          DATA  32,#143       Character value 32 - 144.
B013 A24860         532          ADD   >60,@FAC+1    Add offset to char number.
B016 BCE002         533          ST    @FAC+1,RAM(2(SPSAL))  Store the character value
B019 084B
B01B 00             534          RTN
                    535
                    536
                    537  *       ***************
                    538  *****   SPMOVE ROUTINE   *****
                    539  *       ***************
                    540  SPMOVE
B01C 06B058         541          CALL  COMMA         Parse up to comma and skip.
B01F 06B034         542          CALL  RANGEV        Check if numeric and convert to int
B022 BC0A48         543          ST    @FAC+1,@SPTMP     Store y velocity.
B025 0F74           544          XML   PARSE         Get x velocity.
B027 86             545          DATA  RPAR$         Check for ")" or less.
B028 06B034         546          CALL  RANGEV        Numeric check and conter to integer
B02B BC4A0A         547          ST    @SPTMP,@FAC @FAC=y velocity, @FAC+1=x velocity.
B02E BDE480         548          DST   @FAC,RAM(>480(SPSAL)) Store velocities in SAL.
B031 084A
B033 00             549          RTN
                    550
                    551
                    552
                    553  RANGEV
B034 C64C63         554          $IF @FAC2 .H. >63 GOTO ERRSNM     The same as FTINT
B037 6D57
B039 8654           555          CLR   @FAC10
B03B 876C           556          DCLR  @FPERAD
B03D 0F12           557          XML   FLTINT
B03F 8E544D         558          $IF @FAC10 .NE. 0 GOTO ERRBV
B042 87
B043 D34A00         559          $IF @FAC .DGE. 0 THEN               If positive number,
B046 005051
B049 C74A00         560            $IF @FAC .DH. >7F GOTO ERRBV      should be 0 - 127.
B04C 7F6D87
B04F 5057           561          $SELSE                              If negative number,
B051 CB4AFF         562            $IF @FAC .DL. >FF80 GOTO ERRBV Should be -1 to -12
B054 804D87
                    563          $END IF
B057 00             564          RTN                       Otherwise its o.k.
                    565
                    566
                    567  *       ***************
                    568  *****   COMMA ROUTINE   *****
                    569  *       ***************
                    570  COMMA
B058 0F74           571          XML   PARSE
B05A B3             572          DATA  COMMA$
```

```
005B 0A42B3   573 COMMA2 $IF @CHAT .NE. COMMA$ GOTO ERRSYN
005E 4063
0060 0F79     574        XML  PGMCHR      Get next character.
0062 0C       575        RTN
              576
              577
              578 *      ******************
              579 *****  LINK BACK TO BASIC  *****
              580 *      ******************
0063 06B06B   581 NULRTN CALL PREPN
0066 0F7C     582 ASSRTN XML  ASSGNV
0068 05A01C   583 LNKRTN B    RETURN
              584
              585
              586 *      ******************************
              587 *****  PREPARE FOR PASSING ARGUMENT  *****
              588 *      ******************************
006B 0A4280   589 PREPN  $IF @CHAT .HE. >80 GOTO ERRSYN Do not allow token.
006E 6D53
0070 0F7A     590        XML  SYM                 Pick up name & search table
0072 0F78     591        XML  SMB                 Evaluate any subscripts
0074 0A4063   592        $IF @FAC2 .H. >63 GOTO ERRIAL If not numeric-error
0077 6D8B
0079 0F77     593        XML  VPUSH               Save entry on stack
007B 864A     594        CLR  @FAC                Clear FAC for new value
007D 350007   595        MOVE 7 FROM @FAC TO @FAC1
0080 4B4A
0082 00       596        RTN
```

```
                    598  ********************************************************************
                    599  *
                    600  *                SSSS      A     Y   Y
                    601  *               S         A A     Y Y
                    602  *                SSSS    AAAAA     Y
                    603  *                    S   A   A     Y
                    604  *                SSSS    A   A     Y
                    605  *
                    606  *  CALL SAY(........)
                    607  *       Decode given parameter(s).  Store all data first,
                    608  *       then go speak it all at once.
                    609  ********************************************************************
                    610  SAY
B0B3 D642B7         611          $IF  @CHAT .NE. LPAR$ GOTO ERRSYN Must start w/ '('
B0B6 4D53
B0B8 BD4C6E         612          DST  @VSPTR,@FAC2       Save current top of stack on
B0BB 0F77           613          XML  VPUSH               the stack
B0BD BF0C00         614          DST  255,@BYTE          255 bytes = 85  3-byte entries
B0C0 FF
B0C1 0F71           615          XML  GETSTR             Get temp speech list string
B0C3 BF4A00         616          DST  SREF,@FAC          Indicate it is temp string
B0C6 1C
B0C7 BF4C65         617          DST  >6500,@FAC2        Indicate it is string entry
B0CA 00
B0CB BD4E1C         618          DST  @SREF,@FAC4        Save pointer to temp string
B0CE BD500C         619          DST  @BYTE,@FAC6        Length is 255
B0A1 0F77           620          XML  VPUSH              Make it semi-permenant
                    621  * Set up pointers into the speak list
B0A3 BD004E         622          DST  @FAC4,@PTFBSL      Front points to beginning
B0A6 BD024E         623          DST  @FAC4,@PTLBSL      Last now points to beginning
B0A9 BD0400         624          DST  @PTFBSL,@PTEBSL
B0AC A10450         625          DADD @FAC6,@PTEBSL      End points to the end+1
B0AF 06B4D2         626          CALL SETRW              Set PHROM read/write addrs
B0B2 06B4C7         627          CALL WAIT               Wait till no one is speaking
                    628  DIRSPK
B0B5 06B29F         629          CALL GETPRM             Get next parameter
B0B8 712C           630          BS   NEXT1              If non_null ascii string
B0BA BD064E         631            DST  @FAC4,@PTFCIS    Set up pointer to first char
B0BD BD0A50         632            DST  @FAC6,@PTLCIS    Set ptr-to-last-char-in-strin
B0C0 A10A06         633            DADD @PTFCIS,@PTLCIS    by adding length-of-string
B0C3 930A           634            DDEC @PTLCIS            and subtracting 1
                    635  * Make a speech list
B0C5 06B4D2         636            CALL SETRW              Set speech read/write addrs
B0C8 BD0806         637            DST  @PTFCIS,@PTCCIS    Start at beginning of strin
B0CB 864C           638            CLR  @TOTTIM            Clear Total time delay
B0CD 06B31C         639            CALL GETTIM             Get first timing mark
B0D0 06B30C         640            CALL TIMING             Get any subsequent marks
                    641  * The total first time delay is in TOTTIM now
B0D3 C5080A         642            $WHILE @PTCCIS .NOT. .DH. @PTLCIS While more strin
B0D6 7122
B0D8 06B2B5         643              CALL PHRASE           Get next phrase
                    644  * If spell flag is 0, try to look the phrase up.  If it
                    645  * cann't be found, then set the spell flag, and it will
                    646  * be spelled out.  If found, save on speak list.
B0DB 8E4B50         647              $IF @SPLFLG .EQ. 0 THEN   There is a phrase
```

```
B0DE EE
B0DF 06B3B1     648                 CALL LOOKUP       Try to look it up in the PHROS
B0E2 BF4D50     649                 $IF @DATAAD .DEQ. 0 THEN  If not found the
B0E5 EB
B0E6 BE4901     650                   ST 1,@SPLFLG              Set the spell flag
B0E9 10EE       651                 $ELSE                     else ( found )
B0EB 0684B7     652                   CALL STDATA               Store data in list
                653                 $END
                654               $END
                655   * If spell flag is 1, set time delay to >3C, and take the
                656   * phrase one character at a time( spell it ).  Look up each
                657   * character: if not found, use 'UHOH' data instead.
                658   * Regardless, store data on speak list.
B0EE D64B01     659               $IF @SPLFLG .EQ. 1 THEN   Need to spell it out?
B0F1 511B
B0F3 BD4F10     660                 DST @PTLCIP,@PTLCIL  Set last char to spell ou
B0F6 BE4C3C     661                 ST >3C,@TOTTIM   >3C used because sounds good
                662                 $REPEAT    Take each single character
                663   * Skip over any embedded spaces encountered in a phrase
B0F9 06B00C     664                   $WHILE RAM(@PTFCIP) .EQ. SPACE
B0FC 2091C3
B0FF 910C       665                     DINC @PTFCIP
B101 50F9       666                   $SEND
                667   * Set first and last pointers to same one character
B103 BD100C     668                   DST @PTFCIP,@PTLCIP
B106 06B3B1     669                   CALL LOOKUP  try to look it up
                670   * If not found, use data to 'UHOH'
B109 BF4D51     671                   $IF @DATAAD .DEQ. 0 THEN
B10C 11
B10D BF4D71     672                     DST >71F4,@DATAAD  Put addr of 'UHOH'
B110 F4
                673                   $END
B111 06B4B7     674                   CALL STDATA    Store data on speak list
B114 910C       675                   DINC @PTFCIP   Go on to next character
B116 C50C4F     676                 $UNTIL @PTFCIP .DH. @PTLCIL  Until done all
B119 50F9
                677               $END
                678   * At this point, get next timing group.  The first timing
                679   * character has already been found, and it's value is still
                680   * in TIMLEN.  Therefore, initiatory call to GETTIM not
                681   * needed.  Simply clear TOTTIM and call TIMING.
B11B 864C       682               CLR @TOTTIM
B11D 06B30C     683               CALL TIMING
B120 50D3       684             $SEND
                685   * At this point, finished all the phrases in this string.
                686   * TOTTIM should equal >FE to indicate end of string.  If i
                687   * doesn't equal >FE, it indicates that a timing group was
                688   * put on the end of the string.  Therefore, save the timin
                689   * group with a null data address to show it is only timing
B122 D64CFE     690             $IF @TOTTIM .NE. >FE THEN
B125 712C
B127 874D       691               DCLR @DATAAD
B129 06B4B7     692               CALL STDATA
                693             $END
                694   NEXT1
```

```
                    695  * Next item could be direct string.
B12C D642B3         696          $IF @CHAT .EQ. COMMA$ THEN  If direct string presenti
B12F 0146
B131 D83F9F         697            CALL GETPRM               Get the next parameter      1
B134 7191           698            EB   NEXT2                If non-null direct string   1
B136 3E40FF         699              ST   >FF,@TOTTIM        Mark TOTTIM as direct string 1
B139 0F77           700              XML  VPUSH              Save direct string on stack  1
B13B BD4D6E         701              DST  @VSPTR,@DATAAD Store stack addr of string       1
B13E C6B4B7         702              CALL STDATA             And add to the speak list   1
                    703  NEXT2
                    704  * If the next char is a comma, loop thru it again
B141 D642B3         705            $IF @CHAT .EQ. COMMA$ GOTO DIRSPK                    1
B144 70B5
                    706          $END if and fall into SPEAK
                    707  ********************************************************
                    708  * SPEAK will actually speak the speech list.  It tests the *
                    709  * timing byte to see if it is an >FF.  If it is, then the  *
                    710  * data following it points to a direct speech data string  *
                    711  * in VDP.  If it is not, then the data following it points *
                    712  * to a PHROM speech data list.  In the first case, this    *
                    713  * routine will issue a speak external command to the PHROM *
                    714  * and then feed bytes out to the PHROM as it requests them.*
                    715  * In the second case, the address will be loaded out to    *
                    716  * the PHROM, and then a speak command will be issued.      *
                    717  ********************************************************
                    718  SPEAK
B146 06B4D2         719    CALL SETRW                              Set read/write addrs
B149 C9C002         720    $WHILE @PTFBSL .NOT. .DHE. @PTLBSL More speech list to go
B14C 71D3
B14E 06B4C7         721      CALL WAIT     Yes, wait until previous speech is through
B151 D6B000         722      $IF RAM(@PTFBSL) .NE. >FF THEN External speech data?
B154 FF7179
B157 BC79B0         723        ST RAM(@PTFBSL),@TIMER        No, load timer
B15A 00
B15B 8279           724        NEG @TIMER                     and neg it to correct
B15D BD12E0         725        DST RAM(1(PTFBSL)),@PTFBPH   Put addr into PTFBPH
B160 0100
B162 A30000         726        DADD 3,@PTFBSL                 and skip to nxt node
B165 03
B166 D27900         727  LOOP1 $IF @TIMER .LT. 0 GOTO LOOP1 Wait for time delay
B169 5166
B16B 8E1271         728        $IF @PTFBPH .NE. 0 THEN       If there is data
B16E 77
B16F 06B46D         729          CALL LOADAD                  Load the addr to PHROM
B172 BEC000         730          ST >50,@0(WRITE)             and issue speak command
B175 5A50
                    731        $END
B177 51D0           732      $SELSE
B179 9100           733        DINC @PTFBSL  Speak external, skip over >FF
B17B BD5EB0         734        DST RAM(@PTFBSL),@PTCBED  Set up pointer to 1st byte
B17E 00
B17F BD5EE0         735        DST RAM(4(PTCBED)),@PTCBED  in ext speech data
B182 045E
B184 9500           736        DINCT @PTFBSL                Skip addr bytes
B186 BC62EF         737        ST RAM(-1(PTCBED)),@LENWST Get Len of whole string
```

```
B199 FFFF5E
                738  DIRSPH
B19C A6A203     739        SUB 3,@LENWST                 minus 3 bytes overhead
                740  * All external speech strings start with a >60
B19F D1B05E     741        $IF RAM(@PTCBED) .NE. >60 GOTO ERRBV Bad speech strin
B1A2 5C4287
B1A5 D1B40F     742        CALL WAIT                     Wait for go ahead
B1A8 955E       743        DINCT @PTCBED                 Skip spk ext & 1st byte len
B1AA BC60B0     744        ST RAM(@PTCBED),@LENCST       Get len of curr string
B1AD 5E
B1AE 915E       745        DINC  @PTCBED              Skip len byte to 1st real byt
B1A0 BE5610     746        ST 16,@TEMP2                  Do 1st 16 bytes (fill buff)
B1A3 BEC000     747        ST >60,@0(WRITE)              Start Speak External
B1A6 5A60
B1A8 BCC000     748  LOOPR   ST RAM(@PTCBED),@0(WRITE) Write byte to PHROM
B1AB 5A805E
B1AE 915E       749          DINC @PTCBED              Go to next byte
B1B0 9262       750          DEC  @LENWST              1 less char in whole string
B1B2 7100       751          BS   CONTIN               Finished whole string?
B1B4 9260       752          DEC  @LENCST              1 less char in curr string
B1B6 7130       753          BS   DIRSPH               Finished current string?
B1B8 9256       754          DEC  @TEMP2               1 less char in this loop
B1BA 51A8       755          BR   LOOPR                Not finished curr loop yet?
                756        $REPEAT                     Finished, wait till ready
B1BC BC69C0     757          ST @0(READ),@SPKSTS       Read status from PHROM
B1BF 005B
                758  * If the next statement is true, it means that speak was
                759  * probably interupted and that it is shot at this point.
                760  * Therefore, we are going to quit now.
B1C1 DA6980     761          $IF .BIT7 @SPKSTS .EQ. 0 GOTO CONTIN
B1C4 71D0
B1C6 DA6940     762        $UNTIL .BIT6 @SPKSTS .EQ. 1 Loop till buff below half
B1C9 71BC
B1CB BE5608     763        ST 8,@TEMP2                 Put 8 more bytes to PHROM
B1CE 51A8       764        BR LOOPR                    and go do these
                765      $END
                766  CONTIN
B1D0 05B149     767    $END                            We've said it all!
                768  * Now pop all entries off stack that we put on!
                769         $REPEAT
B1D3 0F78       770           XML  VPOP                Free up a temporary string
B1D5 D56E4C     771         $UNTIL @VSPTR .DEQ. @FAC2
B1D8 51D3
B1DA 5068       772         BR   LNKRTN                And return to the caller
```

```
                    774  ********************************************************************
                    775  *
                    776  *          SSSS   PPPP     GGGG    EEEEE   TTTTT
                    777  *         S       P   P   G    G   E         T
                    778  *          SSSS   PPPP    G        EEEE      T
                    779  *              S  P       G   GG   E         T
                    780  *          SSSS   P        GGGG    EEEEE     T
                    781  *
                    782  *       SPGET subprogram.  Load speech data from external
                    783  *       device.  Use standard file I/O.
                    784  *
                    785  ********************************************************************
                    786
B1DC D642B7         787  SPGET  $IF  @CHAT .NE. LPAR$ GOTO ERRSYN Must have left par
B1DF 4B53
B1E1 06B4D2         788         CALL SETRW                 Set PHROM read/write addr
B1E4 06B4C7         789         CALL WAIT                  Wait till no one is speaking
                    790  NXTPAR
B1E7 06B29F         791         CALL GETPRM                Get the next parameter
B1EA BF3072         792         $IF @FAC6 .DNE. 0 THEN If non-null ASCII string
B1ED F8
B1EE BD064E         793           DST  @FAC4,@PTFCIS   Pointer to 1st char in string
B1F1 BD0A50         794           DST  @FAC6,@PTLCIS   Pointer-to-last-char-in-strin
B1F4 A10A06         795           DADD @PTFCIS,@PTLCIS    by adding length-of-string
B1F7 930A           796           DDEC @PTLCIS             and subtracting 1
B1F9 06B4D2         797           CALL SETRW           Set the speech read/write add
B1FC BD0806         798           DST  @PTFCIS,@PTCCIS Set curr char to first char
B1FF 864C           799           CLR  @TOTTIM           Clear total time delay
B201 06B31C         800           CALL GETTIM            Get first timing mark
B204 06B30C         801           CALL TIMING            Get any subswquent marks
                    802  * Get one phrase, and look it up.  If the phrase is not
                    803  * found, substitute in 'UHOH'.
B207 C5080A         804           $IF @PTCCIS .NOT. .DH. @PTLCIS THEN Possible phras
B20A 7298
B20C 06B2B5         805             CALL PHRASE                              Yes, go get it
B20F D64B01         806             $IF @SPLFLG .EQ. 1 THEN Spell flag set then set
B212 5217
B214 BD100C         807               DST @PTFCIP,@PTLCIP last ptr to first (1 char)
                    808             $END
B217 06B3B1         809             CALL LOOKUP   Look up the phrase
B21A 8F4D52         810             $IF @DATAAD .DEQ. 0 THEN  If not there,
B21D 25
B21E BF4D71         811               DST >71F4,@DATAAD          use 'UHOH' data addr
B221 F4
B222 BE6451         812               ST >51,@STRLEN                'UHOH' data len
                    813             $END
                    814  * Data must be in PHRADD and PHLEN, so move it.
B225 BD014D         815             DST  @DATAAD,@PHRADD
B228 BC0064         816             ST   @STRLEN,@PHLEN
B22B A20003         817             ADD  3,@PHLEN                        For overhead info
                    818  * There must be a variable to put this data in.  If not, er
B22E 0F7E           819             XML  SPEED
B230 00             820             DATA SYNCHK
B231 B3             821             DATA COMMA$
B232 CA4280         822             $IF @CHAT .HE. >80 GOTO ERRSYN Do not allow toke
```

```
B235 6D53
B237 CF7A      823          XML  SYM              Find symbol in table
B239 CF7B      824          XML  SMB              Evaluate any subscripts
B23B CF77      825          XML  VPUSH            Save for assignment
B23D B60C      826          CLR  @BYTE            Two byte value
B23F BC0000    827          ST   @PHLEN,@BYTE+1 Length of string needed
B242 CF71      828          XML  GETSTR           Get a string for the data
B244 068AD2    829          CALL SETRW            Set up speech read/write addr
B247 BF4A00    830          DST  SREF,@FAC        Now build string FAC entry
B24A 1C
B24B BF4C65    831          DST  >6500,@FAC2      String ID
B24E 00
B24F BD4E1C    832          DST  @SREF,@FAC4      Pointer to string
B252 BD500C    833          DST  @BYTE,@FAC6      Length of string
B255 9FB01C    834          DST  >6000,RAM(@SREF) Mark string as speech dat
B258 6000
B25A BCE002    835          ST   @PHLEN,RAM(2(SREF))  Put in string length
B25D 1000
B25F A7E001    836          DSUB 3,RAM(1(SREF))       minus this info
B262 100003
               837     * LOADAD expects addr to be in PTFBPH, so move it.
B265 BD1201    838               DST  @PHRADD,@PTFBPH
B268 068A6D    839               CALL LOADAD
               840     * Going to copy string from PHROM to VDP.  The actual data
               841     * from PHROM is in bit-reversed order, so must reverse the
               842     * order after reading in the order.  Remember that 3 bytes
               843     * PHLEN are our own overhead, so don't copy all
B26B C60003    844               $WHILE @PHLEN .H. 3
B26E 5296
B270 BEC000    845                  ST >10,@0(WRITE)    Issue read byte command
B273 5A10
B275 BC68C0    846                  ST @0(READ),@BYTE3  Read the byte
B278 0058
               847     * the following code is somewhat tricky.  It will bit
               848     * reverse the contents of BYTE3 into BYTE1 through
               849     * BYTE2 by means of word shifts.  Note the definitions of
               850     * of BYTE1, BYTE2, & BYTE3 in the EQU's.  You might try
               851     * an example if it isn't clear what is going on.
B27A 8667      852                  CLR @BYTE2
B27C BE5408    853                  ST >8,@TEMP1
B27F EB6700    854     RNDAG        DSRC @BYTE2,1
B282 01
B283 E36600    855                  DSLL @BYTE1,1
B286 01
B287 9254      856                  DEC @TEMP1
B289 527F      857                  BR RNDAG
               858     * Store the bit-corrected byte into the string & inc str p
B28B BCE003    859                  ST @BYTE1,RAM(3(SREF))
B28E 1C66
B290 911C      860                  DINC @SREF
B292 9200      861                  DEC @PHLEN       Dec the string len
B294 5268      862                $SEND              Go do next char if there is on
B296 CF7C      863                XML ASSGNV         Assign the string to variable
               864              $END
               865           $END
```

```
B29B D642B3    866          $IF @CHAT .EQ. COMMA$ GOTO NXTPAR  If more go do
B29B 71E7
B29D 5068      867          BR  LNKRTN
```

```
                869  ********************************************************
                870  * GETPRM gets the next string parameter passed to the   *
                871  * routine.  If that parameter is non-exist or null, the *
                872  * condition bit is set.  If the parameter is there then *
                873  * condition bit is reset and the FAC entry describes the*
                874  * string.  In either case, return with condition is done.*
                875  ********************************************************
32FF 0F79       876  GETPRM XML   PGMCHR              Get next token
32A1 D64EB3     877         $IF   @CHAT .EQ. COMMA$ GOTO SETCB Go set cond no parm
32A4 7281
32A6 0F74       878         XML   PARSE              Parse the value
32A8 26         879         DATA  RPAR$
32A9 D64C65     880         $IF   @FAC2 .NE. >65 GOTO ERRSNM If not string-error
32AC 4D57
32AE 8F50       881         DCZ   @FAC6              Set cond if null string
32B0 01         882         RTNC                     Else return
32B1 D40000     883  SETCB  CEQ @0,@0                Set condition bit
32B4 01         884         RTNC
```

```
                    886 ***************************************************************
                    887 * Get the next phrase out of the current string.  The        *
                    888 * phrase may begin with a #, which means it will continue    *
                    889 * to the next #, or it may begin with an ordinary char,      *
                    890 * in which case it will end with the char just before the    *
                    891 * first timing char encountered.  In either case, the end    *
                    892 * of the string will indicate a legal end of phrase if it   *
                    893 * occurs before the usual indicator!                         *
                    894 ***************************************************************
B2B5 D64A23         895 PHRASE $IF @CCHAR .EQ. NUMBER THEN  Phrase start with #?    1
B2B8 52F0
B2BA 9108           896           DINC @PTCCIS                  Yes, inc CC ptr past # 1
B2BC D6B008         897           $WHILE RAM(@PTCCIS) .EQ. SPACE Skip spaces        2
B2BF 2052C6
B2C2 9108           898             DINC @PTCCIS                                    2
B2C4 52BC           899           $SEND                                             1
B2C6 D6B008         900           $IF RAM(@PTCCIS) .EQ. NUMBER THEN  All spaces?    2
B2C9 2352CF
B2CC 9108           901             DINC @PTCCIS        Yes, skip this # too        2
B2CE 00             902             RTN                 And ignore this phrase      2
                    903           $END
B2CF BDCC08         904           DST @PTCCIS,@PTFCIP  Save 1st char in phrase      1
                    905           $REPEAT
B2D2 9108           906             DINC @PTCCIS              Go on to next char    2
                    907 * Got to watch for end of string.  If encountered before a
                    908 * #, act like char after string is #.  Then last char will
                    909 * be char before, or the last char in the string!!
B2D4 C5080A         910             $IF @PTCCIS .DH. @PTLCIS GOTO FNDNUM            2
B2D7 72E2
B2D9 BC4AB0         911             ST RAM(@PTCCIS),@CCHAR  No, get char in CCHAR   2
B2DC 08
B2DD D64A23         912           $UNTIL @CCHAR .EQ. NUMBER  If not # cont looking
B2E0 52D2
                    913
B2E2 BD1008         914 FNDNUM    DST  @PTCCIS,@PTLCIP Last char in phrase is one
B2E5 9310           915           DDEC @PTLCIP          before the #
B2E7 9108           916           DINC @PTCCIS          Point to char after #
B2E9 06B31C         917           CALL GETTIM           Get 1st timing char after phr
B2EC 864B           918           CLR  @SPLFLG          Indicate don't spell
B2EE 5308           919         $SELSE                No # as 1st char in phrase
B2F0 BD0C08         920           DST  @PTCCIS,@PTFCIP Curr char is 1st char phrase
B2F3 864B           921           CLR  @SPLFLG              Assume don't spell
B2F5 CA4A41         922           $IF @CCHAR .L. :A: THEN  If not alphabetic
B2F8 72FC
B2FA 904B           923             INC   @SPLFLG           set spell flag
                    924           $END
                    925 * Need to find end of phrase, which is char before next
                    926 * timing char we find.  Therefore, look for a timing char!
                    927           $REPEAT
B2FC 9108           928             DINC @PTCCIS
B2FE 06B31C         929             CALL GETTIM
B301 D651FF         930           $UNTIL @TIMLEN .NE. >FF PTCC If not timing, loop
B304 72FC
B306 BD1008         931           DST  @PTCCIS,@PTLCIP  Char before curr char is
B309 9310           932           DDEC @PTLCIP           the last char in phrase
```

```
                    933         $END
5308 00             934         RTN
```

```
                    936  ********************************************************
                    937  * TIMING will loop through chars in string until it finds a*
                    938  * non-timing char.  Non-timing chars have TIMLEN values of *
                    939  * >FE OR >FF.  GETTIM must be called before this routine to*
                    940  * establish a correct value of TIMLEN.  Also, most likely  *
                    941  * TOTTIM should have been cleared.                         *
                    942  ********************************************************
B30C CA51FE         943  TIMING $WHILE @TIMLEN .NOT. .HE. >FE                      1
B30F 7313
B311 A14C51         944          DADD @TIMLEN,@TOTTIM                             1
B314 9108           945          DINC @PTCCIS                                     1
B316 06B31C         946          CALL GETTIM                                      1
B319 530C           947        $SEND
B31B 00             948        RTN
                    949
                    950  ********************************************************
                    951  * GETTIM will examine the current char in the string and  *
                    952  * set TIMLEN to the appropriate time delay value.  TIMLEN  *
                    953  * can take on the following values:                        *
                    954  *           >00 if char is timing '+'                      *
                    955  *           >06 if char is timing ' '                      *
                    956  *           >0C if char is timing '-'                      *
                    957  *           >12 if char is        ','                      *
                    958  *           >1E if char is        ';'                      *
                    959  *           >30 if char is        ':'                      *
                    960  *           >3C if char is timing '.'                      *
                    961  *           >FE if char is out of string bounds            *
                    962  *           >FF if char is not timing                      *
                    963  * Note that to test timing, some manipulation of PTCCIS    *
                    964  * would be neccesary, so it is stored and used in TEMP1.   *
                    965  ********************************************************
                    966  GETTIM
B31C BC4AB0         967        ST   RAM(@PTCCIS),@CCHAR   Get the char
B31F 08
B320 BD540B         968        DST  @PTCCIS,@TEMP1        store curr ptr in TEMP1
B323 C5540A         969        $IF @TEMP1 .DH. @PTLCIS THEN  out of string bounds?
B326 532C
B328 BE51FE         970          ST >FE,@TIMLEN           Yes, load value and retur
B32B 00             971          RTN
                    972        $END
B32C C64A3B         973        $IF @CCHAR .NOT. .H. SEMICO THEN  cann't be timing
B32F 7396
B331 D64A20         974          $IF @CCHAR .EQ. SPACE THEN
B334 5345
B336 BE5106         975            ST >06,@TIMLEN
B339 D6E001         976            $WHILE RAM(1(PTCCIS)) .EQ. SPACE  While spaces
B33C 082053
B33F 44
B340 9108           977              DINC @PTCCIS                          Skip them
B342 5339           978            $SEND
B344 00             979            RTN
                    980          $END
B345 D64A2B         981          $IF @CCHAR .EQ. PLUS THEN
B348 5354
B34A 9154           982            DINC @TEMP1          Need to test the next char
```

```
B34C D6B09A   983              CALL NUMERC          Is it numeric
B34F 7396     984              BS NOTIME            Was numeric => not timing
B351 D651     985              CLR @TIMLEN          Not numeric => set as no
B353 00       986              RTN
              987            $END
B354 D64A2C   988            $IF @CCHAR .EQ. COMMAT THEN
B357 5350
B359 BE512    989              ST >12,@TIMLEN
B35C 00       990              RTN
              991            $END
B35D D64A2E   992            $IF @CCHAR .EQ. PERIOD THEN
B360 5374
B362 9354     993              DDEC @TEMP1          Go back to preceding char
B364 06B09A   994              CALL NUMERC          Is it numeric?
B367 5370     995              BR PTIME             No, so it is timing
B369 9554     996              DINCT @TEMP1         Yes, on to following char
B36B 06B09A   997              CALL NUMERC          Is it numeric too?
B36E 7396     998              BS NOTIME            Yes,both numeric => not timing
B370 BE513C   999  PTIME       ST >3C,@TIMLEN       Both not numeric => timing
B373 00       1000             RTN
              1001           $END
B374 D64A2D   1002           $IF @CCHAR .EQ. HYPEN THEN
B377 5364
B379 9154     1003             DINC @TEMP1        Check next char
B37B 06B09A   1004             CALL NUMERC        Is it numeric?
B37E 7396     1005             BS NOTIME          Was numeric => not a timing char
B380 BE510C   1006             ST >0C,@TIMLEN     Was not numeric => set as timing
B383 00       1007             RTN
              1008           $END
B384 D64A3A   1009           $IF @CCHAR .EQ. COLON THEN
B387 538D
B389 BE5130   1010             ST >30,@TIMLEN
B38C 00       1011             RTN
              1012           $END
B38D D64A3B   1013           $IF @CCHAR .EQ. SEMICO THEN
B390 5396
B392 BE511E   1014             ST >1E,@TIMLEN
B395 00       1015             RTN
              1016           $END
              1017         $END
B396 BE51FF   1018  NOTIME ST >FF,@TIMLEN  Set as no timing char present
B399 00       1019         RTN
```

```
                1021  ************************************************************
                1022  *    NUMERC tests the char pointed to by PTCCIS and        *
                1023  *    verifies the following:                               *
                1024  *     1- it is within the current string boundaries        *
                1025  *     2- it is numeric  (ie between '0' and '9')           *
                1026  *    If both of the above conditions are true, COND is     *
                1027  *    set upon return, otherwise COND is reset              *
                1028  ************************************************************
B39A C6540A     1029  NUMERC $IF @TEMP1 .NOT. .DH. @PTLCIS THEN
B39D 73B0
B39F C50654     1030            $IF @PTFCIS .NOT. .DH. @TEMP1 THEN
B3A2 73B0
B3A4 CAB054     1031              $IF RAM(@TEMP1) .HE. ZERO THEN
B3A7 3053B0
B3AA C6B054     1032                $IF RAM(@TEMP1) .NOT. .H. NINE GOTO SETCB
B3AD 3952B1
                1033              $END
                1034            $END
                1035          $END
B3B0 01         1036          RTNC
```

```
                    1038  ********************************************************
                    1039  * LOOKUP is a prolong routine to SEARCH.  In each PHROM  *
                    1040  * there may be 2 trees, one starting at >0000 and the o  r*
                    1041  * at >8000.  Either may or may not be present.  Presences *
                    1042  * is determined if a >AA byte is at the starting location.*
                    1043  * LOOKUP determines if the tree at >0000 is in, and if so, *
                    1044  * calls SEARCH with that addr.  If that tree is not present*
                    1045  * or the phrase couldn't be found in it, LOOKUP then checks*
                    1046  * if the tree at >8000 is present, and again, if so, calls *
                    1047  * SEARCH with that tree address.  If the word was found in *
                    1048  * the first tree, or after searching the second tree, the  *
                    1049  * routine will return.                                     *
                    1050  ********************************************************
B3B1 8766           1051  LOOKUP DCLR  @BYTE1           BYTE1 contains addr of curr tree
                    1052
B3B3 BD1266         1053  TRYAGN DST  @BYTE1,@PTFBPH  Look for >AA tree header
B3B6 06B46D         1054         CALL LOADAD           LOADAD expects addr in PTFBPH
B3B9 BEC000         1055         ST   >10,@0(WRITE)    Put out read byte command
B3BC 5A10
B3BE D60000         1056         $IF @0(READ) .EQ. >AA THEN  Tree out there?
B3C1 5BAA53
B3C4 CE
B3C5 9112           1057           DINC @PTFBPH        Skip the tree header
B3C7 06B3D9         1058           CALL SEARCH         Go search this PHROM tree
B3CA 8F4D53         1059           $IF @DATAAD .DNE. 0 GOTO FOUND Phrase found => exi
B3CD D8
                    1060         $END
B3CE A36680         1061         DADD >8000,@BYTE1     Go to start of next PHROM tree
B3D1 00
                    1062  * Note >8000 + >8000 = >0000 => tried both trees
B3D2 8F6653         1063         $IF @BYTE1 .DNE. 0 GOTO TRYAGN
B3D5 B3
B3D6 874D           1064         DCLR @DATAAD          Didnt find phrase in either tre
B3D8 00             1065  FOUND  RTN
                    1066
                    1067  ********************************************************
                    1068  * SEARCH actually searches the PHROM tree for the phrase.
                    1069  * The PHROM tree organization is as follows:
                    1070  *     (ie. this is one phrase node)
                    1071  *           phrase ascii length      1 byte
                    1072  *           actual ascii chars       n bytes
                    1073  *           less then pointer        2 bytes
                    1074  *           greater then pointer     2 bytes
                    1075  *           speech data pointer      3 bytes
                    1076  *           speech data length       1 byte
                    1077  * The comparision of two words proceeds on a char by char
                    1078  * basis, where length is secondary to char values, ie
                    1079  * more > answer; number < we; eight < eighty; etc.
                    1080  ********************************************************
                    1081  SEARCH
B3D9 06B46D         1082         CALL LOADAD           Set PHROM to start phrase no
B3DC BEC000         1083         ST   >10,@0(WRITE)    Issue read byte command
B3DF 5A10
B3E1 8616           1084         CLR  @PTLCPH          Length of phrase => PTLCPH
B3E3 BC17C0         1085         ST   @0(READ),@PTLCPH+1 ( stored as 2 byte value
```

```
B3E6 0058
B3E8 A11612  1086          DADD @PTFBPH,@PTLCPH   Add front ptr giving end ptr
B3EB BD1412  1087          DST  @PTFBPH,@PTCCPH   Set up curr char as 1 beyond
B3EE 9114    1088          DINC @PTCCPH            length byte
B3F0 B0CE0C  1089          DST  @PTFCIP,@PTCCIP   Reset current ptr into phrase
             1090  * Compare two characters
             1091  NEXT
B3F3 BEC000  1092          ST   >10,@0(WRITE)     Issue read byte command
B3F6 5A10
B3F8 BC5DC0  1093          ST  @0(READ),@PHDATA   Get char in from PHROM
B3FB 0058
B3FD D45DB0  1094          $IF @PHDATA .EQ. RAM(@PTCCIP) THEN  Compare the char1
B400 0E5451
B403 9114    1095            DINC @PTCCPH         Equal, advance both pointers 1
B405 910E    1096            DINC @PTCCIP                                      1
B407 D6B00E  1097            $IF RAM(@PTCCIP) .EQ. SPACE THEN  Skip extra space2
B40A 205421
B40D D6E001  1098              $WHILE RAM(1(PTCCIP)) .EQ. SPACE  While spaces  3
B410 0E2054
B413 18
B414 910E    1099                DINC @PTCCIP                          Skip them  3
B416 540D    1100              $SEND                                               2
             1101  * By skipping extra spaces, might have reached end of
             1102  * phrase.  If this is true, next char in Phrase = #.  If
             1103  * so, advance the ptr to be beyond end of phrase.
B418 D6E001  1104              $IF RAM(1(PTCCIP)) .EQ. NUMBER THEN                3
B41B 0E2354
B41E 21
B41F 910E    1105                DINC @PTCCIP
             1106              $END
             1107            $END
B421 C51416  1108            $IF @PTCCPH .DH. @PTLCPH THEN End of PHROM word?
B424 5446
B426 C50E10  1109              $IF @PTCCIP .DH. @PTLCIP THEN Yes-end of phrase?
B429 5440
B42B BD1216  1110                DST  @PTLCPH,@PTFBPH  Yes, Word found
             1111  * Skip 5 bytes down from last char to data ptr
B42E A31200  1112                DADD >6,@PTFBPH
B431 06
B432 06B49F  1113                CALL READAD           Set data addr => DATAAD
B435 BEC000  1114                ST   >10,@0(WRITE)    Issue read byte command
B438 5A10
B43A BC64C0  1115                ST   @0(READ),@STRLEN Get length of speech dat
B43D 0058
B43F 00      1116                RTN
             1117              $END                    Phrase longer-get GT lir
B440 BF1200  1118                DST  3,@PTFBPH        Move 3 bytes past PTLCPH
B443 03
B444 545D    1119                BR   NXTPHR
             1120            $END
B446 C50E10  1121              $IF @PTCCIP .NOT. .DH. @PTLCIP GOTO NEXT 2 chars
B449 53F3
B44B BF1200  1122              DST 1,@PTFBPH         PHROM phrase longer: use LT p
B44E 01
B44F 545D    1123              BR NXTPHR
```

```
                    1124          $END
                    1125  * two characters compared were not equal
8451 8F1E00         1126            DST  3,@PTFBPH        3 bytes past last to GT
8454 02
8455 C45D80         1127            $IF @PHDATA .H. RAM(@PTCCIP) THEN PHROM after phra
8458 C5545D
845B 5712           1128              DDECT @PTFBPH       Back up 2 bytes to LT link
                    1129            $END
                    1130  NXTPHR
                    1131  * go get next phrase out of the PHROM to compare
845D A11216         1132            DADD @PTLCPH,@PTFBPH Add displacement to last char
8460 06B49F         1133            CALL READAD           and get the new addr
8463 9F4D54         1134            $IF @DATAAD .DEQ. #0 THEN More leaves on this tree
8466 68
8467 00             1135              RTN                 No, return empty handed
                    1136            $END                  Yes, set up to check next wor
8468 8D124D         1137            DST  @DATAAD,@PTFBPH Store new addr in PTFBPH
846B 53D9           1138            BR   SEARCH          Go compare this new word!
                    1139  * The program should never reach this point!!  It should
                    1140  * return somewhere up above.
```

```
                1142  ********************************************************
                1143  * LOADAD will set the addr out in the PHROM to the addr  *
                1144  * found in PTFBPH.  Note that the PHROM is expecting five *
                1145  * nybbles to be written out as the address.              *
                1146  ********************************************************
                1147  LOADAD
B46D BD5412     1148    DST   @PTFBPH,@TEMP1     This is destructive, so copy
B470 BD5612     1149    DST   @PTFBPH,@TEMP2      address into temporary areas
B473 E65404     1150    SRL   @TEMP1,4           Isolate the MSN of the MSB
B476 E65504     1151    SRL   @TEMP1+1,4         Isolate the MSN of the LSB
B479 B3560F     1152    DAND  >0F0F,@TEMP2       Isolate the LSN of both MSB, LSB
B47C 0F
B47D B75440     1153    DOR   >4040,@TEMP1       Include a 4 as MSN of all 4 nybbles
B480 40
B481 B75640     1154    DOR   >4040,@TEMP2        to indicate a Load Address Comman
B484 40
B485 BCC000     1155    ST    @TEMP2+1,@0(WRITE)  Write out the LSN of the LSB
B488 5A57
B48A BCC000     1156    ST    @TEMP1+1,@0(WRITE)  Write out the LSN of the MSB
B48D 5A55
B48F BCC000     1157    ST    @TEMP2,@0(WRITE)    Write out the MSN of the LSB
B492 5A56
B494 BCC000     1158    ST    @TEMP1,@0(WRITE)    Write out the MSN of the MSB
B497 5A54
B499 BEC000     1159    ST    >40,@0(WRITE)       Write out 0 as fifth nybble
B49C 5A40
B49E 00         1160    RTN
                1161
                1162  ********************************************************
                1163  * READAD will read an address from the PHROM and store   *
                1164  * it in DATAAD.  Note that PTFBPH should contain the addr *
                1165  * of the PHROM location to be read so LOADAD will work.  *
                1166  ********************************************************
                1167  READAD
B49F 06B46D     1168    CALL LOADAD              Set the addr of the PHROM
B4A2 BEC000     1169    ST >10,@0(WRITE)         Get high byte of addr
B4A5 5A10
B4A7 BC4DC0     1170    ST @0(READ),@DATAAD      Store it in DATAAD
B4AA 0058
B4AC BEC000     1171    ST >10,@0(WRITE)         Get low byte of addr
B4AF 5A10
B4B1 BC4EC0     1172    ST @0(READ),@DATAAD+1    Store it in DATAAD+1
B4B4 0058
B4B6 00         1173    RTN
```

```
                    1175  ********************************************************************
                    1176  * STDATA will store the data in DATAAD and TOTTIM                   *
                    1177  * onto the speech list.  It will also check that there              *
                    1178  * room on the speech list for this entry, and abort with            *
                    1179  * error if not.                                                     *
                    1180  ********************************************************************
                    1181  STDATA
 34B7 030204        1182    $IF @PTLBSL .DEQ. @PTEBSL GOTO ERRSSL  Is there room?
 34BA 7409
 34BC 350003        1183    MOVE 3 FROM @TOTTIM TO RAM(@PTLBSL)    Put data in list
 34BF 80024C
 34C2 A30200        1184    DADD 3,@PTLBSL                         and inc top of list
 34C5 03
 34C6 00            1185    RTN
                    1186
                    1187  ********************************************************************
                    1188  * WAIT loops until the speech periferal goes idle.                  *
                    1189  ********************************************************************
                    1190  WAIT
                    1191    $REPEAT                  Loop until nobody is talking
 34C7 D06F00        1192      ST @0(READ),@SPKSTS    Read status from PHROM
 34CA 0088
 34CC CA6F80        1193    $UNTIL .BIT7 @SPKSTS .EQ. 0
 34CF 34C7
 34D1 00            1194    RTN
                    1195
                    1196  ********************************************************************
                    1197  * SETRW moves addrs of speech read/write from GROM to VDP           *
                    1198  ********************************************************************
                    1199  SETRW
 34D2 310004        1200    MOVE 4 FROM ROM(>46) TO @READ
 34D5 580046
 34D8 00            1201    RTN
```

```
                1203  ********************************************************
                1204  *                        Error messages                *
                1205  ********************************************************
                1206  *      The following calls are in EXECS file.
                1207  *ERRSYN CALL ERR$$       * SYNTAX ERROR
                1208  *       DATA 3
                1209  *
                1210  *ERRSNM CALL ERR$$       * STRING-NUMBER MISMATCH
                1211  *       DATA 7
                1212  *
                1213  *ERRBV  CALL ERR$$       * BAD VALUE
                1214  *       DATA 30
                1215  *
                1216  *ERRIAL CALL ERR$$       * INCORRECT ARGUMENT LIST
                1217  *       DATA 31
                1218  ********************************************************
3409 066AB4     1219  ERRSSL CALL ERR$$
340C 15         1220         DATA 21          * SPEECH STRING TOO LONG
                1221
                1222         END

ERRORS=     0

LENGTH= 1861 (>0745)


225 SYMBOLS USED
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
ALC	0010	226	225
ALL$	00EC	194	220 332
ASSGNV	0070	58	318 582 863
ASSRTN	B0EE	582	348 429
BASE	0042	129	
BUFLEV	0046	132	
BYTE	0000	97	614 619 826 827 833
BYTE1	0066	165	855 859 1051 1053 1061 1063
BYTE2	0067	166	852 854
BYTE3	0068	167	846
CB	0008	1	
CCHAR	004A	141	895 911 912 922 967 973 974 981 988 992 1002 1009 1013
CHAR	0004	1	
CHAR1	AC90	207	
CHAR2	AD92	208	
CHAR2$	001B	36	469
CHAT	0042	128	251 253 269 276 310 332 378 392 405 439 445 478 487 573 589 611 696 705 787 822 866 877
CHKEND	6A7B	39	466
CHRSET	AFA5	466	225
CIF	0080	62	314 428
CODIST	AF65	436	337 417
COLON	003A	180	1009
COMMA	B050	570	338 490 507 541
COMMA$	00B3	191	240 244 251 276 297 310 378 392 405 572 573 696 705 821 866 877
COMMA2	B053	573	334 450
COMMAT	002C	183	988
CONTIN	B1C8	766	751 761
DATA	0034	121	
DATAAD	004D	146	649 671 672 691 701 810 811 815 1059 1064 1134 1137 1170 1172
DIRSPH	B184	738	753
DIRSPK	B0AD	628	705
DISPLA	0009	1	
DIST	AF3D	414	222
ENLN	0032	120	
ERR$$	6A84	38	1219
ERRBV	AD87	47	558 560 562 741
ERRCOD	0022	112	
ERRIAL	AD8B	48	592
ERRSNM	AD57	46	554 880
ERRSSL	B4D1	1219	1182
ERRSYN	AD53	45	439 467 478 487 573 589 611 787 822
EXTRAM	002E	118	
FAC	004A	134	135 136 137 138 139 301 308 316 342 343 347 364 365 424 426 468 495 496 497 498 511 512 518 519 532 533 543 547 548 559 560 562 594 595 616 830
FAC1	004B	135	299 300 303 307 595
FAC10	0054	139	555 558
FAC2	004C	136	554 592 612 617 771 831 880
FAC4	004E	137	618 622 623 631 793 832

SYMBOL	VALUE	DEF	REFERENCE TABLE
FAC6	006C	138	619 625 632 792 794 803 881
FIGURE	0002	1	
FLAG	0046	131	
FLTINT	0012	50	557
FNDNUM	3EDA	914	910
FOUND	3DD0	1065	1059
FPERAD	006C	170	556
FREPTR	0040	127	
GETPRM	B297	876	629 697 791
GETSTR	0071	51	615 828
GETTIM	B314	966	639 800 917 929 946
GRAMFL	0069	173	
HYPEN	002D	184	1002
IOSTRT	003C	125	
JOYX	0077	69	
JOYY	0076	68	
KEYBD	0074	66	
LENCST	0060	160	744 752
LENNST	0062	161	737 739 750
LINKS1	AD96	214	
LINKS2	ADA1	215	214
LINKS3	ADAF	216	215
LINKS4	ADBC	217	216
LINKS5	ADC6	218	217
LINKS6	ADD2	219	218
LINKS7	ADDD	220	219
LINKS8	ADE8	221	220
LINKS9	ADF4	222	221
LINKSA	AE01	223	222
LINKSB	AE09	224	223
LINKSC	AE13	225	224
LNBUF	0036	122	
LNKRTN	B060	583	251 283 311 366 379 393 406 772 867
LOADAD	B465	1147	729 839 1054 1082 1168
LOOKUP	B3A9	1051	648 669 809
LOOP1	B15E	727	727
LOOPR	B1A0	748	755 764
LPAR$	0087	193	478 611 787
LSUBP	0048	133	
M$EDIT	6010	32	41
M$EXEC	A000	31	43 44
M$MON	0010	33	35 36
M$PSCN	6A70	30	38 39
MOTION	007A	72	
NEXT	B3EB	1091	1121
NEXT1	B124	694	630
NEXT2	B139	703	698
NINE	0039	186	1032
NOTIME	B38E	1018	984 998 1005
NUDTAB	0028	115	
NULRTN	B05B	581	335 342 343
NUMBE$	00FD	195	253 269 439 445 487
NUMBER	0023	178	895 900 912 1104
NUMERC	B392	1029	983 994 997 1004
NXTPAR	B1DF	790	866

SYMBOL	VALUE	DEF	REFERENCE TABLE
NXTPHR	B455	1130	1119 1123
OVER	AF5D	426	423
PARSE	0074	52	271 358 513 527 544 571 878
PERIOD	002E	131	992
PGMCHR	0079	55	252 268 270 331 333 357 416 486 489 574 876
PGMPTR	002C	117	
PHADDR	0052	149	
PHDATA	005D	158	1093 1094 1127
PHLEN	0000	90	816 817 827 835 844 861
PHRADD	0001	91	815 838
PHRASE	B2AD	695	643 805
PLUS	002B	182	981
POP	0005	1	
PREPN	B063	589	294 306 346 418 581
PRGFLG	0044	130	
PTC2ED	005E	159	734 735 735 737 741 743 744 745 748 749
PTCCIP	000E	99	1089 1094 1096 1097 1098 1099 1104 1105 1109 1121 1127
PTCCIS	0008	94	637 642 798 804 896 897 898 900 901 904 906 910 911 914 916 920 928 931 945 967 968 976 977
PTCCPH	0014	103	1087 1088 1095 1108
PTEBSL	0004	86	624 625 1182
PTFBPH	0012	102	725 728 838 1053 1057 1086 1087 1110 1112 1118 1122 1126 1128 1132 1137 1148 1149
PTFBSL	0000	84	622 624 720 722 723 725 726 733 734 736
PTFCIP	000C	98	664 665 668 675 676 807 904 920 1089
PTFCIS	0006	93	631 633 637 793 795 798 1030
PTIME	B368	999	995
PTLBSL	0002	85	623 720 1182 1183 1184
PTLCIL	004F	147	660 676
PTLCIP	0010	100	660 668 807 914 915 931 932 1109 1121
PTLCIS	000A	95	632 633 634 642 794 795 796 804 910 969 1029
PTLCPH	0016	104	1084 1085 1086 1108 1110 1132
RAM	0001	1	246 274 275 298 391 441 447 470 471 471 533 548 664 722 723 725 734 735 737 741 744 748 834 835 836 859 897 900 911 967 976 1031 1032 1094 1097 1098 1104 1127 1183
RANDOM	0078	70	
RANGE	0002	61	340 361 493 509 516 530
RANGEV	B02C	553	542 546
READ	0058	154	757 846 1056 1085 1093 1115 1170 1172 1192 1200
READAD	B497	1167	1113 1133
RETURN	A01C	44	583
RKEY	0075	67	
RNDAG	B277	854	857
ROM	000A	1	1200
RPAR$	00B6	192	272 359 514 528 545 879
RPL	0012	35	472
RTNG	0026	114	
SAY	B07B	610	223
SEARCH	B3D1	1081	1058 1138
SEMICO	003B	179	973 1013

SYMBOL	VALUE	DEF	REFERENCE TABLE
SETCO	82A9	883	877 1032
SETRN	B4CA	1199	626 636 719 788 797 829
SMS	007B	57	591 824
SNTSRT	001E	110	
SPOO	0000	77	298 299 307 315 342 419 419 437 438 438
			441 442 453 454
SPO2	0002	78	421 424 425 443 443 444 456
SPO4	0004	79	246 343 391 420 420 447 452 453 455 456
			457 458 512
SPO6	0006	80	421 458 519
SPACE	0020	177	664 897 974 976 1097 1098
SPCHR	B002	526	237 404
SPCOL	A01E	43	241
SPDEL1	AE53	267	276
SPEAK	B13E	718	
SPEED	007E	59	238 242 278 295 339 360 492 508 515 529
			819
SPGET	B1D4	787	224
SPKSTS	0069	168	757 761 762 1192 1193
SPLFLG	004B	142	647 650 659 806 918 921 923
SPLOC	AFE5	506	245 390 449
SPMOVE	B014	540	257 377
SPNUM1	AFBF	477	234 266 291 330 356 374 387 401 415
SPNUM2	AFC5	485	208 235 293 376 389 403
SPNUM3	AFCC	488	207 254 440 446
SPNUM4	AFD1	492	273
SPRCOI	AEBC	329	217
SPRDEL	AE50	265	215
SPRINT	6016	41	281
SPRITE	0006	1	
SPRLC2	AF1C	388	392
SPRLOC	AF19	386	220
SPRMAG	AEF1	355	218
SPRMOV	AF09	373	219
SPRMV2	AF0C	375	378
SPRPAT	AF2D	400	221
SPRPO2	AE7E	292	310
SPRPO3	AEAF	314	305 309
SPRPOS	AE7B	290	216
SPRPT2	AF30	402	405
SPRT3	AE25	236	255
SPRT4	AE3A	250	258
SPRTE	AE1F	233	214
SPSAL	0008	81	246 274 275 298 391 441 447 498 533 548
SPTMP	000A	82	543 547
SREF	001C	109	616 618 830 832 834 835 836 859 860
STACK	0072	64	
STDATA	B4AF	1181	652 674 692 702
STLN	0030	119	
STREND	001A	108	
STRLEN	0064	162	812 816 1115
STRSP	0018	107	
STVSPT	0024	113	
SUBSTK	0073	65	
SUBTAB	003A	124	

SYMBOL	VALUE	DEF	REFERENCE TABLE
SYM	007A	56	590 923
SYMTAB	008E	126	
[illegible]	0000	60	209 243 279 296 820
TABLE	0003	1	
TEMP1	0054	151	853 856 968 969 982 993 996 1003 1029 1030 1031 1032 1148 1150 1151 1153 1156 1158
TEMP2	0056	152	746 754 763 1149 1152 1154 1155 1157
TIMER	0079	71	723 724 727
TIMING	0004	943	640 683 801
TIMLEN	0051	148	930 943 944 970 975 985 989 999 1006 1010 1014 1018
TOP	0003	1	
TOTTIM	004C	145	638 661 682 690 699 799 944 1183
TRYAGN	00AB	1053	1063
VARA	002A	116	
VARN	0020	111	
VOP	0000	1	365
VOPSTT	0078	70	335
VSL	0007	1	
XPOP	0078	54	770
XPUSH	0077	53	593 613 620 700 825
XSPTR	006E	171	612 701 771
WAIT	64BF	1190	627 721 742 789
WRITE	005A	156	730 747 748 845 1055 1083 1092 1114 1155 1156 1157 1158 1159 1169 1171
XPT	000E	1	
YPT	000D	1	
ZERO	0030	185	1031