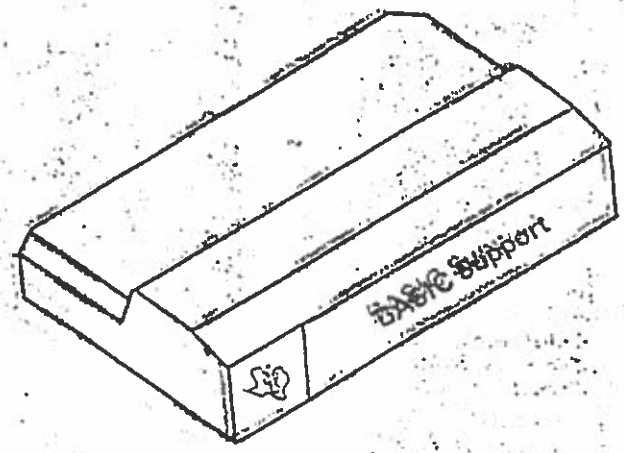
MORE POWER

from PCIF



TO YOUR 99/4

BLOCK MOVES, CHAINING,
CRU ACCESS, 2K BYTES CPU RAM FOR
MACHINE CODE SUBROUTINES,
AND MUCH, MUCH, MORE

TI BASIC Support Module Pricing

with 2K RAM - $150.00
Without 2K RAM - $100.00

Available to TI Cost Centers and individual TI's and their families.

To order call Sally Hayes in Dallas, 995-7131.

Allow two weeks for delivery.

BASIC SUPPORT LIBRARY USER'S GUIDE
(PRELIMINARY)

INTRODUCTION

In order to form a more perfect union between the TI 99/4 and the BASIC programming language, this library provides access to various memory resources via subroutine calls from BASIC. These memory resources include VDP memory, CPU memory, CRU interface registers, VDP registers, and disk sectors.

The names of these support subroutines each begin with the character "@" in order to distinguish them as belonging to this collection.

Since two modules won't fit in one slot, the routines SAY and SPGET are also included for speech synthesizer interface.

SUMMARY

The BASIC support module was produced to enable an experienced programmer to get at the features of the 99/4 through BASIC. It does not require a memory extension.

Some of the major features are:

BLOCK MOVES: Allows for several screen images to be created and subsequently displayed rapidly. An example of use includes the agriculture demonstration where a screen full of data was quickly displayed rather than BASIC updating the screen line by line. Apparent screen motion can be obtained by rapidly updating the display memory.

EXECUTE 9900 CPU CODE: Allows a BASIC program to execute a very short (48 byte) 9900 program. This feature was used in generating a BASIC program to test the CPU memory extension.

PACKING UTILITIES: Allows for saving memory space. A 99/4 numeric variable takes 8 bytes. These routines allow 64 bits, 8 one-byte integers, or 4 two-byte integers to be stored into one location. This very useful if the program requires a large number of small integers.

LOGICAL ROUTINES: BASIC does not allow logical operations, a frequent technique used by programmers. This will allow one to draw lines and curves and other graphics quickly from BASIC.

CRU ACCESS: Used in VTR to read tape position. Used in detecting telephone rings in TIVUE system. This is generally used for controlling devices and detecting external stimuli.

PROGRAM CHAINING: A facility is provided for saving a program on a peripheral device and subsequently re-starting the program when invoked from another program.

PLAY Sound Table: The programmer may use these routines to construct and play sounds of more than the usual single chord, at maximum speed (60 sounds/second) and in parallel to other program activities.

BLOCK MOVE

Block move routines are supplied for moving large blocks of data within or between memory resources. The types of memory include VDP, RAM, CPU RAM or ROM, G-ROM.

The general form of a CALL to one of these routines is:

```
CALL @X2Y (Len, Src, Dest)
```

where X and Y are each V, C or G. X represents the type of memory from which to move; Y represents the type into which to move.

V indicates VDP RAM
C indicates CPU (9900) memory address
G indicates a GROM

Len represents a variable whose value is the number of bytes to be transferred.

Src represents a variable whose value is the source (from) address.

Dest represents a variable whose value is the destination (to) address.

Note that Len, Src and Dest are all variable names and may not be general expressions.

VDP Addressing (V)

A variable containing a VDP address may have a value from 0 to 16383 (>3FFF). Moving to or from addresses outside this range can have undesired effects.

CPU Addressing (C)

CPU (9900) memory has addresses from 0 to 65535 (>FFFF).

GROM, Addressing (G)

GROM addressing only applies to library module 0.
GROM has addresses from 0 to 65535 (>FFFF).
GROM as Dest is NOT ALLOWED.

Example 1:

To move a screen image into CPU memory at > 2000,

```
L  = 768
V  = 0
C  = 8192
CALL @V2C (L,V,C)
```

NOTE: The screen image is at VDP RAM location 0 thru 767 (> 2FF).

Example 2:

To move the screen up 1 line (32 characters)

```
L  = 768 -32
V1 = 32
V2 = 0
CALL @V2V (L,V1,V2)
```

BASIC VARIABLE ADDRESS

To acquire the VDP RAM address of a BASIC variable, use the @LOC routine.

```
CALL @LOC (Var,Addvar)
```

Var and Addvar represent two BASIC variable names. The VDP RAM address of the first (low address) byte of Var is placed in Addvar.

Example: Save the screen image in array A.

```
DIM A (95)
CALL @LOC (A(o),AL)
L = 768
V = 0
CALL @V2V (L,V,AL)
```

NOTE: Each numeric variable requires 8 bytes so that 96 array elements thru 95) are sufficient to hold 768 bytes.

NOTE: CALL @V2V (L,AL,V) will replace the saved screen image.

NOTE: CALL @LOC (AS,AL) will give the address of the string AS in AL, but this value of AL must be used before the string AS is re-allocated by the system. Numeric variables are not re-allocated!

CRU ACCESS

There are 2 routines for CRU access, one for writing, one for reading.

```
CALL @CRIN (Addr,Bits, Data)
CALL @CROUT (Addr,Bits,DATA)
```

will cause the number of bits specified by Bits to be transferred from (TO) the BASIC variable specified by Data into ~~the CRU at the address~~ (from) specified by Addr. The bits to be written should be right justified in DATA. The number of bits should be between 1 and 16.

Execute 9900 (CPU) Code:

CALL @CEX (R0,R1,R2)

Causes the 48 bytes at VDP RAM location >7D0 to execute as if it were a 9900 program at CPU RAM location >8300. This is strictly "user beware" and the 9900 program must have the following form:

- Locations >7D0, >7D1 must contain a pointer to the first instruction to be executed.
- If the workspace pointer is changed, then it must be restored to 83E0 before returning to BASIC.
- In order to return to BASIC, the last 9900 instruction executed must be:

  B *R11 (>045B)

The register values at >8300, >8302, >8304 will be returned in R0,R1,R2 respectively.

The simplest way to get a program into VDP RAM location >7D0 is to use the CHAR function with character codes 154 thru 159.

Example:

```
CALL CHAR (154, "8302045B")
CALL @CEX (A,B,C)
```

will execute a return (B *R11) to BASIC, i.e., a NO-OP.

EXIT BASIC

To simulate a "SHIFT Q" from your program, execute the statement

```
CALL @EXIT
```

nothing in memory will be recoverable.

MODIFY VDP REGISTER (Handle with Care)

To put the 1-byte value "Value" into the VDP register "Reg",

```
CALL @VDRPUT (Reg, Value)
```

Where Reg and Value are BASIC variables containing the appropriate values.

Adjust Character Codes

To modify the contents of a string of bytes in VDP RAM by adding a constant to each one,

```
CALL @VADD(L,Delta,Bytes)
```

LOGICAL OPERATIONS

```
CALL @VAND(L,V1,V2)
```

The string of bytes beginning at the address contained in V1 will be logically "AND" ed with (and the result stored into) the string of L bytes at the address contained in V2.

```
CALL @VOR (L,V1,V2)
```

does the corresponding "OR" operation.

ONE-BIT FLAGS

To treat a numeric variable as 64 individual 1-bit flags;

```
CALL @ON(V,B)
```

to set bit B in variable V to one;

```
CALL @OFF(V,B)
```

to set bit B in variable V to zero;

```
CALL @TEST(V1,B,V2)
```

to get the value of bit B in variable V1 and put it in V2;

```
CALL @VON(V,B)
CALL @VOFF(V,B)
CALL @VTEST(V,B)
```

function the same as @ON, @OFF, @TEST respectively, except that V is a numeric variable whose value is the address of the 8-byte VDP RAM area whose bit is to be set.

Packing Bytes

If in any array of 8 numeric variables, each variable has an integer value less than 256, then these 8 values can be packed into a single numeric variable by

```
CALL @PUTB (A(X),B)
```

The 8 values in A(X),A(X+1) thru A(X+7) will be packed into the variable B, left to right (exponent first).

The reverse operation is accomplished by

```
CALL @GETB(B,A(X))
```

Packing Words

Similar to Packing Bytes, except that 4 variables of integer value up to 2¹⁶-1 (65535) are packed into a single numeric variable.

Pack them in by

CALL @PUTW(A(X),B)

and unpack them by

CALL @GETW(B,A(X))

CHAINING

The BASIC SUPPORT MODULE provides a checkpoint and chaining facility. At any time after a BASIC program has been loaded into the TI 99/4, (whether it has been executed or not), the user may "checkpoint" that program onto an external device. This "checkpoint" file may be re-started from another BASIC program (chaining) at some later time. A re-started program will have retained all previously computed values of variables and all special character graphics for character codes above 95.

CHECKPOINT

CALL CHECK(VARS)

VARS is the name of a string variable whose value is the name of the file onto which to save the current program and associated parameters.

NOTE: The call to CHECK must be an imperative statement, not a program statement.

RESTART

CALL CHAIN(VARS)

VARS is the name of a string variable whose value is the name of the file from which to retrieve a program to be re-started.

NOTE: The call to CHAIN must be a program statement, not an imperative statement.

PLAY Sound Table

CALL @PLAY(VADD)

VADD is the name of numeric variable whose value is the VDP RAM address at which is located the sound table to be "played". This "playing" will continue until a "stop" code is encountered or until another CALL @PLAY is executed, or CALL Sound, or any system function which causes a "beep".

CALL @PLABK(VADD)

If this statement is executed immediately after a CALL SOUND statement, on 11-byte string of sound definition and a 6-byte "stop", string will be placed at the VDP RAM location whose address is the value of VADD.

ACQUIRE SYSTEM DEPENDENT VALUES

The following subroutines may be called to acquire the corresponding values:

@MEM	CURRENT AVAILABLE VDP RAM
@MAXI	MAXIMUM POSITIVE VALUE
@MINI	LARGEST NEGATIVE VALUE
@EPS	SMALLEST POSITIVE NUMBER
@BIGI	LARGEST CONSECUTIVE INTEGER

Example:

CALL @MEM(A)

will cause the value of A to be changed to the number of bytes remaining available to strings, subroutine stack, etc., at the time of the CALL.

Display a Line of Text

CALL @AT(R,C,T$) will display the string value of the variable T$ horizontally from left to right on the display screen (with "LF/CR" after column 32) beginning at the row and column indicated by the values of variables R and C respectively.

Specifically, @AT acts exactly like HCHAR, with T$ subsituting for the string of multiple copies of the same character.

CALL a Known GPL Subroutine

CALL @GSUB(GADDR) will transfer to the GPL subroutine located at the address indicated by the value of the variable GADDR. Some examples of useful subroutines are:

ADDRESS	ROUTINE
>201C	REDEFINE DEFAULT CHARACTER SET
>4D18	STRING-SPACE GARBAGE COLLECTION

```
                  1              TITLE 'BSM'
                  2   *     BASIC SUPPORT MODULE
                  3   *     PHASE I FINAL VERSION
                  4   *         11/26/80
                  5   *
                 41              GROM  3
                 42              ORG   0
6000 AA0101      43              DATA  >AA.1.1.0.#0.#PROG1.#0.#SUBS.#0.#0
6003 000000
6006 667000
6009 006010
600C 000000
600F 00
     6010        44   SUBS       EQU   $
6010 602560      45   V2V        DATA  #CRIN.#V2V01
6013 19
6014 044056      46              DATA  4.:@V2V:
6017 3256
6019 066125      47   V2V01      CALL  P
601C 340280      48              MOVE  @01 FROM RAM(@02) TO RAM(@03)
601F 048006
6022 060012      49              CALL  RPL
                 50   ************************************************************
6025 607560      51   CRIN       DATA  #CROUT.#CRIN01
6028 35
6029 054045      52              DATA  5.:@CRIN:
602C 524942
                 53   *
                 54   *          MACHINE LANGUAGE (CRINDATA):
                 55   * >830C       MOV   @>8302.R12      - CRU ADDRESS
                 56   * >8310       SLA   R12.1             TIMES 2
                 57   * >8312       STCR  @>8300.0          READ 16 BITS INTO >8300
                 58   * >8316       B     *R11              RETURN TO G-LANGUAGE
                 59   *
602F C32083      60   CRINDATA DATA  #>C320.#>8302;#>0A1C.#>3420,#>8300.#>045B
6032 020A1C
6035 342083
6038 00045B
                 61   *          ENTRY POINT FROM BASIC
603B 066125      62   CRIN01     CALL  P
603E 800A00      63              DST   @0.@03          SAVE RTN BUFFER POINTER
6041 800206      64              DST   @02.@02+2
6044 066050      65              CALL  CRIN02
6047 CF0800      66              $IF   @02+2 .DGT. #>8 GOTO CRIN03
604A 086051
604D E70000      67              DSRL  @0.8
6050 08
6051 0665A1      68   CRIN03     CALL  CIF             CONV INTEGER TO FLOATING
6054 350008      69              MOVE  8 FROM @FAC TO RAM(@03)
6057 500A4A
605A 060012      70              CALL  RPL
                 71   *
                 72   *          ENTRY POINT FROM G-LANGUAGE
                 73   *          INPUT:
                 74   *               CRU ADDR IS IN       01.01+1
```

```
                  75  >                # BITS   IS IN       @2+1
                  76  *          OUTPUT:
                  77  *                CRU PATTERN          @>00
                  78  *
6050 310000       79  CRIN02    MOVE  12 FROM ROM(#CRINDATA) TO @>C
6050 0C602F
6053 BF0083       80            DST   >830C.@0        POINTER TO MACHINE LANG CODE
6056 0C
6057 E30600       81            DAND  >000F.@02       MASK OFF 12 BITS
605A 0F
605B E30600       82            DSLL  @02.6           SHIFT # BITS INTO POSITION
605E 06
605F B51206       83            DOR   @02.@>12        COMBINE STOR INSTR & # BITS
6062 0FF0         84            XML   >F0
6064 00           85            RTN
                  86  ************************************************************
6065 609960       87  CROUT     DATA  #CEX.#CROTO1
6068 60
6069 064043       88            DATA  6.:@CROUT:
606C 534F55
606F 54
                  89  *         ENTRY POINT FROM BASIC
6070 066125       90  CROTO1    CALL  P
6073 066099       91            CALL  CROTO2
6076 060012       92            CALL  RPL
                  93  *
                  94  *         ENTRY POINT FROM G-LANGUAGE
                  95  *         INPUT:
                  96  *               CRU ADDR IS IN       @1.@1+1
                  97  *               # BITS   IS IN       @2+1
                  98  *               CRU PATTERN IS IN    @3.@3+1
                  99  *         OUTPUT:
                 100  *               NONE
                 101  *
6089 B06802      102  CROTO2   DST   @01.@10B
608C BC6A07      103           ST    @02+1.@10B+2
608F BE686C      104           ST    BUF.@10B+3
6092 B06CCA      105           DST   @03.@BUF
6095 F66803      106           I/O   @10B.3
6098 00          107           RTN
                 108  ************************************************************
6099 60A260      109  CEX       DATA  #CEXP.#CEXO1
609C 0F
609D 044043      110            DATA  4.:@CEX:
60A0 4953
60A2 618260      111  CEXP      DATA  #LOC.#CEXPO1
60A5 AC
60A6 054043      112            DATA  5.:@CEXP:
60A9 455550
60AC 066125      113  CEXPO1    CALL  P
60AF B00202      114            DST   @01.@>0
60B2 0FF0        115            XML   >F0
60B4 8F0060      116            EIF   @>0 .000. 0 GOTO CEXPO2
60B7 0C
60B8 B00006      117            DST   @02.@>0
```

```
60BB 0665A1   118            CALL  CIF
60BE 350008   119            MOVE  8 FROM @FAC TO RAM(@02+2)
6001 B0084A
6004 B00000A  120            DST   @03.@>0
60C7 0665A1   121            CALL  CIF
60CA 350008   122            MOVE  8 FROM @FAC TO RAM(@03+2)
60CD B00C4A
6000 B00000E  123            DST   @04.@>0
6003 0665A1   124            CALL  CIF
6006 350008   125            MOVE  8 FROM @FAC TO RAM(@04+2)
6009 B0104A
600C 060012   126  CEXP02    CALL  RPL
600F 066625   127  CEX01     CALL  PGMCTR
60E2 35001A   128            MOVE  26 FROM @>18 TO RAM(ROLOUT)
60E5 A3C218
60E8 350030   129            MOVE  48 FROM RAM(>700) TO @0
60EB 00A700
60EE 0FF0     130            XML   >F0
60F0 35001A   131            MOVE  26 FROM RAM(ROLOUT) TO @>18
60F3 18A300
60F6 350006   132            MOVE  6 FROM @0 TO @>10
60F9 1000
60FB 066128   133            CALL  PA
60FE B00010   134            DST   @>10.@0
6101 0665A1   135            CALL  CIF
6104 350008   136            MOVE  8 FROM @FAC TO RAM(@01+2)
6107 B0044A
610A B00012   137            DST   @>12.@0
610D 0665A1   138            CALL  CIF
6110 350008   139            MOVE  8 FROM @FAC TO RAM(@02+2)
6113 B0084A
6116 B00014   140            DST   @>14.@0
6119 0665A1   141            CALL  CIF
611C 350008   142            MOVE  8 FROM @FAC TO RAM(@03+2)
611F B00C4A
6122 060012   143            CALL  RPL
     6125     144  P         EQU   $
6125 066625   145            CALL  PGMCTR
6128 064287   146  PA        $IF   @CHAT .NE. LPAR$ GOTO ERR7
612B 4600
612D 0F1B     147            XML   PGMCHR
612F 066171   148            CALL  PARM
6132 B0024A   149            DST   @FAC.@01
6135 B00400   150            DST   @0.@01+2
6138 064286   151            $IF   @CHAT .EQ. RPAR$ GOTO FINIS
613B 6167
613D 06616A   152            CALL  COMMA2
6140 B0064A   153            DST   @FAC.@02
6143 B00800   154            DST   @0.@02+2
6146 064286   155            $IF   @CHAT .EQ. RPAR$ GOTO FINIS
6149 6167
614B 06616A   156            CALL  COMMA2
614E B0044A   157            DST   @FAC.@03
6151 B00C00   158            DST   @0.@03+2
6154 064286   159            $IF   @CHAT .EQ. RPAR$ GOTO FINIS
```

```
0157 0167
0159 066160   160            CALL  COMMA2
015C 80004A   161            DST   @FAC.@04
015F B01000   162            DST   @0.@04+2
0162 064286   163            $IF   @CHAT .NE. RPAR$ GOTO ERR7
0165 4600
0167 0F1B     164  FINIS     XML   PGMCHR
0169 00       165            RTN
              166  *
              167  *         PARAMETER FETCH
              168  *
016A 064283   169  COMMA2    $IF   @CHAT .NE. COMMA$ GOTO ERR7
0160 4600
016F 0F1B     170            XML   PGMCHR
0171 0F13     171  PARM      XML   SYM
0173 0F14     172            XML   SMB
0175 B00048   173            DST   @FAC+4.@0
0178 350008   174            MOVE  8 FROM RAM(@0) TO @FAC
017B 4AB000
017E 066FCF   175            CALL  CVFLIN
0181 00       176            RTN
0182 019061   177  LOC       DATA  #G2V.#LOC01
0185 2B
0186 044040   178            DATA  4.:@LOC:
0189 4F43
018B 066125   179  LOC01     CALL  P
018E B00004   180            DST   @01+2.@0
0191 066EA1   181            CALL  CIF
0194 350008   182            MOVE  8 FROM @FAC TO RAM(@02+2)
0197 80084A
019A 060012   183            CALL  RPL
019D 01B3A1   184  G2V       DATA  #C2V.#G2V01
01A0 A6
01A1 044047   185            DATA  4.:@G2V:
01A4 3256
01A6 066125   186  G2V01     CALL  P
01A9 320280   187            MOVE  @01 FROM ROM(@02) TO RAM(@03)
01AC 0A0000
01AF 06
01B0 060012   188            CALL  RPL
01B3 01CD61   189  C2V       DATA  #V2G.#C2V01
01B6 BC
01B7 044043   190            DATA  4.:@C2V:
01BA 3256
01BC 066125   191  C2V01     CALL  P
01BF A7C283   192            DSUB  CBIAS.@02
01C2 00
01C3 340280   193            MOVE  @01 FROM @0(02) TO RAM(@03)
01C6 0AC000
01C9 06
01CA 060012   194            CALL  RPL
01CD 01E3A1   195  V2G       DATA  #V2C.#V2G01
01D0 06
01D1 044056   196            DATA  4.:@V2G:
01D4 3247
```

```
61D6 066125   197  V2G01    CALL  P
61D9 200200   198           MOVE  @01 FROM RAM(@02) TO ROM(@03)
61DC 000AB0
61DF 06
61E0 060012   199           CALL  RPL
61E3 61F061   200  V2C      DATA  #C2C.#V2C01
61E6 EC
61E7 044056   201           DATA  4.:@V2C:
61EA 3243
61EC 066125   202  V2C01    CALL  P
61EF A70A83   203           DSUB  CBIAS.@03
61F2 00
61F3 340200   204           MOVE  @01 FROM RAM(@02) TO @0(03)
61F6 000AB0
61F9 06
61FA 060012   205           CALL  RPL
61FD 621062   206  C2C      DATA  #G2G.#C2C01
6200 06
6201 044043   207           DATA  4.:@C2C:
6204 3242
6206 066125   208  C2C01    CALL  P
6209 A70A83   209           DSUB  CBIAS.@02
620C 00
620D A70A83   210           DSUB  CBIAS.@03
6210 00
6211 340200   211           MOVE  @01 FROM @0(02) TO @0(03)
6214 000AC0
6217 0006
6219 060012   212           CALL  RPL
621C 623362   213  G2G      DATA  #C2G.#G2G01
621F 2S
6220 044047   214           DATA  4.:@G2G:
6223 3247
6225 066125   215  G2G01    CALL  P
6228 2A0200   216           MOVE  @01 FROM ROM(@02) TO ROM(@03)
622B 000A00
622E 0006
6230 060012   217           CALL  RPL
6233 624E62   218  C2G      DATA  #G2C.#C2G01
6236 3C
6237 044043   219           DATA  4.:@C2G:
623A 3247
623C 066125   220  C2G01    CALL  P
623F A70A83   221           DSUB  CBIAS.@02
6242 09
6243 200200   222           MOVE  @01 FROM @0(02) TO ROM(@03)
6246 000AC0
6249 0006
624B 060012   223           CALL  RPL
624E 626962   224  G2C      DATA  #CRUN.#G2C01
6251 57
6252 044047   225           DATA  4.:@G2C:
6255 3243
6257 066125   226  G2C01    CALL  P
625A A70A83   227           DSUB  CBIAS.@03
```

```
625D 00
625E 3202C0    228            MOVE  @01 FROM ROM(@02) TO @0(@03)
6261 C00A00
6264 0006
6266 060012    229            CALL  RPL
6269 627B62    230  CRUN       DATA  #VDRPUT.#CRUN01
626C 73
626D 054043    231            DATA  5.:@CRUN:
6270 525545
6273 066625    232  CRUN01    CALL  PGMCTR
6276 0F20      233            XML   C2K
6278 060012    234            CALL  RPL
627B 62B562    235  VDRPUT    DATA  #VADD.#VDRP01
627E 99
627F 074056    236            DATA  7.:@VDRPUT:
6282 446250
6285 5554
6287 C06063    237  VDRDATA   DATA  #>C060.#>8302                    MOV >8302.R
628A 02
628B 06C108    238            DATA  #>06C1.#>0801.#>8C02            SWPB: MOVB
628E 018C02
6291 06C108    239            DATA  #>06C1.#>0801.#>8C02.#>045B     SWPB: MOVB
6294 018C02
6297 045B
6299 066125    240  VDRP01    CALL  P
629C E30200    241            DSLL  @01.8
629F 03
62A0 8C0307    242            ST    @02+1.@01+1
62A3 860280    243            SB    @01.7
62A6 310012    244            MOVE  18 FROM ROM(#VDRDATA) TO @>4
62A9 046287
62AC 8F0063    245            DST   >8304.@0
62AF 04
62B0 0FF0      246            XML   >F0
62B2 060012    247            CALL  RPL
               248  *
62B5 62D662    249  VADD       DATA  #VOR.#VADD01
62B8 BF
62B9 054056    250            DATA  5.:@VADD:
62BC 414444
62BF 066125    251  VADD01    CALL  P
62C2 A1020A    252            DADD  @03.@01
62C5 D10A02    253            $WHILE @03 .DLT. @01                  1
62C8 6203
62CA A0800A    254            ADD   @02+1.RAM(@03)                  1
62CD 07
62CE 910A      255            DINC  @03                             1
62D0 0562C5    256            $END
62D3 060012    257            CALL  RPL
62D6 62F962    258  VOR        DATA  #VAND.#VOR01
62D9 0F
62DA 044056    259            DATA  4.:@VOR:
62DD 4F52
62DF 066125    260  VOR01     CALL  P
62E2 A1020A    261            DADD  @03.@01
```

```
62E5 D10A02   262          $WHILE @03 .DLT. @01                          1
62E8 62F4
62EA B4600A   263           OR    RAM(@02).RAM(@03)                      1
62ED B006
62EF 910A     264           DINC  @03                                    1
62F1 9106     265           DINC  @02                                    1
62F3 0562E5   266          $END
62F6 060012   267           CALL RPL
62F9 631D63   268  VAND     DATA  #RSECT.#VAND01
62FC 03
62FD 054056   269           DATA  5.:@VAND:
6300 414544
6303 066125   270  VAND01   CALL  P
6306 A1020A   271           DADD  @03.@01
6309 D10A02   272          $WHILE @03 .DLT. @01                          1
630C 631A
630E B0800A   273           AND   RAM(@02).RAM(@03)                      1
6311 B006
6313 910A     274           DINC  @03                                    1
6315 9106     275           DINC  @02                                    1
6317 056337   276          $END
631A 060012   277           CALL  RPL
              278  *
     0030     279  STLN     EQU   >30
     0032     280  ENLN     EQU   >32
     0000     281  SRC      EQU   0
     0002     282  DEST     EQU   2
     0070     283  PTOP     EQU   >70
     2840     284  FRGLST   EQU   >2840
     03EC     285  SAVEPC   EQU   >3EC
     0024     286  STVSPT   EQU   >24
     05F8     287  VRAMVS   EQU   >5F8
     0044     288  BUFFY    EQU   >44
     0034     289  DATA     EQU   >34
     006E     290  VSPTR    EQU   >6E
     0006     291  PLEN     EQU   @2
     0008     292  BLEN     EQU   PLEN+2
              293  *
              294  *** CALL @RSEC(UNIT.SECTOR.BUFFER,ERROR)
              295  *
631D 635763   296  RSECT    DATA  #WSECT.#RSEC01
6320 27
6321 054052   297           DATA  5.:@RSEC:
6324 534543
6327 066125   298  RSEC01   CALL  P
632A BF4C00   299           DST   >00FF.@FAC+2
632D FF
632E BF4A00   300  RSEC02   DST   >0000.@FAC
6331 00
6332 B04C03   301           ST    @01+1.@FAC+2
6335 B04E0A   302           DST   @03.@FAC+4
6338 B05006   303           DST   @02.@FAC+6
633B BFA000   304           DST   >0110.RAM(0)
633E 0110
6340 BF5600   305  RSEC03   DST   >0000.@>56
```

```
6343 00
6344 060010   306            CALL  >10
6347 0A       307            DATA  >0A
6348 B00050   308            DST   @FAC+6.@0
634B 066BA1   309            CALL  CIF
634E 350008   310            MOVE  8 FROM @FAC TO RAM(@04+2)
6351 B0104A
6354 060012   311            CALL  RPL
              312  *
              313  ***  CALL @WSEC(UNIT.SECTOR.BUFFER.ERROR)
              314  *
6357 636863   315  WSECT    DATA  #RENAME.#WSEC01
635A 61
635B 054057   316            DATA  5.:@WSEC:
635E 534543
6361 066125   317  WSEC01   CALL  P
6364 BF4C00   318            DST   >0000.@FAC+2
6367 00
6368 056326   319            B     RSEC02
              320  *
              321  ***  CALL @RENAME(UNIT.OLDNAME,NEWNAME.ERROR)
              322  *
636B 638863   323  RENAME   DATA  #DSKFMT.#RENM01
636E 77
636F 074052   324            DATA  7.:@RENAME:
6372 454541
6375 4045
6377 066125   325  RENM01   CALL  P
637A BC4C03   326            ST    @01+1.@FAC+2          UNIT NUMBER
6370 B05008   327            DST   @02+2.@FAC+6          OLD NAME
6380 B04E0C   328            DST   @03+2.@FAC+4          NEW NAME
6383 BFA000   329            DST   >0113.RAM(0)
6386 0113
6388 056340   330            B     RSEC03
              331  *
              332  ***  CALL @DSKFMT(UNIT.TRACKS,BUFFER,ERROR)
              333  *
638B 63AF63   334  DSKFMT   DATA  #ON.#DFMT01
638E 97
638F 074044   335            DATA  7.:@DSKFMT:
6392 534846
6395 4054
6397 066125   336  DFMT01   CALL  P
639A BF4A00   337            DST   >0C00.@FAC
639D 00
639E B04C06   338            DST   @02.@FAC+2            NUMBER OF TRACKS
63A1 BC4C03   339            ST    @01+1.@FAC+2          UNIT NUMBER
63A4 B04E0A   340            DST   @03.@FAC+4            BUFFER AREA
63A7 BFA020   341            DST   >0111.RAM(0)
63AA 0111
63AC 056340   342            B     RSEC03
              343  *
63AF 638763   344  ON       DATA  #VON.#ON01      SET BIT "ON"
63B2 C9
63B3 03404F   345            DATA  3.:@ON:
```

```
6386 4E
6387 630C63   346  VON      DATA  #OFF.#VON01    SET BIT "ON" INDIRECT
638A C0
638B 044056   347           DATA  4.:@VON:
638E 4F4E
6390 066125   348  VON01    CALL  P
6393 B00402   349           DST   @01.@01+2
6396 0563C0   350           B     ON02
6399 066125   351  ON01     CALL  P
639C 06650C   352  ON02     CALL  BYTBIT
639F BE0220   353           ST    >20.@01
63A2 E80207   354           SRC   @01.@02+1
63A5 B48004   355           OR    @01.RAM(@01+2)
63A8 02
63A9 060012   356           CALL  RPL
              357  *
63AC 63E563   358  OFF      DATA  #VOFF.#OFF01
63AF F8
63B0 04404F   359           DATA  4.:@OFF:
63B3 4646
63B5 640863   360  VOFF     DATA  #TEST.#VOFF01
63B8 BF
63B9 054056   361           DATA  5.:@VOFF:
63BC 4F4646
63BF 066125   362  VOFF01   CALL  P
63F2 B00402   363           DST   @01.@01+2
63F5 0563FB   364           B     OFF02
63F8 066125   365  OFF01    CALL  P
63FB 06650C   366  OFF02    CALL  BYTBIT
63FE BE027F   367           ST    >7F.@01
6401 E80207   368           SRC   @01.@02+1
6404 B08004   369           AND   @01.RAM(@01+2)
6407 02
6408 060012   370           CALL  RPL
              371  *
640B 641564   372  TEST     DATA  #VTEST.#TEST01
640E 26
640F 054054   373           DATA  5.:@TEST:
6412 455354
6415 644C64   374  VTEST    DATA  #GETWRD.#VTST01
6418 20
6419 064056   375           DATA  6.:@VTEST:
641C 544553
641F 54
6420 B00402   376  VTST01   DST   @01.@01+2
6423 056429   377           B     TEST02
6426 066125   378  TEST01   CALL  P
6429 06650C   379  TEST02   CALL  BYTBIT
642C BE0280   380           ST    >80.@01
642F E80207   381           SRC   @01.@02+1
6432 560E     382           CLR   @04
6434 350007   383           MOVE  7 FROM @04 TO @04+1
6437 0F0E
6439 080280   384           CLOG  RAM(@01+2).@01
643C 04
```

```
6430 6443      385            BS     UNSET
643F BF0EBF    386            DST    >BFFF.@04
6442 FF
6443 350008    387  UNSET     MOVE   8 FROM @04 TO RAM(@03+2)
6446 B00C0E
6449 060012    388            CALL   RPL
644C 647C64    389  GETWRD    DATA   #PUTWRD.#GETW01
644F 56
6450 054047    390            DATA   5.:@GETW:
6453 455457
6456 066125    391  GETW01   CALL   P
6459 870A      392            DCLR   @03
645B C80A00    393           $WHILE @03 .DL. >4                                1
645E 046479
6461 B00080    394            DST    RAM(@01+2).@0                             1
6464 04
6465 0665A1    395            CALL   CIF                                       1
6468 350008    396            MOVE   8 FROM @FAC TO RAM(@02+2)                 1
646B B00B4A
646E 9504      397            DINCT  @01+2                                     1
6470 A30800    398            DADD   8.@02+2                                   1
6473 08
6474 910A      399            DINC   @03                                       1
6476 056458    400           $END
6479 060012    401            CALL   RPL
647C 64AB64    402  PUTWRD    DATA   #GETBYT.#PUTW01
647F 86
6480 054050    403            DATA   5.:@PUTW:
6483 555457
6486 066125    404  PUTW01    CALL   P
6489 870A      405            DCLR   @03
648B C80A00    406           $WHILE @03 .DL. >4                                1
648E 0464A8
6491 350008    407            MOVE   8 FROM RAM(@01+2) TO @FAC                 1
6494 4AB004
6497 0F12      408            XML    CFI                                       1
6499 B0B008    409            DST    @FAC.RAM(@02+2)                           1
649C 4A
6490 9508      410            DINCT  @02+2                                     1
649F A30400    411            DADD   8.@01+2                                   1
64A2 08
64A3 910A      412            DINC   @03                                       1
64A5 056488    413           $END
64A8 060012    414            CALL   RPL
64AB 640064    415  GETBYT    DATA   #PUTBYT.#GETB01
64AE B5
64AF 054047    416            DATA   5.:@GETB:
64B2 455442
64B5 066125    417  GETB01    CALL   P
64B8 870A      418            DCLR   @03
64BA C80A00    419           $WHILE @03 .DL. >8                                1
64BD 0264DA
64C0 8700      420            DCLR   @0                                        1
64C2 BC01B0    421            ST     RAM(@01+2).@1                             1
64C5 04
```

```
64C6 0665A1   422          CALL  CIF                                  1
64C9 350008   423          MOVE  8 FROM @FAC TO RAM(@02+2)            1
64CC B0024A
64CF 9104     424          DINC  @01+2                                1
64D1 A30200   425          DADD  8.@02+2                              1
64D4 08
64D5 910A     426          DINC  @03                                  1
64D7 0564BA   427         $END
64DA 060012   428          CALL  RPL
64DD 651A64   429  PUTBYT  DATA  #RUN.#PUTBO1
64E0 E7
64E1 054050   430          DATA  5.:8PUTB:
64E4 555442
64E7 066125   431  PUTBO1  CALL  P
64EA 670A     432          DCLR  @03
64EC C80A00   433         $WHILE @03 .DL. >8                          1
64EF 026509
64F2 350008   434          MOVE  8 FROM RAM(@01+2) TO @FAC            1
64F5 4A0004
64F8 0F12     435          XML   CFI                                  1
64FA B00002   436          ST    @FAC+1.RAM(@02+2)                    1
64FD 42
64FE 9108     437          DINC  @02+2                                1
6500 A30400   438          DADD  8.@01+2                              1
6503 08
6504 910A     439          DINC  @03                                  1
6506 0564EC   440         $END
6509 060012   441          CALL  RPL
              442  *
650C AE0608   443  BYTBIT  DIV   8.@02
650F 2608     444          CLR   @02+2
6511 B00906   445          ST    @02.@02+3
6514 2606     446          CLR   @02
6516 A10408   447          DADD  @02+2.@01+2
6519 00       448          RTN
              449  *
651A 663065   450  .RUN    DATA  #SAVE.#RUNO1
6510 22
651E 035255   451          DATA  3.:RUN:
6521 4E
6522 D7703F   452  RUNO1   $IF   @PTOP .DEQ. >3FFF GOTO RUNO2
6525 FF658A
6528 B00270   453          DST   @PTOP.@DEST
652B 9102     454          DINC  @DEST
652D B00002   455          DST   @DEST.@SRC
6530 BF083F   456          DST   >3FFF.@BLEN
6533 FF
6534 A50870   457          DSUB  @PTOP.@BLEN
6537 BF703F   458          DST   >3FFF.@PTOP
653A FF
653B A50008   459          DSUB  @BLEN.@SRC
653E C90030   460         $WHILE @SRC .DHE. @STLN                     1
6541 4552
6543 340380   461          MOVE  @BLEN FROM RAM(@SRC) TO RAM(@DEST)   1
6546 028000
```

```
6549 B00200   462          DST   @SRC.@DEST                                  1
654C A50008   463          DSUB  @BLEN.@SRC                                  1
654F 05653E   464         $END
6552 B00A02   465          DST   @DEST.@03
6555 A50A30   466          DSUB  @STLN.@03
6558 B00230   467          DST   @STLN.@DEST
655B A10208   468          DADD  @BLEN.@DEST
655E 340AB0   469          MOVE  @03 FROM RAM(@STLN) TO RAM(@DEST)
6561 02B030
6564 A13208   470          DADD  @BLEN.@ENLN
6567 A13008   471          DADD  @BLEN.@STLN
656A B00A30   472          DST   @STLN.@03
656D 950A     473          DINCT @03              POINTER TO LINE ADDR
656F 010A32   474         $WHILE @03 .DLT. @ENLN                             1
6572 658A
6574 350002   475          MOVE  2 FROM RAM(@03) TO @0                        1
6577 00B00A
657A A10008   476          DADD  @BLEN.@0                                     1
657D 350002   477          MOVE  2 FROM @0 TO RAM(@03)                        1
6580 B00A00
6583 A30A00   478          DADD  4.@03                                        1
6586 04
6587 05656F   479         $END
658A B03432   480  RUN02   DST   @ENLN.@DATA     SET-UP FOR FIRST LINE
658D A73400   481          DSUB  >03.@DATA
6590 03
6591 87A3EC   482          DCLR  RAM(SAVEPC)     DISALLOW CONTINUE
6594 BF6E05   483          DST   VRAMVS.@VSPTR   SET STACK EMPTY
6597 F8
6598 B0246E   484          DST   @VSPTR.@STVSPT
659B BE4401   485          ST    1.@BUFFY
659E 052840   486          B     PRGLST
              487  * CIF  -CONVERT INTEGER TO FLOATING    -
              488  *      -INTEGER  IN @>00               -
              489  *      -FLOATING IN FAC.FAC+1.....FAC+7-
     65A1     490  CIF    EQU   $
65A1 874A     491          DCLR  @FAC
65A3 B04C00   492          DST   @0,@FAC+2
65A6 874E     493          DCLR  @FAC+4
65A8 8750     494          DCLR  @FAC+6
65AA 8F4C     495          DCZ   @FAC+2
65AC 65CE     496          BS    CIFEND
65AE AF4A27   497          DDIV  10000.@FAC
65B1 10
65B2 AE4C64   498          DIV   100.@FAC+2
65B5 BE4A42   499          ST    >42.@FAC
65B8 8E48     500          CZ    @FAC+1
65BA 45CE     501          BR    CIFEND
65BC 350003   502          MOVE  3 FROM @FAC+2 TO @FAC+1
65BF 484C
65C1 A64A01   503          SUB   1.@FAC
65C4 8E48     504          CZ    @FAC+1
65C6 45CE     505          BR    CIFEND
65C8 B04B4C   506          DST   @FAC+2.@FAC+1
65CB A64A01   507          SUB   1.@FAC
```

```
     65CE      508  CIFEND   EQU   $
65CE 00        509           RTN
               510  *
               511  ***  CONVERT FLOATING TO INTEGER
               512  ***  PLACE INTEGER IN FLOATING(LOW WORD)
               513  ***  MARK BITS 32(INTEGER PRESENT)
               514  ***            40(FLOATING PRESENT)
               515  *
     65CF      516  CVFLIN   EQU   $
65CF DA4E60    517           CLOG  >80.@FAC+4                IS INTEGER PRESENT?
65D2 4609      518           BR    CVEND                     YES. SHORT-CUT
65D4 874E      519           DCLR  @FAC+4                    NO.  CALCULATE INTEG
65D6 8750      520           DCLR  @FAC+6
65D8 CE4A42    521           $IF   @FAC .GT. >42 GOTO CVEND
65DB 6609
65DD 024A40    522           $IF   @FAC .LT. >40 GOTO CVEND
65E0 4609
65E2 BC5148    523           ST    @FAC+1.@FAC+7
65E5 D64A40    524           $IF   @FAC .EQ. >40 GOTO CVEND
65E8 6609
65EA BC4F4C    525           ST    @FAC+2.@FAC+5
65ED A85000    526           DMUL  100.@FAC+6
65F0 64
65F1 B05052    527           DST   @FAC+8.@FAC+6
65F4 A1504E    528           DADD  @FAC+4.@FAC+6
65F7 D64A41    529           $IF   @FAC .EQ. >41 GOTO CVEND
65FA 6609
65FC BC4F4D    530           ST    @FAC+3.@FAC+5
65FF A85000    531           DMUL  100.@FAC+6
6602 64
6603 B05052    532           DST   @FAC+8.@FAC+6
6606 A1504E    533           DADD  @FAC+4.@FAC+6
6609 B04A50    534  CVEND    DST   @FAC+6.@FAC               MOVE INTEGER TO FAC
660C 00        535           RTN
     660D      536  ERR7     EQU   $
660D 06001C    537           CALL  ERR$$
6610 6612      538           DATA  #SYNERR
               539           BASE  0.0.>300.>300.0.0.>60
6612 128A8A    540  SYNERR   DATA  18.:** SYNTAX ERROR **:
6615 80B3B9
6618 AEB4A1
661B B280A5
661E B2B2AF
6621 B280BA
6624 8A
               541           BASE  0.0.>300.>300.0.0.0
6625 8E8089    542  PGMCTR   CI    @>89
6628 4620      543           BR    PGMCT1
662A B02C56    544           DST   @>56.@>2C
662D 0F1B      545  PGMCT1   XML   PGMCHR
662F 00        546           RTN
6630 665E66    547  SAVE     DATA  #LOAD.#SAVE01
6633 3A
6634 054053    548           DATA  5.:@SAVE:
6637 415645
```

```
                 549  ***********************************************
                 550  **** CALL @SAVE(LEN.ADDR,NAME-LEN,NAME-ADDR) *****
                 551  ***********************************************
663A 066125      552  SAVE01   CALL  P
663D BEA3C0      553           ST    6.RAM(ROLOUT)              CODE FOR 'SAVE'
6640 06
6641 BDA3C6      554  SAVE02   DST   @01.RAM(ROLOUT+6)          LENGTH
6644 02
6645 BDA3C2      555           DST   @02.RAM(ROLOUT+2)          ADDRESS
6648 06
6649 BF5603      556           DST   ROLOUT+9.@PABLOC           DSR PTR
664C C9
664D 340AA3      557           MOVE  @03 FROM RAM(@04) TO RAM(ROLOUT+10)   FILE NA
6650 CA000E
                 558  *  03 MUST BE LESS THAN 16
6653 BCA3C9      559           ST    @03+1.RAM(ROLOUT+9)        NAME LENGTH
6656 0B
6657 060010      560           CALL  CALDSR                     CALL DSR
665A 02          561           DATA  8
665B 060012      562           CALL  RPL                        RETURN TO BASIC
665E 667266      563  LOAD     DATA  #EXIT.#LOAD01
6661 68
6662 05404C      564           DATA  5.:@LOAD:
6665 4F4144
6668 066125      565  LOAD01   CALL  P
666B BEA3C0      566           ST    5.RAM(ROLOUT)              CODE FOR 'LOAD'
666E 05
666F 056641      567           B     SAVE02
6672 66D466      568  EXIT     DATA  #SASUB.#EXIT01
6675 7C
6676 054045      569           DATA  5.:@EXIT:
6679 584954
667C 0B          570  EXIT01   EXIT
667D 6698        571  PROG1    DATA #PROG2      PTR TO NEXT
667F 6696        572           DATA #PROG11     ENTRY
6681 144350      573           DATA  20.:CPU PROG AT C(>6000):
6684 552050
6687 524F47
668A 204154
668D 204328
6690 3E3630
6693 303029
6696 0F70        574  PROG11   XML   >70        CPU >6010
6698 6688        575  PROG2    DATA  #PROG3     PTR TO NEXT
669A 6681        576           DATA  #PROG21    ENTRY
669C 144350      577           DATA  20.:CPU PROG AT G(>7000):
669F 552050
66A2 524F47
66A5 204154
66A8 204728
66AB 3E3730
66AE 303029
6681 311000      578  PROG21   MOVE  >1000 FROM ROM(#>7000) TO @>9000    C2000-830
6684 8F9000
6687 7000
```

```
66B9 0F20      579            XNL   >20        CPU :2000
               580   *
66BB 0000      581   PROG3    DATA  #0
66BD 7000      582            DATA  #>7000
66BF 144750    583            DATA  20.:GPL PROG AT G(>7000):
66C2 4C2050
66C5 524F47
66C8 204154
66CB 204726
66CE 3E3730
66D1 303029
               584   *        END
```

```
                586  * Use of CPU ram locations
                587  *
0000            588  PTFBSL   EQU >00     Pointer To First Byte Speak List
0002            589  PTLBSL   EQU >02     Pointer To Last Byte Speak List
0004            590  PTEBSL   EQU >04     Pointer To End Byte Speak List
0000            591  PHLEN    EQU >00     PHrom data lenght
0001            592  PHRADD   EQU >01     PHRom ADDress
                593
0006            594  PTFCIS   EQU >06     Pointer To First Char In String
0008            595  PTCCIS   EQU >08     Pointer To Current Char In String
000A            596  PTLCIS   EQU >0A     Pointer To Last Char In String
                597
000C            598  PTFCIP   EQU >0C     Pointer To First Char In Phrase
000E            599  PTCCIP   EQU >0E     Pointer To Current Char In Phrase
0010            600  PTLCIP   EQU >10     Pointer To Last Char In Phrase
                601
0012            602  PTFBPH   EQU >12     Pointer To First Byte In PHrom
0014            603  PTCCPH   EQU >14     Pointer To Current Char in PHrom
0016            604  PTLCPH   EQU >16     Pointer to Last Char in PHrom
                605
004A            606  CCHAR    EQU >4A     Current CHAR
004B            607  TOTTIM   EQU >4B     TOTal wait TIme
004C            608  SPLFLG   EQU >4C     SPelL out phrase FLaG
004D            609  DATAAD   EQU >4D     speech DATA ADdr
004F            610  PTLCIL   EQU >4F     Pointer To Last Char In List
0051            611  TIMLEN   EQU >51     TIme LENght of timing char
0052            612  PHADDR   EQU >52     PHrom ADDRess
                613
0054            614  TEMP1    EQU >54     TEMPorary cpu location 1
0056            615  TEMP2    EQU >56     TEMPorary cpu location 2
                616
0058            617  READ     EQU >58     Contains addr of speech periferal
                618  *                        READ byte interface
005A            619  WRITE    EQU >5A     Contains addr of speech periferal
                620  *                        WRITE byte interface
005C            621  COND     EQU >5C     CONDition flag used in testing
005D            622  PHDATA   EQU >5D     PHrom DATA
005E            623  PTCBED   EQU >5E     Pointer To Current Byte Ext Data
0060            624  PTLBED   EQU >60     Pointer to Last Byte Ext Data
0062            625  PTEBED   EQU >62     Pointer To End Byte Ext Data
0064            626  STRLEN   EQU >64     STRing LENght
0066            627  BYTE1    EQU >66     BYTE 1
0067            628  BYTE2    EQU >67     BYTE 2
0068            629  BYTE3    EQU >68     BYTE 3
0069            630  SPKSTS   EQU >69     SPeak StaTuS
```

```
                 632  ***********************************************************
                 633  * ASCII symbol definitions
                 634
0041             635  ASCIIA  EQU :A:
0020             636  SPACE   EQU : :
0023             637  NUMBER  EQU :#:
003B             638  SEMICO  EQU :;:
003A             639  COLON   EQU ::::
002E             640  PERIOD  EQU :.:
002B             641  PLUS    EQU :+:
002C             642  COMMA   EQU :,:
002D             643  HYPEN   EQU :-:
0030             644  ZERO    EQU :0:
0039             645  NINE    EQU :9:
                 646  ***********************************************************
                 647  * External BASIC subroutine and other standard definitions
0017             648  VPUSH   EQU  23          Push value onto the stack
0018             649  VPOP    EQU  24          Pop value off the stack
0010             650  CPL     EQU >10          Call link
0038             651  GETSTR  EQU >38          Get string
0079             652  TIMER   EQU >79          Location of interval timer
0089             653  GROMFL  EQU >89          Grom flag
002C             654  CPU2C   EQU >2C
001C             655  SREF    EQU >1C          String reference
                 656  ***********************************************************
```

```
66D4 66DC6A   658  SASUB    DATA  #SPSUB,#SAY,3,:SAY:
66D7 280353
66DA 4159
66DC 68D26A   659  SPSUB    DATA  #ENDLNK,#SPGET,5,:SPGET:
66DF 9F0553
66E2 504745
66E5 54
              660  MAKESL
66E6 066920   661    CALL SETRW
66E9 B00806   662    DST @PTFCIS,@PTCCIS
66EC 0667BD   663    CALL GETTIM
66EF 864B     664    CLR @TOTTIM
66F1 0667AD   665    CALL TIMING
66F4 C5080A   666    $WHILE @PTCCIS .NOT. .DH. @PTLCIS                    1
66F7 6748
66F9 066752   667      CALL PHRASE                                        1
66FC 8E4C47   668      $IF @SPLFLG .EQ. 0 THEN                            2
66FF 11
6700 066859   669        CALL LOOKUP                                      2
6703 8F4D47   670        $IF @DATAAD .DEQ. 0 THEN                         3
6706 0C
6707 8E4C01   671          ST >1,@SPLFLG                                  3
670A 4711     672        $ELSE                                            3
670C 066968   673          CALL STDATA                                    3
670F 864B     674          CLR @TOTTIM                                    3
              675        $END
              676      $END
6711 D64C01   677      $IF @SPLFLG .EQ. 1 THEN                            2
6714 4743
6716 B04F10   678        DST @PTLCIP,@PTLCIL                              2
6719 8E483C   679        ST >3C,@TOTTIM                                   2
              680        $REPEAT
671C D6800C   681          $WHILE RAM(@PTFCIP) .EQ. SPACE                 4
671F 204729
6722 8E483C   682              ST >3C,@TOTTIM                             4
6725 910C     683              DINC @PTFCIP                               4
6727 471C     684          $END                                           3
6729 B0100C   685          DST @PTFCIP,@PTLCIP                            3
672C 066859   686          CALL LOOKUP                                    3
672F 8F4D47   687          $IF @DATAAD .DEQ. 0 THEN                       4
6732 37
6733 8F4071   688              DST >71F4,@DATAAD  Put addr of 'UHOH' in DATAAD4
6736 F4
              689          $END
6737 066968   690          CALL STDATA                                    3
673A 864B     691          CLR @TOTTIM                                    3
673C 910C     692          DINC @PTFCIP                                   3
673E C50C4F   693        $UNTIL @PTFCIP .DH. @PTLCIL                      2
6741 471C
              694      $END
6743 0667AD   695      CALL TIMING                                        1
6746 46F4     696    $END
6748 8E4867   697    $IF @TOTTIM .NE. 0 THEN                              1
674B 51
674C 8740     698      DCLR @DATAAD                                       1
```