```
SOURCE ACCESS NAME=      PCD2.BASLCP.SRC.INTSIM
OBJECT ACCESS NAME=      PCD2.BASLCP.OBJ.INTSIM
LISTING ACCESS NAME=     PCD2.BASLCP.LST.INTSIM
ERROR ACCESS NAME=       PCD2.BASLCP.LST.ERR
OPTIONS=                 BUNLST,TUNLST
MACRO LIBRARY PATHNAME=
```

```
0001                    IDT  'INTERP3'
0002              *
0003              *                  >>> GRAPHICS LANGUAGE INTERPRETER <<<      *
0004              ***************************************************************
0005              *                                                             *
0006              *          GRAPHICS PROGRAMMING LANGUAGE INTERPRETER          *
0007              *                                                             *
0008              ***************************************************************
0009              *          STATUS BLOCK CONTENTS RELATIVE TO PAD              *
0010              *       >70->71                    LAST VDP ADDRESS FROM MONITOR*
0011              *       >72          STKDAT        DATA STACK POINTER          *
0012              *       >73          STKADD        ADDRESS STACK POINTER       *
0013              *       >74          PLAYER        PLAYER NUMBER               *
0014              *       >75          KEYBRD        CODE FOR KEY PUSHED         *
0015              *       >76          JOYY          Y VALUE OF JOYSTICK         *
0016              *       >77          JOYX          X VALUE OF JOYSTICK         *
0017              *       >78          RANDOM        RANDOM NUMBER               *
0018              *       >79          TIME          TIMING VARIABLE             *
0019              *       >7A          MOTION        MOTION PARAMETER            *
0020              *       >7B          VDPST         VDP STATUS                  *
0021              *       >7C          STATUS        STATUS REGISTER             *
0022              *       >7D          CHRBUF        CHARACTER BUFFER            *
0023              *       >7E          YPT           Y POINTER IN VIDEO DISPLAY  *
0024              *       >7F          XPT           X POINTER IN VIDEO DISPLAY  *
0025              ***************************************************************
0026              *                  EXTERNAL DEFINITIONS
0027                        DEF   NEXT,VDPRD,VDPWD,WRVDP,VDPWA
0028                        DEF   GRMWA,GRMRA,GRMWD,GRMRD,H20
0029                        DEF   SET,RESET,ENTRY,PUTSTK,GETSTK
0030                        DEF   SROM,SGROM,C2,C1000,LNKVRM,LNKCRM
0031                        DEF   GETDA,GETMAB,SETV,SPGROM,HO300
0032                        DEF   RND,GPLLNK,XXML,VDPIO,CPUIO
0033                        REF   PARSEG,FLTTAB,WRITE,READ,TIMER,XTAB
0034                        REF   CONTG,EXECG,RTNG,VERIFY,CHTAB
0035                        REF   XTAB2,XTAB3,XTAB4,XTAB5
0036                        REF   XTAB6,XTAB7,XTAB8,XTAB9
0037                        REF   MAPX3,MAPX4,MAPX5,SCRNX
0038              *                  I/O DEVICE ADDRESSES
0039        8810  MAPPER  EQU   >8810                MAPPER COMMAND/STATUS PORT
0040        8000  MFO     EQU   >8000
0041        8000  VDPREG  EQU   >8000
0042        4000  WRVDP   EQU   >4000
0043        8C02  VDPWA   EQU   >8C02                LOAD VDP ADDRESS
0044        FC00  VDPSTA  EQU   >8802-VDPWA          READ VDP STATUS
0045        FFFE  VDPWD   EQU   >8C00-VDPWA          WRITE DATA TO VDP
0046        FBFE  VDPRD   EQU   >8800-VDPWA          READ DATA FROM VDP
0047        9800  GRMRD   EQU   >9800                READ GROM DATA
0048        0402  GRMWA   EQU   >9C02-GRMRD          LOAD GROM ADDRESS
0049        0002  GRMRA   EQU   >9802-GRMRD          READ GROM ADDRESS
0050        0400  GRMWD   EQU   >9C00-GRMRD          WRITE DATA TO GRAM
0051        8400  SGCADR  EQU   >8400                LOAD SOUND CHIP ADDRESS
0052        9400  SPKCMD  EQU   >9400                SPEECH WRITE
0053        9000  SPKSTA  EQU   >9000                SPEECH READ
0054              *---------------------------------------------------------------*
0055              *
0056              *                  STATUS BLOCK DEFINITIONS
0057        8300  PAD     EQU   >8300
0058        834A  FAC     EQU   PAD+>4A
0059        8355  SCLEN   EQU   PAD+>55
0060        8356  SCNAM   EQU   PAD+>56
```

```
0061   836C  TEMP2  EQU  PAD+>6C
0062   836D  TYPE   EQU  PAD+>6D
0063   8372  STKDAT EQU  PAD+>72
0064   8373  STKADD EQU  PAD+>73
0065   8374  PLAYER EQU  PAD+>74
0066   8375  KEYBRD EQU  PAD+>75
0067   8376  JOYY   EQU  PAD+>76
0068   8377  JOYX   EQU  PAD+>77
0069   8378  RANDOM EQU  PAD+>78
0070   8379  TIME   EQU  PAD+>79
0071   837A  MOTION EQU  PAD+>7A
0072   837B  VDPST  EQU  PAD+>7B
0073   837C  STATUS EQU  PAD+>7C
0074   837D  CHRBUF EQU  PAD+>7D
0075   837E  YPT    EQU  PAD+>7E
0076   837F  XPT    EQU  PAD+>7F
0077         *
0078         *              WORKSPACE DEFINITIONS
0079   83C0  INTWSP EQU  PAD+>C0          INTERRUPT WORKSPACE AREA
0080   83E0  GPLWS  EQU  PAD+>E0          RESERVE WORKSPACE AREA
0081         *
0082         *
0083         *              ABSOLUTE 9900 LOCATIONS
0084   83C0  RAND16 EQU  PAD+>C0      SEED FOR RANDOM NUMBER
0085   83C2  INTFLG EQU  PAD+>C2      Flags for VDP interrupt handler
0086   83C4  INTPTR EQU  PAD+>C4      Pointer to VDP interrupt handler
0087   83C6  KBDFLG EQU  PAD+>C6      KBD FLAG (0=99/4,1=PASCAL,2=TE II)
0088   83C7  OLDMOD EQU  PAD+>C7      LAST MODIFIER FLAGS
0089   83C8  DBNCE  EQU  PAD+>C8      POSITION DEBOUNCE FOR CONSOLE KBD
0090   83C9  DBNC1  EQU  PAD+>C9      POSITION DEBOUNCE FOR SPLIT KBD 1
0091   83CA  DBNC2  EQU  PAD+>CA      POSITION DEBOUNCE FOR SPLIT KBD 2
0092   83CB  SAVEG  EQU  PAD+>CB      SAVE GROM ADDRESS OF HEADER
0093   83CC  SNDADD EQU  PAD+>CC      SOUND LIST ADDRESS
0094   83CE  STFLGS EQU  PAD+>CE      NUMBER OF SOUND BYTES
0095   83D0  CRULST EQU  PAD+>D0
0096   83D2  SADDR  EQU  PAD+>D2
0097   83D4  SAVVDP EQU  PAD+>D4
0098   83D6  TIMOUT EQU  PAD+>D6
0099   83D8  RSAVE  EQU  PAD+>D8      SAVE R11 IN SCAN ROUTINE
0100         *      EQU  PAD+>DA
0101         *      EQU  PAD+>DC
0102         *      EQU  PAD+>DE
0103         *
0104         *              LOCATIONS OF LOW ORDER BYTES OF REGISTERS
0105   83E1  R0LB   EQU  PAD+>E1
0106   83E3  R1LB   EQU  PAD+>E3
0107   83E5  R2LB   EQU  PAD+>E5
0108   83E7  R3LB   EQU  PAD+>E7
0109   83E9  R4LB   EQU  PAD+>E9
0110   83EB  R5LB   EQU  PAD+>EB
0111   83ED  R6LB   EQU  PAD+>ED
0112   83F1  R8LB   EQU  PAD+>F1
0113   83EF  R7LB   EQU  PAD+>EF
0114   83F3  R9LB   EQU  PAD+>F3
0115   83F7  R11LB  EQU  PAD+>F7
0116   83F9  R12LB  EQU  PAD+>F9
0117   83D9  AR12LB EQU  PAD+>D9
0118   83FB  R13LB  EQU  PAD+>FB
0119         *
0120         **** NEW SCRATCHPAD DEFINITIONS
```

```
0121                    *
0122        8430    SCRLBF  EQU  >8430
0123        8458    SCRWS   EQU  SCRLBF+40
0124        8000    MAP0    EQU  >8000      MAP FILE 0
0125        8040    MAP1    EQU  >8040               1
0126        8080    MAP2    EQU  >8080               2
0127        80C0    MAP3    EQU  >80C0               3
0128        8410    MAPWS   EQU  >8410          WORKSPACE FOR MAPPER XOPS
0129        849E    DELAY1  EQU  >849E          60-CYCLE DELAY COUNTER FOR SPEED MAT
0130        849C    DELAY2  EQU  >849C          DELAY COUNTER FOR KBD DEBOUNCE
0131        849A    VDPSPD  EQU  >849A          50 (EUROPE) OR 60 (USA)
0132        849B    CLKSPD  EQU  >849B          SYSTEM CLK FREQ ON THE BUS
0133        84A4    GLINK0  EQU  >84A4          USED FOR GPL LINK FROM ASSEMBLY
0134        84A6    GLINK1  EQU  >84A6          USED FOR GPL LINK FROM ASSEMBLY
0135                ***CLICK  EQU  >84A8           keyboard click flag
0136                *--------------------------------------------------------------*
0137                *                SPECIAL EQUATE VALUES
0138        0000    MONIT   EQU >0000                    MONITOR START ADDRESS IN GROM
0139        83C3    ALPHLK  EQU >83C3
0140        000E    ROWBAS  EQU  14
0141        000C    JOYBAS  EQU  12
0142        0020    COLBAS  EQU  32
0143                ****BBJOY   EQU  MONIT+>16E0          Joystick tables
0144                ****KEYTAB  EQU  MONIT+>1700          Table of unmodified keycod
0145                ****KSHIFT  EQU  MONIT+>1734          TABLE OF SHIFTED KEYCODES
0146                ****KFNCTN  EQU  MONIT+>1768          Table of function keycodes
0147                ****KCNTRL  EQU  MONIT+>179C          Table of control keycodes
0148                ****KSPLIT  EQU  MONIT+>17D0          TABLE OF SPLIT KEYBOARD KE
0149        2000    VDELTA  EQU  >2000               >C0 - >E0 INVALID VERTICAL
0150                *****RSMOT   EQU  >0780               SPRITE MOTION LIST ADDRESS
0151                *****QSAML   EQU  >0480               SML - SAL
0152        84A0    SAL     EQU  >84A0              POINTER TO SPRITE ATTRIBUTE LIS
0153        84A2    SVL     EQU  >84A2              POINTER TO SPRITE VELOCITY  LIS
0154        2200    SHIFTQ  EQU  >2200              HHU SHIFT Q CODE
0155        400C    H400C   EQU  >400C              INTERRUPT ADDRESS VECTOR
0156        4000    H4000   EQU  >4000              ROM ID LOCATION
0157        0000    INTROM  EQU  >0000              CRU ADDRESS FOR PERIPH. ROM
0158        2800    DSRWSP  EQU  >2800              DSR WORKSPACE FOR 99/4 DEBUGGER
0159        280A    XOPWSP  EQU  >280A              XOP WORKSPACE FOR AL BPS
0160        2836    SSFLG   EQU  >2836              SINGLE STEP FLAG
0161        2838    MALPC   EQU  >2838              MODIFIED ALPC FLAG
0162        4024    SSTEP   EQU  >4024              DSR ENTRY POINT FOR SNGL STEP
0163        4028    ALBRK   EQU  >4028              DSR ENTRY POINT AFTER AL BP
```

```
0165                    *--------------------------------------------------------------*
0166                    *          ***** ENTRY POINT OF EMULATOR *****                 *
0167                    *
0168 0000                       RORG 0
0169 0000 83E0                  DATA GPLWS,ENTRY      RESET VECTOR                  >00
     0002 0D2A'
0170 0004 83C0                  DATA INTWSP,REMOTE     ALL 9901 INTERRUPTS           >0
     0006 0840'
0171 0008 83C0                  DATA INTWSP,NULLRT     UNIMPLEMENTED INSTRUCTION     >0
     000A 0A26'
0172                    *                              (OR ARITHMETIC OVERFLOW)
0173                    *
0174 000C   80          IDBYTE  BYTE >80               (WAS CLOCK, NOW CONSOLE MODEL)
0175 000D   AA          HAA     BYTE >AA               VALID ROM ID                  >0
0176                    ********************************************************************
0177                    *       CALL SCAN FROM ROM                                         *
0178                    ********************************************************************
0179 000E 0460  0566'   SCNKEY  B     @KSCAN         ALWAYS BL HERE FROM ANY ROM   >00
0180                    ********************************************************************
0181 0012 0008          BIT12   DATA >0008                                           >001
0182                    *
0183                    **      BRANCH TO RETURN FROM BREAKPOINT FROM 99/4 DSR
0184                    *
0185 0014 1E00                  SBZ   0               DISABLE 99/4 DSR
0186 0016 0460  007A'           B     @PROCSS         RE-ENTRY ON BP IN 99/4
0187 001A 1E00                  SBZ   0               DISABLE 99/4 DSR
0188 001C 0460  0078'           B     @TRYAGN         RE-ENTRY ON ILLEGAL OPCODE IN 99/4
0189                    ********************************************************************
0190 0020 0460  0294'           B     @CHKBRK         CHECK FOR BASIC & RS232 "BREAK" KEY
0191                    ********************************************************************
0192 0024 0460  0C02'   GPLLNK  B     @GPLLKO                                       >0
0193                    ********************************************************************
```

```
0195                *-------------------------------------------------------------*
0196                *              000     001     010     011     100
0197                *MODTAB DATA KEYTAB,KSHIFT,KCNTRL,KCNTRL,KFNCTN              >002
0198 0028 0F3E'     RMDTAB DATA RKEYTB,RKSHFT,RKCNTL,RKCNTL,RKFCTN                **
     002A 0F72'
     002C 0FDA'
     002E 0FDA'
     0030 0FA6'
0199                *              101     110     111
0200                *       DATA KFNCTN,KFNCTN,KFNCTN                           >0
0201 0032 0FA6'          DATA RKFCTN,RKFCTN,RKFCTN                              **
     0034 0FA6'
     0036 0FA6'
```

```
0203                    *===============================================================*
0204 0038 1E00                  SBZ  0                  DISABLE 99/4 DSR              >
0205 003A 02E0                  LWPI XOPWSP             SELECT CORRECT WORKSPACE      >
     003C 280A
0206 003E 0380                  RTWP                    RETURN TO AL PROGRAM          >
0207 0040 280A                  DATA XOPWSP,PRCXOP      XOP VECTOR FOR AL BP  XOP0    >
     0042 0BEE'
0208                    * PASCAL TIBUG XOP
0209 0044 FFD8                  DATA >FFD8,>FFF8                              XOP1    >0
     0046 FFF8
0210                    * A SPARE XOP JUST FOR FUN
0211 0048 83A0                  DATA >83A0,>8300                              XOP2    >0
     004A 8300
0212                    * MEMORY MAPPER XOPS
0213 004C 8410                  DATA MAPWS,MAPX3                              XOP3    >0
     004E 0000
0214 0050 8410                  DATA MAPWS,MAPX4                              XOP4    >0
     0052 0000
0215 0054 8410                  DATA MAPWS,MAPX5                              XOP5    >0
     0056 0000
0216                    * SCREEN HANDLER XOPS
0217 0058 8458                  DATA SCRWS,SCRNX                              XOP6    >0
     005A 0000
0218 005C 1000  C1000   DATA >1000              PRESERVE ADDRESSES                    >0
0219      005F' HOB     EQU  $+1
0220 005E 070B  FNCEQ   DATA >070B    TEST BITS FOR QUIT KEY                        >0
0221 0060 D846  LDKADD  MOVB R6,@GRMWA(R13)                                         >0
     0062 0402
0222 0064 DB60          MOVB @R6LB,@GRMWA(R13)  LOAD GROM ADDRESS                   >0
     0066 83ED
     0068 0402
0223 006A 5820  RESET   SZCB @BIT2,@STATUS      RESET CONDITION BIT               >00
     006C 0119'
     006E 837C
0224      0072' IN2     EQU  $+2
0225      0072' C2      EQU  IN2
0226      0073' RMAP1   EQU  $+3
0227      0070' H0300   EQU  $              Allows squish of 1 word in SCREEN.
0228 0070 0300  NEXT    LIMI 2              ALLOW INTERRUPTS BETWEEN
     0072 0002
0229      0074' LMAP1   EQU  $
0230      0074' H03     EQU  $
0231 0074 0300          LIMI 0              GPL INSTRUCTIONS -
     0076 0000
0232 0078 D25D  TRYAGN  MOVB *R13,R9        LOAD INSTRUCTION FROM GAME ROM
0233 007A 1105  PROCSS  JLT  ABOPS          JUMP ON SIGN BIT
0234 007C D109          MOVB R9,R4          MOVE INST INTO WORK REGISTER
0235 007E 09C4          SRL  R4,12          SHIFT TO LEAVE TOP 3 BITS * 2
0236 0080 C164          MOV  @ITAB(R4),R5 GET BRANCH ADDRESS
     0082 0C36'
0237 0084 0455          B    *R5            JUMP ON TOP THREE BITS
0238                    *---------------------------------------------------------------*
0239                    *          INSTRUCTIONS WITH A AND B OPERANDS
0240 0086 04C4  ABOPS   CLR  R4             CLEAR VDP RAM FLAGS
0241 0088 C149          MOV  R9,R5          LOAD R5 WITH DOUBLE FLAG
0242 008A 0245          ANDI R5,>100
     008C 0100
0243 008E 06A0          BL   @GETMAD        GET FIRST OPERAND
     0090 03CC'
0244 0092 06C4          SWPB R4
```

```
0245 0094 C0C1          MOV  R1,R3          SAVE VARIABLE ADDRESS
0246 0096 C080          MOV  R0,R2          SAVE VARIABLE VALUE
0247 0098 0289          CI   R9,>A000       SINGLE OPERAND ?
     009A A000
0248 009C 1A09          JL   AOPS           YES
0249 009E 2260          COC  @IMMOPS,R9     IMMEDIATE OR VARIABLE ?
     00A0 0D7C'
0250 00A2 160C          JNE  ABOPA          VARIABLE
0251 00A4 C04D          MOV  R13,R1         GET  IMMEDIATE VALUE
0252 00A6 D011          MOVB *R1,R0         GET FIRST BYTE
0253 00A8 0601          DEC  R1             MODIFY FOR COMMON ROUTINE
0254 00AA 06A0          BL   @MADC2         GO TO COMMON ROUTINE
     00AC 03FC'
0255 00AE 1008          JMP  BOPS
0256 00B0 C209  AOPS    MOV  R9,R8          BRANCH THROUGH BRANCH TABLE
0257 00B2 0988          SRL  R8,8
0258 00B4 0700          SETO R0
0259 00B6 C228          MOV  @ATAB(R8),R8
     00B8 0BFE'
0260 00BA 0458          B    *R8
0261 00BC 06A0  ABOPA   BL   @GETMAD        GET SECOND VARIABLE OPERAND
     00BE 03CC'
0262 00C0 C209  BOPS    MOV  R9,R8          BRANCH THROUGH BRANCH TABLE
0263 00C2 0998          SRL  R8,9
0264 00C4 C228          MOV  @BTAB(R8),R8 LOAD BRANCH ADDRESS
     00C6 0C4E'
0265 00C8 8002  G       C    R2,R0          COMPARE FOR IF PRIMITIVES
0266 00CA 0458          B    *R8
0267            *-------------------------------------------------------------*
0268            *                PERFORM IF COMPARISONS
0269 00CC 11CE  GREQ    JLT  RESET          A ALGEBRAIC .GE. B
0270 00CE F820  SET     SOCB @BIT2,@STATUS      SET CONDITION BIT
     00D0 0119'
     00D2 837C
0271 00D4 10CD          JMP  NEXT
0272 00D6 1BFB  HIGH    JH   SET            A LOGICAL .GT. B
0273 00D8 10C8          JMP  RESET
0274 00DA 14F9  HIGHEQ  JHE  SET            A LOGICAL .GE. B
0275 00DC 10C6          JMP  RESET
0276 00DE 15F7  GREATR  JGT  SET            A ALGEBRAIC .GT. B
0277 00E0 10C4          JMP  RESET
0278 00E2 0540  IFAND   INV  R0             PERFORM LOGICAL AND
0279 00E4 4080          SZC  R0,R2
0280 00E6 13F3          JEQ  SET
0281 00E8 10C0          JMP  RESET
0282 00EA C082  IFZ     MOV  R2,R2          COMPARE TO ZERO
0283 00EC 02C4  EQUAL   STST R4             STORE STATUS IN STATUS BYTE
0284 00EE D804          MOVB R4,@STATUS
     00F0 837C
0285 00F2 10BE          JMP  NEXT
0286 00F4 C009  TSTST   MOV  R9,R0          MOVE INST TO SHIFT REGISTER
0287 00F6 0AC0          SLA  R0,12          REMOVE HIGH ORDER BITS
0288 00F8 09D0          SRL  R0,13          POSITION 3 REMAINING BITS
0289 00FA D160          MOVB @STATUS,R5     LOAD STATUS BYTE
     00FC 837C
0290 00FE 0A05          SLA  R5,R0          SHIFT TO TEST BIT
0291 0100 18E6          JOC  SET
0292 0102 10B3          JMP  RESET
0293            *-------------------------------------------------------------*
0294            *          BRANCH INSTRUCTIONS
```

```
0295 0104 D19D  BLONG  MOVB *R13,R6       RECALL ADDRESS FROM GROM
0296 0106 D81D         MOVB *R13,@R6LB
     0108 83ED
0297 010A 10AA         JMP  LDKADD        LOAD GAME ROM ADDRESS
0298 010C D120  BSET   MOVB @STATUS,R4    IS CONDITION SET ?
     010E 837C
0299 0110 0A24         SLA  R4,2
0300 0112 1106         JLT  BRAD          YES
0301 0114 D11D  NOBR   MOVB *R13,R4       INCREMENT PROGRAM COUNTER
0302 0116 10A9         JMP  RESET
0303      0119' BIT2   EQU  $+1           CONSTANT >20
0304 0118 D120  BRESET MOVB @STATUS,R4    IS CONDITION SET ?
     011A 837C
0305 011C 0A24         SLA  R4,2
0306 011E 11FA         JLT  NOBR          YES
0307 0120 D81D  BRAD   MOVB *R13,@R9LB    GET SECOND BYTE OF ADDRESS
     0122 83F3
0308      0126' H1FFF  EQU  $+2
0309 0124 0249         ANDI R9,>1FFF      MASK TO LEAVE BRANCH ADDRESS
     0126 1FFF
0310 0128 D1AD         MOVB @GRMRA(R13),R6    READ OLD MSB OF GROM ADDRES
     012A 0002
0311 012C 0246         ANDI R6,>E000      LEAVE TOP 3 BITS OF ADDRESS
     012E E000
0312 0130 E189         SOC  R9,R6         COMBINE TO GIVE NEW ADDRESS
0313 0132 D02D         MOVB @GRMRA(R13),R0        get second byte for spec
     0134 0002
0314 0136 1094         JMP  LDKADD
0315              *--------------------------------------------------------------*
0316              *          SINGLE OPERAND INSTRUCTIONS
0317 0138 0742  ABS    ABS  R2             ABSOLUTE VALUE
0318 013A 1079         JMP  TRAP
0319 013C 0502  NEG    NEG  R2             NEGATE VALUE
0320 013E 1077         JMP  TRAP
0321 0140 0702  CLR    SETO R2             SET TO ZERO
0322 0142 0542  INV    INV  R2             INVERT VALUE
0323 0144 1074         JMP  TRAP
0324 0146 C184  FETCH  MOV  R4,R6          SAVE ADDRESSING MODE FLAG
0325 0148 06A0         BL   @PUTSTK        SAVE CURRENT PROGRAM ADDRESS
     014A 048C'
0326 014C 0644         DECT R4             SET I TO OLD STACK POINTER
0327 014E 06A0         BL   @GTSTK         GET ADDRESS FROM STACK TOP
     0150 0498'
0328 0152 D09D         MOVB *R13,R2        LOAD BYTE FROM GROM
0329 0154 0882         SRA  R2,8           EXTEND SIGN
0330 0156 05A4         INC  @PAD(R4)       INCREMENT RETURN ADDRESS
     0158 8300
0331 015A 05C4         INCT R4             SET I BACK TO NEW STACK POINTER
0332 015C 06A0         BL   @GTSTK1        RESTORE GROM ADDRESS
     015E 049C'
0333 0160 C106         MOV  R6,R4          RESTORE ADDRESSING MODE FLAG
0334 0162 1065         JMP  TRAP
0335 0164 0602  CASE   DEC  R2             VARIABLE DISPLACEMENT
0336 0166 1781         JNC  RESET
0337 0168 D15D         MOVB *R13,R5        SKIP TO NEW ADDRESS
0338 016A D15D         MOVB *R13,R5        SKIP TO NEW ADDRESS
0339 016C 10FB         JMP  CASE
0340 016E B80E  PUSH   AB   R14,@STKDAT    LOAD STACK ADDRESS  R14=01XX
     0170 8372
0341 0172 D1A0         MOVB @STKDAT,R6
```

```
      0174 8372
0342  0176 0986          SRL   R6,8
0343  0178 D9A0          MOVB  @R2LB,@PAD(R6)      PUSH LSBYTE
      017A 83E5
      017C 8300
0344  017E 0460          B     @NEXT
      0180 0070'
0345                     *--------------------------------------------------------------*
0346                     *          TWO OPERAND INSTRUCTIONS
0347  0182 09E0  DECT    SRL   R0,14         DECREMENT VALUE BY 2
0348  0184 0600  INCT    DEC   R0            INCREMENT INSTRUCTIONS
0349  0186       INC
0350  0186 0500  MINUS   NEG   R0            SUBTRACT OPERATION
0351  0188       DEC
0352  0188 D145  ADD     MOVB  R5,R5         SINGLE OR DOUBLE
0353  018A 134B          JEQ   ADDB
0354  018C A080          A     R0,R2         ADD OPERATION
0355  018E 104C          JMP   FILLST
0356  0190 0540  AND     INV   R0            LOGICAL AND OPERATION
0357  0192 4080          SZC   R0,R2
0358  0194 1049          JMP   FILLST
0359  0196 E080  OR      SOC   R0,R2         LOGICAL OR OPERATION
0360  0198 1047          JMP   FILLST
0361  019A 2880  XOR     XOR   R0,R2         LOGICAL EXCLUSIVE OR OPERATION
0362  019C 1045          JMP   FILLST
0363  019E C080  STORE   MOV   R0,R2         STORE SOURCE INTO DESTINATION
0364  01A0 1046          JMP   TRAP
0365  01A2 C242  EXCH    MOV   R2,R9         EXCHANGE A AND B
0366  01A4 C080          MOV   R0,R2
0367  01A6 06A0          BL    @TRAPA
      01A8 0232'
0368  01AA 06C4          SWPB  R4
0369  01AC C0C1          MOV   R1,R3
0370  01AE 101B          JMP   MUL2
0371  01B0 0802  SRA     SRA   R2,R0         SHIFT RIGHT ARITHMETIC
0372  01B2 103D          JMP   TRAP
0373  01B4 0A02  SLL     SLA   R2,R0
0374  01B6 103B          JMP   TRAP
0375  01B8 D145  SRL     MOVB  R5,R5         SHIFT RIGHT LOGICAL
0376  01BA 1601          JNE   SRL1          DOUBLE INSTRUCTION
0377  01BC 7082          SB    R2,R2
0378  01BE 0902  SRL1    SRL   R2,R0
0379  01C0 1036          JMP   TRAP
0380  01C2 D145  SRC     MOVB  R5,R5         SHIFT RIGHT CIRCULAR
0381  01C4 1602          JNE   SRC1          DOUBLE INSTRUCTION
0382  01C6 D0A0          MOVB  @R2LB,R2
      01C8 83E5
0383  01CA 0B02  SRC1    SRC   R2,R0
0384  01CC 1030          JMP   TRAP
0385  01CE C202  MUL     MOV   R2,R8         MULTIPLY
0386  01D0 D145          MOVB  R5,R5         SINGLE OR DOUBLE
0387  01D2 1601          JNE   MUL0          DOUBLE
0388  01D4 7208          SB    R8,R8         CLEAR EXTENDED SIGN BITS
0389  01D6 3A00  MUL0    MPY   R0,R8
0390  01D8 D145          MOVB  R5,R5         SINGLE OR DOUBLE ?
0391  01DA 1602          JNE   MUL1          DOUBLE
0392  01DC D809          MOVB  R9,@R8LB
      01DE 83F1
0393  01E0 C088  MUL1    MOV   R8,R2         STORE FIRST HALF OF RESULT
0394  01E2 06A0          BL    @TRAPA
```

```
      01E4 0232'
0395  01E6 C089  MUL2   MOV  R9,R2          STORE SECOND HALF OF RESULT
0396  01E8 1022        JMP  TRAP
0397  01EA D805  DIV    MOVB R5,@STATUS     CLEAR STATUS
      01EC 837C
0398  01EE C202        MOV  R2,R8
0399  01F0 C080        MOV  R0,R2
0400  01F2 C043        MOV  R3,R1          SET DESTINATION ADDRESS
0401  01F4 0581        INC  R1
0402  01F6 D145        MOVB R5,R5          SINGLE OR DOUBLE ?
0403  01F8 1301        JEQ  DIV1           SINGLE
0404  01FA 0581        INC  R1
0405  01FC D104  DIV1   MOVB R4,R4          CPU OR VDP ?
0406  01FE 1303        JEQ  DIVC           CPU
0407  0200 06A0        BL   @MVDPA         GET REST OF DIVIDEND FROM VDP
      0202 044C'
0408  0204 1002        JMP  DIVB
0409  0206 06A0  DIVC   BL   @MADC          GET REST OF DIVIDEND FROM CPU
      0208 03FA'
0410  020A C240  DIVB   MOV  R0,R9
0411  020C D145        MOVB R5,R5          SINGLE OR DOUBLE ?
0412  020E 1603        JNE  DIV2
0413  0210 D260        MOVB @R8LB,R9       DO SIGN EXTENSION FOR BYTE
      0212 83F1
0414  0214 0888        SRA  R8,8           OPERANDS
0415  0216 3E02  DIV2   DIV  R2,R8          PERFORM DIVISION
0416  0218 19E3        JNO  MUL1
0417  021A F820        SOCB @BIT4,@STATUS     SET DIVIDE OVERFLOW
      021C 1016'
      021E 837C
0418  0220 10DF        JMP  MUL1
0419               *                FILL IN STATUS BITS
0420  0222 B820  ADDB   AB   @R0LB,@R2LB
      0224 83E1
      0226 83E5
0421  0228 02CB  FILLST STST R11           STORE STATUS WORD
0422  022A D80B        MOVB R11,@STATUS    STORE STATUS BITS
      022C 837C
0423               *
0424               *              *** TRAP OUT CHANGES IN CB, XPT, AND YPT ***
0425               *
0426  022E 020B  TRAP   LI   R11,NEXT       LOAD R11 TO RETURN TO NEXT
      0230 0070'
0427  0232 D104  TRAPA  MOVB R4,R4          DESTINATION IS CPU OR VDP ?
0428  0234 130F        JEQ  STCPU          CPU
0429  0236 D7E0        MOVB @R3LB,*R15     LOAD VDP ADDRESS
      0238 83E7
0430  023A 0263        ORI  R3,WRVDP       SET BIT TO WRITE TO VDP
      023C 4000
0431  023E D7C3        MOVB R3,*R15
0432  0240 D145        MOVB R5,R5          SINGLE OR DOUBLE
0433  0242 1303        JEQ  TRAP1          SINGLE
0434  0244 DBC2        MOVB R2,@VDPWD(R15) MOVE 2 BYTES TO VDP
      0246 FFFE
0435  0248 0583        INC  R3
0436  024A DBE0  TRAP1  MOVB @R2LB,@VDPWD(R15) MOVE 1 BYTE TO VDP
      024C 83E5
      024E FFFE
0437  0250 0583        INC  R3
0438       0252' H04    EQU  $
```

```
0439 0252 045B  TEXIT  RT
0440 0254 D145  STCPU  MOVB R5,R5          SINGLE OR DOUBLE ?
0441 0256 1301         JEQ  TRAP2          SINGLE
0442 0258 DCC2         MOVB R2,*R3+        STORE 2 BYTE RESULT
0443 025A 06C2  TRAP2  SWPB R2
0444 025C DCC2         MOVB R2,*R3+        STORE 1 BYTE RESULT
0445 025E 0283         CI   R3,PAD+>7E     IS CB THE DESTINATION ?
     0260 837E
0446 0262 16F7         JNE  TEXIT          NO
0447 0264 C18B         MOV  R11,R6
0448 0266 06A0         BL   @GETDA         GET VDP DISPLAY ADDRESS
     0268 04D8'
0449 026A DBC2  RTN    MOVB R2,@VDPWD(R15) WRITE CHARACTER TO VDP
     026C FFFE
0450 026E 0456         B    *R6            RETURN
0451                   *---------------------------------------------------------------*
0452                   *          MISCELLANEOUS INSTRUCTIONS
0453 0270 0A39  MISCLN SLA  R9,3           REMOVE TOP THREE BITS OF INST
0454 0272 09A9         SRL  R9,10          LEAVE BOTTOM FIVE BITS * 2
0455 0274 C129         MOV  @MSCTAB(R9),R4  GET BRANCH ADDRESS
     0276 0C3E'
0456 0278 0454         B    *R4
0457                   *
0458                   *---------------------------------------------------------------*
0459                   *          GENERATE A RANDOM NUMBER
0460 027A D19D  RANDNO MOVB *R13,R6        LOAD MODULO VALUE
0461 027C 0986         SRL  R6,8           SHIFT TO LOW ORDER POSITION
0462 027E 06A0         BL   @RND
     0280 0BBE'
0463 0282 1006         JMP  BKGR1
0464                   *
0465                   *          SET BACKGROUND COLOR
0466 0284 0207  BKGRND LI   R7,>700+VDPREG      CHANGE BACKGROUND COLOR
     0286 8700
0467 0288 D81D         MOVB *R13,@R7LB     IMMEDIATE  OPERAND
     028A 83EF
0468 028C 06A0         BL   @SETVDP        OUTPUT ADDRESS TO VDP
     028E 04F2'
0469 0290 0460  BKGR1  B    @NEXT
     0292 0070'
```

```
0471                    *
0472                    *** Check for the BASIC and RS232 "BREAK" key
0473                    *
0474 0294 020C  CHKBRK  LI    R12,COLBAS        Select output for rows
     0296 0020
0475                    ***   LDCR  @H00,4      Select line 0
0476 0298 310C          LDCR  R12,4             Select line 0
0477 029A 0B7C          SRC   R12,7             Kill some time
0478 029C 020C          LI    R12,ROWBAS        Select input for columns
     029E 000E
0479 02A0 350C          STCR  R12,4             Read data
0480 02A2 9320          CB    @FNCEQ+1,R12      Is proper key down? (fctn)
     02A4 005F'
0481 02A6 160A          JNE   CHKBEX            No so exit
0482 02A8 020C          LI    R12,COLBAS        Select output for rows
     02AA 0020
0483 02AC 3120          LDCR  @H04,4            Select '4' line
     02AE 0252'
0484 02B0 0B7C          SRC   R12,7             Kill some time
0485 02B2 020C          LI    R12,ROWBAS        Select input for columns
     02B4 000E
0486 02B6 350C          STCR  R12,4             Read data
0487 02B8 9320          CB    @FNCEQ,R12        Is proper key down? (4)
     02BA 005E'
0488 02BC 045B  CHKBEX  RT                      Return condition to the caller
```

```
0490                    *=============================================== ================*
0491                    *          INPUT / OUTPUT INSTRUCTION
0492 02BE C080  INOUT   MOV  R0,R2          GET INDEX INTO I/O TABLE
0493 02C0 C043          MOV  R3,R1          MOVE LIST ADDRESS FOR CASSETTE
0494 02C2 A082          A    R2,R2
0495 02C4 C122          MOV  @IOTAB(R2),R4      LOAD ADDRESS OF I/O FUNCTION
     02C6 0CEC'
0496 02C8 04C9          CLR  R9             CLEAR R9 FOR CRU INSTRUCTIONS
0497 02CA 0454          B    *R4            BRANCH TO I/O FUNCTION
0498                    *          SOUND INSTRUCTION
0499 02CC 024E  SOUND   ANDI R14,>FFFE      CLEAR SOUND SOURCE FLAG
     02CE FFFE
0500 02D0 E380          SOC  R0,R14         SET SOUND SOURCE FLAG
0501 02D2 C813          MOV  *R3,@SNDADD    LOAD SOUND LIST ADDRESS
     02D4 83CC
0502 02D6 D80E          MOVB R14,@STFLGS    SET TIME DELAY TO 1 UNIT R14=01XX
     02D8 83CE
0503 02DA 10DA  QEXIT   JMP  BKGR1
0504                    *          CRU INSTRUCTIONS                          *
0505 02DC 0589  CRUIN   INC  R9
0506 02DE C331  CRUOUT  MOV  *R1+,R12       LOAD CRU BASE ADDRESS
0507 02E0 A30C          A    R12,R12
0508 02E2 04C2          CLR  R2
0509 02E4 D0B1          MOVB *R1+,R2        LOAD NUMBER OF BITS TO MOVE
0510 02E6 0A42          SLA  R2,4
0511 02E8 E242          SOC  R2,R9          COMBINE NUMBER AND IN/OUT FLAG
0512 02EA 0B69          SRC  R9,6           REPOSITION BITS
0513 02EC 0269          ORI  R9,>3012       R9 IS NOW LDCR OR STCR
     02EE 3012
0514 02F0 D091          MOVB *R1,R2         LOAD CPU SOURCE/DEST ADDRESS
0515 02F2 06C2          SWPB R2
0516 02F4 0222          AI   R2,PAD
     02F6 8300
0517 02F8 0489          X    R9             EXECUTE INSTRUCTION IN R9
0518 02FA 10EF          JMP  QEXIT
```

```
0520                    *=============================================================*
0521                    *          MOVE DATA INSTRUCTION
0522 02FC D14E  MOVDAT  MOVB  R14,R5        SET DOUBLE FLAG
0523 02FE 0999          SRL   R9,9          IS LENGTH VAR OR IMMEDIATE ?
0524 0300 1804          JOC   LIMM          IMMEDIATE
0525 0302 06A0          BL    @GETMAD       GET VARIABLE LENGTH
     0304 03CC'
0526 0306 C200          MOV   R0,R8
0527 0308 1003          JMP   MVSRC
0528 030A D21D  LIMM    MOVB  *R13,R8       LOAD IMMEDIATE LENGTH
0529 030C D81D          MOVB  *R13,@R8LB
     030E 83F1
0530 0310 04C4  MVSRC   CLR   R4
0531 0312 0AC9          SLA   R9,12
0532 0314 06A0          BL    @MVADDR       GET DEST ADDRESS
     0316 03AA'
0533 0318 C081          MOV   R1,R2         SAVE DEST ADDRESS
0534 031A B249          AB    R9,R9         WAS DEST VDP REGISTER ?
0535 031C 1702          JNC   DTYPE         NO
0536 031E 0224          AI    R4,3          DESTINATION IS VDP REGISTERS
     0320 0003
0537 0322 C1C4  DTYPE   MOV   R4,R7         SAVE DEST TYPE
0538 0324 04C4          CLR   R4
0539 0326 06A0          BL    @MVADDR       GET SOURCE ADDRESS
     0328 03AA'
0540                    *          LOAD BRANCH REGISTERS
0541 032A A104          A     R4,R4         DOUBLE R4
0542 032C C1A4          MOV   @MSRC(R4),R6 LOAD ADDRESS TO SOPURCE OF MOVE
     032E 0CCE'
0543 0330 A1C7          A     R7,R7         DOUBLE R7
0544 0332 C1E7          MOV   @MDST(R7),R7 LOAD ADDRESS TO DESTINATION
     0334 0CD4'
0545 0336 06A0          BL    @PUTSTK       SAVE PROGRAM COUNTER
     0338 04BC'
0546 033A 0456  BSRC    B     *R6           BRANCH TO SOURCE
0547                    *          LOAD ONE BYTE FROM THE SOURCE OF THE MOVE
0548 033C D2F1  CPUSRC  MOVB  *R1+,R11      LOAD BYTE FROM CPU ADDR SPACE
0549 033E 0457          B     *R7
0550 0340 D7E0  VDPSRC  MOVB  @R1LB,*R15    LOAD VDP WITH ADDRESS
     0342 83E3
0551 0344 D7C1          MOVB  R1,*R15
0552 0346 0581          INC   R1            INCREMENT SOURCE ADDRESS
0553 0348 D2EF          MOVB  @VDPRD(R15),R11 LOAD BYTE FROM VDP
     034A F8FE
0554 034C 0457          B     *R7
0555 034E DB41  GRMSRC  MOVB  R1,@GRMWA(R13) LOAD GROM WITH ADDRESS
     0350 0402
0556 0352 DB60          MOVB  @R1LB,@GRMWA(R13)
     0354 83E3
     0356 0402
0557 0358 0581          INC   R1            INCREMENT SOURCE ADDRESS
0558 035A D2DD          MOVB  *R13,R11 LOAD BYTE FROM VDP
0559 035C 0457          B     *R7
0560                    *          DESTINATIONS FOR THE MOVE INSTRUCTION
0561 035E DC8B  CPUDST  MOVB  R11,*R2+      STORE BYTE IN CPU ADDR SPACE
0562 0360 1020          JMP   TESTLP
0563 0362 DB42  GRMDST  MOVB  R2,@GRMWA(R13)
     0364 0402
0564 0366 DB60          MOVB  @R2LB,@GRMWA(R13)
     0368 83E5
```

```
      036A 0402
0565  036C 0582          INC  R2
0566  036E DB4B          MOVB R11,@GRMWD(R13)
      0370 0400
0567  0372 1017          JMP  TESTLP
0568  0374 93A0  REGDST  CB   @R2LB,R14
      0376 83E5
0569  0378 1604          JNE  REGADJ
0570               ***   ALWAYS 16K MEMORY
0571               ***    COC @BIT12,R14    TEST VDP MEMORY TYPE
0572               ***    JNE REG100
0573       037C' H8000   EQU $+2
0574  037A 026B          ORI  R11,>8000    SET 16K MEMORY SELECT
      037C 8000
0575  037E D80B  REG100  MOVB R11,@SAVVDP  SAVE VDP REGISTER
      0380 83D4
0576  0382 D7CB  REGADJ  MOVB R11,*R15     STORE BYTE IN VDP REGISTER
0577  0384 C0C2          MOV  R2,R3
0578  0386 0263          ORI  R3,>80       SET FLAG FOR VDP REGISTER LOAD
      0388 0080
0579  038A D7E0          MOVB @R3LB,*R15   STORE REGISTER NUMBER
      038C 83E7
0580               ****   ANDI R2,7         IN CASE VR0 WAS FIRST, SO WE FIND VR
0581  038E 0582          INC  R2           INCREMENT REGISTER NUMBER
0582  0390 1008          JMP  TESTLP
0583  0392       VDPDST
0584  0392 D7E0          MOVB @R2LB,*R15
      0394 83E5
0585  0396 0262          ORI  R2,WRVDP     SET FLAG TO WRITE TO VDP
      0398 4000
0586  039A D7C2          MOVB R2,*R15
0587  039C 0582          INC  R2           INCREMENT DESTINATION ADDRESS
0588  039E DBCB          MOVB R11,@VDPWD(R15) STORE BYTE IN VDP
      03A0 FFFE
0589               *         TEST END OF LOOP FOR THE MOVE INSTRUCTION
0590  03A2 0608  TESTLP  DEC  R8           DECREMENT NUMBER TO MOVE
0591  03A4 15CA          JGT  BSRC         LOOP UNTIL NUMBER IS ZERO
0592  03A6 0460  TSTRTN  B    @RETNC       RESTORE PROGRAM COUNTER
      03A8 048E'
```

```
0594                    *===============================================================*
0595                    *          COMPUTE ADDRESS FOR MOVE INSTRUCTION
0596 03AA B249  MVADDR  AB    R9,R9             IS ADDRESS IN GROM ?
0597 03AC 180F          JOC   GETMAD            YES
0598 03AE D0DD  INGROM  MOVB  *R13,R3           LOAD GROM ADDRESS
0599 03B0 0584          INC   R4                SET GROM FLAG
0600 03B2 D81D          MOVB  *R13,@R3LB
     03B4 83E7
0601 03B6 C30B          MOV   R11,R12
0602 03B8 B249          AB    R9,R9             IS GROM ADDRESS INDEXED ?
0603 03BA 1704          JNC   NOGNDX            NO
0604 03BC D05D          MOVB  *R13,R1           INDEX IS ONLY ONE BYTE
0605 03BE 06A0          BL    @MADDA            GET INDEX VALUE
     03C0 03D0'
0606 03C2 A0C0          A     R0,R3             ADD INDEX TO ADDRESS
0607 03C4 C043  NOGNDX  MOV   R3,R1             RETURN ADDRESS IN R1
0608 03C6 0919          SRL   R9,1
0609 03C8 045C          B     *R12              RETURN
0610                    *===============================================================*
0611                    *          GLOBAL ADDRESS DECODE:
0612                    *               R1 = ADDRESS OF VARIABLE
0613                    *               R0 = VALUE OF VARIABLE
0614                    *               SINGLE BYTE VALUES ARE SIGN EXTENDED
0615                    *               IF ADDRESS IS IN VDP, THEN THE MSBYTE OF R4
0616                    *                 IS SET TO >80
0617 03CA 04C5  GETMAB  CLR   R5                BYTE FLAG
0618 03CC D05D  GETMAD  MOVB  *R13,R1           GET NEXT BYTE FROM GAME ROM
0619 03CE 111E          JLT   MLONG             LONG ADDRESS
0620 03D0 0981  MADDA   SRL   R1,8              SINGLE BYTE ADDRESS
0621 03D2 0221  MADD    AI    R1,PAD            ADD CPU RAM ADDRESS
     03D4 8300
0622 03D6 0281          CI    R1,PAD+>7D        CHARACTER BUFFER ?
     03D8 837D
0623 03DA 160F          JNE   MADC
0624                    *          LOAD CB FROM VDP MEMORY
0625 03DC 04CA          CLR   R10               SET FLAG TO READ FROM VDP
0626 03DE C18B          MOV   R11,R6            SAVE RETURN ADDRESS
0627 03E0 06A0          BL    @GETDAD           SET VPD RAM ADDRESS
     03E2 04DC'
0628 03E4 C2C6          MOV   R6,R11            RESTORE RETURN ADDRESS
0629 03E6 D02F          MOVB  @VDPRD(R15),R0 LOAD CB FROM DISPLAY
     03E8 F8FE
0630 03EA 23A0          COC   @IN2,R14          IS IT MULTICOLOR MODE??????????
     03EC 0072'
0631 03EE 1603          JNE   MADE              NO, WHO CARES???
0632 03F0 1701          JNC   $+4               WHICH NYBBLE????
0633 03F2 0A40          SLA   R0,4              LOW NYBBLE
0634 03F4 0940          SRL   R0,4              HIGH NYBBLE
0635 03F6 D800  MADE    MOVB  R0,@CHRBUF        STORE CHARACTER
     03F8 837D
0636 03FA D011  MADC    MOVB  *R1,R0            LOAD VALUE OF VARIABLE
0637 03FC D145  MADC2   MOVB  R5,R5             SINGLE OR DOUBLE INSTRUCTION ?
0638 03FE 1602          JNE   MDOUB             DOUBLE
0639 0400 0880  MSIGN   SRA   R0,8              SIGN EXTEND BYTE VALUE
0640 0402 045B          RT
0641      0406' C1      EQU   $+2
0642 0404 D821  MDOUB   MOVB  @1(R1),@R0LB GET SECOND HALF OF DOUBLE
     0406 0001
     0408 83E1
0643 040A 045B          RT
```

```
0644 040C D81D  MLONG  MOVB *R13,@R1LB   GET SECOND HALF OF ADDRESS
     040E 83E3
0645 0410 C281         MOV  R1,R10       SAVE CONTROL BITS
0646 0412 0241         ANDI R1,>FFF      DELETE CONTROL BITS
     0414 0FFF
0647 0416 0281         CI   R1,>F00      LOOK FOR EXTENDED ADDRESS
     0418 0F00
0648 041A 1103         JLT  MLONG1       NO, JUST A REGULAR ADDRESS
0649 041C 0A81         SLA  R1,8         EXTENDED ADDRESS. MAKE ROOM
0650 041E D81D         MOVB *R13,@R1LB   GET ADDRESS
     0420 83E3
0651 0422 0A2A  MLONG1 SLA  R10,2        BASE ADDRESS NEEDED ?
0652 0424 1708         JNC  MVDP         NO
0653 0426 D19D         MOVB *R13,R6      GET ADDRESS OF BASE ADDRESS
0654 0428 0986         SRL  R6,8
0655 042A D026         MOVB @PAD(R6),R0  LOAD BASE ADDRESS
     042C 8300
0656 042E D826         MOVB @PAD+1(R6),@R0LB
     0430 8301
     0432 83E1
0657 0434 A040         A    R0,R1
0658 0436 0A1A  MVDP   SLA  R10,1        VDP OR 9985 RAM SPACE ?
0659 0438 1714         JNC  MCPU         9985 RAM SPACE
0660 043A 05C4         INCT R4           SET VDP INDICATOR
0661 043C 0A1A         SLA  R10,1        INDIRECT ADDRESS ?
0662 043E 1706         JNC  MVDPA        NO
0663 0440 D021         MOVB @PAD(R1),R0  LOAD INDIRECT ADDRESS
     0442 8300
0664 0444 D821         MOVB @PAD+1(R1),@R0LB
     0446 8301
     0448 83E1
0665 044A C040         MOV  R0,R1
0666 044C D7E0  MVDPA  MOVB @R1LB,*R15   LOAD ADDRESS INTO VDP
     044E 83E3
0667 0450 D7C1         MOVB R1,*R15
0668           ****    SLA  R0,8         NO OP
0669 0452 D02F         MOVB @VDPRD(R15),R0 RECALL CONTENTS OF VDP RAM
     0454 FBFE
0670 0456 D145         MOVB R5,R5        SINGLE OR DOUBLE INSTRUCTION ?
0671 0458 13D3         JEQ  MSIGN        SINGLE
0672 045A D82F         MOVB @VDPRD(R15),@R0LB RECALL LSBYTE FROM VDP RAM
     045C FBFE
     045E 83E1
0673 0460 045B         RT
0674 0462 0A1A  MCPU   SLA  R10,1        INDIRECT ADDRESS ?
0675 0464 17B6         JNC  MADD         NO
0676 0466 0281         CI   R1,>7C       INDIRECT ON STATUS BYTE ?
     0468 007C
0677 046A 1605         JNE  MADB         NO
0678 046C D060         MOVB @STKDAT,R1   GET TOP STACK ADDRESS
     046E 8372
0679 0470 780E         SB   R14,@STKDAT  R14=01XX
     0472 8372
0680 0474 10AD         JMP  MADDA
0681 0476 D061  MADB   MOVB @PAD(R1),R1  LOAD INDIRECT ADDRESS
     0478 8300
0682 047A 10AA         JMP  MADDA
0683           *=============================================~==============*
0684           *               RETURN ADDRESS FROM LIBRARY OR PROGRAM
0685 047C 06A0  RGBA   BL   @GETSTK      RECALL GROM BASE ADDRESS
```

```
      047E 0492'
0686  0480 C364          MOV   @PAD(R4),R13
      0482 8300
0687  0484 DB44          MOVB  R4,@GRMWD(R13)      SYNCHRONIZE YOUR GROMS
      0486 0400
0688  0488 5820  RETURN  SZCB  @BIT2,@STATUS        RESET CONDITION BIT
      048A 0119'
      048C 837C
0689  048E 020B  RETNC   LI    R11,NEXT
      0490 0070'
0690  0492 D120  GETSTK  MOVB  @STKADD,R4    LOAD ADDRESS OF SUBROUTINE STACK
      0494 8373
0691  0496 0984          SRL   R4,8            MOVE TO LOW ORDER BYTE
0692  0498 0660  GTSTK   DECT  @STKADD         NEW VALUE OF STACK POINTER
      049A 8373
0693  049C DB64  GTSTK1  MOVB  @PAD(R4),@GRMWA(R13)  LOAD RETURN ADDRESS
      049E 8300
      04A0 0402
0694  04A2 DB64          MOVB  @PAD+1(R4),@GRMWA(R13)
      04A4 8301
      04A6 0402
0695  04A8 045B          RT
0696  04AA F820  RTNSET  SOCB  @BIT2,@STATUS       SET CONDITION BIT
      04AC 0119'
      04AE 837C
0697  04B0 10EE          JMP   RETNC
0698                     *
0699                     *                PUSH PROGRAM COUNTER IN K ONTO STACK
0700  04B2 D19D  CALL    MOVB  *R13,R6        GET BRANCH ADDRESS FROM GROM
0701  04B4 020B          LI    R11,LDKADD     SET RETURN POINTER
      04B6 0060'
0702  04B8 D81D          MOVB  *R13,@R6LB
      04BA 83ED
0703  04BC 05E0  PUTSTK  INCT  @STKADD         NEW VALUE OF STACK POINTER
      04BE 8373
0704  04C0 D120          MOVB  @STKADD,R4     LOAD ADDRESS OF STACK
      04C2 8373
0705  04C4 0984          SRL   R4,8            MOVE TO LOW ORDER BYTE
0706  04C6 D92D          MOVB  @GRMRA(R13),@PAD(R4)     SAVE ADDRESS ON STACK
      04C8 0002
      04CA 8300
0707  04CC D92D          MOVB  @GRMRA(R13),@PAD+1(R4)
      04CE 0002
      04D0 8301
0708  04D2 0624          DEC   @PAD(R4)
      04D4 8300
0709  04D6 045B          RT
0710                     *================================================*
0711                     *                LOAD XPTR AND RETURN CHRBUF
0712  04D8 020A  GETDA   LI    R10,WRVDP      LOAD VDP WRITE FLAG
      04DA 4000
0713  04DC D1E0  GETDAD  MOVB  @XPT,R7         LOAD VALUE OF XPT
      04DE 837F
0714  04E0 23A0          COC   @IN2,R14       TEST IF MULTICOLOR MODE
      04E2 0072'
0715  04E4 130E          JEQ   MCMDA
0716  04E6 0A37          SLA   R7,3           MASK OUT TRUE XPT VALUE
0717  04E8 0987          SRL   R7,8           MOVE TO LOW ORDER BYTE
0718  04EA D1E0          MOVB  @YPT,R7        LOAD VALUE OF YPT
      04EC 837E
```

```
0719 04EE 0937          SRL  R7,3          MOVE TO CORRECT POSITION
0720 04F0 A1CA          A    R10,R7        READ OR WRITE FLAG
0721 04F2 D7E0  SETVDP  MOVB @R7LB,*R15    LOAD CHARACTER BUFFER
     04F4 83EF
0722 04F6 D7C7          MOVB R7,*R15       STORE ADDRESS
0723 04F8 61CA          S    R10,R7
0724 04FA 045B  GETRTN  RT
0725 04FC 0207  SETV    LI   R7,WRVDP      WRITE TO VDP MEMORY
     04FE 4000
0726 0500 10F8          JMP  SETVDP
0727              *        MULTICOLOR MODE ADDRESS COMPUTATION
0728 0502 D020  MCMDA   MOVB @YPT,R0       LOAD YPT
     0504 837E
0729 0506 C200          MOV  R0,R8
0730 0508 0A58          SLA  R8,5          GET 3 LOW BITS AS LSB
0731 050A 09D8          SRL  R8,13
0732 050C 0980          SRL  R0,11         POSITION 5 HIGH BITS
0733 050E 0A80          SLA  R0,8
0734 0510 A008          A    R8,R0
0735 0512 C207          MOV  R7,R8         SAVE XPT
0736 0514 0247          ANDI R7,>3E00      POSITION XPT BITS
     0516 3E00
0737 0518 0967          SRL  R7,6
0738 051A A1C0          A    R0,R7         ADD X VALUE TO DISPLAY ADDRESS
0739 051C 0227          AI   R7,>800
     051E 0800
0740 0520 D7E0          MOVB @R7LB,*R15    LOAD VDP ADDRESS
     0522 83EF
0741 0524 0A88          SLA  R8,8          SET CARRY TO LSB OF XPT
0742 0526 D7C7          MOVB R7,*R15
0743 0528 028B          CI   R11,RTN       STORE OR RETRIEVE CB ?
     052A 026A'
0744 052C 16E6          JNE  GETRTN        STORE
0745 052E D02F          MOVB @VDPRD(R15),R0 LOAD BYTE FROM VDP
     0530 F8FE
0746 0532 D220          MOVB @CHRBUF,R8    LOAD CB
     0534 837D
0747      0538' HOF     EQU  $+2
0748 0536 0248          ANDI R8,>0F00      MASK OUT LOW DIGIT
     0538 0F00
0749 053A 1804          JOC  USTRCB        JMP IF CB IS LOW DIGIT
0750 053C 0240          ANDI R0,>0F00      MASK OUT LOW DIGIT
     053E 0F00
0751 0540 0A48          SLA  R8,4          MOVE CB TO HIGH DIGIT
0752 0542 1002          JMP  VSTRCB
0753 0544 0240  USTRCB  ANDI R0,>F000      MASK OUT HIGH DIGIT
     0546 F000
0754 0548 0267  VSTRCB  ORI  R7,WRVDP      SET BIT TO WRITE TO VDP
     054A 4000
0755 054C 06A0          BL   @SETVDP       LOAD VDP ADDRESS
     054E 04F2'
0756 0550 A008          A    R8,R0         COMBINE DIGITS
0757 0552 DBC0          MOVB R0,@VDPWD(R15) STORE BYTE IN VDP
     0554 FFFE
0758 0556 0456          B    *R6                  RETURN TO NEXT
0759              *
```

```
0761                    *=============================================== ================
0762                    ***     KEYBOARD SCAN
0763                    *
0764                    *** Initialization
0765                    *
0766 0558                       EVEN
0767                    *
0768                    *** Fire button positions
0769                    *
0770       0557' FIRE   EQU  $-1
0771 0558   01          BYTE 1
0772 0559   15          BYTE 21
0773       0559' LINE   EQU  $-1
0774 055A   0E          BYTE >0E,>0F
0775 055C   00  CRULOW  BYTE 0,1,6     LOWER LIMIT FOR KBD 0,1,2
0776 055F   0D  CRUHI   BYTE 13,5,10   UPPER LIMIT FOR KBD 0,1,2
0777 0562               EVEN
0778       0563' D11    EQU  $+1                  CONSTANT >0B = 11
0779 0562 0208  KEYSCN LI   R11,NEXT             RETURN TO GPL INTERPRETER
     0564 0070'
0780 0566 C80B  KSCAN  MOV  R11,@RSAVE           save return address
     0568 83D8
0781 056A D160         MOVB @PLAYER,R5           Is it console keyboard?
     056C 8374
0782 056E 0985         SRL  R5,8                 Right justify
0783 0570 C185         MOV  R5,R6                Make a copy of keyboard number
0784 0572 1323         JEQ  KSCAN2               Scan console keyboard
0785 0574 0286         CI   R6,3                 No, is it 1 or 2?
     0576 0003
0786 0578 1A0C         JL   JSCAN                if so, scan joysticks first
0787 057A 0286         CI   R6,5                 Illegal keyboard number?
     057C 0005
0788 057E 1B64         JH   NOKEY                Yes, return no key
0789 0580 0226         AI   R6,-3
     0582 FFFD
0790 0584 D806         MOVB R6,@PLAYER           Reset keyboard number to zero
     0586 8374
0791 0588 06C6         SWPB R6                   Get low byte for flag
0792 058A D806         MOVB R6,@KBDFLG           Store new keyboard flag
     058C 83C6
0793 058E 04C5         CLR  R5
0794 0590 1014         JMP  KSCAN2
0795                    *
0796                    *** Scan Joystick
0797                    *
0798 0592 020C  JSCAN  LI   R12,COLBAS           CRU base for column/joystick se
     0594 0020
0799 0596 3125         LDCR @LINE(R5),4          SELECT LINE TO SCAN
     0598 0559'
0800 059A 0BF4         SRC  R4,15                kill time to ensure setup
0801                    *                        (make sure we have 10 mic.sec.
0802                    *                        (for Milton Bradley expander)
0803 059C 020C         LI   R12,JOYBAS           CRU base to read directions + f
     059E 000C
0804 05A0 04C3         CLR  R3                   assume no fire
0805 05A2 04C4         CLR  R4                   clean register for read
0806 05A4 3544         STCR R4,5                 Read line
0807 05A6 0994         SRL  R4,9                 Get the fire button
0808 05A8 1802         JOC  JSCAN1               No fire button
0809 05AA D0E5         MOVB @FIRE(R5),R3         Get fire button
```

```
      05AC 0557'
0810  05AE 0A14  JSCAN1 SLA  R4,1                 Times 2 = index into table
0811               ****   MOVB @RBBJOY(R4),@JOYY read y value into joyy
0812               ****   MOVB @RBBJOY+1(R4),@JOYX read x value into joyx
0813  05B0 C824           MOV  @RBBJOY(R4),@JOYY read y and x values
      05B2 0F1E'
      05B4 8376
0814  05B6 D0C3           MOVB R3,R3              Fire button down?
0815  05B8 165C           JNE  DEBOU1             Yes
0816               *
0817               *** Scan Keyboard for a Key
0818               *
0819  05BA D065  KSCAN2 MOVB @CRULOW(R5),R1       starting column to scan
      05BC 055C'
0820  05BE 020C  ROWLP  LI   R12,COLBAS           CRU base for column selection
      05C0 0020
0821  05C2 3101           LDCR R1,4               Output row scan to KB
0822  05C4 020C           LI   R12,ROWBAS         Base for column input
      05C6 000E
0823  05C8 3504           STCR R4,4               input one column of data
0824  05CA 0544           INV  R4                 MAKE ONE'S REPRESENT KEYS DOWN
0825  05CC 0244           ANDI R4,>0F00           LEAVE ONLY SIGNIFICANT BITS
      05CE 0F00
0826  05D0 D041           MOVB R1,R1              column 0?
0827  05D2 1624           JNE  NOTONE
0828               * must look at column 11 also, for other shift key
0829  05D4 020C           LI   R12,COLBAS
      05D6 0020
0830  05D8 0201           LI   R1,>B00
      05DA 0B00
0831  05DC 3101           LDCR R1,4               scan column for second shift
0832  05DE 020C           LI   R12,ROWBAS
      05E0 000E
0833  05E2 3506           STCR R6,4
0834  05E4 0546           INV  R6
0835  05E6 0246           ANDI R6,>100            save only shift key
      05E8 0100
0836  05EA F106           SOCB R6,R4              and or it with column 0 shift
0837  05EC 04C1           CLR  R1                 make it column 0 again
0838  05EE 0A54           SLA  R4,5               test alpha lock bit
0839  05F0 170A           JNC  NOALPH
0840  05F2 D1A0           MOVB @ALPHLK,R6         debounce the alpha lock key
      05F4 83C3
0841  05F6 1110           JLT  NA1                still down...do nothing
0842  05F8 29A0           XOR  @H81,R6            indicate key is down, and toggl
      05FA 09F6'
0843  05FC D806           MOVB R6,@ALPHLK
      05FE 83C3
0844  0600 06A0           BL   @KL10MS            wait for physical debounce
      0602 074C'
0845  0604 1009           JMP  NA1
0846  0606 D1A0  NOALPH MOVB @ALPHLK,R6          was it down before?
      0608 83C3
0847  060A 0A16           SLA  R6,1
0848  060C 1705           JNC  NA1                yes, do nothing
0849  060E 06A0           BL   @KL10MS
      0610 074C'
0850  0612 5820           SZCB @H80,@ALPHLK       indicate that alpha lock key is
      0614 0718'
      0616 83C3
```

```
0851 0618 D1C4  NA1     MOVB R4,R7              save other three qualifier keys
0852 061A 1011          JMP  NXTROW
0853 061C 9801  NOTONE  CB   R1,@D11            Is this column 11? (second shif
     061E 0563'
0854 0620 1602          JNE  NOT1
0855 0622 0244          ANDI R4,>0E00           ignore second shift key
     0624 0E00
0856 0626 D104  NOT1    MOVB R4,R4
0857 0628 130A          JEQ  NXTROW             IF NO KEY DOWN, TRY NEXT LINE
0858                    *
0859                    *** GOT A KEY! DETERMINE WHICH ONE!
0860                    *
0861 062A C0C1          MOV  R1,R3              which column
0862 062C 0983          SRL  R3,8               make a word
0863 062E 0603          DEC  R3                 correct for fact column 0 is no
0864 0630 0A23          SLA  R3,2               multiply by 4 keys per column
0865 0632 0A44          SLA  R4,4               get bits in position to test
0866 0634 0603          DEC  R3
0867 0636 0583  CNTLP   INC  R3                 add in another row offset
0868 0638 0A14          SLA  R4,1               SHIFT NEXT BIT OUT INTO CARRY
0869 063A 17FD          JNC  CNTLP              BIT = 1 IS KEY DEPRESSED
0870 063C 1019          JMP  DEBOUN             and debounce it
0871                    *
0872                    *** SELECT NEXT LINE TO POLL
0873                    *
0874 063E 0221  NXTROW  AI   R1,>100            select next column
     0640 0100
0875 0642 9941          CB   R1,@CRUHI(R5)
     0644 055F'
0876 0646 12BB          JLE  ROWLP              Not finished- scan next row
0877                    *
0878                    *** No Keys Down
0879                    *
0880 0648 04C6  NOKEY   CLR  R6                 Condition reset
0881 064A D806          MOVB R6,@OLDMOD         Clear old modifiers
     064C 83C7
0882 064E 0700          SETO R0                 Invalid keycode is >FF
0883 0650 9940          CB   R0,@DBNCE(R5)      Was key just released?
     0652 83C8
0884 0654 1302          JEQ  NOKEY2             No so no delay
0885 0656 06A0          BL   @KL10MS            Kill some time for debounce
     0658 074C'
0886 065A D800  NOKEY2  MOVB R0,@DBNCE          Always clear KBD 0 debounce
     065C 83C8
0887 065E D940          MOVB R0,@DBNCE(R5)      Always clear KBD being scanned
     0660 83C8
0888 0662 C145          MOV  R5,R5              Is this KBD 0 ?
0889 0664 165D          JNE  OLDCHR             No so done
0890 0666 D800          MOVB R0,@DBNC1          Clear KBD 1 debounce
     0668 83C9
0891 066A D800          MOVB R0,@DBNC2          Clear KBD 2 debounce
     066C 83CA
0892 066E 1058          JMP  OLDCHR
0893                    *
0894                    *** Debounce key
0895                    *
0896                    ***CLKSND DATA >8903,>939F    1975 HZ, ON THEN OFF
0897 0670 06C3  DEBOUN  SWPB R3                  GET POSITION IN MSB
0898 0672 04C6  DEBOU1  CLR  R6                  ASSUME IT IS OLD KEY
0899 0674 9943          CB   R3,@DBNCE(R5)      Same key position?
```

```
      0676 83C8
0900  0678 1317          JEQ   MODIFY              Yes, go get old key code
0901       067C' H020    EQU   $+2
0902  067A 0206          LI    R6,>2000            Set status to new
      067C 2000
0903               ***   MOVB  @CLICK,R0
0904               ***   JEQ   DEBOU2
0905               ***   LI    R2,CLKSND
0906               ***   MOVB  *R2+,@>8400         frequency
0907               ***   MOVB  *R2+,@>8400
0908               ***   MOVB  *R2+,@>8400         volume on
0909  067E 06A0    DEBOU2 BL   @KL10MS
      0680 074C'
0910               ***   MOVB  R0,R0
0911               ***   JEQ   DEBOU3
0912               ***   MOVB  *R2+,@>8400         volume off
0913  0682 D803    DEBOU3 MOVB R3,@DBNCE           Always debounce KBD 0
      0684 83C8
0914  0686 D943          MOVB  R3,@DBNCE(R5)       Debounce KBD being scanned
      0688 83C8
0915  068A C145          MOV   R5,R5               Is this KBD 0 ?
0916  068C 160D          JNE   MODIFY              No so done
0917               * KBD 0 - MAY ALSO NEED TO DEBOUNCE A SPLIT KBD
0918  068E 0283          CI    R3,>1300            LEFT?
      0690 1300
0919  0692 1B03          JH    DB2
0920  0694 D803          MOVB  R3,@DBNC1           DEBOUNCE LEFT
      0696 83C9
0921               * DON'T SAVE R7 IF SPLIT KBD - IT IS NOT VALID
0922  0698 1005          JMP   NEWMOD
0923  069A 0283    DB2   CI    R3,>2800            NEITHER?
      069C 2800
0924  069E 1402          JHE   NEWMOD
0925  06A0 D803          MOVB  R3,@DBNC2           DEBOUNCE RIGHT
      06A2 83CA
0926  06A4 D807    NEWMOD MOVB R7,@OLDMOD
      06A6 83C7
0927               *
0928               *** Got the key position.  Now get the key code.
0929               *
0930  06A8 D1E0    MODIFY MOVB @OLDMOD,R7
      06AA 83C7
0931  06AC 06C3          SWPB  R3                  RETURN POSITION TO LSB
0932  06AE 0201          LI    R1,RKSPLT           Assume split keyboard
      06B0 100E'
0933  06B2 C145          MOV   R5,R5               Split keyboard?
0934  06B4 1603          JNE   MAPIT               Yes
0935  06B6 09C7          SRL   R7,12               3 modifier bits * 2
0936  06B8 C067          MOV   @RMDTAB(R7),R1      SELECT PROPER TABLE
      06BA 0028'
0937  06BC A043    MAPIT A     R3,R1               Add table offset
0938  06BE D011          MOVB  *R1,R0              Get key code from table
0939  06C0 9800          CB    R0,@HFF             unassigned station?
      06C2 09FA'
0940  06C4 13C1          JEQ   NOKEY               if so, do not report it
0941  06C6 C145          MOV   R5,R5               Is it split keyboard?
0942  06C8 162B          JNE   OLDCHR              Yes, no special mapping
0943  06CA D0E0          MOVB  @KBDFLG,R3          Get keyboard flag
      06CC 83C6
0944  06CE 0983          SRL   R3,8                make it a word
```

```
0945 06D0 06A0          BL   @RNGCHK            Is it a lowercase letter
     06D2 073C'
0946 06D4 617A          DATA 'az'
0947 06D6 1608          JNE  WHCHKB             No so no alpha lock
0948 06D8 C0C3          MOV  R3,R3              Is this keyboard 0 ?
0949 06DA 1304          JEQ  ALOCK              Yes so map to upper case
0950 06DC D060          MOVB @ALPHLK,R1
     06DE 83C3
0951 06E0 0991          SRL  R1,9               TEST THE LOCK STATUS
0952 06E2 1702          JNC  WHCHKB             No alpha lock
0953 06E4 7020  ALOCK   SB   @H020,R0           Map to upper case
     06E6 067C'
0954 06E8 C0C3  WHCHKB  MOV  R3,R3              Is it keyboard 0 ?
0955 06EA 1609          JNE  WHCH2              No, so no mapping
0956                    * THIS IS A CONVENIENT PLACE TO INITIALIZE ALPHA LOCK CONDITI
0957 06EC D80E          MOVB R14,@ALPHLK  START WITH ALPHA LOCK SET
     06EE 83C3
0958 06F0 06A0          BL   @RNGCHK            Bad key?
     06F2 073C'
0959 06F4 101F          DATA >101F
0960 06F6 13A8          JEQ  NOKEY              Yes, return no key
0961 06F8 9800          CB   R0,@H5F            Is it higher than "_" ?
     06FA 07FF'
0962 06FC 1BA5          JH   NOKEY              Yes, return no key
0963 06FE 0603  WHCH2   DEC  R3                 Is this keyboard 4 ?
0964 0700 160F          JNE  OLDCHR             No, it's 5. So return code.
0965 0702 9800          CB   R0,@H0D            Is it RETURN key?
     0704 0D55'
0966 0706 130C          JEQ  OLDCHR             Yes so return it
0967 0708 9800          CB   R0,@H0F            Is it GPL control key (1-15) ?
     070A 0538'
0968 070C 1B03          JH   MAP4               No
0969 070E F020          SOCB @H80,R0            Yes so set sign bit
     0710 0718'
0970 0712 1006          JMP  OLDCHR             Return it
0971 0714 06A0  MAP4    BL   @RNGCHK            Is it ASCII printable ?
     0716 073C'
0972 0718 809F  H80     DATA >809F
0973 071A 1602          JNE  OLDCHR             Yes, return it
0974 071C 5020          SZCB @H80,R0            Control code, reset sign bit
     071E 0718'
0975 0720 D800  OLDCHR  MOVB R0,@KEYBRD         Output ASCII character
     0722 8375
0976 0724 D806          MOVB R6,@STATUS         Store status
     0726 837C
0977 0728 1306          JEQ  EXSCN              No new key
0978 072A D7E0          MOVB @SAVVDP,*R15       Turn on the screen
     072C 83D4
0979 072E 04E0          CLR  @TIMOUT            Reset screen timer
     0730 83D6
0980 0732 D7E0          MOVB @H81,*R15          VDP register 1
     0734 09F6'
0981 0736 C2E0  EXSCN   MOV  @RSAVE,R11         Restore return address
     0738 83D8
0982 073A 045B          RT
0983                    *
0984                    *** Range check routine
0985                    *
0986 073C C33B  RNGCHK  MOV  *R11+,R12          Fetch range
0987 073E 9300          CB   R0,R12             Compare to low end of range
```

```
0988 0740 1A04          JL    RNGRT             Out of range low
0989 0742 06CC          SWPB  R12
0990 0744 9300          CB    R0,R12            Compare to high end of range
0991 0746 1B01          JH    RNGRT             Out of range high
0992 0748 9000          CB    R0,R0             In range so set EQ bit
0993 074A 045B  RNGRT   RT                      Return to caller
0994                    *
0995                    *** Kill time routine
0996                    *
0997 074C C320  KL10MS  MOV   @DELAY2,R12       Load up 10 m-secs of counts
     074E 849C
0998 0750 060C  KILTIM  DEC   R12               Count it down
0999 0752 15FE          JGT   KILTIM            Not done yet
1000 0754 045B          RT                      Return to caller
```

```
1002                    *--------------------------------------------------------------*
1003                    *                FORMAT FOR STRING
1004 0756 04C9  FORMAT  CLR   R9            RESET BLOCK COUNT TO 0
1005 0758 04C3          CLR   R3            ZERO BIAS BYTE
1006 075A 06A0          BL    @GETDA        ENCODE VDP ADDRESS
     075C 04D8'
1007 075E D21D  FUNCTN  MOVB  *R13,R8       GET NEXT BYTE FROM GAME ROM
1008 0760 020C          LI    R12,STKADD
     0762 8373
1009 0764 C148          MOV   R8,R5         SAVE DATA BYTE
1010 0766 0A38          SLA   R8,3          SHIFT TO FIND FUNCTION TO PERFORM
1011 0768 09B8          SRL   R8,11         MOVE TO LOW ORDER POSITION
1012 076A 0548          INV   R8            LEAVE N IN RANGE OF MINUS 1 - 32
1013 076C 09C5          SRL   R5,12         ISOLATE FUNCTION BITS
1014 076E C165          MOV   @FBTAB(R5),R5     GET BRANCH ADDRESS
     0770 0CDC'
1015 0772 0202          LI    R2,LOOP
     0774 0782'
1016 0776 0704          SETO  R4            SET I TO -1
1017 0778 0455          B     *R5
1018                    *                   REPEAT CHARACTER N TIMES DOWN THE SCREEN
1019 077A 0A54  RPTDWN  SLA   R4,5          SET I TO -32
1020                    *                   REPEAT CHARACTER N TIMES ACROSS THE SCREEN
1021 077C 8CB2  RPTACR  C     *R2+,*R2+     LOAD BRANCH ADDRESS FOR LOOPING
1022 077E 1001          JMP   LOOP
1023                    *                   STORE N CHARACTERS DOWN THE SCREEN
1024 0780 0A54  STRDWN  SLA   R4,5          SET I TO -32
1025                    *                   STORE N CHARACTERS ACROSS THE SCREEN
1026 0782               STRACR
1027                    *                   LOOP TO STORE CHARACTERS
1028 0782 D19D  LOOP    MOVB  *R13,R6       GET NEXT BYTE FROM GAME ROM
1029 0784 A183  LOOPB   A     R3,R6         ADD BIAS TO CHARACTER
1030 0786 DBC6  LOOPA   MOVB  R6,@VDPWD(R15) MOVE BYTE INTO DISPLAY
     0788 FFFE
1031 078A 61C4          S     R4,R7         DISPLACEMENT TO NEXT CHARACTER
1032 078C 0287          CI    R7,>320       DO I WANT WRAPAROUND?
     078E 0320
1033 0790 1405          JHE   ZEND          NO
1034 0792 0287  UPXY    CI    R7,>300       DISPLAY ADDRESS OUT OF RANGE?
     0794 0300
1035 0796 1A02          JL    ZEND          NO
1036 0798 0227          AI    R7,->300      SUBTRACT LENGTH OF DISPLAY
     079A FD00
1037 079C 06A0  ZEND    BL    @RESTOR
     079E 0830'
1038 07A0 06A0          BL    @GETDA
     07A2 04D8'
1039 07A4 0588          INC   R8            INCREMENT LOOP COUNTER
1040 07A6 13D8          JEQ   FUNCTN        DONE WITH LOOP IF N = 0
1041 07A8 0452          B     *R2           LOOP UNTIL N = 0
1042                    *                   SKIP N LINES DOWN THE SCREEN
1043 07AA 0A58  SKPDWN  SLA   R8,5          MULTIPLY LINES TO SKIP BY 32
1044                    *                   SKIP N SPACES ACROSS THE SCREEN
1045 07AC 61C8  SKPACR  S     R8,R7         MOVE NUMBER OF SPACES TO SKIP INTO I
1046 07AE 0708          SETO  R8            SET LOOP COUNTER TO -1
1047 07B0 10F0          JMP   UPXY          GO PERFORM SKIP
1048                    *                   REPEAT BLOCK N TIMES
1049 07B2 0589  RPTBLK  INC   R9            INCREMENT BLOCK COUNTER
1050 07B4 05DC          INCT  *R12          INCREMENT STACK POINTER
1051 07B6 D19C          MOVB  *R12,R6       GET DATA STACK POINTER
```

```
1052 07B8 0986          SRL   R6,8           MOVE TO LOW ORDER BYTE
1053 07BA 06C8          SWPB  R8
1054 07BC D988          MOVB  R8,@PAD(R6)       PUSH N ONTO DATA STACK
     07BE 8300
1055 07C0 10CE          JMP   FUNCTN
1056              *                REPETITION OF BLOCK HAS BEEN COMPLETED
1057 07C2 065C  FINISH  DECT  *R12           DECREMENT DATA STACK POINTER R14=01X
1058 07C4 0609          DEC   R9             DECREMENT BLOCK COUNT
1059 07C6 10CB          JMP   FUNCTN
1060              *                END BLOCK FUNCTION
1061 07C8 C249  ENDBLK  MOV   R9,R9          IS BLOCK COUNT ZERO ?
1062 07CA 1330          JEQ   ENDFMT         YES - END OF FORMAT INSTRUCTION
1063 07CC D11D          MOVB  *R13,R4        LOAD LOOP ADDRESS
1064 07CE D19C          MOVB  *R12,R6        GET DATA STACK POINTER
1065 07D0 D15D          MOVB  *R13,R5
1066 07D2 0986          SRL   R6,8
1067 07D4 B98E          AB    R14,@PAD(R6) DECREMENT REPITITION COUNT R14=01XX
     07D6 8300
1068 07D8 13F4          JEQ   FINISH         DONE WITH BLOCK IF COUNT IS ZERO
1069 07DA DB44          MOVB  R4,@GRMWA(R13)      LOAD GROM WITH LOOP ADDRESS
     07DC 0402
1070 07DE DB45          MOVB  R5,@GRMWA(R13)
     07E0 0402
1071 07E2 10BD          JMP   FUNCTN
1072              *
1073              *                SPECIAL FORMAT FUNCTIONS
1074      07E7' HE4     EQU   $+3
1075 07E4 0288  SPECL   CI    R8,-28         WHICH FUNCTION ?
     07E6 FFE4
1076 07E8 13EF          JEQ   ENDBLK         END BLOCK
1077 07EA 1511          JGT   VARBLE         VARIABLE CHARACTERS
1078 07EC C04D          MOV   R13,R1
1079 07EE 0288          CI    R8,-30         WHICH FUNCTION ?
     07F0 FFE2
1080 07F2 1309          JEQ   BIASV          SET COLOR BIAS
1081 07F4 150A          JGT   BIASI          SET COLOR BIAS IMMEDIATE
1082 07F6 06A0          BL    @RESTOR        SET XPT OR YPT
     07F8 0830'
1083 07FA 0508          NEG   R8
1084      07FF' H5F     EQU   $+3
1085 07FC DA1D          MOVB  *R13,@PAD+>5F(R8) LOAD VALUE OF XPT OR YPT
     07FE 835F
1086 0800 06A0          BL    @GETDA         COMPUTE VALUE OF ADDRESS
     0802 04D8'
1087 0804 10AC          JMP   FUNCTN
1088              *                SET COLOR BIAS
1089 0806 06A0  BIASV   BL    @GETMAB        GET MEMORY ADDRESS
     0808 03CA'
1090 080A D0D1  BIASI   MOVB  *R1,R3         LOAD COLOR BIAS
1091 080C 10A8          JMP   FUNCTN
1092              *                VARIABLE CHARACTERS
1093 080E 06A0  VARBLE  BL    @GETMAB        GET MEMORY ADDRESS
     0810 03CA'
1094 0812 0202          LI    R2,LOOPV       SET LOOP REGISTER
     0814 0816'
1095 0816 D1B1  LOOPV   MOVB  *R1+,R6        LOAD VARIABLE CHARACTER
1096 0818 10B5          JMP   LOOPB
```

```
1098                *---------------------------------------------------------------*
1099                *                  CLEAR SCREEN
1100 081A D15D  ALL     MOVB *R13,R5       GET CHARACTER TO STORE
1101 081C 06A0          BL   @SETV
     081E 04FC'
1102 0820 0207          LI   R7,768        NUMBER OF BYTES TO TRANSFER
     0822 0300
1103 0824 DBC5  ALLOOP  MOVB R5,@VDPWD(R15) STORE BYTE IN DISPLAY
     0826 FFFE
1104 0828 0607          DEC  R7
1105 082A 16FC          JNE  ALLOOP
1106                *                  RESTORE XPT AND YPT FROM DISPLAY ADDRESS
1107 082C 020B  ENDFMT  LI   R11,NEXT      END OF FORMAT
     082E 0070'
1108 0830 0A37  RESTOR  SLA  R7,3          SHIFT YPT VALUE TO TOP BYTE
1109 0832 D807          MOVB R7,@YPT       STORE VALUE OF YPT
     0834 837E
1110 0836 0A87          SLA  R7,8          SHIFT OFF VALUE OF YPT
1111 0838 0937          SRL  R7,3          LEAVE VALUE OF XPT
1112 083A D807          MOVB R7,@XPT       STORE VALUE OF XPT
     083C 837F
1113 083E 0458          RT
```

```
1115                    *
1116 0840 0300  REMOTE  LIMI  0
     0842 0000
1117 0844 02E0          LWPI  GPLWS       USE REGULAR REGISTERS
     0846 83E0
1118 0848 04CC          CLR   R12          LOAD CRU ADDRESS
1119 084A 23A0          COC   @H20,R14     IS TIMER FLAG ON?
     084C 0D7E'
                        JNE   TIM1         NO, SEE WHAT IS INTERRUPTING ME
1120 084E 1602
1121 0850 0460          B     @TIMER       HANDLE TIMER INTERRUPT
     0852 0000
1122 0854 1F02  TIM1    TB    2            VDP INTERRUPT??
1123 0856 161B          JNE   VDPINT       YES
1124                    *
1125                    *       PERIPHERAL ROM INTERRUPT ROUTINES MAY CHANGE ALL
1126                    *       REGISTERS EXCEPT R0,R11-R15.
1127                    *       R11 HAS THE RETRUN ADDRESS FOR THE INTERRUPT ROUTINE
1128                    *       R12 IS LOADED FOR THE CRU SPACE OF THE PERIPHERAL
1129                    *       R13 HAS THE CURRENT GROM SLOT ADDRESS
1130                    *       R14 HAS A >01XX WHERE THE XX IS USED BY THE INTERPRETE
1131                    *       R15 HAS THE ADDRESS OF THE VDP WRITE ADDRESS ADDRESS
1132                    *       INTERRUPT ROUTINE SHOULD DO A RETURN WHEN DONE
1133                    *
1134 0858 D020          MOVB  @MAPPER,R0   *debug* turn off mapper interrupt
     085A 8810
1135 085C 020C          LI    R12,>0F00    TABLE OF CRU BITS TO SELECT ROM
     085E 0F00
1136 0860 1D01          SBO   1            TURN OFF EXTERNAL INTERRUPT
1137 0862 1E00  ILOOP   SBZ   0            TURN OFF LAST ROM
1138 0864 022C          AI    R12,>0100    NEXT ROM
     0866 0100
1139 0868 028C          CI    R12,>3000    FINISHED?
     086A 3000
1140 086C 130E          JEQ   EMERG2       END OF ROM CRU BITS
1141 086E 1D00          SBO   0            LOAD CRU BITS TO SELECT ROM
1142 0870 9820          CB    @H4000,@HAA  VALID ROM?
     0872 4000
     0874 000D'
                        JNE   ILOOP        NO
1143 0876 16F5
1144 0878 C0A0          MOV   @H400C,R2    GET INTERRUPT ROUTINE ADDRESS
     087A 400C
1145 087C 13F2  LOOP1   JEQ   ILOOP        NO ROUTINE, GO TO NEXT ROM
1146 087E C002          MOV   R2,R0        SAVE POINTER TO NEXT ROUTINE
1147 0880 C0A2          MOV   @2(R2),R2    GET ENTRY ADDRESS
     0882 0002
1148 0884 0692          BL    *R2          JUMP TO INTERRUPT ROUTINE
1149 0886 C090          MOV   *R0,R2       NEXT ROUTINE'S ADDRESS
1150 0888 10F9          JMP   LOOP1        GO TO NEXT ROUTINE
1151 088A 0460  EMERG2  B     @EMERGE
     088C 0A20'
1152                    *=======================================================*
1153                    *       V D P   I N T E R R U P T   H A N D L E R
1154                    *
1155 088E 1D02  VDPINT  SBO   2            RESET VDP INTERRUPT ON 9901
1156 0890 D060          MOVB  @INTFLG,R1
     0892 83C2
1157 0894 0A11          SLA   R1,1
1158 0896 1702          JNC   TSTMOT
1159 0898 0460          B     @VSTAT
     089A 09E2'
```

```
1160                    *===============================================================*
1161                    *          AUTOMATIC SPRITE MOTION      (INTERRUPT ROUTINE)
1162 089C 0A11  TSTMOT SLA   R1,1
1163 089E 1848         JOC   TSTSND
1164 08A0 D320         MOVB  @MOTION,R12
     08A2 837A
1165 08A4 1345         JEQ   TSTSND
1166 08A6 098C         SRL   R12,8
1167 08A8 0202         LI    R2,VDPRD+VDPWA
     08AA 8800
1168 08AC 0203         LI    R3,VDPWD+VDPWA
     08AE 8C00
1169 08B0 C220         MOV   @SVL,R8
     08B2 84A2
1170 08B4 C2C8         MOV   R8,R11          R11=OFFSET FROM SAL TO SVL
1171 08B6 62E0         S     @SAL,R11
     08B8 84A0
1172 08BA D7E0  MLOOP  MOVB  @R8LB,*R15      LOAD ADDRESS TO READ SML
     08BC 83F1
1173 08BE D7C8         MOVB  R8,*R15
1174 08C0 04C4         CLR   R4
1175 08C2 D112         MOVB  *R2,R4          READ DELTA Y
1176 08C4 04C6         CLR   R6
1177 08C6 D192         MOVB  *R2,R6          READ DELTA X
1178 08C8 0844         SRA R4,4              MOVE RIGHT ONE DIGIT
1179 08CA D152         MOVB  *R2,R5          READ TEMP Y
1180 08CC 0845         SRA   R5,4            MOVE RIGHT ONE DIGIT
1181 08CE A144         A     R4,R5
1182 08D0 D1D2         MOVB  *R2,R7
1183 08D2 0846         SRA   R6,4            MOVE RIGHT ONE DIGIT
1184 08D4 0847         SRA   R7,4            MOVE RIGHT ONE DIGIT
1185 08D6 A1C6         A     R6,R7
1186 08D8 620B         S     R11,R8          CHANGE ADDRESS TO SAL
1187 08DA D7E0         MOVB  @R8LB,*R15      LOAD ADDRESS TO READ SAL
     08DC 83F1
1188 08DE D7C8         MOVB  R8,*R15
1189 08E0 04C4         CLR   R4
1190 08E2 D112         MOVB  *R2,R4          READ Y POSITION
1191 08E4 A105         A     R5,R4           ADD Y MOVEMENT TO POSITION
1192                    ***************************************************************
1193                    ***     To fix the sprite flicker problem,
1194                    ***     add following code back (it has been taken out
1195                    ***     of the /4A shipped 03/16/81) 07/29/81.
1196 08E6-0284         CI    R4,6*VDELTA+255
     08E8 C0FF
1197 08EA 1209         JLE   ONSCRN
1198 08EC-0284         CI    R4,7*VDELTA
     08EE E000
1199 08F0 1B06         JH    ONSCRN
1200 08F2 C145         MOV   R5,R5
1201 08F4 1502         JGT   $+6
1202 08F6-0224         AI    R4,6*VDELTA
     08F8 C000
1203 08FA 0224         AI    R4,VDELTA
     08FC 2000
1204                    ***************************************************************
1205 08FE 04C6  ONSCRN CLR   R6
1206 0900 D192         MOVB  *R2,R6
1207 0902 A187         A     R7,R6
1208 0904 0268         ORI   R8,WRVDP
```

```
                0906 4000
1209 0908 D7E0          MOVB @R8LB,*R15   LOAD ADDRESS TO WRITE SAL
                090A 83F1
1210 090C D7C8          MOVB R8,*R15
1211 090E D4C4          MOVB R4,*R3
1212 0910 A20B          A    R11,R8       BACK TO SVL FOR FRACTIONS
1213 0912 05C8          INCT R8
1214 0914 D4C6          MOVB R6,*R3
1215 0916 06C5          SWPB R5
1216 0918 D7E0          MOVB @R8LB,*R15   LOAD ADDRESS TO WRITE SML
                091A 83F1
1217 091C D7C8          MOVB R8,*R15
1218 091E 0945          SRL  R5,4
1219 0920 D4C5          MOVB R5,*R3
1220 0922 06C7          SWPB R7
1221 0924 0947          SRL  R7,4
1222 0926 D4C7          MOVB R7,*R3
1223 0928 0228          AI   R8,2-WRVDP
                092A C002
1224 092C 060C          DEC  R12
1225 092E 15C5          JGT  MLOOP
1226                  *=======================================================*
1227                  *       AUTO-SOUND FEATURE            (INTERRUPT ROUTINE)
1228 0930 0A11  TSTSND  SLA  R1,1
1229 0932 183D          JOC  TSTQT
1230 0934 D0A0          MOVB @STFLGS,R2
                0936 83CE
1231 0938 133A          JEQ  TSTQT
1232 093A 780E          SB   R14,@STFLGS       R14=01XX
                093C 83CE
1233 093E 1637          JNE  TSTQT
1234 0940 C0E0          MOV  @SNDADD,R3
                0942 83CC
1235 0944 C14E          MOV  R14,R5
1236 0946 0915          SRL  R5,1
1237 0948 180A          JOC  SNDRAM
1238 094A 06A0          BL   @PUTSTK
                094C 04BC'
1239 094E 0205          LI   R5,GRMWA
                0950 0402
1240 0952 A14D          A    R13,R5
1241 0954 D543          MOVB R3,*R5
1242 0956 D560          MOVB @R3LB,*R5
                0958 83E7
1243 095A C18D          MOV  R13,R6
1244 095C 1007          JMP  USND
1245 095E 0205  SNDRAM  LI   R5,VDPWA
                0960 8C02
1246 0962 D560          MOVB @R3LB,*R5
                0964 83E7
1247 0966 D543          MOVB R3,*R5
1248 0968 0206          LI   R6,VDPRD+VDPWA
                096A 8800
1249 096C D216  USND    MOVB *R6,R8
1250 096E 130F          JEQ  NEWADD
1251 0970 9220          CB   @HFF,R8       DO I SWITCH SOURCE TYPE ?
                0972 09FA'
1252 0974 130A          JEQ  NEW1          YES
1253 0976 0988          SRL  R8,8
1254 0978 A0C8          A    R8,R3
```

```
1255 097A D816  SLOOP  MOVB *R6,@SGCADR  SEND BYTE TO SOUND CHIP
     097C 8400
1256 097E 0608         DEC  R8
1257 0980 16FC         JNE  SLOOP        LOOP UNTIL COUNT IS ZERO
1258 0982 05C3         INCT R3
1259 0984 D096         MOVB *R6,R2
1260 0986 1309         JEQ  XSOUND
1261 0988 1009         JMP  SNDXIT
1262 098A 2BA0  NEW1   XOR  @C1,R14      CHANGE VDP TO GROM OR GROM TO VDP
     098C 0406'
1263 098E D0D6  NEWADD MOVB *R6,R3       GET HIGH BYTE OF NEW ADDRESS
1264 0990 0202         LI   R2,>100
     0992 0100
1265 0994 D816         MOVB *R6,@R3LB    GET LOW BYTE OF NEW ADDRESS
     0996 83E7
1266 0998 1001         JMP  SNDXIT
1267 099A 7082  XSOUND SB   R2,R2        TURN OFF SOUND PROCESSING
1268 099C C803  SNDXIT MOV  R3,@SNDADD
     099E 83CC
1269 09A0 D802         MOVB R2,@STFLGS
     09A2 83CE
1270 09A4 0285         CI   R5,VDPWA
     09A6 8C02
1271 09A8 1302         JEQ  TSTQT
1272 09AA 06A0         BL   @GETSTK
     09AC 0492'
1273                   *==============================================~.==============*
1274 09AE 0A11  TSTQT  SLA  R1,1
1275 09B0 1818         JOC  VSTAT
1276 09B2 020C         LI   R12,COLBAS   LOAD CONSOLE ADDRESS IN CRU BASE
     09B4 0020
1277 09B6 3120         LDCR @H00,4       LOAD FOR COLUMN 0
     09B8 0F1E'
1278 09BA 0B7C         SRC  R12,7        KILL TIME
1279 09BC 020C         LI   R12,ROWBAS   CRU ADDRESS FOR CHARACTER LINES
     09BE 000E
1280 09C0 3505         STCR R5,4         RECALL COLUMN INFORMATION
1281 09C2 06C5         SWPB R5           GET READY TO READ RIGHT-HAND COLUMN
1282 09C4 020C         LI   R12,COLBAS
     09C6 0020
1283 09C8 3120         LDCR @H0B,4       LOAD FOR "=" COLUMN
     09CA 005F'
1284 09CC 020C         LI   R12,ROWBAS
     09CE 000E
1285 09D0 3505         STCR R5,4         READ COLUMN 13
1286 09D2 8160         C    @FNCEQ,R5    IS IT A  "FNCT'N  BACKSLASH" ?
     09D4 005E'
1287 09D6 1605         JNE  VSTAT
1288 09D8 020C  BUGOUT LI   R12,>2702    try a hardware reset
     09DA 2702
1289 09DC 1D00         SBO  0
1290                   * but just in case, do a soft reset
1291 09DE 0420         BLWP @0           RESET VECTOR TRAP LOCATION
     09E0 0000
1292                   *==============================================~.==============*
1293 09E2 D82F  VSTAT  MOVB @VDPSTA(R15),@VDPST LOAD VDP STATUS IN STATUS BLO
     09E4 FC00
     09E6 837B
1294 09E8 02E0         LWPI INTWSP       USE INTERRUPT WORK SPACE
     09EA 83C0
```

```
1295 09EC 05CB          INCT R11            INCREMENT SCREEN TIMER
1296 09EE 160B          JNE  TIMIN1         NO TIMEOUT
1297 09F0 D30A  TIMIN   MOVB R10,R12
1298 09F2 098C          SRL  R12,8
1299      09F6' H81     EQU  $+2
1300 09F4 026C          ORI  R12,>8160      TURN OFF VDP
     09F6 8160
1301      09FA' HFF     EQU  $+2
1302 09F8 024C          ANDI R12,>FFBF
     09FA FFBF
1303 09FC D820          MOVB @AR12LB,@VDPWA
     09FE 83D9
     0A00 8C02
1304 0A02 D80C          MOVB R12,@VDPWA
     0A04 8C02
1305 0A06 02E0  TIMIN1  LWPI GPLWS
     0A08 83E0
1306 0A0A B80E          AB   R14,@TIME      INCREMENT TIME IN STATUS BLOCK
     0A0C 8379
1307 0A0E C060          MOV  @DELAY1,R1      DELAY COUNTER TO MATCH 4A SPEED
     0A10 849E
1308 0A12 1302          JEQ  USERI
1309 0A14 0601  LOOP4A  DEC  R1
1310 0A16 16FE          JNE  LOOP4A
1311 0A18 C320  USERI   MOV  @INTPTR,R12    USER INTERRUPT ROUTINE?
     0A1A 83C4
1312 0A1C 1301          JEQ  EMERGE
1313 0A1E 069C          BL   *R12
1314 0A20 04C8  EMERGE  CLR  R8             CLEAR REGISTER FOR BASIC
1315 0A22 02E0          LWPI INTWSP         RETURN TO CALLING ROUTINE
     0A24 83C0
1316 0A26 0380  NULLRT  RTWP                FROM ALTERNATE WORKSPACE
```

```
1318                    *
1319                    *       LINK TO ROUTINE IN ROM
1320                    *
1321                    *     ENTER WITH PAB POINTER IN >8356, (SCNAM,WORD)
1322                    *     AND ROUTINE TYPE IN >836D (TYPE,BYTE)
1323                    *        LNKVRM:  PAB AND BUFFER ARE IN VDP RAM
1324                    *        LNKCRM:  PAB AND BUFFER ARE IN CPU RAM
1325                    *     FALLS THROUGH INTO SROM IF NAME IS BETWEEN 1 AND 10 CHAR
1326                    *      (WHICH RETURNS FROM A GPL CALL IS A MATCH IS FOUND)
1327                    *     OTHERWISE RETURNS WITH CONDITION BIT SET
1328                    *
1329 0A28 0460  LNKR9   B     @SET
     0A2A 00CE'
1330                    * actual entry point *****
1331                    ***LRVM    MOVB *R4,R1
1332                    ***LRCM    MOVB *R4+,R1
1333 0A2C D7E0  LNKVRM MOVB  @SCNAM+1,*R15    Fetch pointer into PAB
     0A2E 8357
1334 0A30 D7E0          MOVB  @SCNAM,*R15
     0A32 8356
1335 0A34 D0EF          MOVB  @VDPRD(R15),R3    fetch the name length byte
     0A36 F8FE
1336                    ***     MOV   @LRVM,R0
1337 0A38 0200          LI    R0,>D054    (MOVB *R4,R1)
     0A3A D054
1338                    ***     MOV   R15,R4
1339                    ***     AI    R4,VDPRD
1340 0A3C 0204          LI    R4,VDPRD+VDPWA
     0A3E 8800
1341 0A40 04E0          CLR   @SCLEN-1
     0A42 8354
1342 0A44 1008          JMP   LNKR0
1343 0A46 C120  LNKCRM MOV   @SCNAM,R4
     0A48 8356
1344 0A4A D0F4          MOVB  *R4+,R3
1345                    ***     MOV   @LRCM,R0
1346 0A4C 0200          LI    R0,>D074    (MOVB *R4+,R1)
     0A4E D074
1347 0A50 C820          MOV   @H8000,@SCLEN-1
     0A52 037C'
     0A54 8354
1348 0A56 0983  LNKR0   SRL   R3,8                Make length a word value
1349 0A58 13E7          JEQ   LNKR9               if length=0, done
1350 0A5A 0202          LI    R2,FAC              buffer for name
     0A5C 834A
1351 0A5E 04C1          CLR   R1
1352 0A60 0480  LNKR1   X     R0                  Fetch a byte of name
1353 0A62 0281          CI    R1,'. '*256         Is is a period?
     0A64 2E00
1354 0A66 1306          JEQ   LNKR2               if so, stop accumulating name
1355 0A68 0282          CI    R2,FAC+9            is the name too long?
     0A6A 8353
1356 0A6C 18DD          JH    LNKR9               if > 10 bytes
1357 0A6E DC81          MOVB  R1,*R2+             Put it in the buffer
1358 0A70 0603          DEC   R3                  Finished with name?
1359 0A72 15F6          JGT   LNKR1               if not
1360 0A74 0222  LNKR2   AI    R2,-FAC             R2 contains length up to period
     0A76 7CB6
1361 0A78 E802          SOC   R2,@SCLEN-1         Save the length, but save memor
     0A7A 8354
```

```
1362 0A7C 0582          INC  R2
1363 0A7E A802          A    R2,@SCNAM          And update the name pointer
     0A80 8356
1364 0A82 04E0          CLR  @CRULST            indicate beginning of search
     0A84 83D0
1365                    *  fall through into SROM
1366                    *
1367                    *      SEARCH ROM CROM GROM FOR DSR OR LINK
1368                    *
1369 0A86 04C1  SROM    CLR  R1                 VERSION FOUND OF DSR ETC
1370 0A88 C320          MOV  @CRULST,R12        SEARCH ROM FOR ROUTINE
     0A8A 83D0
1371 0A8C 1617          JNE  SGO                IF <> 0 THEN CONTINUE SEARCH
1372 0A8E 020C          LI   R12,>0F00          START OVER AGAIN
     0A90 0F00
1373 0A92 C30C  NOROM   MOV  R12,R12            ANYTHING TO TURN OFF?
1374 0A94 1301          JEQ  NOOFF              NO
1375 0A96 1E00          SBZ  0                  YES, TURN IT OFF
1376                    ***NOOFF  AI   R12,>0100    NEXT ROM'S TURN ON
1377 0A98 B30E  NOOFF   AB   R14,R12            NEXT ROM'S TURN ON
1378 0A9A 04E0          CLR  @CRULST            CLEAR JUST IN CASE WE'RE FINISHED
     0A9C 83D0
1379                    *<><><><><><><><><><><><><><><><><><><><><><><><><><>
1380                    * CRU SPACE EXTENDED FOR 99/8 (9995)
1381                    *<><><><><><><><><><><><><><><><><><><><><><><><><><>
1382 0A9E 028C          CI   R12,>3000          AT THE END
     0AA0 3000
1383 0AA2 1320          JEQ  NOSET              NO MORE ROMS TO TURN ON
1384 0AA4 C80C          MOV  R12,@CRULST        SAVE ADDRESS OF NEXT CRU
     0AA6 83D0
1385 0AA8 1D00          SBO  0                  TURN ON ROM
1386 0AAA 0202          LI   R2,>4000           START AT BEGINNING
     0AAC 4000
1387 0AAE 9812  SG01    CB   *R2,@HAA           IS IT A VALID ROM?
     0AB0 000D'
1388 0AB2 16EF          JNE  NOROM              NO
1389 0AB4 B820  SG01A   AB   @TYPE,@R2LB        GO TO FIRST POINTER
     0AB6 836D
     0AB8 83E5
1390 0ABA 1003          JMP  SG02
1391 0ABC C0A0  SGO     MOV  @SADDR,R2          CONTINUE WHERE WE LEFT OFF
     0ABE 83D2
1392 0AC0 1D00          SBO  0                  TURN ROM BACK ON
1393 0AC2 C092  SG02    MOV  *R2,R2             IS ADDRESS A ZERO
1394 0AC4 13E6          JEQ  NOROM              YES, NO PROGRAM TO LOOK AT
1395 0AC6 C802          MOV  R2,@SADDR          REMEMBER WHERE WE GO NEXT
     0AC8 83D2
1396 0ACA 05C2          INCT R2                 GO TO ENTRY POINT
1397 0ACC C272          MOV  *R2+,R9            GET ENTRY ADDRESS
1398 0ACE 06A0          BL   @NAME              SEE IF NAME MATCHES
     0AD0 089A'
1399 0AD2 10F4          JMP  SGO                NO MATCH, TRY NEXT PROGRAM
1400 0AD4 0581          INC  R1                 NEXT VERSION FOUND
1401 0AD6 0699          BL   *R9                MATCH, CALL SUBROUTINE
1402 0AD8 10F1          JMP  SGO                NOT RIGHT VERSION
1403 0ADA 1E00          SBZ  0                  TURN OFF ROM
1404 0ADC 1001          JMP  NOGR2              SET STATUS AND GO TO GPL
1405 0ADE 04D8  NOGR1   CLR  *R8                NO MORE TO DO
1406 0AE0 06A0  NOGR2   BL   @GETSTK            RESTORE GROM ADDRESS
     0AE2 0492'
```

```
1407 0AE4 0460  NOSET  B     @RESET        INDICATE NO MORE
     0AE6 006A'
1408            *==============================================================*
1409            * spgrom is an xml3
1410 0AE8 0200  SPGROM LI    R0,>F500      P-SYSTEM VALIDATION FLAG
     0AEA F500
1411 0AEC 1002         JMP   SGROM0        SHARE CODE
1412 0AEE 0200  SGROM  LI    R0,>AA00      GPL VALIDATION FLAG
     0AF0 AA00
1413 0AF2 0207  SGROM0 LI    R7,SADDR
     0AF4 83D2
1414 0AF6 0208         LI    R8,CRULST
     0AF8 83D0
1415 0AFA 06A0         BL    @PUTSTK       SAVE GROM ADDRESS
     0AFC 04BC'
1416 0AFE C057  SGROMA MOV   *R7,R1        START WHERE WE LEFT OFF
1417 0B00 C098         MOV   *R8,R2        IS IT A RESTART?
1418 0B02 1604         JNE   SGROM3        NO
1419 0B04 0202         LI    R2,GRMRD      START OF GROMS
     0B06 9800
1420 0B08 0201  SGROM1 LI    R1,>E000      START OF GROM
     0B0A E000
1421 0B0C 2460  SGROM3 CZC   @H1FFF,R1     IS IT A NEW GROM OR A CONTINUATION
     0B0E 0126'
1422 0B10 160D         JNE   SGROM2
1423 0B12 C602         MOV   R2,*R8        SAVE GROM BASE ADDRESS
1424 0B14 D881         MOVB  R1,@GRMWA(R2) LOAD ADDRESS
     0B16 0402
1425 0B18 D8A0         MOVB  @R1LB,@GRMWA(R2)
     0B1A 83E3
     0B1C 0402
1426 0B1E B820         AB    @TYPE,@R1LB   LOOK FOR PROGRAM ADDRESS
     0B20 836D
     0B22 83E3
1427 0B24 D801         MOVB  R1,@SAVEG     SAVE GROM ADDRESS OF HEADER
     0B26 83CB
1428 0B28 9012         CB    *R2,R0        VALID GROM?
1429 0B2A 162D         JNE   NOGR          NO GROM HERE
1430 0B2C D881  SGROM2 MOVB  R1,@GRMWA(R2) LOOK FOR PROGRAM
     0B2E 0402
1431 0B30 D8A0         MOVB  @R1LB,@GRMWA(R2)
     0B32 83E3
     0B34 0402
1432 0B36 D0D2         MOVB  *R2,R3        READ PROGRAM ADDRESS
1433 0B38 D812         MOVB  *R2,@R3LB
     0B3A 83E7
1434 0B3C C5C3         MOV   R3,*R7        GET NEXT HEADER'S ADDRESS
1435 0B3E 1323         JEQ   NOGR          IF ZERO, GO TO NEXT GROM
1436 0B40 05C3         INCT  R3            GO TO PROGRAM ENTRY ADDRESS
1437 0B42 D883         MOVB  R3,@GRMWA(R2) GO TO PROGRAM ENTRY ADDRESS
     0B44 0402
1438 0B46 D8A0         MOVB  @R3LB,@GRMWA(R2)
     0B48 83E7
     0B4A 0402
1439 0B4C D252         MOVB  *R2,R9        ENTRY ADDRESS
1440 0B4E D812         MOVB  *R2,@R9LB
     0B50 83F3
1441 0B52 06A0         BL    @NAME         SEE IF NAME MATCHES
     0B54 0B9A'
1442 0B56 10D3         JMP   SGROMA        NO LOOK FOR NEXT PROGRAM
```

```
 1443 0B58 B820          AB   @H2,@STKDAT  FOUSP NAME SO PUSH IT
      0B5A 0D7C'
      0B5C 8372
 1444 0B5E B80E          AB   R14,@TEMP2   INCREASE PROGRAM COUNT
      0B60 836C
 1445 0B62 D120          MOVB @STKDAT,R4
      0B64 8372
 1446 0B66 0984          SRL  R4,8
 1447 0B68 0643          DECT R3           POINT BACK TO START OF HEADER
 1448 0B6A 9820          CB   @TYPE,@H06   IS IT A USER PROGRAM  LOOKUP?
      0B6C 836D
      0B6E 0B86'
 1449 0B70 1601          JNE  SGROM4       YES
 1450 0B72 C243          MOV  R3,R9        PUSH HEADER ADDRESS FOR USER PROGRAM
 1451 0B74 D909   SGROM4 MOVB R9,@PAD(R4)  NO, PUSH ENTRY ADDRESS
      0B76 8300
 1452 0B78 D920          MOVB @R9LB,@PAD+1(R4)
      0B7A 83F3
      0B7C 8301
 1453             *****  MOV  R2,R13       GO TO THAT LIBRARY
 1454 0B7E 06A0   SGSET  BL   @GETSTK      RESTORE GROM ADDRESS
      0B80 0492'
 1455 0B82 0460          B    @SET         SET STATUS AND RETURN
      0B84 00CE'
 1456 0B86 04C1   NOGR   CLR  R1           GET ADDRESS OF GROM HEADER
 1457 0B88 D060          MOVB @SAVEG,R1
      0B8A 83CB
 1458 0B8C 0221          AI   R1,->2000    NEXT GROM DOWN
      0B8E E000
 1459 0B90 C5C1          MOV  R1,*R7       SAVE ADDRESS OF WHERE WE'RE AT
 1460 0B92 0281          CI   R1,>E000     FINISHED?
      0B94 E000
 1461 0B96 16BA          JNE  SGROM3       NO, CHECK THIS GROM
 1462             *<><><><><><><><><><><><><><><><><><><><><><><><><><><>
 1463             *  GROM LIBRARY SUPPORT REMOVED FROM 99/8
 1464             *<><><><><><><><><><><><><><><><><><><><><><><><><><><>
 1465             ***    C    *R2+,*R2+    INCREMENT GROM MAPPED ADDRESS BY FOU
 1466             ***    MOV  R2,*R8       SAVE THE NEW MAP ADDRESS
 1467             ***    CI   R2,GRMRD+>40 AT END OF LIBRARY?
 1468             ***    JEQ  NOGR1        YES
 1469 0B98 10A2          JMP  NOGR1        FINISHED
 1470             ***    MOVB @SCLEN,R5    ARE WE LOOKING FOR A MENU?
 1471             ***    JNE  SGROM1       YES SO DO ONLY ONE SLOT
 1472             ***    JMP NOGR2         NO CONTINUE SEARCH
 1473             *
 1474             *** TRY TO MATCH NAME FOR SROM,SGROM
 1475             *
 1476 0B9A D160   NAME   MOVB @SCLEN,R5    GET LENGTH AS COUNTER
      0B9C 8355
 1477 0B9E 130D          JEQ  NAME2A       ZERO LENGTH, DON'T DO MATCH
 1478 0BA0 9485          CB   R5,*R2       DOES LENGTH MATCH?
 1479 0BA2 160C          JNE  NAME3        NO
 1480 0BA4 0985          SRL  R5,8         MOVE TO RIGHT PLACE
 1481 0BA6 0206          LI   R6,FAC
      0BA8 834A
 1482 0BAA 0282   NAME1  CI   R2,GRMRD     IS IT GROM?
      0BAC 9800
 1483 0BAE 1401          JHE  NAME2        YES, DON'T INCREMENT ADDRESS
 1484 0BB0 0582          INC  R2
 1485 0BB2 94B6   NAME2  CB   *R6+,*R2     IS NAME THE SAME?
```

```
1486 0BB4 1603          JNE  NAME3          NO
1487 0BB6 0605  H06     DEC  R5             MORE TO LOOK AT ?
1488 0BB8 16F8          JNE  NAME1          YES
1489 0BBA 05CB  NAME2A  INCT R11            RETURN , NAME FOUND
1490 0BBC 045B  NAME3   RT                  RETURN
1491                    *
1492                    *  RANDOM NUMBER GENERATOR, FOR RAND INSTRUCTION IN GPL
1493                    *  AND RND FUNCTION IN BASICS
1494 0BBE 0204  RND     LI   R4,28645
     0BC0 6FE5
1495 0BC2 3920          MPY  @RAND16,R4
     0BC4 83C0
1496 0BC6 0225          AI   R5,31417
     0BC8 7AB9
1497 0BCA C805          MOV  R5,@RAND16
     0BCC 83C0
1498 0BCE 0586          INC  R6
1499 0BD0 04C4          CLR  R4             CLEAR UPPER HALF OF DIVIDEND
1500 0BD2 06C5          SWPB R5
1501 0BD4 3D06          DIV  R6,R4          PERFORM DIVISION
1502 0BD6 D820          MOVB @R5LB,@RANDOM      STORE REMAINDER
     0BD8 83EB
     0BDA 8378
1503 0BDC 045B          RT
1504                    *
1505                    ** INTERFACE TO DEBUGGER
1506                    *
1507      4020  OPBR    EQU  >4020          ADDRESS TO HANDLF OPERR IN DSR
1508 0BDE 06A0  OPERR   BL   @SETUP
     0BE0 0BFA'
1509 0BE2 0460          B    @OPBR
     0BE4 4020
1510      401C  BPBR    EQU  >401C          ADDRESS TO HANDLF BP IN DSR
1511 0BE6 06A0  BP      BL   @SETUP
     0BE8 0BFA'
1512 0BEA 0460          B    @BPBR
     0BEC 401C
1513 0BEE 02E0  PRCXOP  LWPI DSRWSP
     0BF0 2800
1514 0BF2 06A0          BL   @SETUP
     0BF4 0BFA'
1515 0BF6 0460          B    @ALBRK
     0BF8 4028
1516 0BFA 020C  SETUP   LI   R12,>1B00      SET UP CRU
     0BFC 1B00
1517 0BFE 1D00          SBO  0
1518 0C00 045B          RT
1519                    *
1520                    ***  GPLLNK  - ROUTINE TO CALL GPL FROM 9900 CODE
1521                    ***  CALLER SHOULD SET UP GPL ADDRESS IN GLINK1 (>84A6)
1522                    ***  >84A4 IS USED.
1523                    ***  CALL SEQUENCE IS BL @GPLLNK FROM GPLWS
1524                    *
1525 0C02 C80B  GPLLKO MOV  R11,@GLINKO  RETURN TO CALLER IS VIA XML >A0 FROM
     0C04 84A4
1526 0C06 0206          LI   R6,>42+MONIT   GPL ADDRESS OF LINK ROUTINE
     0C08 0042
1527 0C0A 0460          B    @LDKADD        START EXECUTING GPL
     0C0C 0060'
1528                    *          0     1     2     3
```

```
1529 0C0E FF00  MAPDAT DATA >FF00,>FF00,>0008,>0018
     0C10 FF00
     0C12 0008
     0C14 0018
1530                *            4     5     6     7
1531 0C16 FF40           DATA >FF40,>FF50,>FF60,>FF70
     0C18 FF50
     0C1A FF60
     0C1C FF70
1532                *            8     9     A     B
1533 0C1E FF00           DATA >FF00,>FF00,>0028,>0038
     0C20 FF00
     0C22 0028
     0C24 0038
1534                *            C     D     E     F
1535 0C26 0048           DATA >0048,>0058,>0068,>0078
     0C28 0058
     0C2A 0068
     0C2C 0078
1536                *=============================================================*
1537                *** CAUTION !!!!! BE CAREFUL OF CODE OVERRUN
1538 0C36                RORG >C36          (COMPATABILITY WITH 99/4 AND 4A)
1539                *                BRANCH TABLES
1540 0C36 0270' ITAB     DATA MISCLN,MOVDAT,BRESET,BSET                    >0
     0C38 02FC'
     0C3A 0118'
     0C3C 010C'
1541 0C3E 0488' MSCTAB   DATA RETURN,RETNC,RANDNO,KEYSCN,BKGRND,BLONG,CALL,ALL
     0C40 048E'
     0C42 027A'
     0C44 0562'
     0C46 0284'
     0C48 0104'
     0C4A 04B2'
     0C4C 081A'
1542 0C4E 0756'           DATA FORMAT,TSTST,TSTST,BUGOUT,TSTST,TSTST,PARSEG,XML
     0C50 00F4'
     0C52 00F4'
     0C54 09D8'
     0C56 00F4'
     0C58 00F4'
     0C5A 0000
     0C5C 0D9C'
1543                *        DATA CONTG,EXECG,RTNG,RGBA,OPERR,OPERR,OPERR,OPERR
1544                ****     XML2 ADDED ATA 4/9/82 -- MORE XML TABLES FOR ARMADILLO
1545 0C5E 0000           DATA CONTG,EXECG,RTNG,RGBA,XML2,XML3,RTNSET,OPERR
     0C60 0000
     0C62 0000
     0C64 047C'
     0C66 0DAA'
     0C68 0DC8'
     0C6A 04AA'
     0C6C 0BDE'
1546 0C6E 0BDE'           DATA OPERR,OPERR,OPERR,OPERR,OPERR,OPERR,OPERR,BP
     0C70 0BDE'
     0C72 0BDE'
     0C74 0BDE'
     0C76 0BDE'
     0C78 0BDE'
     0C7A 0BDE'
```

```
      0C7C 0BE6'
1547        0BFE' ATAB   EQU  $->80
1548 0C7E 0138'          DATA ABS,NEG,INV,CLR,FETCH,CASE,PUSH,IFZ
      0C80 013C'
      0C82 0142'
      0C84 0140'
      0C86 0146'
      0C88 0164'
      0C8A 016E'
      0C8C 00EA'
1549 0C8E 0186'          DATA INC,DEC,INCT,DECT,OPERR,OPERR,OPERR,OPERR
      0C90 0188'
      0C92 0184'
      0C94 0182'
      0C96 0BDE'
      0C98 0BDE'
      0C9A 0BDE'
      0C9C 0BDE'
1550        0C4E' BTAB   EQU  $->50
1551 0C9E 0188'          DATA ADD,MINUS,MUL,DIV,AND,OR,XOR,STORE
      0CA0 0186'
      0CA2 01CE'
      0CA4 01EA'
      0CA6 0190'
      0CA8 0196'
      0CAA 019A'
      0CAC 019E'
1552 0CAE 01A2'          DATA EXCH,HIGH,HIGHEQ,GREATR,GREQ,EQUAL,IFAND
      0CB0 00D6'
      0CB2 00DA'
      0CB4 00DE'
      0CB6 00CC'
      0CB8 00EC'
      0CBA 00E2'
1553 0CBC 01B0'          DATA SRA,SLL,SRL,SRC,COINC,OPERR,INOUT,DGBA,OPERR
      0CBE 01B4'
      0CC0 01B8'
      0CC2 01C2'
      0CC4 0E5A'
      0CC6 0BDE'
      0CC8 02BE'
      0CCA 0D86'
      0CCC 0BDE'
1554 0CCE 033C' MSRC   DATA CPUSRC,GRMSRC,VDPSRC
      0CD0 034E'
      0CD2 0340'
1555 0CD4 035E' MDST   DATA CPUDST,GRMDST,VDPDST,REGDST
      0CD6 0362'
      0CD8 0392'
      0CDA 0374'
1556 0CDC 0782' FBTAB  DATA STRACR,STRDWN,RPTACR,RPTDWN
      0CDE 0780'
      0CE0 077C'
      0CE2 077A'
1557 0CE4 07AC'          DATA SKPACR,SKPDWN,RPTBLK,SPECL
      0CE6 07AA'
      0CE8 07B2'
      0CEA 07E4'
1558 0CEC 02CC' IOTAB  DATA SOUND,SOUND,CRUIN,CRUOUT
      0CEE 02CC'
```

```
      0CF0 02DC'
      0CF2 02DE'
1559  0CF4 0000          DATA WRITE,READ,VERIFY
      0CF6 0000
      0CF8 0000
1560  0CFA 0000  XMLTAB  DATA FLTTAB,XTAB,>2000,>3FC0
      0CFC 0000
      0CFE 2000
      0D00 3FC0
1561  0D02 3FE0          DATA >3FE0,>4010,>4030,>6010
      0D04 4010
      0D06 4030
      0D08 6010
1562  0D0A 6030          DATA >6030,>7000,GLINK0,>A000
      0D0C 7000
      0D0E 84A4
      0D10 A000
1563       0D14' HC000   EQU  $+2
1564  0D12 B000          DATA >B000,>C000,>D000,PAD
      0D14 C000
      0D16 D000
      0D18 8300
1565  0D1A 0000  XMLTB2  DATA XTAB2,XTAB3,XTAB4,XTAB5
      0D1C 0000
      0D1E 0000
      0D20 0000
1566  0D22 0000          DATA XTAB6,XTAB7,XTAB8,XTAB9
      0D24 0000
      0D26 0000
      0D28 0000
```

```
1568                    ************************************************************
1569                    *  POWER-UP AND RESTART INITIALIZATION                     *
1570                    ************************************************************
1571          0D2A'     ENTRY   EQU   $
1572 0D2A 04CC                   CLR   R12
1573 0D2C 1D14                   SBO   >14             MMD AT >8000, ROM ENABLED P4
1574 0D2E 1D15                   SBO   >15             NO P-CODE GROMS           P5
1575 0D30 D020                   MOVB  @>2000,R0       dummy read to initialize state of ma
     0D32 2000
1576                    * NOW INITIALIZE MEMORY MAPPER
1577 0D34 0201                   LI    R1,MF0
     0D36 8000
1578 0D38 0202                   LI    R2,MAPDAT
     0D3A 0C0E'
1579 0D3C 0203                   LI    R3,16
     0D3E 0010
1580 0D40 04C4                   CLR   R4
1581 0D42 DC44            ENTR0  MOVB  R4,*R1+
1582 0D44 DC72                   MOVB  *R2+,*R1+
1583 0D46 DC72                   MOVB  *R2+,*R1+
1584 0D48 DC44                   MOVB  R4,*R1+
1585 0D4A 0603                   DEC   R3
1586 0D4C 16FA                   JNE   ENTR0
1587 0D4E D820                   MOVB  @LMAP0,@MAPPER
     0D50 0D68'
     0D52 8810
1588          0D55'     H0D     EQU   $+1
1589 0D54 020D            ENTR1  LI    R13,GRMRD      GPL REGISTERS
     0D56 9800
1590 0D58 C08D                   MOV   R13,R2         SYNCHRONIZE ALL GROM LIBRARIES
1591 0D5A D012            ENTR2  MOVB  *R2,R0
1592 0D5C 0222                   AI    R2,4
     0D5E 0004
1593 0D60 0282                   CI    R2,GRMRD+>40
     0D62 9840
1594 0D64 16FA                   JNE   ENTR2
1595          0D68'     LMAP0   EQU   $+2            COMMAND CODE TO LAD MAP FROM FILE 0
1596          0D69'     RMAP0   EQU   $+3
1597 0D66 020E                   LI    R14,>0100
     0D68 0100
1598 0D6A 020F                   LI    R15,VDPWA
     0D6C 8C02
1599 0D6E 04E0                   CLR   @DELAY1        FULL SPEED
     0D70 849E
1600 0D72 0200                   LI    R0,1500
     0D74 05DC
1601 0D76 C800                   MOV   R0,@DELAY2     KEYBOARD DEBOUNCE
     0D78 849C
1602                    ****     CLR   @CLICK         assume no click
1603 0D7A 2D44                   XOP   4,5            FIND TOP OF MEMORY
1604          0D7C'     H2      EQU   $
1605          0D7C'     IMMOPS  EQU   $
1606          0D7E'     H20     EQU   $+2
1607 0D7C 0200                   LI    R0,>20+MONIT    FIRST  BYTE IN GROM
     0D7E 0020
1608 0D80 04E0                   CLR   @INTPTR         CLEAR USER INTERRUPT POINTER
     0D82 83C4
1609 0D84 1007                   JMP   DGBADD
1610                    *--------------------------------------------------------------*
1611                    *       ***   BEGIN EXECUTION OF GPL INSTRUCTIONS ***
```

```
1612                    *                   LIBRARY CALL ROUTINE
1613 0D86 06A0  DGBA    BL    @PUTSTK       SAVE RETURN ADDRESS
     0D88 04BC'
1614 0D8A 06A0          BL    @PUTSTK       INCREMENT STACK POINTER
     0D8C 04BC'
1615 0D8E C90D          MOV   R13,@PAD(R4)  SAVE GROM BASE ADDRESS
     0D90 8300
1616 0D92 C342          MOV   R2,R13        NEW GROM BASE ADDRESS
1617 0D94 D11D  DGBADD  MOVB  *R13,R4       SYNCHRONIZE YOUR GROMS
1618 0D96 C180          MOV   R0,R6         BRANCH TO ADDRESS IN GROMS
1619 0D98 0460          B     @LDKADD
     0D9A 0060'
```

```
1621                    *===============================================================*
1622                    *          EXECUTE 9900 CODE IN EXTERNAL ROM
1623 0D9C 0202  XML     LI    R2,XMLTAB
     0D9E 0CFA'
1624 0DA0 06A0  XML0    BL    @XMLCOM
     0DA2 0DCE'
1625 0DA4 0694          BL    *R4
1626 0DA6 0460  GEXIT2  B     @NEXT
     0DA8 0070'
1627                    * SECOND XML INSTRUCTION ADDED FOR ARMADILLO FOR MORE TABLES
1628                    * ESPECIALLY FOR ROM AT >FF8000.  MAP IT TO >2000
1629 0DAA 06A0  XML2    BL    @X2COM
     0DAC 0E2C'
1630 0DAE 06A0          BL    @XMLCOM
     0DB0 0DCE'
1631 0DB2 0694          BL    *R4
1632 0DB4 06A0          BL    @X2END
     0DB6 0DBA'
1633 0DB8 10F6          JMP   GEXIT2
1634 0DBA               X2END
1635 0DBA 020C          LI    R12,>2700
     0DBC 2700
1636 0DBE 1E00          SBZ   0
1637 0DC0 D820          MOVB  @LMAP0,@MAPPER
     0DC2 0D68'
     0DC4 8810
1638 0DC6 045B          RT
1639                    * THIRD XML INSTRUCTION TO ACCESS ROUTINES AT >2000
1640                    * WITHOUT REMAPPING MEMORY
1641 0DC8 0202  XML3    LI    R2,XMLTB2
     0DCA 0D1A'
1642 0DCC 10E9          JMP   XML0
1643 0DCE D25D  XMLCOM  MOVB  *R13,R9       FETCH SELECTOR BYTE FROM GROM
1644 0DD0 C109  XMLCM1  MOV   R9,R4
1645 0DD2 09B9          SRL   R9,11         R9=LIST SELECTOR * 2      (LSB DOESN'
1646 0DD4 0A44          SLA   R4,4
1647 0DD6 09B4          SRL   R4,11         R4= SELECTOR WITHIN LIST (LSB DOESN'
1648 0DD8 A242          A     R2,R9         PLUS TABLE OFFSET
1649 0DDA A119          A     *R9,R4        GET TABLE POINTER IN R4
1650 0DDC C114          MOV   *R4,R4        NOW ACTUAL CODE POINTER
1651 0DDE 045B          RT
1652                    *
1653                    *** XOP ENTRY POINT FOR CALLING CERTAIN XML ROUTINES
1654                    *** FROM ASSEMBLY LANGUAGE
1655                    *** WORKSPACE IS SCRWS
1656                    *
1657 0DE0 D05D  XXML    MOVB  *R13,R1   FIRST DATA BYTE = WHICH XML  (R0)
1658 0DE2 0971          SRL   R1,7
1659 0DE4 C061          MOV   @XXTAB(R1),R1
     0DE6 0DE8'
1660 0DE8 0451          B     *R1
1661      0DE8' XXTAB   EQU   $-2
1662 0DEA 0DF0'         DATA  XXML1,XXML2,XXML3
     0DEC 0E0E'
     0DEE 0E08'
1663 0DF0 0202  XXML1   LI    R2,XMLTAB   STANDARD XML
     0DF2 0CFA'
1664 0DF4 06A0  XXML10  BL    @XXCOM      GET POINTER
     0DF6 0E26'
1665 0DF8 C804          MOV   R4,@GPLWS+8   ALL ASSUME GPLWS
```

```
      0DFA 83E8
1666  0DFC 02E0          LWPI GPLWS
      0DFE 83E0
1667  0E00 0694          BL   *R4          DO ROUTINE
1668  0E02 02E0  XXML11  LWPI SCRWS        NOW RETURN
      0E04 8458
1669  0E06 0380          RTWP
1670  0E08 0202  XXML3   LI   R2,XMLTB2    SECONDARY TABLES
      0E0A 0D1A'
1671  0E0C 10F3          JMP  XXML10       (ASSUME ALREADY MAPPED IN)
1672  0E0E 06A0  XXML2   BL   @X2COM       MAP IN ROMS
      0E10 0E2C'
1673  0E12 06A0          BL   @XXCOM       GET ADDRESS
      0E14 0E26'
1674  0E16 C804          MOV  R4,@GPLWS+8    ALL ASSUME GPLWS
      0E18 83E8
1675  0E1A 02E0          LWPI GPLWS
      0E1C 83E0
1676  0E1E 0694          BL   *R4          DO ROUTINE
1677  0E20 06A0          BL   @X2END
      0E22 0DBA'
1678  0E24 10EE          JMP  XXML11       AND RETURN
1679  0E26 D26D  XXCOM   MOVB @1(R13),R9    FETCH ROUTINE SELECTOR (ROLB)
      0E28 0001
1680  0E2A 10D2          JMP  XMLCM1       (RETURNS)
1681  0E2C 0202  X2COM   LI   R2,XMLTB2
      0E2E 0D1A'
1682  0E30 0203          LI   R3,ROMFF8
      0E32 0F06'
1683  0E34 0205          LI   R5,MAPPER
      0E36 8810
1684  0E38 D560          MOVB @RMAP0,*R5
      0E3A 0D69'
1685  0E3C D560          MOVB @RMAP1,*R5
      0E3E 0073'
1686  0E40 0201          LI   R1,>8048        >2000 WINDOW IM MAP 1
      0E42 8048
1687  0E44 0204          LI   R4,12           12 WORDS FOR >2000 TO >7FFF
      0E46 000C
1688  0E48 CC73  XML2C   MOV  *R3+,*R1+
1689  0E4A 0604          DEC  R4
1690  0E4C 16FD          JNE  XML2C
1691  0E4E D560          MOVB @LMAP1,*R5
      0E50 0074'
1692  0E52 020C          LI   R12,>2700
      0E54 2700
1693  0E56 1D00          SBO  0
1694  0E58 045B          RT
1695                     ********************************************************************
1696                     *                                                    4/14/78
1697                     *          REVISED FROM COINC5 FOR GB REV E   8/09/78
1698                     * COINCIDENCE ROUTINE FOR INSERTION INTO REL4 INTERPRETER.
1699                     * UPON ENTRANCE TO THIS ROUTINE AT LABEL 'COINC' THE
1700                     *  REGISTERS ARE ASSUMED TO BE SET UP:
1701                     *MSBY  R2 = Y2 IN MSBY AND X2 IN LSBY;
1702                     *MSBY  R0 = Y1 IN MSBY AND X1 IN LSBY;
1703                     *
1704                     * IT IS ALSO ASSUMED THAT THE GROM'S INTERNAL ADDRESS IS SET
1705                     * UP PREPARED TO READ (FOLLOWING THE COINC INSTRUCTION   ):
1706                     *      - A ONE BYTE GRANULARITY VALUE, FOLLOWED BY:
```

```
1707                *       - A TWO BYTE ADDRESS POINTING TO THE COINCIDENCE
1708                *           TABLE.  THE TABLE IS ASSUMED TO RESIDE IN GROM,
1709                *           AND HAVE THE FOLLOWING FORMAT:
1710                *
1711                *           BYTE 0- TV = VERTICAL BIT SIZE OF TABLE LESS 1;
1712                *           BYTE 1- TH = HORIZ.   BIT SIZE OF TABLE LESS 1;
1713                *           BYTE 2- V1 = VERTICAL DOT SIZE OF OBJECT 1/2**GRA
1714                *           BYTE 3- H1 = HORIZ.   DOT SIZE OF OBJECT 1/2**GRA
1715                *           BYTES 4 ON - THE BIT TABLE ITSELF;  THE BITS ARE
1716                *                ARRANGED SUCH THAT THE FIRST (TH+1) BITS
1717                *                REPRESENT BOOLEAN COINCIDENCE VALUES
1718                *                CORRESPONDING TO A DELTA Y (Y1-Y2) OF
1719                *                -V1 THRU -V1+TV AND DELTA X (X1-X2) VALUES O
1720                *                -H1 THROUGH -H1+TH.
1721                *
1722 0E5A C200    COINC   MOV   R0,R8
1723 0E5C C0C8            MOV   R8,R3           FIRST GET DELTA Y AND DELTA X:
1724 0E5E 70C2            SB    R2,R3           R3 = Y1-Y2 = DELTA Y.
1725 0E60 06C8            SWPB  R8              GET X1 IN MSBY.
1726 0E62 06C2            SWPB  R2              GET X2 IN MSBY.
1727 0E64 7202            SB    R2,R8           R8 X1-X2 = DELTA X.
1728 0E66 D01D            MOVB  *R13,R0          SET RESOLUTION AND TABLE POINTER
1729 0E68 0980            SRL   R0,8            R0 = GRAN.
1730 0E6A D15D            MOVB  *R13,R5
1731 0E6C D81D            MOVB  *R13,@R5LB
     0E6E 83EB
1732 0E70 06A0            BL    @PUTSTK            SAVE GROM PC.
     0E72 04BC'
1733                *
1734                * NOW GET TV,TH,V1,H1 OUT OF THE FIRST 4 BYTES OF TABLE.
1735                *
1736 0E74 DB45            MOVB  R5,@GRMWA(R13)   PUT OUT TABLE PTR. LSBY.
     0E76 0402
1737 0E78 DB60            MOVB  @R5LB,@GRMWA(R13)   PUT OUT TABLE PTR. MSBY.
     0E7A 83EB
     0E7C 0402
1738 0E7E D09D            MOVB  *R13,R2          R2 = TV (MSBY).
1739 0E80 D05D            MOVB  *R13,R1          R1 = TH (MSBY).
1740 0E82 D19D            MOVB  *R13,R6          R6 = V1 (MSBY).
1741 0E84 D1DD            MOVB  *R13,R7          R7 = H1 (MSBY).
1742                *
1743                * NOW ON WITH THE SHOW.   THE REGISTERS ARE NOW SET UP AS:
1744                *
1745                *       R0 = GRANULARITY;
1746                *MSBY   R1 = TH = COINCIDENCE TABLE HORIZONTAL SIZE - 1.
1747                *MSBY   R2 = TV = COINCIDENCE TABLE VERTICAL   SIZE - 1.
1748                *MSBY   R3 = Y1 - Y2 = DELTA Y;
1749                *MSBY   R8 = X1 - X2 = DELTA X;
1750                *       R5 = POINTER TO COINCIDENCE TABLE IN GROM.
1751                *MSBY   R6 = V1 = VERTICAL SIZE OF OBJECT ONE IN DOTS.
1752                *MSBY   R7 = H1 = HORIZ.    SIZE OF OBJECT ONE IN DOTS.
1753                *       R13= GRMRD.
1754                *
1755 0E86 C000            MOV   R0,R0          IF GRANULARITY IS 0, DON'T SHIFT
1756 0E88 1302            JEQ   DNTSHF              BECAUSE 9900 SHIFT BY 0 IS FU
1757 0E8A 0803            SRA   R3,R0       DIVIDE DELTA Y BY (2** GRAN).
1758 0E8C 0808            SRA   R8,R0       DIVIDE DELTA X BY (2** GRAN).
1759 0E8E B207  DNTSHF    AB    R7,R8       R8 = B = H1 + DELTA X.
1760 0E90 111E            JLT   NOCOIN
1761 0E92 B0C6            AB    R6,R3       R3 = A = V1 + DELTA Y.
```

```
1762 0E94 111C          JLT   NOCOIN
1763 0E96 9083          CB    R3,R2             A::TV
1764 0E98 151A          JGT   NOCOIN
1765 0E9A 9048          CB    R8,R1             B::TH
1766 0E9C 1518          JGT   NOCOIN          RANGE TEST PASSED?
1767 0E9E 0981          SRL   R1,8             NOW COMPUTE TABLE INDEX.
1768 0EA0 0581          INC   R1            R1 = TH +1.
1769 0EA2 0983          SRL   R3,8          R3 = A.
1770 0EA4 3843          MPY   R3,R1         R2 = A * (TH + 1).
1771 0EA6 0988          SRL   R8,8          R8 = B.
1772 0EA8 A088          A     R8,R2         R2 = INDEX.  COMPUTE TABLE AND BIT POSIT
1773 0EAA C002          MOV   R2,R0         R0 = INDEX ALSO.
1774 0EAC 0242          ANDI  R2,>FFF8      R2 = ROUNDED DOWN TO LOWER MULT OF 8.
     0EAE FFF8
1775 0EB0 6002          S     R2,R0         R0 = BIT DISPLACEMENT (0 = LEFTMOST).
1776 0EB2 0832          SRA   R2,3          R2 = BYTE INDEX INTO TABLE.
1777 0EB4 A085          A     R5,R2         R2 = ACTUAL ADDRESS OF BYTE.
1778 0EB6 8CB2          C     *R2+,*R2+   INCREMENT PTR BY 4 FOR 4 BYTE HEADER
1779 0EB8 DB42          MOVB  R2,@GRMWA(R13)    PULL PROPER BYTE FROM GROM
     0EBA 0402
1780 0EBC 0580          INC   R0
1781 0EBE 06C2          SWPB  R2
1782 0EC0 DB42          MOVB  R2,@GRMWA(R13)
     0EC2 0402
1783 0EC4 0202          LI    R2,>2000
     0EC6 2000
1784 0EC8 D0DD          MOVB  *R13,R3     R3 = THE BYTE FROM THE TABLE.
1785 0ECA 0A03          SLA   R3,R0       GET PROPER BIT INTO THE STATUS CARRY.
1786 0ECC 1801          JOC   YUP         IF BIT IS 0, NO COINCIDENCE.
1787 0ECE 04C2  NOCOIN  CLR   R2                  NO, WE HAVE NO COINCIDENCE
1788 0ED0 D802  YUP     MOVB  R2,@STATUS          YES, WE HAVE COINCIDENCE
     0ED2 837C
1789 0ED4 0460  SOUT9   B     @RETNC        RETURN
     0ED6 048E'
1790                    * VDP IO CALL FROM 9900 CODE
1791                    * CALLED AS XOP 7,6 (SCREEN WORK SPACE)
1792                    * PAB NAME POINTER IN CALLER'R R0
1793                    * DATA TYPE (8 OR 10) IN CALLER'S R1
1794 0ED8 0200  VDPIO   LI    R0,>54+MONIT      VDP IO LINK W/O TYPE FETCH
     0EDA 0054
1795 0EDC C800  IOCOM   MOV   R0,@GLINK1         POINTER TO GPL ROUTINE
     0EDE 84A6
1796 0EE0 C81D          MOV   *R13,@SCNAM       POINTER TO NAME LENGTH (R0)
     0EE2 8356
1797 0EE4 D82D          MOVB  @3(R13),@TYPE     FETCH TYPE (R1 LSB)
     0EE6 0003
     0EE8 836D
1798 0EEA 02E0          LWPI  GPLWS
     0EEC 83E0
1799 0EEE 06A0          BL    @PUTSTK
     0EF0 04BC'
1800 0EF2 06A0          BL    @GPLLKO           CALL I/O LINKAGE
     0EF4 0C02'
1801 0EF6 06A0          BL    @GETSTK
     0EF8 0492'
1802 0EFA 02E0          LWPI  SCRWS             AND RETURN
     0EFC 8458
1803 0EFE 0380          RTWP
1804 0F00 0200  CPUIO   LI    R0,>56+MONIT
     0F02 0056
```

```
1805 0F04 10EB          JMP   IOCOM
1806                 *** MAP DATA FOR XML2
1807 0F06 00FF  ROMFF8 DATA >00FF,>A000,>00FF,>B000
     0F08 A000
     0F0A 00FF
     0F0C B000
1808 0F0E 00FF          DATA >00FF,>4000,>00FF,>5000
     0F10 4000
     0F12 00FF
     0F14 5000
1809 0F16 00FF          DATA >00FF,>C000,>00FF,>D000
     0F18 C000
     0F1A 00FF
     0F1C D000
1810                 * J O Y S T I C K   T A B L E
1811      0F1E' RBBJOY  EQU $
1812      0F1E' HOO EQU $               DRLU (0=TRUE)
1813 0F1E 0000          DATA >0000    0000
1814 0F20 FC00          DATA >FC00    0001
1815 0F22 0004          DATA >0004    0010
1816 0F24 FC04          DATA >FC04    0011
1817 0F26 00FC          DATA >00FC    0100
1818 0F28 FCFC          DATA >FCFC    0101
1819 0F2A 0000          DATA >0000    0110
1820 0F2C FC00          DATA >FC00    0111
1821 0F2E 0400          DATA >0400    1000
1822 0F30 0000          DATA >0000    1001
1823 0F32 0404          DATA >0404    1010
1824 0F34 0004          DATA >0004    1011
1825 0F36 04FC          DATA >04FC    1100
1826 0F38 00FC          DATA >00FC    1101
1827 0F3A 0400          DATA >0400    1110
1828 0F3C 0000          DATA >0000    1111
1829                 * K E Y B O A R D    T A B L E S
1830      0F3E' RKEYTB EQU  $               UNMODIFIED KEYS
1831 0F3E   31          BYTE '1','q','a','z','2','w','s','x'
1832 0F46   33          BYTE '3','e','d','c','4','r','f','v'
1833 0F4E   35          BYTE '5','t','g','b','6','y','h','n'
1834 0F56   37          BYTE '7','u','j','m','8','i','k',','
1835 0F5E   39          BYTE '9','o','l','.','0','p',';','/'
1836 0F66   3D          BYTE '=','[',''',>FF,'-',']',>0D,' '
1837 0F6E   5C          BYTE '\','`',>FF,>FF
1838      0F72' RKSHFT EQU  $               SHIFTED KEYS
1839 0F72   21          BYTE '!','Q','A','Z','@','W','S','X'
1840 0F7A   23          BYTE '#','E','D','C','$','R','F','V'
1841 0F82   25          BYTE '%','T','G','B','^','Y','H','N'
1842 0F8A   26          BYTE '&','U','J','M','*','I','K','<'
1843 0F92   28          BYTE '(','O','L','>',')','P',':','?'
1844 0F9A   2B          BYTE '+','{','"',>FF,'_','}',>0D,' '
1845 0FA2   7C          BYTE '|','~',>FF,>FF
1846      0FA6' RKFCTN EQU  $               Function keys
1847 0FA6   03          BYTE >03,>C5,>FF,>FF,>04,>FF,>08,>0A    Q W
1848 0FAE   07          BYTE >07,>0B,>09,>FF,>02,>FF,>FF,>7F    E R
1849 0FB6   0E          BYTE >0E,>FF,>FF,>BE,>0C,>C6,>BF,>C4    T Y
1850 0FBE   01          BYTE >01,>FF,>C0,>C3,>06,>FF,>C1,>B8    U I
1851 0FC6   0F          BYTE >0F,>FF,>C2,>B9,>BC,>FF,>BD,>BA    O P
1852 0FCE   05          BYTE >05,>FF,>FF,>FF,>FF,>FF,>0D,' '    [ ]
1853 0FD6   FF          BYTE >FF,>FF,>FF,>FF
1854      0FDA' RKCNTL EQU  $               CONTROL KEYS
1855 0FDA   B1          BYTE >B1,>91,>81,>9A,>B2,>97,>93,>98
```

```
 1856 0FE2   B3          BYTE >B3,>85,>84,>83,>B4,>92,>86,>96
 1857 0FEA   B5          BYTE >B5,>94,>87,>82,>B6,>99,>88,>8E
 1858 0FF2   B7          BYTE >B7,>95,>8A,>8D,>9E,>89,>8B,>80
 1859 0FFA   9F          BYTE >9F,>8F,>8C,>9B,>B0,>90,>9C,>BB
 1860 1002   9D          BYTE >9D,>FF,>FF,>FF,>FF,>FF,>0D,' '
 1861 100A   FF          BYTE >FF,>FF,>FF,>FF
 1862      100E' RKSPLT EQU  $                SPLIT KEYBOARDS
 1863 100E   13          BYTE >13,>12,>01,>0F,>07,>04,>02,>00       LEFT
 1864      1016' BIT4 EQU $
 1865 1016   08          BYTE >08,>05,>03,>0E,>09,>06,>0C,>0D
 1866 101E   0A          BYTE >0A,>0B,>11,>10
 1867 1022   13          BYTE >13,>12,>01,>0F,>07,>04,>02,>00       RIGHT
 1868 102A   08          BYTE >08,>05,>03,>0E,>09,>06,>0C,>0D
 1869 1032   0A          BYTE >0A,>0B,>11,>10
 1870                    END
NO ERRORS,    0003 WARNINGS
```