```
*
GPLINT IDT
       TITL 'T.I. 99/4A GRAPHICS LANGUAGE INTERPRETER'
* MEMORY ALLOCATION:
* 0 - >1FFF       INTERNAL ROM
* >2000 - >3FFF   MEMORY EXPANSION PERIPHERAL
* >4000 - >5FFF   PERIPHERAL EXPANSION (DECODED TO I/O CON)
* >6000 - >7FFF   CARTRIDGE ROM/RAM (GROM CONNECTOR)
* >8000 - >83FF   INTERNAL RAM (ONLY 8300 - 83FF USED)
* >8300 - >83FF   SCRATCH PAD RAM
* >8400           SOUND
* >8800           VDP READ DATA
* >8802           VDP READ STATUS
* >8C00           VDP WRITE DATA
* >8C02           VDP WRITE ADDRESS
* >9000           SPEECH READ
* >9400           SPEECH WRITE
* >9800           GROM READ DATA
* >9802           GROM READ ADDRESS
* >9C00           GROM WRITE DATA
* >9C02           GROM WRITE ADDRESS
* >A000 - >FFFF   MEMORY EXPANSION PERIPHERAL
*
* CRU ALLOCATION
* 0000-0FFE INTERNAL USE
* 1000-10FE UNASSIGNED
* 1100-11FE DISK CONTROLLER CARD
* 1200-12FE MODEMS
* 1300-13FE RS232 (PRIMARY)
* 1400-14FE UNASSIGNED
* 1500-15FE RS232 (SECONDARY)
* 1600-16FE UNASSIGNED
* 1700-17FE HEX-BUS
* 1800-18FE THERMAL PRINTER
* 1900-19FE EPROM PROGRAMMER
* 1A00-1BFE UNASSIGNED
* 1C00-1CFE VIDEO CONTROLLER CARD
* 1D00-1DFE IEE 488 CONTROLLER CARD
* 1E00-1EFE UNASSIGNED
* 1F00-1FFE P-CODE CARD
*
* 9901 CRU BIT ALLOCATIONS
* 0    CONTROL
* 1    EXTERNAL INTERRUPT
* 2    VDP VERT. SYNC INTERRUPT
* 3    9901 INTERNAL TIMER INTERRUPT
*      KEYBOARD '=' LINE
*      JOYSTICK 'FIRE'
* 4    KEYBOARD 'SPACE' LINE
*      JOYSTICK 'LEFT'
* 5    KEYBOARD 'ENTER' LINE
*      JOYSTICK 'RIGHT'
* 6    KEYBOARD 'O' LINE
*      JOYSTICK 'DOWN'
* 7    KEYBOARD 'FCTN' LINE
*      JOYSTICK 'UP'
* 8    KEYBOARD 'SHIFT' LINE
* 9    KEYBOARD 'CTRL' LINE
* 10   KEYBOARD 'Z' LINE
* 11   NOT USED AS INTERRUPT
* 12   RESERVED, HIGH LEVEL
* 13-15 NOT USED AS INTERRUPT
```

```
*
* 16   RESERVED
* 17   RESERVED
* 18   BIT 2 OF KEYBOARD SELECT
* 19   BIT 1 OF KEYBOARD SELECT
* 20   BIT 0 (MSB) OF KEYBOARD SELECT
* 21   KEYBOARD ALPHA LOCK
* 22   CASSETTE CONTROL 1 (MOTOR CONTROL)
* 23   CASSETTE CONTROL 2 (MOTOR CONTROL)
* 24   AUDIO GATE
* 25   MAG TAPE OUT
* 26   RESERVED
* 27   MAG TAPE INPUT
* 28-31 NOT USED IN I/O MAPPING
*
****************************************
*
* DEFINITIONS FOR BASIC INTERPRETER
*
       DEF  HX0002,C020,RESET,SET
       DEF  GETSTK,PUTSTK
*
* DEFINTIONS FOR FLOATING POINT PACK
*
       DEF  HX0020,NEXT,SROM,SGROM
*
* REFERENCES INTO FPT ROM
*
XTAB   EQU  S+>12A0
WRITE  EQU  S+>1346
TIMER  EQU  S+>1404
VERIFY EQU  S+>1426
READ   EQU  S+>142E
*
* REFERENCES IN BASIC INTERPRETER
*
PARSEG EQU  S+>18C8
CONTG  EQU  S+>1920
EXECG  EQU  S+>1968
RTNG   EQU  S+>19F0
*
*
*EQUATES
*
VWDOFF EQU  -2              VDP WRITE DATA OFFSET (FROM R15)
GRAOFF EQU  2               GROM READ ADDRESS OFFSET (FROM R13)
GWDOFF EQU  >400            GROM WRITE DATA OFFSET (FROM R13)
VRSOFF EQU  ->400           VDP READ STATUS OFFSET (FROM R15)
GWAOFF EQU  >402            WRITE ADDRESS OFFSET(FROM R13)
VRDOFF EQU  ->402           VDP READ DATA OFFSET (FROM R15)
SGCADR EQU  >8400           SOUND CHIP
*
* RAM EQUATES
*
PAD    EQU  >8300           START OF 256 BYTES OF RAM
*
*  STATUS BLOCK DEFINITIONS
*
FAC    EQU  PAD+>4A
SCLEN  EQU  PAD+>55
TEMP2  EQU  PAD+>6C
TYPE   EQU  PAD+>6D
```

```
STKDAT EQU  PAD+>72
STKADD EQU  PAD+>73
PLAYER EQU  PAD+>74
RANDOM EQU  PAD+>78
TIME   EQU  PAD+>79          TIME
MOTION EQU  PAD+>7A
VDPST  EQU  PAD+>7B          VDP STATUS
STATUS EQU  PAD+>7C
CHRBUF EQU  PAD+>7D
YPT    EQU  PAD+>7E
XPT    EQU  PAD+>7F
*
* WORKSPACE DEFINITIONS
*
WKSC   EQU  PAD+>C0          INT. L WORKSPACE
SAVEG  EQU  PAD+>CB          SAVE GROM ADR. OF HEADER
SNDADD EQU  PAD+>CC          SOUND LIST ADDRESS
STFLGS EQU  PAD+>CE          NUMBER OF SOUND BYTES
CRULST EQU  PAD+>D0          R8, INT WKS
SADDR  EQU  PAD+>D2
SAVVDP EQU  PAD+>D4
RSAVE  EQU  PAD+>D8
WKSE   EQU  PAD+>E0          MAIN WORKSPACE
*
R0LSB  EQU  WKSE+R0+R0+1
R1LSB  EQU  WKSE+R1+R1+1
R2LSB  EQU  WKSE+R2+R2+1
R3LSB  EQU  WKSE+R3+R3+1
R4LSB  EQU  WKSE+R4+R4+1
R5LSB  EQU  WKSE+R5+R5+1
R6LSB  EQU  WKSE+R6+R6+1
R7LSB  EQU  WKSE+R7+R7+1
R8LSB  EQU  WKSE+R8+R8+1
R9LSB  EQU  WKSE+R9+R9+1
R12LSB EQU  WKSE+R12+R12+1
*
VDPREG EQU  >8000
*
GR     EQU  >9800            GROM READ (DATA)
VRD    EQU  >8800            VDP READ DATA
VRS    EQU  >8802            VDP READ STATUS
VWD    EQU  >8C00            VDP WRITE DATA
VWA    EQU  >8C02            VDP WRITE ADDRESS
*
* EXTERNAL ROM EQUATES
*
H4000  EQU  >4000
H400C  EQU  >400C            ROM HEADER, INTERRUP LINK
*
* MISC. EQUATES
VDELTA EQU  >2000
QSAML  EQU  >0480
WRVDP  EQU  >4000            WRITE TO VDP BIT
*
       RORG
S      EQU  $
*
* INTERUPT/RESET VECTORS
*
       DATA WKSE,ENTRY
       DATA WKSC,REMOTE      INTERRUPT ONE  (9901)
       DATA WKSC,TIMIN       INTERRUPT TWO
```

```
*
HX30AA BYTE >30              3 MHZ CLOCK
       BYTE >AA              VALID ROM ID
*
* CALL SCAN FROM ROM
*
SCNKEY B    @KSCAN
*
HX0008 DATA >0008            USED IN COC & LDCR
* ENTRY NOT KNOWN
C014   SBZ  0                 DEBUG BOARD INSTRUCTIONS
       B    @>007A
*ENTRY NOT KNOWN
C01A   SBZ  0                 DEBUG BOARD INSTRUCTIONS
       B    @>0078
* ENTRY = BL
C020   B    @C4B2
C025   EQU  $+1               NASTY
ENTRY  LI   R13,GR            RESET ENTRY
       LI   R14,>0100
       LI   R15,VWA           VDP WRITE ADDRESS
C030   LI   R0,>0020          C030 REFD (0200) NASTY
HX0020 EQU  $-2
       JMP  DGBADD
HX1000 DATA >1000            USED IN CZC
* ENTRY NOT KNOWN
C038   SBZ  0                 DEBUG BOARD INSTRUCTIONS
       LWPI >280A
       RTWP
* XOP 0 VECTORS (DEBUGGER)
       DATA >280A
       DATA >0C1C
* XOP 1 VECTORS (USER DEFINED)
       DATA >FFD8
       DATA >FFF8
* XOP 2 VECTORS (USER DEFINED)
       DATA >83A0
       DATA PAD
*
SHQ    DATA >1100            USED IN CZC
* ----- BEGIN EXECUTION OF GPL INSTRUCTIONS -----
*           LIBRARY CALL ROUTINE
* = BR TABLE
DGBA   BL   @PUTSTK          SAVE RETURN ADDRESS
       BL   @PUTSTK          INCREMENT STACK POINTER
       MOV  R13,@PAD(R4)     SAVE GROM BASE ADDRESS
       MOV  R2,R13           NEW GROM BASE ADDRESS
DGBADD MOVB *R13,R4          (SYNC GROMS) DO GROM DUMMY READ
       MOV  R0,R6            BRANCH TO ADDRESS IN GROMS
LDKADD MOVB R6,@GWA0FF(R13)                )WRITE R6 TO GROM
       MOVB @R6LSB,@GWA0FF(R13)            )ADDRESS
RESET  SZCB @HX20,@STATUS    CLEAR CONDITION BIT
* ENTRY = BR
NEXT   LIMI 2                ALLOW INT'S BETWEEN GPL INSTN'S
HX0002 EQU  $-2
C074   LIMI 0
       MOVB *R13,R9          LOAD INSTN FROM GAME ROM.
       JLT  ABOPS            JMP IF MS BIT SET
       MOVB R9,R4            MOVE INSTN TO WORK REG'R
       SRL  R4,12            LEAVE TOP 3 BITS *2
       MOV  @ITAB(R4),R5     GET BRANCH ADDRESS
       B    *R5              B @MICSLN, MOVDAT, BRESET, BSET
```

```
*---------INSTRUCTIONS WITH A & B OPERANDS---------
ABOPS  CLR  R4               CLEAR VDP RAM FLAGS
       MOV  R9,R5            LOAD R5 WITH DOUBLE FLAG
       ANDI R5,>0100         IF 0, BYTE OPND, IF 1 = WORD OPND
       BL   @GETMAD          GET FIRST OPERAND
       SWPB R4
       MOV  R1,R3            SAVE VARIABLE ADDRESS
       MOV  R0,R2            SAVE VAR. VALUE (OPERAND)
       CI   R9,>A000         SINGLE OPND? (8X OR 9X = SINGLE)
       JL   AOPS             YES
       COC  @C030,R9         IMMEDIATE OR VAR? (C030 =>0200)
       JNE  ABOPA            VARIABLE
       MOV  R13,R1           GET IMMEDIATE VALUE
*                            (2ND OPERAND IS IMMEDIATE)
       MOVB *R1,R0           GET FIRST BYTE
       DEC  R1               MODIFY FOR COMMON ROUTINE
       BL   @MADC2           GO TO COMMON ROUTINE (PUT 2ND IMM.
*                            OPND BYTE IN R0 LSB, IF WORD.
*                            OTHERWISE SIGN EXTEND BYTE TO LSB)
       JMP  BOPS
AOPS   MOV  R9,R8            BRANCH THROUGH BRANCH TABLE
       SRL  R8,8
       SETO R0
       MOV  @ATAB(R8),R8
       B    *R8
ABOPA  BL   @GETMAD
BOPS   MOV  R9,R8            BRANCH THROUGH BRANCH TABLE
       SRL  R8,9
       MOV  @BTAB(R8),R8     LOAD BRANCH TABLE
       C    R2,R0            COMPARE FOR IF PRIMITIVES
       B    *R8
*---------PERFORM IF COMPARISONS---------
*
*  = BR TABLE (BTAB)
GREQ   JLT  RESET            A ALGEBRAIC .GE. B
SET    SOCB @HX20,@STATUS    SET CONDTION BIT
       JMP  NEXT
*  = BR TABLE (BTAB)
HIGH   JH   SET              A LOGICAL .GT. B
       JMP  RESET
*  = BR TABLE (BTAB)
HIGHEQ JHE  SET              A LOGICAL .GE. B
       JMP  RESET
*  = BR TABLE (BTAB)
GREATR JGT  SET              A ALGEBRAIC .GT. B
       JMP  RESET
*  = BR TABLE (BTAB)
IFAND  INV  R0               PERFORM LOGICAL AND
       SZC  R0,R2
       JEQ  SET
       JMP  RESET
*  = BR TABLE (ATAB)
IFZ    MOV  R2,R2            COMPARE TO ZERO
*  = BR TABLE (BTAB)
EQUAL  STST R4               STORE STATUS IN STATUS BYTE
       MOVB R4,@STATUS
       JMP  NEXT
*  = BR TABLE (MSCTAB)
TSTST  MOV  R9,R0            MOVE INST TO SHIFT REG
       SLA  R0,12            REMOVE HIGH ORDER BITS
       SRL  R0,13            POSITION 3 REMAINING BITS
       MOVB @STATUS,R5       LOAD STATUS BYTE
```

```
       SLA  R5,0             SHIFT TO TEST BIT
       JOC  SET
       JMP  RESET
*
* = BR TABLE
* READ GROM TO R6, EXIT
* AND WRITE R6 TO GROM ADDRESS
*
BLONG  MOVB *R13,R6
       NOP
       MOVB *R13,@R6LSB
       JMP  LDKADD
* ----- BRANCH INSTRUCTIONS -------
*  = BR TABLE
BSET   MOVB @STATUS,R4       IS CONDITION SET?
       SLA  R4,2
       JLT  BRAD             YES
NOBR   MOVB *R13,R4          INCREMENT PGM COUNTER
       JMP  RESET
HX20   EQU  $+1              =>20 (NASTY)
* = BR TABLE
BRESET MOVB @STATUS,R4       IS CONDITION SET?
       SLA  R4,2
       JLT  NOBR             YES
BRAD   MOVB *R13,@R9LSB      GET 2ND BYTE OF ADR
       ANDI R9,>1FFF         MASK TO LEAVE BRANCH ADR.
HX1FFF EQU  $-2
       MOVB @GRA0FF(R13),R6  READ GROM ADDRESS MSB
       ANDI R6,>E000         LEAVE TOP 3 BITS (GROM NO.)
       SOC  R9,R6            COMBINE TO GIVE NEW ADR.
       JMP  LDKADD           WRITE R6 TO GROM ADDRESS
*
* ---------SINGLE OPERAND INSTRUCTIONS ---------
*
* ENTRY = BR TABLE
ABS    ABS  R2               ABSOLUTE VALUE
       JMP  TRAP
* ENTRY = BR TABLE
NEG    NEG  R2               NEGATE VALUE
       JMP  TRAP
* ENTRY = BR TABLE
CLR    SETO R2               SET TO ONES
* ENTRY = BR TABLE
INV    INV  R2               INVERT VALUE
       JMP  TRAP
* ENTRY = BR TABLE
FETCH  MOV  R4,R6            SAVE ADRESSING MODE FLAG
       BL   @PUTSTK          SAVE CURRENT PGM ADDRESS
       DECT R4               SET I TO OLD STACK POINTER
       BL   @GTSTK           GET ADR. FROM STACK TOP
       MOVB *R13,R2          LOAD BYTE FROM GROM
       SRA  R2,8             EXTEND SIGN
       INC  @PAD(R4)         INCREMENT RETURN ADDRESS
       INCT R4               SET I BACK TO NEW STACK PNTR
       BL   @GTSTK1          RESTORE GROM ADDRESS
       MOV  R6,R4            RESTORE ADR'G MODE FLAG
       JMP  TRAP
* ENTRY = BR TABLE
CASE   DEC  R2               VARIABLE DISPLACEMENT
       JNC  RESET
       MOVB *R13,R5          SKIP TO NEW ADDRESS
       NOP
```

```
       MOVB *R13,R5          SKIP TO NEW ADDRESS
       JMP  CASE
* ENTRY = BR TABLE
PUSH   AB   R14,@STKDAT      LOAD STACK ADDRESS (R14=01XX)
       MOVB @STKDAT,R6
       SRL  R6,8
       MOVB @R2LSB,@PAD(R6)  PUSH LSBYTE
       B    @NEXT
*---------TWO OPERAND INSTRUCTIONS------------
*
* = BR TABLE
DECT   SRL  R0,14            DECREMENT VALUE BY 2
* = BR TABLE
INCT   DEC  R0               INCREMENT INSTRUCTIONS
* = BR TABLE
MINUS  NEG  R0               SUBTRACT OPERATION
* = BR TABLE
ADD    MOVB R5,R5            SINGLE OR DOUBLE
       JEQ  ADDB
       A    R0,R2            ADD OPERATION
       JMP  FILLST
* = BL TABLE
AND    INV  R0               LOGICAL AND OPERATION
       SZC  R0,R2
       JMP  FILLST
* = BR TABLE
OR     SOC  R0,R2            LOGICAL OR OPERATION
       JMP  FILLST
XOR    XOR  R0,R2            LOGICAL XOR OPERATION
       JMP  FILLST
* = BR TABLE
STORE  MOV  R0,R2            STORE SOURCE INTO DESTINATION
       JMP  TRAP
* = BR TABLE
EXCH   MOV  R2,R9            EXCHANGE A AND B
       MOV  R0,R2
       BL   @TRAPA
       SWPB R4
       MOV  R1,R3
       JMP  MUL2
* = BR TABLE
SRA    SRA  R2,0             SHIFT RIGHT ARITHMETIC
       JMP  TRAP
* = BR TABLE
SLL    SLA  R2,0
       JMP  TRAP
* = BR TABLE
SRL    MOVB R5,R5            SHIFT RIGHT LOGICAL
       JNE  SRL1             DOUBLE INSTRUCTION
       SB   R2,R2
SRL1   SRL  R2,0
       JMP  TRAP
* = BR TABLE
SRC    MOVB R5,R5            SHIFT RIGHT CIRCULAR
       JNE  SRC1             DOUBLE INSTRUCTION
       MOVB @R2LSB,R2
SRC1   SRC  R2,0
       JMP  TRAP
* = BR TABLE
MUL    MOV  R2,R8            MULTIPLY
       MOVB R5,R5            SINGLE OR DOUBLE?
       JNE  MUL0             DOUBLE
```

```
       SB   R8,R8            CLEAR EXT'D SIGN BITS
MUL0   MPY  R0,R8
       MOVB R5,R5            SINGLE OR DOUBLE?
       JNE  MUL1
       MOVB R9,@R8LSB
MUL1   MOV  R8,R2            STORE FIRST HALF OF RESULT
       BL   @TRAPA
MUL2   MOV  R9,R2            STORE 2ND HALF OF RESULT
       JMP  TRAP
* = BR TABLE
DIV    MOVB R5,@STATUS       CLEAR STATUS
       MOV  R2,R8
       MOV  R0,R2
       MOV  R3,R1            SET DESTINATION ADR.
       INC  R1
       MOVB R5,R5            SINGLE OR DOUBLE?
       JEQ  DIV1             SINGLE
       INC  R1
DIV1   MOVB R4,R4            CPU OR VDP?
       JEQ  DIVC             CPU
       BL   @MVDPA           GET REST OF DIVIDEND FROM VDP
       JMP  DIVB
DIVC   BL   @MADC            GET REST OF DIVIDEND FROM CPU
DIVB   MOV  R0,R9
       MOVB R5,R5            SINGLE OR DOUBLE?
       JNE  DIV2
       MOVB @R8LSB,R9        DO SIGN EXT'N FOR BYTE
       SRA  R8,8                 OPERANDS
DIV2   DIV  R2,R8            PERFORM DIVISION
       JNO  MUL1
       SOCB @HX0008+1,@STATUS               SET DIVIDE O'FLOW
       JMP  MUL1
ADDB   AB   @R0LSB,@R2LSB
FILLST STST R11              STORE STATUS WORD
       MOVB R11,@STATUS      STORE STATUS BITS
TRAP   LI   R11,NEXT         LOAD R11 TO RETURN TO 'NEXT'
* = BL
TRAPA  MOVB R4,R4            DEST'N IS CPU OR VDP?
       JEQ  STCPU            CPU
       MOVB @R3LSB,*R15      LOAD VDP ADR.
       ORI  R3,>4000         SET BIT TO WRITE TO VDP
       MOVB R3,*R15
       MOVB R5,R5            SINGLE OR DOUBLE?
       JEQ  TRAP1            SINGLE
       MOVB R2,@VWD0FF(R15)  MOVE 2 BYTES TO VDP
       INC  R3
TRAP1  MOVB @R2LSB,@VWD0FF(R15)             MOVE 1 BYTE TO VDP
       INC  R3
TEXIT  RT
STCPU  MOVB R5,R5            SINGLE OR DOUBLE?
       JEQ  TRAP2            SINGLE
       MOVB R2,*R3+          STORE 2 BYTE RESULT
TRAP2  SWPB R2
       MOVB R2,*R3+          STORE 1 BYTE RESULT
       CI   R3,PAD+>7E       IS CB THE DESTINATION?
       JNE  TEXIT            NO
       MOV  R11,R6           SAVE RETURN
       BL   @GETDA           GET VDP DISPLAY ADR.
RTN    MOVB R2,@VWD0FF(R15)  WRITE CHAR. TO VDP
       B    *R6              RETURN
*-------- MISCELLANEOUS INSTRUCTIONS --------
* = BRANCH TABLE
```

```
MICSLN SLA  R9,3             REMOVE TOP 3 BITS OF INSTN
       SRL  R9,10            MAKES MAX VAL =>3F
       MOV  @MSCTAB(R9),R4   GET BRANCH ADDRESS
       B    *R4
*-------- GENERATE A RANDOM NUMBER ----------
* = BR TABLE
RANDNO LI   R4,>6FE5         (=28645)
       MPY  @WKSC,R4
       AI   R5,>7AB9         (=31417)
       MOV  R5,@WKSC
       MOVB *R13,R6          LOAD MODULO NUMBER
       SRL  R6,8             SHIFT TO LSB
       INC  R6
       CLR  R4               CLEAR UPPER HALF OF DIVIDEND
       SWPB R5               SWAP BYTES IN J
       DIV  R6,R4            PERFORM DIVISION
       MOVB @R5LSB,@RANDOM   STORE REMAINDER
       JMP  BKGR1
* = BR TABLE
BKGRND LI   R7,VDPREG+>700   CHANGE BACKGROUND COLOUR
       MOVB *R13,@R7LSB      IMMEDIATE OPERAND
       BL   @SETVDP          OUTPUT ADDRESS TO VDP
BKGR1  B    @NEXT
* -------KEYBOARD SCANNER -------------
* = BR TABLE
KEYSCN LI   R11,NEXT         RTN TO NEXT IF INTERPRETER CALL
KSCAN  MOV  R11,@RSAVE       SAVE R11 FOR RTN
       BL   @PUTSTK          SAVE CURRENT GROM ADDRESS
       CLR  R12              /4A
       SBO  21               /4A PUT ALPHA LOCK BIT UP
       MOVB @PLAYER,R5
       SRL  R5,8             TO LSB
       MOV  R5,R6            SAVE PLAYER
       JEQ  C2EC             NO PLAYER NO, JUMP
       LI   R0,>0FFF
HX0FFF EQU  $-2
       DEC  R6               ADJUST PLAYER NO
       JEQ  C2F4             ZERO, JUMP (PLAYER 1)
       LI   R0,>F0FF
       DEC  R6               ADJUST PLAYER
       JEQ  C2F4             ZERO (PLAYER 2) JUMP
       DEC  R6               DEC PLAYER
       C    R6,@HX0002       IF INIT VALUE OF PLAYER
       JH   C382              GT 5, JUMP
       MOVB R6,@PLAYER       CLEAR PLAYER NO.
       SWPB R6
       MOVB R6,@WKSC+R3+R3   SAVE PLAYER (DEC'D BY 3)
       CLR  R5               ASSUME PLAYER ZERO
C2EC   CLR  R0
       CLR  R6               CLEAR PLAYER NO. SAVE TOO
       JMP  C32E
C2F1   EQU  $-1              R5 = 1 OR 2 HERE
       DATA >2925
C2F4   LI   R12,>24           POINT TO KBD O/P SELECT
       LDCR @>0405(R5),3     PLAYER 1 LOADS 6, 2 LOADS 7
       LI   R12,6            POINT TO J/S BITS
       CLR  R3
       SETO R4
       STCR R4,5             READ J/S BITS
       SRL  R4,9             CHECK FIRE BUTTON
       JOC  C310             SET, JUMP
       MOVB @C2F1(R5),@R3LSB BYTES ARE >29 & >25 (R5 = 1 OR 2)
```

```
C310    SLA  R4,1             RE-ALIGN
        AI   R4,>16E0         CALC ADR. OF BYTE IN TABLE IN GROM
*                              (FOR FIRE, L,R,U,D)
        MOVB R4,@GWAOFF(R13)  LOAD GROM ADR.
        MOVB @R4LSB,@GWAOFF(R13)
        NOP
        MOVB *R13,@>8376      GET GROM DATA
        MOVB *R13,@>8377      GET GROM DATA
        MOV  R3,R3            FIRE BUTTON PRESSED?
        JNE  C3AA             NO
*
* FIRE PRESSED OR PLAYER NOT 1 OR 2
*
C32E    LI   R1,5             SET UP TO DO KEYBD
        CLR  R2
        CLR  R7
C336    LI   R12,>24          POINT TO J.S/KEYBD
        SWPB R1               TO MSB FOR LDCR
        LDCR R1,3             GND A STROBE LINE
        SWPB R1
        LI   R12,6            POINT TO KBD O/P'S
        SETO R4               INVERT FOLLOWS
        STCR R4,8             STORE KBD OUTPUTS
        INV  R4               MAKE ANY STROBE A 1
        MOV  R1,R1            LAST KBD STROBE
        JNE  C354             NO
        MOVB R4,R7            SAVE BITS FROM KBD
        ANDI R4,>0F00
C354    SZC  R0,R4
        JEQ  C37A
        MOV  R1,R1            ON LAST STROBE ?
        JNE  C360             NO
        MOV  R5,R5            PLAYER 0?
        JNE  C37A             NO
C360    MOV  R2,R2            1ST TIME ROUND LOOP
        JNE  C37A             NO
        SETO R2               SET 'NOT FIRST TIME ROUND' FLAG
        MOV  R1,R3            MOVE LOOP (STROBE NO) TO R3
        SLA  R3,3             MPY BY 8
        DEC  R3               COMP FOR FOLLOWING INC
C36C    INC  R3
        SLA  R4,1
        JNC  C36C
* R3= 8*KBD STROBE+BIT POSITION
        MOV  R1,R1            LAST STROBE?
        JEQ  C37A             YES
        LI   R1,1
HX0001  EQU  $-2
C37A    DEC  R1
        JOC  C336
        MOV  R2,R2
        JNE  C3AA
C382    CLR  R6
        MOVB R6,@WKSC+R3+R3+1
        SETO R0
        CB   R0,@WKSC+R4+R4(R5)
        JEQ  C394
        BL   @C498            DELAY (ONLY)
C394    MOVB R0,@WKSC+R4+R4
        MOVB R0,@WKSC+R4+R4(R5)
        MOV  R5,R5
        JNE  C478
```

```
        MOVB R0,@WKSC+R4+R4+1
        MOVB R0,@WKSC+R5+R5
        JMP  C478
C3AA    CB   @R3LSB,@WKSC+R4+R4(R5)
        JEQ  C3E6
        LI   R6,>2000
HX2000  EQU  $-2
        BL   @C498             DELAY (ONLY)
        MOVB @R3LSB,@WKSC+R4+R4
        MOVB @R3LSB,@WKSC+R4+R4(R5)
        MOV  R5,R5
        JNE  C3E2
        MOV  R3,R12
        AI   R12,>FFF8
        JLT  C3E2
        LI   R1,>0002
        SRL  R12,3
        JOC  C3DC
        DEC  R1
C3DC    MOVB @R3LSB,@WKSC+R4+R4(R1)
C3E2    MOVB R7,@WKSC+R3+R3+1
C3E6    MOVB @WKSC+R3+R3+1,R7
        LI   R1,>17C0
        MOV  R5,R5
        JNE  C40E
        LI   R1,>1790
        SLA  R7,2
        JOC  C40E
        LI   R1,>1760
        SRL  R7,15
        JOC  C40E
        LI   R1,>1730
        DEC  R7
        JEQ  C40E
        LI   R1,>1700
C40E    A    R3,R1
        MOVB R1,@GWA0FF(R13)   LOAD COMPUTED GROM ADR.
        MOVB @R1LSB,@GWA0FF(R13)
        NOP
        MOVB *R13,R0            GET CHAR. FOR KEY
        MOV  R5,R5
        JNE  C478
        MOVB @WKSC+R3+R3,@R3LSB
        BL   @C4A2             IS R0 MSB BETWEEN >61 & >7A
        DATA >617A              LOWER CASE CHAR?
        JNE  C444              NO, JUMP
        CLR  R12
        MOV  R3,R3
        JEQ  C43E
        SBZ  21                 PUT ALPHA LOCK STROBE DOWN
        SRC  R12,14             WAIT
        TB   7                  ALPHA LOCK DOWN?
        JEQ  C442               NO
C43E    SB   @HX2000,R0         CHANGE TO UPPER CASE
C442    SB0  21                 RESET A/LOCK STROBE
C444    MOV  R3,R3
        JNE  C456
        BL   @C4A2             IS R0 MSB BETWEEN >10 & >1F?
        DATA >101F
        JEQ  C382              YES
        CB   R0,@HX5F           GT. >5F?
        JH   C382               YES
```

```
C456   DEC  R3
       JNE  C478
       CB   R0,@C025          ENTER? (>0D)
       JEQ  C478              YES
       CB   R0,@HX0FFF
       JH   C46C
       SOCB @C470,R0          SET MSBIT IF >00 THRU >0F
       JMP  C478
C46C   BL   @C4A2             IS R0 MSB BETWEEN >80 & >9F?
C470   DATA >809F
       JNE  C478              NO
       SZCB @C470,R0          CLEAR MSBIT
C478   MOVB R0,@>8375         SAVE CHAR
       BL   @GETSTK
       MOVB R6,@STATUS
       JEQ  C492
       MOVB @SAVVDP,*R15
       CLR  @WKSC+R11+R11
       MOVB @HX81,*R15
C492   MOV  @RSAVE,R11
       RT
* = BL ( DELAY LOOP )
C498   LI   R12,>04E2
C49C   DEC  R12
       JNE  C49C
       RT
*
* CHECKS THAT R0 MSB IS IN RANGE OF DATA STATEMENT
* FOLLOWING THE CALL (=STATUS=IN RANGE)
*
* = BL ( CALL REQUIRES DATA STATEMENT)
C4A2   MOV  *R11+,R12
       CB   R0,R12
       JL   C4B0
       CB   R0,@R12LSB
       JH   C4B0
       CB   R0,R0
C4B0   RT
* = B
C4B2   LI   R12,>24            POINT TO KBD
       LDCR @HX0008,3          PNT TO CTL/SHFT ETC
       SRC  R12,7              WAIT
       LI   R12,6              POINT TO KBD INPUTS
       STCR R12,8              READ ALL 8 I/PS
       CZC  @HX1000,R12        FUNCTION DOWN?
       JNE  C4DC               YES, EXIT
       LI   R12,>24            POINT TO KBD
       LDCR @C074,3            POINT TO '4' LINE
       SRC  R12,7              WAIT
       LI   R12,6              POINT TO I/PS
       STCR R12,8              GET DATA
       CZC  @HX1000,R12        '4' DOWN?
C4DC   RT
* ---------FORMAT FOR STRING ----------------
* ENTRY FROM BRANCH TABLE
FORMAT CLR  R9                 CLEAR BLOCK COUNT
       CLR  R3                 ZERO BIAS BYTE
       BL   @GETDA             ENCODE VDP ADDRESS
FUNCTN MOVB *R13,R8            GET NEXT BYTE FROM GAME ROM
       LI   R12,STKADD
       MOV  R8,R5              SAVE DATA BYTE
       SLA  R8,3               SHIFT TO FIND FUNCTION TO PERFORM
```

```
       SRL  R8,11            MOVE TO LSB
       INV  R8               LEAVE N IN RANGE OF MINUS 1 - 32
       SRL  R5,12            ISOLATE FUNCTION BITS
       MOV  @FBTAB(R5),R5    GET BRANCH ADR.
       LI   R2,LOOP
       SETO R4               SET I TO -1
       B    *R5
* ------- REPEAT CHARACTER N TIMES DOWN THE SCREEN ------
* ENTRY = BR TABLE
RPTDWN SLA  R4,5             SET I TO -32
* ------- REPEAT CHARACTER N TIMES ACROSS THE SCREEN -----
* ENTRY = BR TABLE
RPTACR C    *R2+,*R2+        LOAD BRANCH ADR. FOR LOOPING
       JMP  LOOP
* ------- STORE N CHARACTERS DOWN THE SCREEN ------
* = BR TABLE
STRDWN SLA  R4,5             SET I TO -32
* ------- STORE N CHARACTERS ACROSS THE SCREEN -----
* ------- LOOP TO STORE CHARACTER -----
* = BR TABLE & JMP
LOOP   MOVB *R13,R6          GET NEXT BYTE FROM GAME ROM
LOOPB  A    R3,R6            ADD BIAS TO CHAR.
       MOVB R6,@VWDOFF(R15)  MOVE BYTE INTO DISPLAY
       S    R4,R7            DISPLACE TO NEXT CHAR.
       CI   R7,>0320         DO I WANT WRAP ROUND?
       JHE  ZEND             NO
UPXY   CI   R7,>0300         DISPLAY ADR. OUT OF RANGE?
       JL   ZEND             NO
       AI   R7,->300         SUBTRACT LENGTH OF DISPLAY
ZEND   BL   @RESTOR
       BL   @GETDA
       INC  R8               INCR. LOOP COUNTER
       JEQ  FUNCTN           DONE WITH LOOP IF N=0
       B    *R2              LOOP UNTIL N=0
* ------- SKIP N LINES DOWN THE SCREEN ----
* = BR TABLE
SKPDWN SLA  R8,5             MULTIPLY LINES TO SKIP BY 32
* ------- SKIP N SPACES ACROSS THE SCREEN ----
* = BR TABLE
SKPACR S    R8,R7            MOVE NUMBER OF SPACES TO SKIP INTO
       SETO R8               SET LOOP COUNTER TO -1
       JMP  UPXY             GO PERFORM SKIP
* ------- REPEAT BLOCK N TIMES ----
* = BR TABLE
RPTBLK INC  R9               INCREMENT BLOCK COUNTER
       INCT *R12             INCREMENT STACK POINTER
       MOVB *R12,R6          GET DATA AT STACK POINTER
       SRL  R6,8             TO LSB
       MOVB @R8LSB,@PAD(R6)  PUSH N ONTO DATA STACK
       JMP  FUNCTN
* ------- REPETITION OF BLOCK HAS BEEN COMPLETED ----
FINISH DECT *R12             DECT DATA STACK PNTR R14=01
       DEC  R9               DECREMENT BLOCK COUNT
       JMP  FUNCTN
* ------ END BLOCK FUNCTION
ENDBLK MOV  R9,R9            IS BLOCK COUNT ZERO?
       JEQ  ENDFMT           YES - END OF FORMAT INSTN

       MOVB *R13,R4          LOAD LOOP ADDRESS
       MOVB *R12,R6          GET DATA STACK POINTER
       MOVB *R13,R5
       SRL  R6,8
```

```
       AB   R14,@PAD(R6)     DEC REPETITION COUNT (R14=01XX)
       JEQ  FINISH           DONE WITH BLOCK IF COUNT = 0
       MOVB R4,@GWAOFF(R13)  LOAD GROM WITH LOOP ADDRESS
       MOVB R5,@GWAOFF(R13)
       JMP  FUNCTN
* ------  SPECIAL FORMAT INSTNS
* = BR TABLE
SPECL  CI   R8,-28           WHICH FUNCTION?
       JEQ  ENDBLK           END BLOCK
       JGT  VARBLE           VARIABLE CHARS.
       MOV  R13,R1
       CI   R8,-30           WHICH FUNCTION?
       JEQ  BIASV            SET COLOUR BIAS
       JGT  BIASI            SET COL. BIAS IMMEDIATE
       BL   @RESTOR          SET XPT OR YPT
       NEG  R8
       MOVB *R13,@PAD+>5F(R8)              GET VAL OF XPT OR YPT
HX5F   EQU  $-1
       BL   @GETDA           COMPUTE VALUE OF ADR.
       JMP  FUNCTN
* ------ SET COLOUR BIAS ----
BIASV  BL   @GETMAB          GET MEMORY ADR.
BIASI  MOVB *R1,R3           LOAD COLOUR BIAS
       JMP  FUNCTN
* ------ VARIABLE CHARACTERS ----
VARBLE BL   @GETMAB          GET MEMORY ADR.
       LI   R2,LOOPV         SET LOOP REGISTER
LOOPV  MOVB *R1+,R6          LOAD VARIABLE CHAR.
       JMP  LOOPB
* ------  CLEAR SCREEN -----
* = BR TABLE
ALL    MOVB *R13,R5          GET CHAR TO STORE
       BL   @SETV
       LI   R7,768           NUMBER OF BYTES TO TRANSFER
ALLOOP MOVB R5,@VWDOFF(R15)  STORE BYTE IN DISPLAY
       DEC  R7
       JNE  ALLOOP
* ------ RESTORE XPT AND YPT FROM DISPLAY ADR. ----
ENDFMT LI   R11,NEXT
* ENTRY = BL
RESTOR SLA  R7,3             SHIFT YPT VALUE TO TOP BYTE
       MOVB R7,@YPT          STORE VALUE OF YPT
       SLA  R7,8             SHIFT OFF VALUE OF YPT
       SRL  R7,3             LEAVE VALUE OF XPT
       MOVB R7,@XPT          STORE VALUE OF XPT
       RT
* -------INPUT/OUTPUT INSTRUCTION -------
*
* = BR TABLE
INOUT  MOV  R0,R2            GET INDEX INTO I/O TABLE
       MOV  R3,R1            MOVE LIST ADR FOR CASSETTE
       A    R2,R2
       MOV  @IOTAB(R2),R4    LOAD ADR. OF I/O FUNCTION
       CLR  R9               CLEAR R9 FOR CRU INSTN'S
       B    *R4              BRANCH TO I/O FUNCTION
*
* ------ SOUND INSTRUCTION -------
* = BR TABLE
SOUND  ANDI R14,>FFFE        CLEAR SOUND SOURCE FLAG
       SOC  R0,R14           SET SOUND SOURCE FLAG
       MOV  *R3,@SNDADD      LOAD SOUND LIST ADR.
       MOVB R14,@STFLGS      SET TIME DELAY TO 1 UNIT (R14=01XX)
```

```
@EXIT  B    @NEXT
*
* ------CRU INSTRUCTIONS ------
* = BR TABLE
CRUIN  INC  R9              /4 IS INCT!!
* = BR TABLE
CRUOUT MOV  *R1+,R12        LOAD CRU BASE ADR.
       A    R12,R12
       CLR  R2
       MOVB *R1+,R2         LOAD NUMBER OF BITS TO MOVE
       SLA  R2,4
       SOC  R2,R9           COMBINE NUMBER AND IN/OUT FLAG
       SRC  R9,6            REPOSITION BITS
       ORI  R9,>3012        R9 IS NOW LDCR OR STCR *R2
       MOVB *R1,R2          LOAD CPU SOURCE/DEST ADR.
       SWPB R2
       AI   R2,PAD
       X    R9              EXECUTE R9
       JMP  @EXIT
*
* EXECUTE 9900 CODE IN EXTERNAL ROM
* ENTRY = BR TABLE
XML    MOVB *R13,R9
       MOV  R9,R4
       SRL  R9,12
       SLA  R9,1
       SLA  R4,4
       SRL  R4,11
       A    @XMLTAB(R9),R4
       MOV  *R4,R4
       BL   *R4             GO TO SUBROUTINE
       JMP  @EXIT
* ENTRY FROM BRANCH TABLE
MOVDAT MOVB R14,R5          SET DOUBLE FLAG
       SRL  R9,9            IS LENGTH VAR OR IMMED?
       JOC  LIMM            IMMEDIATE
       BL   @GETMAD         GET VAR. LENGTH
       MOV  R0,R8
       JMP  MVSRC
LIMM   MOVB *R13,R8         LOAD IMMED. LENGTH
       SLA  R4,15           STALL FOR LONG TIME
       MOVB *R13,@R8LSB
MVSRC  CLR  R4
       SLA  R9,12
       BL   @MVADDR         GET DESTN. ADR.
       MOV  R1,R2           SAVE DESTN. ADR
       AB   R9,R9           WAS DEST VDP REGR.
       JNC  DTYPE           NO
       AI   R4,3            DESTN. IS VDP REGR.
DTYPE  MOV  R4,R7           SAVE DEST TYPE
       CLR  R4
       BL   @MVADDR         GET SOURCE ADR.
       A    R4,R4           DOUBLE R4
       MOV  @MSRC(R4),R6    LOAD ADR. TO SOURCE OF MOVE
       A    R7,R7           DOUBLE R7
       MOV  @MDST(R7),R7    LOAD ADR. TO DESTN.
       BL   @PUTSTK         SAVE PGM COUNTER
BSRC   B    *R6             BRANCH TO SOURCE
*
*------- LOAD 1 BYTE FROM THE SOURCE OF THE MOVE
* ENTRY FROM BRANCH TABLE
CPUSRC MOVB *R1+,R11        LOAD BYTE FROM CPU ADR. SPACE
```

```
       B    *R7
VDPSRC MOVB @R1LSB,*R15      LOAD VDP WITH ADR
       MOVB R1,*R15
       INC  R1
       MOVB @VRD0FF(R15),R11 LOAD BYTE FROM VDP
       B    *R7
* ENTRY FROM BRANCH TABLE
GRMSRC MOVB R1,@GWA0FF(R13)  LOAD GROM WITH ADR
       MOVB @R1LSB,@GWA0FF(R13)
       INC  R1               INCREMENT SOURCE ADR
       MOVB *R13,R11         LOAD BYTE FROM GROM
       B    *R7              BRANCH TO DESTN
*
* DESTINATIONS FOR THE MOVE INSTRUCTION
* ENTRY = BR TABLE
CPUDST MOVB R11,*R2+         STORE BYTE IN CPU ADR SPACE
       JMP  TESTLP
* ENTRY = BR TABLE
GRMDST MOVB R2,@GWA0FF(R13)  LOAD GROM WITH ADR.
       MOVB @R2LSB,@GWA0FF(R13)
       INC  R2               INC DESTN ADR
       MOVB R11,@GWD0FF(R13) STORE BYTE IN GROM
       JMP  TESTLP
* ENTRY = BR TABLE
REGDST CB   @R2LSB,R14
       JNE  REGADJ
       COC  @HX0008,R14      TEST VDP MEMORY TYPE
       JNE  REG100
       ORI  R11,>8000        SET 16K MEMORY SELECT
REG100 MOVB R11,@SAVVDP      SAVE VDP REGR
REGADJ MOVB R11,*R15         STORE BYTE IN BDP REGR
       ORI  R2,>80           SET FLAG FOR VDP REGR LOAD
       MOVB @R2LSB,*R15      STORE REGR NUMBER
       INC  R2               INC REGR NUMBER
       JMP  TESTLP
* ENTRY = BR TABLE
VDPDST MOVB @R2LSB,*R15      LOAD VDP WITH DEST ADR.
       ORI  R2,>4000         SET FLAG TO WRITE TO VDP
       MOVB R2,*R15
       INC  R2
       MOVB R11,@VWD0FF(R15) STORE BYTE IN VDP
* ------ TEST END OF LOOP FOR THE MOVE INSTN
TESTLP DEC  R8               DEC NUMBER TO MOVE
       JGT  BSRC             LOOP TILL ZERO
TSTRTN B    @RETNC           RESTORE PROGRAM COUNTER
* COINCIDENCE ROUTINE FOR INSERTION INTO REL4 INTERPRETER
* UPON ENTRACE TO THIS ROUTINE AT LABEL 'COINC' THE
* REGISTERS ARE ASSUMED TO BE SET UP:
*  MSBY   R2=Y2 IN MSBY AND X2 IN LSBY;
*  MSBY   R0=Y1 IN MSBY AND X1 IN LSBY;
* IT IS ALSO ASSUMED THAT THE GROM'S INTERNAL ADDRESS IS SET
* UP PREPARED TO READ (FOLLOWING THE COINC INSTRUCTION):
*   - A ONE BYTE GRANULARITY VALUE, FOLLOWED BY:
*   - A TWO BYTE ADR. POINTING TO THE COINCIDENCE TABLE.
*     THE TABLE IS ASSUMED TO RESIDE IN GROM, AND HAVE THE
*  FOLLOWING FORMAT:
*      BYTE 0- TV = VERTICAL BIT SIZE OF TABLE LESS 1
*      BYTE 1- TH = HORIZ.   BIT SIZE OF TABLE LESS 1
*      BYTE 2- V1 = VERTICAL DOT SIZE OF OBJECT 1/2**GR
*      BYTE 3- H1 = HORIZ.   DOT SIZE OF OBJECT 1/2**GR
*      BYTES 4 ON - THE BIT TABLE ITSELF; THE BITS ARE
* ARRANGED SUCH THAT THE FIRST (TH+1) BITS REPRESENT BOOLEAN
```

```
* COINCIDENCE VALUES CORRESPONDING TO A DELTA Y (Y1-Y2) OF -V1
* THRU -V1+TV AND DELTA X (X1-CX2) VALUES -H1 THRU -H1+TH
*
* ENTRY = BR TABLE
COINC  MOV  R0,R8
       MOV  R8,R3            FIRST GET DELTA Y AND DELTA X
       SB   R2,R3            R3= Y1-Y2= DELTA Y
       SWPB R8               GET X1 IN MSBY
       SWPB R2               GET X2 IN MSBY
       SB   R2,R8            R8 X1-X2 = DELTA X
       MOVB *R13,R0          SET RESLN AND TABLE POINTER
       SRL  R0,8             R0 = GRAN
       MOVB *R13,R5
       SWPB R5
       MOVB *R13,R5
       SWPB R5               R5 = TABLE POINTER
       BL   @PUTSTK          SAVE GROM PC
*
* NOW GET TV,TH,V1,H1, OUT OF THE LST 4 BYTES OF TABLE
*
       MOVB R5,@GWA0FF(R13)  PUT OUT TABLE POINTER LSBY
       SWPB R5
       MOVB R5,@GWA0FF(R13)  PUT OUT TABLE POINTER MSBY
       SWPB R5
       MOVB *R13,R2          R2=TV(MSBY)
       NOP
       MOVB *R13,R1          R1=TH(MSBY)
       NOP
       MOVB *R13,R6          R6=V1(MSBY)
       NOP
       MOVB *R13,R7          R7=H1(MSBY)
* NOW ON WITH THE SHOW, THE REGISTERS ARE NOW SET UP AS:
*      R0= GRANULARITY;
* MSBY R1= TH = CONICIDENCE TABLE HORIZONTAL SIZE -1
* MSBY R2= TV = CONICIDENCE TABLE VERTICAL   SIZE -1
* MSBY R3= Y1 - Y2 = DELTA Y
* MSBY R8= X1 - X2 = DELTA X
*      R5= PNTR TO COINCIDENCE TABLE IN GROM
* MSBY R6= V1 = VERTICAL SIZE OF OBJECT ONE IN DOTS
* MSBY R7= H1 = HORIZ.   SIZE OF OBJECT ONE IN DOTS
*      R13 = GROM READ ADR.
*
       MOV  R0,R0            IF GRANULARITY IS 0, DON'T SHIFT
       JEQ  DNTSHF            BECAUSE 9900 SHIFT BY 0 IS 16
       SRA  R3,R0            DIVIDE DELTA Y BY (2** GRAN)
       SRA  R8,R0            DIVIDE DELTA X BY (2** GRAN)
DNTSHF AB   R7,R8            R8 = B = H1 + DELTA X
       JLT  NOCOIN
       AB   R6,R3            R3 = A = V1 +DELTA Y
       JLT  NOCOIN
       CB   R3,R2            A::TV
       JGT  NOCOIN
       CB   R8,R1            B::TH
       JGT  NOCOIN           RANGE TEST PASSED?
       SRL  R1,8             NOW COMPUTE TABLE INDEX
       INC  R1               R1=TH+1
       SRL  R3,8             R3=A
       MPY  R3,R1            R2=A*(TH+1)
       SRL  R8,8             R8=B
       A    R8,R2            R2= INDEX. COMPUTE TABLE & BIT POSN
       MOV  R2,R0            R0 = INDEX ALSO
       ANDI R2,>FFF8         R2 = ROUNDED DOWN TO LOWER MULT OF 8
```

```
       S    R2,R0            R0 = BIT DISPLACEMENT (0= LEFTMOST)
       SRA  R2,3             R2 = BYTE INDEX INTO TABLE
       A    R5,R2            R2 = ACTUAL ADDRESS OF BYTE
       C    *R2+,*R2+        INC PNTR BY 4 FOR 4 BYTE HEADER
       MOVB R2,@GWAOFF(R13)  PULL PROPER BYTE FROM GROM
       INC  R0
       MOVB @R2LSB,@GWAOFF(R13)
       LI   R2,>2000
       MOVB *R13,R3          R3 = THE BYTE FROM THE TABLE
       SLA  R3,R0            GET PROPER BIT INTO THE STATUS CARRY
       JOC  YUP              IF BIT IS 0, NO COINCIDENCE
NOCOIN CLR  R2               NO, WE HAVE COINCIDENCE
YUP    MOVB R2,@STATUS       YES, WE HAVE COINCIDENCE
       JMP  TSTRTN
*
* ------ COMPUT ADR. FOR MOVE INSTRUCTION
* = BL
MVADDR AB   R9,R9            IS ADR IN GROM (BIT 3=1)?
       JOC  GETMAD           YES & RT FROM THERE
       MOVB *R13,R3          LOAD GROM ADR
       INC  R4               SET GROM FLAG
       MOVB *R13,@R3LSB
       MOV  R11,R12
       AB   R9,R9            IS GROM ADRESS INDEXED (BIT 4=1)?
       JNC  NOGNDX           NO
       MOVB *R13,R1          INDEX IS ONLY ONE BYTE
       BL   @MADDA           GET INDEX VALUE
       A    R0,R3            ADD INDEX TO ADR
NOGNDX MOV  R3,R1            RETURN ADR. IN R1
       SRL  R9,1
       B    *R12             = RT
*  GLOBAL ADR. DECODE:
*    R1= ADDRESS OF VARIABLE
*    R0= VALUE OF VARIABLE
*    SINGLE BYTE VALUES ARE SIGN EXTENDED
*    IF ADR IS IN VDP, THEN THE MSBYTE OF R4 IS SET TO >80
* = BL
GETMAB CLR  R5               BYTE FLAG
*
GETMAD MOVB *R13,R1          GET NEXT BYTE FROM GROM (2ND BYTE)
       JLT  MLONG            LONG ADR. IN GT >80
* = BL
MADDA  SRL  R1,8             SINGLE BYTE ADR.
MADD   AI   R1,PAD           ADD CPU RAM ADR.
       CI   R1,PAD+>7D       CHAR BUFFER?
       JNE  MADC             NO
* LOAD CB FROM VDP MEM INTO CHAR. BUF, LEAVE COPY IN R0
       CLR  R10              SET FLAG TO READ FROM VDP
       MOV  R11,R6           SAVE RTN ADR
       BL   @GETDAD          SET UP VDP RAM ADR (EXPECTS XPT, YPT
*                             TO BE ALREADY PNTG TO CHAR POSN)
       MOV  R6,R11           RESTORE RT ADR.
       MOVB @VRD0FF(R15),R0  LOAD CB FROM DISPLAY
       COC  @HX0002,R14      IS IT MULTICOLOUR MODE??
       JNE  MADE             NO, WHO CARES??
       JNC  C7A2             WHICH NYBBLE??
       SLA  R0,4             LOW NYBBLE
C7A2   SRL  R0,4             HIGH NYBBLE
MADE   MOVB R0,@CHRBUF       STORE CHAR
* = BL
MADC   MOVB *R1,R0           LOAD VALUE OF VAR.
MADC2  MOVB R5,R5            SINGLE OR DOUBLE INST?
```

```
       JNE  MDOUB            DOUBLE
MSIGN  SRA  R0,8             SIGN EXTEND BYTE VALUE
       RT
*
MDOUB  MOVB @1(R1),@R0LSB    GET 2ND HALF OF DOUBLE
       RT
*
MLONG  MOVB *R13,@R1LSB      GET 2ND HALF OF ADR.
       MOV  R1,R10           SAVE CONTROL BITS
       ANDI R1,>FFF          DELETE CONTROL BITS
       CI   R1,>0F00         LOOK FOR EXTENDED ADR.
       JLT  MLONG1           NO, JUST REGULAR ADR (TO >EFF)
       SLA  R1,8             EXTENDED ADR, MAKE ROOM
       MOVB *R13,@R1LSB      GET ADR (UP TO >FFFF)
MLONG1 SLA  R10,2            BASE ADR. NEEDED?
       JNC  MVDP             NO
       MOVB *R13,R6          GET ADR. OF BASE ADR.
       SRL  R6,8
       MOVB @PAD(R6),R0      LOAD BASE ADR
       MOVB @PAD+1(R6),@R0LSB
       A    R0,R1
MVDP   SLA  R10,1            VDP OR 9900 RAM SPACE?
       JNC  MCPU             9900 RAM SPACE
       INCT R4               SET VDP INDICATOR
       SLA  R10,1            INDIRECT ADR?
       JNC  MVDPA            NO
       MOVB @PAD(R1),R0      LOAD INDCT ADR.
       MOVB @PAD+1(R1),@R0LSB
       MOV  R0,R1
MVDPA  MOVB @R1LSB,*R15      LOAD ADR. INTO VDP
       MOVB R1,*R15
       SLA  R0,8             NO OP
       MOVB @VRD0FF(R15),R0  RECALL CONTENTS OF VDP RAM
       MOVB R5,R5            BYTE OR WORD INSTRN?
       JEQ  MSIGN            BYTE
       MOVB @VRD0FF(R15),@R0LSB            GET LSB FROM VDP
       RT
*
MCPU   SLA  R10,1            INDIRECT ADR?
       JNC  MADD             NO
       CI   R1,>7C           INDIRECT ON STATUS BYTE?
       JNE  MADB             NO
       MOVB @STKDAT,R1       GET TOP STACK ADR.
       SB   R14,@STKDAT      R14=01XX
       JMP  MADDA
MADB   MOVB @PAD(R1),R1      LOAD INDIRECT ADR.
       JMP  MADDA
*------ RETURN ADR. FROM LIBRARY OR PROGRAM ------
* = BR TABLE
RGBA   BL   @GETSTK          RECALL GROM BASE ADR.
       MOV  @PAD(R4),R13
       MOVB R4,@GWD0FF(R13)  SYNC THE GROMS
* = BR TABLE
RETURN SZCB @HX20,@STATUS    RESET CONDITION BIT
* = BR TABLE
RETNC  LI   R11,NEXT
* = BL
GETSTK MOVB @STKADD,R4       LOAD ADR. OF SUBROUTINE STACK
       SRL  R4,8             MOVE TO LSB
* ENTRY = BL
GTSTK  DECT @STKADD          NEW VALUE OF STACK POINTER
* ENTRY = BL
```

```
GTSTKL MOVB @PAD(R4),@GWAOFF(R13)          LOAD RTN ADR.
       MOVB @PAD+1(R4),@GWAOFF(R13)
       RT
* ------ PUSH PGM COUNTER IN KONTO STACK
* = BR TABLE
CALL   MOVB *R13,R6           GET BRANCH ADR FROM GROM
       LI   R11,LDKADD        SET RTN POINTER
       MOVB *R13,@R6LSB
* ENTRY = BL
PUTSTK INCT @STKADD           NEW VALUE OF STACK POINTER
       MOVB @STKADD,R4        LOAD ADR. OF STACK
       SRL  R4,8              TO LSB
       MOVB @GRAOFF(R13),@PAD(R4)          SAVE ADR ON STACK
       MOVB @GRAOFF(R13),@PAD+1(R4)
       DEC  @PAD(R4)
       RT
*  ----- LOAD XPTR AND RETURN CHRBUF -----
* ENTRY = BL
GETDA  LI   R10,>4000         LOAD VDP WRITE FLAG
* = BL
GETDAD MOVB @XPT,R7           LOAD VALUE OF XPT
       COC  @HX0002,R14       TEST IF MULTICOLOUR MODE
       JEQ  MCMDA             YES MULTI
       SLA  R7,3              MASK OUT TRUE XPT VALUE
       SRL  R7,8              MOVE TO LSB
       MOVB @YPT,R7           LOAD VALUE OF YPT
       SRL  R7,3              MOVE TO CORRECT POSITION
       A    R10,R7            READ/WRITE FLAG (R10=0=READ)
* ENTRY = BL
SETVDP MOVB @R7LSB,*R15       LOAD CHAR. BUFFER
       MOVB R7,*R15           STORE ADR.
       S    R10,R7            RESTORE R/W FLAG
GETRTN RT
* = BL
SETV   LI   R7,>4000          WRITE TO VDP MEMORY
       JMP  SETVDP
* ----- MULTICOLOUR MODE ADR. COMPUTATION
MCMDA  MOVB @YPT,R0           LOAD YPT
       MOV  R0,R8
       SLA  R8,5              GET 3 LOW BITS AS LSB
       SRL  R8,13
       SRL  R0,11             POSITION 5 HIGH BITS
       SLA  R0,8
       A    R8,R0
       MOV  R7,R8             SAVE XPT
       ANDI R7,>3E00          POSITION XPT BITS
       SRL  R7,6
       A    R0,R7             ADD X VALUE TO DISPLAY ADR
       AI   R7,>0800
       MOVB @R7LSB,*R15       LOAD VDP ADR.
       SLA  R8,8              SET CARRY TO LAD OF XPT
       MOVB R7,*R15
       CI   R11,RTN           STORE OR RETRIEVE CB?
       JNE  GETRTN            STORE
       MOVB @VRD0FF(R15),R0   LOAD BYTE FROM VDP
       MOVB @CHRBUF,R8        LOAD CHAR BUF
       ANDI R8,>0F00          MASK OUT LOW DIGIT
       JOC  USTRCB            JMP IF CB IS LOW DIGIT
       ANDI R0,>0F00          MASK OUT LOW DIGIT
       SLA  R8,4              MOVE CB TO HIGH DIGIT
       JMP  VSTRCB
USTRCB ANDI R0,>F000          MASK OUT HIGH DIGIT
```

```
VSTRCB ORI  R7,>4000         SET BIT TO WRITE TO VDP
       BL   @SETVDP          LOAD VDP ADR.
       A    R8,R0            COMBINE DIGITS
       MOVB R0,@VWD0FF(R15)  STORE BYTE IN VDP
       B    *R6              RETURN TO NEXT
*
*  HAND HELD UNIT DSR
* THERE IS A SEPARATE VERSION OF THE INTERPRETER FOR THE
* T.I. VERSION AND THE MILTON BRADLEY VERSION OF THE HAND
* HELD UNITS. (MB VERSION: HC2.ALBERT.SRC.MBINTR
*
REMOTE LIMI >0000            INTERRUPT 1 ENTRY POINT
       LWPI WKSE             USE REGULAR REGISTERS
       CLR  R12              POINT TO 9901
       COC  @HX0020,R14      IS TIMER FLAG ON?
       JNE  TIM1             NO, SEE WHAT IS INT'G ME
       B    @TIMER           HANDLE TIMER INTERRUPT
*
TIM1   TB   2                VDP INTERUPT?
       JNE  TIMING           YES
*
* PERIPHERAL ROM INTERRUPT ROUTINES MAY CHANGE ALL REGS
* EXCEPT R0, AND R11 TO R15
* R11 HAS THE RETURN ADR. FOR THE INT. ROUTINE
* R12 IS LOADED FOR THE CRU SPACE OF THE PERIPHERAL
* R13 HAS THE CURRENT GROM SLOT ADR.
* R14 HAS A >01XX WHERE THE XX IS USED BY THE INTERPRETER
* R15 HAS THE ADR. OF THE VDP WRITE ADR. ADR.
* INTERRUPT ROUTINE SHOULD DO A RETURN WHEN DONE
*
       LI   R12,>0F00        TABLE OF CRU BITS TO SELECT ROM
       SBO  1                TURN OF EXT. INTERRUPT
ILOOP  SBZ  0                TURN OFF LAST ROM
       AI   R12,>0100        NEXT ROM
       CI   R12,>2000        FINISHED?
       JEQ  EMERGE           END OF ROM CRU BITS
       SBO  0                LOAD CRU BITS TO SELECT ROM
       CB   @H4000,@HX30AA+1 VALID ROM?
       JNE  ILOOP            NO
       MOV  @H400C,R2        GET INT. ROUTINE ADR.
LOOP1  JEQ  ILOOP            NO ROUTINE, GO TO NEXT ROM
       MOV  R2,R0            SAVE POINTER TO NEXT ROUTINE
       MOV  @2(R2),R2        GET ENTRY ADR.
       BL   *R2              JUMP TO INTERRUPT ROUTINE
       MOV  *R0,R2           NEXT ROUTINE'S ADR.
       JMP  LOOP1            GO TO NEXT ROUTINE
EMERGE B    @SNDEXT          END INTERRUPT ROUTINE
* DIFF. ON /4--------
TIMING SBO  2                RESET VDP INT. ON 9901
       MOVB @WKSC+R1+R1,R1
       SLA  R1,1
       JNC  C958
       B    @TIMEXT
C958   SLA  R1,1
       JOC  TSTSND
* END OF /4 DIFFS-----
* ------- AUTO SPRITE MOTION ------- (INT. ROUTINE)
       MOVB @MOTION,R12
       JEQ  TSTSND
       SRL  R12,8
       LI   R2,>8800         =VDPRD+VDPWA
       LI   R3,>8C00         =VDPWD+VDPWA
```

```
       LI   R8,>0780          =RSM0T
ML00P  M0VB @R8LSB,*R15       L0AD ADR. T0 READ SML
       M0VB R8,*R15
       CLR  R4
       M0VB *R2,R4            READ DELTA Y
       CLR  R6
       M0VB *R2,R6            READ DELTA X
       SRA  R4,4              M0VE RIGHT 0NE DIGIT
       M0VB *R2,R5            READ TEMP Y
       SRA  R5,4              M0VE RIGHT 0NE DIGIT
       A    R4,R5
       M0VB *R2,R7
       SRA  R6,4              M0VE RIGHT 0NE DIGIT
       SRA  R7,4              M0VE RIGHT 0NE DIGIT
       A    R6,R7
       AI   R8,-@SAML         CHANGE ADR. T0 SAL
       M0VB @R8LSB,*R15       L0AD ADR. T0 READ SAL
       M0VB R8,*R15
       CLR  R4
       M0VB *R2,R4            READ Y P0SITI0N
       A    R5,R4             ADD Y M0VEMENT T0 P0SITI0N
       CI   R4,6*VDELTA+255
       JLE  0NSCRN
       CI   R4,7*VDELTA
       JH   0NSCRN
       M0V  R5,R5
       JGT  C9B2              JMP IS $+6
       AI   R4,6*VDELTA
C9B2   AI   R4,VDELTA
0NSCRN CLR  R6
       M0VB *R2,R6
       A    R7,R6
       0RI  R8,>4000          =WRVDP
       M0VB @R8LSB,*R15       L0AD ADR. T0 WRITE SAL
       M0VB R8,*R15
       M0VB R4,*R3
       AI   R8,@SAML+2
       M0VB R6,*R3
       SWPB R5
       M0VB @R8LSB,*R15       L0AD ADR T0 WRITE SML
       M0VB R8,*R15
       SRL  R5,4
       M0VB R5,*R3
       SWPB R7
       SRL  R7,4
       M0VB R7,*R3
       AI   R8,2-WRVDP
       DEC  R12
       JGT  ML00P
TSTSND SLA  R1,1              THESE 2 INSTN'S N0T 0N /4
       J0C  SNDEND                    "
       M0VB @STFLGS,R2
       JEQ  SNDEND
       SB   R14,@STFLGS       R14 = 01XX
       JNE  SNDEND
       M0V  @SNDADD,R3
       M0V  R14,R5
       SRL  R5,1
       J0C  SNDRAM
       BL   @PUTSTK
       LI   R5,GWA0FF
       A    R13,R5
```

```
       MOVB R3,*R5
       MOVB @R3LSB,*R5
       MOV  R13,R6
       JMP  USND
SNDRAM LI   R5,VWA
       MOVB @R3LSB,*R5
       MOVB R3,*R5
       LI   R6,>8800          = VDPRD+VDPWA
USND   MOVB *R6,R8
       JEQ  NEWADD
       CB   @HXFFBF,R8        DO I SWITCH SOURCE TYPE?
       JEQ  NEW1              YES
       SRL  R8,8
       A    R8,R3
SLOOP  MOVB *R6,@SGCADR       SEND BYTE TO SOUND CHIP
       DEC  R8
       JNE  SLOOP             LOOP TILL COUNT IS ZERO
       INCT R3
       MOVB *R6,R2
       JEQ  XSOUND
       JMP  SNDXIT
NEW1   XOR  @HX0001,R14       CHANGE VDP TO GROM OR VICE VERSA
NEWADD MOVB *R6,R3
       LI   R2,>100
       MOVB *R6,@R3LSB        GET LOW BYTE OF NEW ADR.
       JMP  SNDXIT
XSOUND SB   R2,R2
SNDXIT MOV  R3,@SNDADD
       MOVB R2,@STFLGS
       CI   R5,VWA
       JEQ  SNDEND
       BL   @GETSTK
SNDEND SLA  R1,1
       JOC  TIMEXT
       LI   R12,>24           LOAD CONSOLEADR. IN CRU BASE
       LDCR @HX0008,3         LOAD FOR COLUMN ZERO
       SRC  R12,7             WAIT???
       LI   R12,6             CRU ADR. FOR CHAR. LINES
       STCR R5,8              RECALL COLUMN INFO.
       CZC  @SHQ,R5           IS IT A SHIFT Q?
       JNE  TIMEXT
       BLWP @>0000            DO A RESET
TIMEXT MOVB @VRSOFF(R15),@VDPST              LOAD VDP ST.
       LWPI WKSC              USE INTERRUPT WKSP
       INCT R11               INCREMENT SCREEN TIMER
       JNE  TIMIN1            NO TIMEOUT
TIMIN  MOVB R10,R12
       SRL  R12,8
       ORI  R12,>8160         TURN OFF VDP
       ANDI R12,>FFBF
HXFFBF EQU  $-2
       MOVB @WKSC+R12+R12+1,@VWA
       MOVB R12,@VWA
TIMIN1 LWPI WKSE              TEST FOR SHIFT Q KEY
       AB   R14,@TIME         INC. TIME IN STATUS BLOCK
       MOV  @WKSC+R2+R2,R12
       JEQ  SNDEXT
       BL   *R12
SNDEXT CLR  R8                CLR REGISTER FOR BASIC
       LWPI WKSC              RESET WKSP
       RTWP                   RETURN TO CALLING ROUTINE
* ------ SEARCH ROM FOR DSR OR LINK -------
```

```
* SEARCH FOR PERIPHERALS, MEM ADR 4000 TO 5FFF.
* ENABLE BY CRU ADR 1000 TO 1F00
* = BR TABLE
SROM    CLR  R1              VERSION FOUND OF DSR ETC
        MOV  @CRULST,R12     SEARCH ROM FOR ROUTINE
        JNE  SG0             IF <> 0, CONTINUE SEARCH
        LI   R12,>0F00       START OVER AGAIN
NOROM   MOV  R12,R12
        JEQ  NOOFF
        SBZ  0
NOOFF   AI   R12,>0100       NEXT ROM'S TURN ON
        CLR  @CRULST         CLR IN CASE WE'RE FINISHED
        CI   R12,>2000       AT THE END (1F00 IS LAST PERIPH)
        JEQ  NOSET           NO MORE PERIPHS TO TURN ON
        MOV  R12,@CRULST     SAVE ADR. OF NEXT CRU
        SB0  0               TURN ON PERIPH
        LI   R2,>4000        START AT BEGINING (PERIPH ADR)
        CB   *R2,@HX30AA+1   IS IT A VALID ROM?
        JNE  NOROM           NO
        AB   @TYPE,@R2LSB
        JMP  SG02
SG0     MOV  @SADDR,R2       CONTINUE WHERE WE LEFT OFF
        SB0  0               TURN PERIPH BACK ON
SG02    MOV  *R2,R2          IS ADR. ZERO?
        JEQ  NOROM           YES, NO PROG. TO LOOK AT
        MOV  R2,@SADDR       REMEMBER WHERE TO GO NEXT
        INCT R2              GO TO ENTRY POINT
        MOV  *R2+,R9         GET ENTRY ADR
        BL   @NAME           SEE IF NAME MATCHES
        JMP  SG0             NO MATCH, TRY NEXT PROGG
        INC  R1              NEXT VERSION FOUND
        BL   *R9             MATCH, CALL SUBROUTINE
        JMP  SG0             NOT RIGHT VERSION
* = BR TABLE
CB16    SBZ  0
        JMP  NOGR2
NOGR1   CLR  *R8
NOGR2   BL   @GETSTK
NOSET   B    @RESET
* ------ SEARCH GROM FOR DSR OR LINK ------
* ENTRY = BR TABLE (FPT)
SGROM   LI   R7,SADDR
        LI   R8,CRULST
        BL   @PUTSTK         SAVE GROM ADR
SGROMA  MOV  *R7,R1          START WHERE WE LEFT OFF
        MOV  *R8,R2          IS IT A RESTART?
        JNE  SGROM3          NO
        LI   R2,>9800        START OF GROMS
SGROM1  LI   R1,>E000        START OF GROM
SGROM3  CZC  @HX1FFF,R1      IS IT A NEW GROM OR CONTIUATION?
        JNE  SGROM2
        MOV  R2,*R8          SAVE GROM ADR
        MOVB R1,@GWA0FF(R2)  LOAD ADR
        MOVB @R1LSB,@GWA0FF(R2)
        AB   @TYPE,@R1LSB    LOOK FOR PGM ADR.
        MOVB R1,@SAVEG       SAVE GROM ADR. OF HEADER
        CB   *R2,@HX30AA+1   VALID GROM?
        JNE  NOGR            NO GROM HERE
HX81    EQU  $+1
SGROM2  MOVB R1,@GWA0FF(R2)  LOOK FOR PGM
        MOVB @R1LSB,@GWA0FF(R2)
        SLA  R10,4           STALL
```

```
       MOVB *R2,R3           READ PGM ADR
       NOP
       MOVB *R2,@R3LSB
       MOV  R3,*R7           GET NEXT HEADER'S ADR
       JEQ  NOGR             IF ZERO, GO TO NEXT PGM
       INCT R3               GO TO PGM ENTRY ADR
       MOVB R3,@GWAOFF(R2)   GO TO PGM ENTRY ADR
       MOVB @R3LSB,@GWAOFF(R2)
       NOP
       MOVB *R2,R9           ENTRY ADR
       SLA  R10,4            STALL
       MOVB *R2,@R9LSB
       BL   @NAME            SEE IF NAME MATCHES
       JMP  SGROMA           NO, LOOK FOR NEXT PGM
       AB   @C030,@STKDAT    FOUND NAME SO PUSH IT
       AB   R14,@TEMP2       INCREASE PGM COUNT
       MOVB @STKDAT,R4
       SRL  R4,8
       DECT R3               POINT BACK TO START OF HEADER
       CB   @TYPE,@HX06      IS IT A USER PGM LOOKUP?
       JNE  SGROM4           YES
       MOV  R3,R9            PUSH HEADER ADR. FOR USER PGM
SGROM4 MOVB R9,@PAD(R4)      NO, PUSH ENTRY ADR
       MOVB @R9LSB,@PAD+1(R4)
       MOV  R2,R13           GO TO THAT LIBRARY
       BL   @GETSTK          RESTORE GROM ADR
       B    @SET             SET STATUS AND RETURN
NOGR   CLR  R1               GET ADR OF GROM HEADER
       MOVB @SAVEG,R1
       AI   R1,->2000        NEXT GROM DOWN
       MOV  R1,*R7           SAVE ADR OF WHERE WE'RE AT
       CI   R1,>E000         FINISHED?
       JNE  SGROM3           NO, CHECK THIS GROM
       C    *R2+,*R2+        INC GROM MAPPED ADR BY 4
       MOV  R2,*R8           SAVE THE NEW MAP ADR
       CI   R2,GR+>40        AT END OF LIBRAY
       JEQ  NOGR1            YES
       MOVB @SCLEN,R5        ARE WE LOOKING FOR A MENU?
       JNE  SGROM1           YES SO DO ONLY ONE SLOT
       JMP  NOGR2            NO, CONTINUE SEARCH
* = BL, CALLED WITH 2 RETURNS
NAME   MOVB @SCLEN,R5        GET LENGTH AS COUNTER
       JEQ  NAME2A           ZERO LENGTH, DON'T DO MATCH
       CB   R5,*R2           DOES LENGTH MATCH?
       JNE  NAME3            NO
       SRL  R5,8             MOVE TO RIGHT PLACE
       LI   R6,FAC
NAME1  CI   R2,GR            IS IT GROM?
       JHE  NAME2            YES, DON'T INC ADR.
       INC  R2
NAME2  CB   *R6+,*R2         IS NAME THE SAME?
       JNE  NAME3            NO
HX06   DEC  R5               MORE TO LOOK AT? REF IS NASTY
       JNE  NAME1            YES
NAME2A INCT R11              RETURN, NAME FOUND
NAME3  RT
*
* ------ NOTE, NOTE, THIS IS NOT THE SAME 'SNAIL' AS /4 ------
SNAIL  BL   @CC28
       B    @>4020
* ENTRY = BR TABLE
CC14   BL   @CC28
```

```
       B    @>401C
* XOP 0 ENTRY VIA VECTORS
CC1C   LWPI >2800
       BL   @CC28
       B    @>4028
* ENTRY = BL
CC28   LI   R12,>1B00
       SBO  0
       RT
* REF NOT FOUND
       DATA 0,0,0
*
* BRANCH TABLE MAX OFFSET = >8 (= 4 ENTRIES)
*
ITAB   DATA MICSLN,MOVDAT,BRESET,BSET
*           00-1F  20-3F  40-5F  60-7F
*
* REFD AS START OF BRANCH TABLE
MSCTAB DATA RETURN,RETNC,RANDNO,KEYSCN,BKGRND,BLONG,CALL,ALL
*             00    01     02     03     04     05   06   07
*
       DATA FORMAT,TSTST,TSTST,ENTRY,TSTST,TSTST,PARSEG,XML
*             08    09    0A    0B     0C   0D    0E     0F
*
       DATA CONTG,EXECG,RTNG,RGBA
*             10   11    12   13
*
       DATA SNAIL,SNAIL,SNAIL,SNAIL,SNAIL,SNAIL
       DATA SNAIL,SNAIL,SNAIL,SNAIL,SNAIL,CC14
*
ATAB   EQU  $->80
       DATA ABS,NEG,INV,CLR,FETCH,CASE,PUSH,IFZ
*           80  82  84   86  88    8A   8C   8E
*
       DATA MINUS,ADD,INCT,DECT
*            90    92  94  96
*
       DATA SNAIL,SNAIL,SNAIL,SNAIL
*
BTAB   EQU  $->50
       DATA ADD,MINUS,MUL,DIV,AND,OR,XOR,STORE
*            A0  A4   A8   AC  B0 B4  B8  BC
*
       DATA EXCH,HIGH,HIGHEQ,GREATR,GREQ,EQUAL,IFAND
*            C0   C4    C8     CC    D0   D4     D8
*
       DATA SRA,SLL,SRL,SRC,COINC,SNAIL,INOUT,DGBA,SNAIL
*            DC  E0  E4  E8   EC   F0    F4    F8   FC
*
MSRC   DATA CPUSRC,GRMSRC,VDPSRC
*
MDST   DATA CPUDST,GRMDST,VDPDST,REGDST
*
FBTAB  DATA LOOP,STRDWN,RPTACR,RPTDWN
       DATA SKPACR,SKPDWN,RPTBLK,SPECL
*
IOTAB  DATA SOUND,SOUND,CRUIN,CRUOUT
       DATA WRITE,READ,VERIFY
*
XMLTAB DATA FLTTAB,XTAB,>2000,>3FC0
       DATA >3FE0,>4010,>4030,>6010
       DATA >6030,>7000,>8000,>A000
```

```
        DATA >B000,>C000,>D000,PAD
FLTTAB EQU  $
```