015

```
OK
  OK
.NLIST
```

```
 70|( CONSTANTS AND IO PORT DEFINITIONS )
 71|( SPECIAL VGS VERBS )
 72|( VGS terse verb       SWAN )
 73|( VGS                  random, ranger )
 74|( VGS                  random, RND, RANGER, RANGERND )
 75|( UPDATE COLOR MAP -- QUICKLY )
 76|( VGS                            FLOOD , VERTICAL , HORIZONTAL )
 77|( VGS              INTHIGHRES , FILL , SCRERASE )
 78|( VGS write routines     pattern representation )
 79|( VGS                  pattern board and magic equates )
 80|( RELABS ) F= relabs SUBR ffrelabs <ASSEMBLE
 81|( VGS                   RELABS )
 82|( VGS                   WRTSET , write )
 83|( VGS                   write con't. )
 84|( VGS                   write con't. )
 85|( VGS                   writep )
 86|( VGS                   WRITEP )
 87|( VGS                   WRITE )
 88|( FRAME, UNFRAME MACROS )
 89|( SPECIAL RELABS FOR BOX )
 90|( PATTERN BOARD BOX COMMAND )
 91|( X Y XS YS MODE BOX ) HEX
 92|( BOX ) E A MOV, 4 CPI, ..XSL4 JRC,     ( JUMP IF LESS THAN 4 )
 93|( BOX ) B DAD, L BX.X Y STX, H BX.X 1+ Y STX, ( UPDATE )
 94|( BOX ) LABEL ..PLOP PIXVAL LDA, M XRA, C ANA, ( PLOP LOOP )
 95|( BOX )
 96|( 16 BIT INTEGER DIVIDE ROUTINE: M N UN/ Q R ) DECIMAL
 97|( SNAP COMMAND )
 98|( 8 X 10 CHARACTER SET - ROTATED )
 99|( MORE CHARS ) C03F , E03F , 700C , 700C , E03F , C03F , ( A )
100|( CHARS ) E01F , F03F , 3020 , 3028 , F01F , E02F , ( Q )
101|( NEW CHARACTER DRAW ROUTINE )
102|( NORMAL BCD ADDITION )
103|( VGS                          CPOST , SPOST )
104|( DISPLAY 6 DIGIT BCD NUMBER -- X Y OPT NUMADDR DISPBCD6 )
105|( n-processor MUSPCU, this is starting load block ) HEX
106|( MUSIC EQUATES FOR VECTOR OFFSETS ) HEX
107|( MUSIC VARIABLES & IY EQUATES FOR OFFSETS ) HEX
108|( STEREO EFFECTS RAM AND VOLUME CRES.-DECRES. RAM ) HEX
109|( MUSIC VARIABLES FOR COMPUTER MUSIC GENERATOR AND SYNCER ) HEX
110|( MUSIC PROCESSOR COMANDS ) HEX ( data,PORT )
111|( MUSIC PROCESSOR COMANDS cont. ) HEX
112|( MUSIC PROCESSOR COMANDS cont., STEREO STUFF and ABCRND ) HEX
113|( NOTE CONSTANTS ) HEX
114|( SIN BTABLE FOR LEFT-RIGHT PAN VOLTAGES ) DECIMAL
115|( HELPING SUBR's for MUSCPU *** NOTICE *** ) HEX
116|( HELPING SUBR's cont. ) HEX
117|( MUSIC PROCESSOR- emusic )
118|( OPCODE SUBR's, 0-4 )
119|( OPCODE SUBR's,  5-6, HL= MUSPC )
120|( OPCODE SUBR's, 8-0B, 10H )
121|( OPCODES 0C-0F )
122|( OPCODES 11-16 I/O PORT OUTPUTS and PAN COUNTING, 1AH ) HEX
123|( STEREO OPCODE 1A, THUMPER 1B, MUSIC GENERATOR 07H ) HEX
124|( OPCODE ADDRESS TABLE and FORWARDS ) HEX
125|( COMPMUSIC's +-disp., 15MOD, NOTABLE and THUMPLOCATION ) HEX
126|( STEREO STUFF, LIMITCOUNTING )
127|( ** MUSCPU **                    **STEREO** ) HEX
128|( ** MUSCPU **                    **STEREO** ) HEX
129|( MUSCPU                        NOTETIMER,MOSYNC )
130|( MUSCPU cont.              MORAMBLE, LOWMOVER, HIGHMOVER )
131|( MUSCPU cont.,                MORAMBLE cont., STEPMOVER )
132|( MUSCPU                       VOLUME MOVING )
133|( MUSCPU                      STEPMOVER, COMPDURMOVER )
134|( MUSPCU cont.,                TBMOVER, NOMOVER )
```

```
135|( ** MUSIC INTERRUPTER **          COMPUTER MUSIC ) HEX
136|( MUSCPU cont.,                    RANDOM NOTES )
137|( MUSCPU cont.,                    PROCESS the score, )
138|( MUSIC PROCESSOR-                 MUSCPUS PUT TOGETHER ) HEX
139|( MUSIC PROCESSOR- ALL xmusics NEED AN IY LOAD ) HEX
140|( MUSCPU SUBROUTINE CALLS )
141|( MUSIC PROCESSOR- EMUSIC, BMUSIC, ... )
142|( MUSIC PROCESSOR- E2MUSIC, B2MUSIC, ... )
143|( JAYS VIDEO GAME GOODIES ) : CL SCRERASE ;
144|( QUEUE - VECTOR MANIPULATION ROUTINES )
145|( VECTOR FIELD EQUATES CONTINUED )
146|( STATUS BIT EQUATES )
147|( VGS                    VWRITE )
148|( VGS                    VERASE )
149|( GLOBAL GAME RAM AREA START )
150|( STORAGE ALLOCATOR GOODIES )
151|( ADD NODE TO QUEUE ROUTINE )
152|( DELETE FROM QUEUE )
153|( ADVANCE TO NEXT NODE ON QUEUE )
154|( INCREMENT TIME BASES - C = TIME BASE, IY = Q HEAD )
155|( NEW, IMPROVED, HOTROD INTERRUPT SYSTEM ) DECIMAL
156|( RESUME BACKGROUND - END INTERRUPT )
157|( TRY TO RUN SOMETHING IN FOREGROUND )
158|( BACKGROUND END INTERRUPT )
159|( INTERRUPT START ROUTINE ) HEX
160|( ROUTINE TO DELETE VECTOR IF STATUS SO INDICATES )
161|( MACROS TO GENERATE ANIMATION OPCODES ) DECIMAL
162|( MORE ANIMATION MACRO STUFF )
163|( ANIMATION INTERPRETER ROUTINES )
164|( MORE ANIMATION INTERPRETER ROUTINES )
165|( YET MORE ANIMATION INTERPRETER ROUTINES )
166|( THE ABSOLUTELY LAST SCREEN OF ANIMATION INTERPRETER STUFF )
167|( JUMP TABLE FOR INTERPRETER ROUTINES )
168|( ANIMATION UPDATOR ROUTINE )
169|( DECREMENT ANIMATION TIMERS, COMPUTE VECTORING TIME )
170|( TIME BASED VECTOR UPDATE - IX=VECTOR ADDR, IY=QUEUE ENTRY )
171|( INITIALIZE INTERRUPT VERBS )
172|( SUBROUTINE TO UPDATE PATTERN USING XOR )
173|( SUBROUTINE TO VECTOR USING SECOND DERIVITIVE )
174|( SUBROUTINE TO UPDATE PATTERN USING XOR AND 2ND DERV VECTOR )
175|( UPDATE VECTOR FROM JOYSTICK ) HEX 11 C= JOYSTICK
176|( SUBROUTINE TO UPDATE PATTERN FROM JOYSTICK )
177|( COMPUTE DELTA FOR 1 COORDINATE )
178|( CLEAR VECTOR ) F= INIZL
179|( SUBROUTINE TO PUT VECTOR ON PROCESS Q )
180|( XVMOVE COMMAND - MOVE AN EXISTING VECTOR )
181|( XSTART COMMAND - START AN EXISTING VECTOR )
182|( START A VECTOR WITH JUST INITIAL X AND Y ) DECIMAL
183|( CHECK FOR INTERCEPT WITH VECTOR )
184|( CHECK GROUP OF VECTORS FOR INTERCEPT )
185|( NUMBER PATTERNS , 5 X 7 ORDERED 0-9 )
186|( ROUTINE TO DISPLAY A BCD NUMBER 3 DIGITS LONG FROM VECTOR )
187|( INTERRUPT WRITE NUMBER ROUTINE )
188|( BASE STATION )
189|( SMALL BASE ) DECIMAL DATA SMALBASE 4 B, 11 B, QUAD
190|( GORF ) DECIMAL DATA GORF 6 B, 15 B, QUAD
191|( GORF2 ) DECIMAL DATA GORF2 6 B, 15 B, QUAD
192|( FIRE BASE EXPLOSION PATTERN )
193|( ANOTHER FIREBASE EXPLOSION PATTERN )
194|( CONTINUATION OF FBEXP2, PHASOR AND NULFAT )
195|( FBEXP3 )
196|( CONTINUED FBEXP3 )
197|( FBEXP4 )
198|( FBEXP4 CONTINUED )
199|( FIREBASE EXPLOSION 5 )
200|( FBEXP5 CONTINUED )
```

```
201|( FIRE BASE EXPLOSION 6 )
202|( FIRE BASE EXPLOSION 6 CONTINUED )
203|( ALIEN EXPLOSION PATTERN )
204|( MORE ALIEN EXPLOSIONS )
205|( EXPLOSION PATTERNS ) DECIMAL
206|( INVADERS- PLAYER SHOOTING SOUND, ID ) HEX
207|( INVADERS- PLAYER EXPLOSION, 1G -- AND PZIP PZ ) HEX
208|( SPACE MISSIONS ZPiP SOUND - ZP ) HEX
209|( DRAW FIREBASES ON SCREEN )
210|( GAME VARIABLES AND CONSTANTS )
211|( INITIALIZE GAME SCREEN ) HEX
212|( RACK UPDATOR )
213|( SUBROUTINES TO CALCULATE DISPLACEMENTS FOR RACK MEMBER ) HEX
214|( WAIT AND ANIMATION TRACKING TABLE ROUTINES ) HEX
215|( RECOMPUTE LIMITS ) HEX
216|( SUBR TO STEP MASTER COORDS ONE TICK AND LIMIT CHECK ) HEX
217|( WE FOUND AN INVADER - WRITE HIM )
218|( REWRITE A RACK MEMBER USING NORMAL PATTERNS )
219|( REENTER RACK ) HEX
220|( INTERRUPT ROUTINE TO REENTER A GALAXIAN ) DECIMAL
221|( CHECK FOR INTERCEPT WITH RACK MEMBER )
222|( ANIMATION LIST AND ROUTINE TO EXPLODE THE FIREBASE )
223|( SCORIN ) HEX TABLE ASTBL 60 , 60 , 80 , 100 , 300 , 200 ,
224|( MORE SCORING GOODIES )
225|( BACKGROUND PHASOR INTERCEPT PROCESSING ROUTINES )
226|( ROUTINE TO CALL FROM SCAN LOOP )
227|( ANIMATION SUBR TO INITIALIZE THE FIRE BASE )
228|( CHECK FOR PLAYER HIT )
229|( COMMON INITIALIZATION GOODIES )
230|( SPECIAL ROUTINE TO MOVE PHASOR BLAST )
231|( START OR RESTART THE PHASOR MOVING )
232|( CHECK FIRE SWITCH )
233|( AWAIT THE ARRIVAL OF THE VERTICAL INTERVAL )
234|( NEW COLOR ROUTINES )
235|( FADE UP/DOWN ROUTINES )
236|( FORCE FIELD DRAWER ) DECIMAL
237|( MORE FORCE FIELD GOODIES )
238|( RADIAL LINE GENERATOR )
239|( RADIAL EFFECT VARIABLES )
240|( NEAT SUBROUTINES )
241|( SUBR TO WRITE NEXT PIXEL IN A LINE )
242|( OTHER NEAT STUFF )
243|( GENERATE A LINE )
244|( LINE GENERATOR - CLIP CHECK )
245|( LINE GENERATOR - SET DELTAS )
246|( ADJUST DELTAS TO QUADRANT, AND BIAS TO EFFECT CENTER )
247|( SET CENTER OF LINE EFFECT )
248|( CHECK FOR INTERCEPT WITH ANY OF THE ATTACKERS )
```

```
 +--------Block     70--------
 0|( CONSTANTS AND IO PORT DEFINITIONS )
 1|{ : C= } CONSTANT { ; } { : V= } VARIABLE { ; }
 2|HEX 0 C= CC? ( 0 TO CROSS COMPILE, 1 FOR NORMAL )
 3|CC? IFTRUE 0F000 C= RAMBASE 0FFFF C= LASTRAMADDR
 4|OTHERWISE 0D000 C= RAMBASE 0DFFF C= LASTRAMADDR IFEND
 5|{ : NC= } 1+ DUP C= { ; } { : SC= } DUP C= { ; }
 6|{ : T= } TABLE { ; }    { : A= } ARRAY { ; }
 7|{ : BT= } BTABLE { ; }    { : BA= } BARRAY { ; }
 8|{ : BV= } BVARIABLE { ; } { : F= } FORWARD { ; }
 9|0 C= COL0R  1 C= COL1R  2 C= COL2R  3 C= COL3R
10|4 C= COL0L  5 C= COL1L  6 C= COL2L  7 C= COL3L
11|0B C= COLBX  9 C= HORCB  0A C= VERBL
12|10 C= TONMO  11 C= TONEA  12 C= TONEB  13 C= TONEC
13|14 C= VIBRA  15 C= VOLAB  15 C= VOLC  17 C= VOLN  18 C= SNDBX
14|0D C= INFBK  0E C= INMOD  0F C= INLIN  8 C= CONCM 0F C= HORAF
15|0C C= MAGIC  19 C= XPAND  8 C= INTST  0E C= VERAF -->
 +--------Block     71--------
 0|( SPECIAL VGS VERBS )
 1|CODE DI DI, NEXT ( disable interrupts )
 2|CODE EI EI, NEXT ( enable interrupts )
 3|CODE XDI DI, A XRA, INMOD OUT, NEXT
 4|: MS 0 DO 4 0 DO LOOP LOOP ;
 5|CC? IFTRUE
 6|: ROMIT DP @ other @ DP ! ; : TIMOR DP @ other ! DP ! ;
 7|IFEND
 8|-->
 9|
10|
11|
12|
13|
14|
15|
 +--------Block     72--------
 0|( VGS terse verb       SWAN )
 1|CODE SWAN   ( swap nibbles in low byte )
 2|  ( in- # to be swapped )
 3|  ( out- swapped result )
 4|  H POP, L A MOV, RLC, RLC, RLC, RLC, A L MOV, H PUSH, NEXT
 5|-->
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block     73--------
 0|( VGS                    random, ranger )
 1|2 A= RND#       ( you must seed RND# !!!!!!!!! )
 2|SUBR random ( 32 bit random # generator )
 3|  ( out- randomly selected # in DEHL )
 4| B PUSH, 0 RND# LBCD, 1321 H LXI, B DAD, H PUSH,
 5| 2776 H LXI, B DADC, 1 RND# LDED, D DAD, XTHL,
 6| B DAD, XTHL, D DADC, XTHL, B DAD, XTHL, D DADC, XTHL,
 7| E D MOV, B E MOV, C B MOV, 0 C MVI, B DAD, 0 RND# SHLD,
 8| XTHL, D DADC, 1 RND# SHLD, D POP, B POP, RET,
 9|SUBR ranger ( pass # in HL, range in DE, ) B PUSH, EXX,
10| 0 H LXI, H D MOV, L E MOV, EXX, D PUSH, B POP, XCHG,
11| 0 H LXI, BEGIN, B SRLR, C RARR, CY, IF, D DAD, EXX, D DADC,
12| EXX, THEN, B A MOV, C ORA, <>, IF, E SLAR, D RALR, EXX, E RALR,
13| D RALR, EXX, ( SWAP ) JMP, THEN, EXX, B POP, RET,
14|-->
15|
 +--------Block     74--------
 0|( VGS                    random, RND, RANGER, RANGERND )
 1|SUBR rnd ( pass range in DE, returns # in HL )
 2| D PUSH, random CALL, D POP, ranger CALL, RET,
 3|CODE RANDOM ( out= # on stack ) random CALL, H PUSH, NEXT
 4|CODE RANGER ( pass range, # on stack )
 5| ( result is # times range / FFFF, ie. 30H 8000H ---- is 18H )
 6| D POP, H POP, ranger CALL, H PUSH, NEXT
 7|CODE RND ( pass range on stack )
 8| random CALL, D POP, ranger CALL, H PUSH, NEXT
 9|-->
10|
11|
12|
13|
14|
15|
 +--------Block     75--------
 0|( UPDATE COLOR MAP -- QUICKLY )
 1|CODE COLOR EXX, H POP, 800 B LXI,
 2|BEGIN, M A MOV, A OUTP, H INX, C INR, LOOP,
 3|EXX, NEXT
 4|-->
 5|
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block     76--------
 0|( VGS                              FLOOD , VERTICAL , HORIZONTAL )
 1|CODE FLOOD   ( set all color ports to the same value )
 2|  ( in- byte color value )
 3|  ( out- screen color ports set to same value )
 4|EXX, H POP, L A MOV, 800 B LXI, BEGIN, A OUTP, C INR, LOOP,
 5|EXX, NEXT
 6|: VERTICAL   ( intialize vertical blank port )
 7|  ( in- # of line )
 8|  ( out- vertical blank port set ) 0A OUTP ;
 9|: HORIZONTAL   ( sets horizontal color boundry and outside
10|                 pixel value )
11|  ( in- x value for boundry , outside screen pixel value
12|    bits 6 and 7 determine value )
13|  ( out- horizontal color boundry port set )
14|  DUP SWAB OR 9 OUTP ;
15|-->
 +--------Block     77--------
 0|( VGS              INTHIGHRES , FILL , SCRERASE )
 1|: INTHIGHRES    ( intialize screen for high resolution mode )
 2|  1 8 OUTP ( con,com port ) C0 VERTICAL 0 HORIZONTAL ;
 3|: FILL   ( fill screen whith constant data )
 4|  ( in- constant , starting address , # of bytes to fill )
 5|  ( out- does sequential fill whith constant specified )
 6|  ROT ROT 2DUP ! SWAP DROP DUP 1+ ROT 1- BMOVE ;
 7|: SCRERASE   ( erase entire screen except last 1 k )
 8|  0 4000 3C00 FILL ;
 9|DECIMAL -->
10|
11|
12|
13|
14|
15|
 +--------Block     78--------
 0|( VGS write routines     pattern representation )
 1|-->            PATTERN REPRESENTATION
 2|  Pattern header requirments are determined by the write
 3|  routine used. The following diagram shows the hierarchy
 4|  used :
 5|\             VWRITER , WRITER        X displacement
 6|                                      Y displacement
 7|                       WRITER         X size
 8|                                      Y size
 9|                       WRITE          pattern data ---
10|
11|  If a pattern is to be written with a shift it must be self
12|  flushing. A pattern is self flushing if the right side 3
13|  bits are all zeros. Pattern padding is required in some cases.
14|  The character set is not self flushing therfor it must be
15|  used only on byte boundrys.
```

```
 +--------Block     79--------
 0|( VGS                    pattern board and magic equates )
 1|HEX
 2|( pattern board ports )
 3|78 C= PBLINADRL 79 C= PBLINADRH 7A C= PBSTAT 7B C= PBAREADRL
 4|7B C= PBXMOD    7C C= PBAREADRH 7D C= PBXWIDE 7E C= PBYHIGH
 5|  ( pattern board status port bits )
 6|0 C= PBDIR 1 C= PBEXP 2 C= PBCONS 3 C= PBFLUSH
 7|4 C= PBFLIP 5 C= PBFLOP
 8|  ( magic register bits )
 9|2 C= MRROT 3 C= MREXP 4 C= MROR 5 C= MRXOR
10|6 C= MRFLOP 7 C= MRFLIP
11|-->
12|
13|
14|
15|
 +--------Block     80--------
 0|( RELABS ) F= relabs SUBR ffrelabs <ASSEMBLE
 1|7 C BIT, 0< >, IF, 1 Y A LDX, H ADD, A DCR, A H MOV,
 2|THEN, 6 C BIT, 0< >, IF, 0 Y A LDX, D ADD, A DCR,
 3|A D MOV, THEN, ( FALL INTO ... )
 4|LABEL relabs ( relative X Y to magic address conversion )
 5|  ( in- BC=exp/mag DE=x HL=y )
 6|  ( out- BC=exp/mag+shift HL=scradr )
 7|  H A MOV, 0 H MVI, A L MOV,
 8|  H DAD, H DAD, H DAD,
 9|  H DAD, D PUSH, L E MOV, H D MOV, H DAD, H DAD, ( *64 )
10|  D DAD, ( *80 ) XCHG, H POP, ( x )
11|  L A MOV,  ( SAVE BIT CNT ) H L MOV, 0 H MVI, D DAD, ( x+y )
12|  RLC, RLC, HEX 3 ANI,
13|  MRFLOP C BIT, 0< >, IF, NEG, 0=, IF, H DCX, THEN, THEN,
14|  3 ANI, A E MOV, C A MOV, FC ANI, E ORA, A C MOV, RET,
15|ASSEMBLE> -->
 +--------Block     81--------
 0|( VGS                     RELABS )
 1|CODE RELABS ( relative to absolute conversion )
 2|  ( in- exp/mag , X , Y )
 3|  ( out- exp/mag+shift , scradr )
 4|  EXX, ( save BC )
 5|  H POP, ( Y ) D POP, ( X ) B POP, ( exp/mag )
 6|  relabs CALL, B PUSH, ( exp/mag+shf )
 7|  H PUSH, ( scradr ) EXX, NEXT
 8|-->
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block     82--------
 0|( VGS                    WRTSET , write )
 1|1 BV= WRTSYS  ( write system flag )
 2|: WRTSET WRTSYS B! ; ( write system byte intialize )
 3|                     ( 0<> for pattern board 0= for software )
 4|
 5|SUBR write   ( write pattern on screen )
 6|    ( in- BC=exp/mag+shift DE=y/x size HL=scradr IY=patadr )
 7|         ( WRTSYS 0<> for pattern board 0= for software write )
 8|    ( out- C=mag+shift ; pattern on screen )
 9|-->
10|
11|
12|
13|
14|
15|
 +--------Block     83--------
 0|( VGS                    write con't. )
 1|  ( pattern board write )
 2|  B A MOV, XPAND OUT, C A MOV, MAGIC OUT, HEX 24 A MVI,
 3|  MRFLIP C BIT, 0<>, IF, PBFLIP A SET, THEN,
 4|  MRFLOP C BIT, 0<>, IF, PBFLOP A RES, THEN,
 5|  MREXP  C BIT, 0<>, IF, PBEXP  A SET, THEN,
 6|  A B MOV, PBSTAT OUT, ( B=status C=magic )
 7|  H PUSH,
 8|  Y PUSHX, H POP, L A MOV, PBLINADRL OUT,
 9|                  H A MOV, PBLINADRH OUT,
10|  H POP,
11|                  L A MOV, PBAREADRL OUT,
12|                  H A MOV, PBAREADRH OUT,
13|-->
14|
15|
 +--------Block     84--------
 0|( VGS                    write con't. )
 1|  E H MOV, ( X size )
 2|  MREXP C BIT, 0<>, IF, H RLCR, ( *2 ) THEN,
 3|  H DCR, ( H=X size zero relative )
 4|  MRFLIP C BIT, 0<>, IF,
 5|    MRFLOP C BIT, 0<>, IF, DECIMAL -80 A MVI, H ADD,
 6|                       ELSE, DECIMAL -80 A MVI, H SUB, THEN,
 7|                    ELSE,
 8|    MRFLOP C BIT, 0<>, IF, DECIMAL  80 A MVI, H ADD,
 9|                       ELSE, DECIMAL  80 A MVI, H SUB, THEN,
10|                    THEN, ( A=Xmod ) PBXMOD OUT,
11|  HEX H A MOV, PBYHIGH OUT,
12|  D A MOV, ( Y size ) A DCR, ( 0 rel ) PBYHIGH OUT,
13|  RET,
14|-->
15|
```

```
 +--------Block     85--------
 0|( VGS                    writep )
 1|SUBR writep  ( does write with pattern size header on pattern )
 2|   ( in- BC=exp/mag+shift DE=y/x size HL=scradr IY=patadr )
 3|        ( WRTSYS 0< > for pattern board 0= for software write )
 4|   ( out- C=mag+shift ; pattern on screen )
 5|  0 Y E LDX, ( X size ) Y INXX,
 6|  0 Y D LDX, ( Y size ) Y INXX, write JMP,
 7|-->
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block     86--------
 0|( VGS                    WRITEP )
 1|CODE WRITEP   ( write with pattern size header on pattern )
 2|  ( in- x , y , patadr , ex/mag )
 3|        ( WRTSYS 0< > for pattern board 0= for software write )
 4|  ( out- pattern on screen )
 5|  Y PUSHX, H POP, EXX, B POP, Y POPX, H POP, D POP,
 6|  relabs CALL,
 7|  writep CALL, EXX, H PUSH, Y POPX, NEXT
 8|CODE FFWRITEP Y PUSHX, H POP, EXX, B POP, Y POPX, H POP, D POP,
 9|ffrelabs CALL, writep CALL, EXX, H PUSH, Y POPX, NEXT
10|-->
11|
12|
13|
14|
15|
 +--------Block     87--------
 0|( VGS                    WRITE )
 1|CODE WRITE   ( write with X Y sizes ; pattern with no header )
 2|  ( in- x , y , patadr , y/x size ex/mag )
 3|        ( WRTSYS 0< > for pattern board 0= for software write )
 4|  ( out- pattern on screen )
 5|  Y PUSHX, H POP, EXX, B POP, ( ex/mag ) H POP, ( sizes )
 6|  Y POPX, ( patadr ) D POP, ( Y ) XTHL, ( H<-X S<-sizes )
 7|  XCHG, ( X<->Y ) relabs CALL, D POP, ( sizes )
 8|  write CALL, EXX, H PUSH, Y POPX, NEXT
 9|-->
10|
11|
12|
13|
14|
15|
```

```
 +--------Block     88--------
 0|( FRAME, UNFRAME MACROS )
 1|2 C= FR.P1 4 C= FR.P2 6 C= FR.P3
 2|{ : FRAME ) { [ } ASM { ] } Y PUSHX, 0 Y LXIX, SP DADY, { ; }
 3|{ : UNFRAME ) { [ } ASM { ] } Y POPX, { ; }
 4|-->
 5|
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block     89--------
 0|( SPECIAL RELABS FOR BOX )
 1|SUBR R2A ( RELATIVE TO ABSOLUTE CONVERSION FOR BOX AND LINE )
 2|E A MOV, 3 ANI, PSW PUSH, D PUSH, D SRLR, E A MOV, RAR,
 3|A ANA, RAR, C L MOV, 0 H MVI, H DAD, H DAD, H DAD,
 4|H DAD, L E MOV, H D MOV, H DAD, H DAD, D DAD,
 5|A E MOV, 0 D MVI, D DAD, D POP, PSW POP, RET,
 6|-->
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block     90--------
 0|( PATTERN BOARD BOX COMMAND )
 1|( X Y XS YS MODE BOX )
 2|( PARAMETER ADDRESS EQUATES )
 3|DECIMAL
 4|2 C= BX.M 4 C= BX.YS 6 C= BX.XS 8 C= BX.Y 10 C= BX.X
 5|( SCRATCH AREA USED BY BOX COMMAND )
 6|0 BVARIABLE WRMODE 0 BVARIABLE PIXVAL
 7|( F=ARD DIFFERENCE DECLARATIONS )
 8|F= ..SKIP F= ..MNE F= ..MSL4 F= ..BLOD F= PBBOX
 9|F= ..XSTL F= ..STMO F= ..XSTF F= ..XSBG F= ..BOXF
10|F= ..PLOT F= OUTPB F= ..CFBB F= ..FFF F= ..DOSF
11|F= ..XSLA F= ..XORL F= ..OUTM F= NULRET
12|HEX DATA MSKTBL 0 C, F8 B, 80 B, FF B,
13|-->
14|
15|
```

```
 +--------Block     91--------
 0|( X Y XS YS MODE BOX ) HEX
 1|CODE BOX <ASSEMBLE
 2|         FRAME   EXX,
 3|         BX.M Y A LDX,
 4|         A C MOV,                  ( IS MODE = 4 )
 5|         4 CPI, ..SKIP JZ,       ( IF SO SKIP IT )
 6|         8 CPI, ..SKIP JNC,       ( SKIP IF >= 8 TOO )
 7|         4 ANI, WRMODE STA,       ( ISOLATE AND STUFF MODE )
 8|         C A MOV, 3 ANI, A C MOV, ( GET A BYTE ALL THE SAME )
 9|         0 B MVI, MSKTBL H LXI,
10|         B DAD, M A MOV, PIXVAL STA, ( AND REMEMBER AS PIXVAL )
11|         BX.XS Y E LDX, BX.XS 1+ Y D LDX, ( DE=XS )
12|LABEL ..BOXP E A MOV, D ORA, ..SKIP JRZ, ( QUIT IF XS=0 )
13|         BX.X Y A LDX, 3 ANI,         ( ON A BYTE BOUNDARY? )
14|         ..MNZ JRNZ,                  ( NO - JUMP )
15|         D A MOV, A ORA, ..CPBB JRNZ,    ( IF >256 USE PB ) -->
 +--------Block     92--------
 0|( BOX ) E A MOV, 4 CPI, ..XSL4 JRC,      ( JUMP IF LESS THAN 4 )
 1|         8 CPI, ..SLOB JRC,              ( IF <8 DON'T USE PB )
 2|LABEL ..CPBB PBBOX CALL, ..XST1 JMPR,    ( CALL DRAW WITH PB )
 3|LABEL ..SLOB FF C MVI, ..STRC CALL,      ( PAINT A FULL STRIPE )
 4|         4 C MVI, ..XSTF JMPR,
 5|LABEL ..MNZ 3 ANI, A B MOV, ( COMPUTE MIN(X,4-MOD(XS-4 )
 6|         4 A MVI, B SUB, D 0 BIT, ..XSBG JNZ, ( JMP IF XS>256 )
 7|         E CMP, ..XSBG JC, E A MOV,     ( OR > MOD )
 8|LABEL ..XSBG A C MOV, B PUSH, B C MOV, ( MOD IS BIGGER )
 9|         A B MOV, A XRA,
10|LABEL ..FFF  RRC, RRC, C0 ORI, ..FFF DJNZ, C B MOV, ( MASK )
11|LABEL ..DOSF RRC, RRC, 3F ANI, ..DOSF DJNZ, ( SHIFT MOD TIMES )
12|         A C MOV, ..STRC CALL, B POP,       ( DRAW PART STRIPE )
13|LABEL ..XSTF 0 B MVI,
14|LABEL ..XST1 BX.X Y L LDX, BX.X 1+ Y H LDX, ( HL=X )
15|-->
 +--------Block     93--------
 0|( BOX ) B DAD, L BX.X Y STX, H BX.X 1+ Y STX, ( UPDATE )
 1|XCHG, A XRA, B DSBC, XCHG, ..BOXP JMP, ( SUBTRACT SIZE )
 2|LABEL ..XSL4 A B MOV, A XRA,             ( PAINT FINAL STRIPE )
 3|LABEL ..XSLA RRC, RRC, C0 ORI, ..XSLA DJNZ, ( FORM FINAL MSK )
 4|         A C MOV, ..STRC CALL,           ( AND DO FINAL STRIPE )
 5|LABEL ..SKIP UNFRAME H POP, H POP, H POP, H POP, H POP,
 6|         EXX, NEXT ( QUIT CIRCLE ROUTINE )
 7|LABEL ..STRC D PUSH, C B MOV, BX.X Y E LDX, ( STRIPE DRAWER )
 8|         BX.M 1+ Y D LDX, BX.Y Y C LDX,  ( GET COORDS )
 9|         R2A CALL, H S SET, B C MOV,     ( R2A, DYNABIC, RESET )
10|         S0 D LXI, BX.XS Y B LDX, WRMODE LDA, A ANA,
11|         ..PLOP JNZ,     ( JUMP IF PLOP, XOR ROUTINE FOLLOWS )
12|LABEL ..XORL PIXVAL LDA, C ANA, M XRA, A M MOV, ( UPDATE PIXL )
13|         D DAD, ..XORL DJNZ,          ( UPDATE ADDR, LOOP BACK )
14|         D POP, NEXT,                  ( XOR STRIPE DONE )
15|-->
```

```
 +--------Block     94--------
 0|( BOX ) LABEL ..PLOP PIXVAL LDA, M XRA, C ANA, ( PLOP LOOP )
 1|         M XRA, A M MOV, D DAD, ..PLOP DJNZ, ( USE XOR TRICK )
 2|         D POP, RET,
 3|LABEL PBBOX
 4|         D PUSH, D SRLR, E RARR, D SRLR, E RARR, ( DE=DE/4 )
 5|         E B MOV,
 6|         BX.Y Y C LDX, BX.X Y E LDX, BX.X 1+ Y D LDX, ( COORDS )
 7|         R2A CALL, WRMODE LDA, 4 ANI, ( CONVERT, CHECK WR TYPE )
 8|         ..OUTM JRZ, 20 A MVI, ( JUMP IF PLOP, ELSE ITS XOR )
 9|LABEL ..OUTM MAGIC OUT, 20 A MVI, PIXVAL D LXI, ( SET MR, ST )
10|         OUTPB CALL, B DCR, 50 A MVI, B SUB, ( COMPUTE XMOD )
11|         PBXMOD OUT, B A MOV, PBXWIDE OUT, ( THEN WIDTH )
12|         BX.YS Y A LDX, A DCR, PBYHIGH OUT, ( THEN HEIGHT )
13|         D POP, B L MOV, 0 H MVI, L INR,  ( COMPUTE BYTES USED )
14|         H DAD, H DAD, L C MOV, H B MOV, RET,
15|-->
 +--------Block     95--------
 0|( BOX )
 1|LABEL OUTPB ( ROUTINE TO OUTPUT STUFF TO PAT BOARD )
 2|         PBSTAT OUT, E A MOV, PBLINADRL OUT, ( STAT AND LINEAR )
 3|         D A MOV, PBLINADRH OUT, L A MOV,
 4|         PBAREADRL OUT, H A MOV, PBAREADRH OUT, ( AREA )
 5|LABEL NULRET RET,
 6|ASSEMBLE>
 7|DECIMAL
 8|-->
 9|
10|
11|
12|
13|
14|
15|
 +--------Block     96--------
 0|( 16 BIT INTEGER DIVIDE ROUTINE: M N UN/ Q R ) DECIMAL
 1|FORWARD .ZERO FORWARD IDV50 FORWARD IDV60
 2|FORWARD IDV10 FORWARD IDV20 FORWARD IDV30 FORWARD IDV40
 3|SUBR unsdiv <ASSEMBLE L C MOV, H B MOV, D A MOV, 0 H LXI,
 4|E ORA, .ZERO JRZ, B A MOV, 16 B MVI,
 5|LABEL IDV10 C RALR, RAL, H DADC, D DSBC,
 6|LABEL IDV20 CMC, IDV50 JRNC,
 7|LABEL IDV30 IDV10 DJNZ, IDV60 JMPT,
 8|LABEL IDV40 C RALR, RAL, H DADC, A ANA, D DADC,
 9|IDV30 JRC, IDV20 JRZ,
10|LABEL IDV60 IDV40 DJNZ, D DAD, A ANA, ( MAKE IT POS )
11|LABEL IDV50 C RALR, RAL, A D MOV, C E MOV,
12|LABEL .ZERO RET, ASSEMBLE>
13|SUBR UNSDIV H PUSH, D DSBC, CY, IF, 0 D LXI, H POP, ELSE,
14|H POP, unsdiv CALL, THEN, RET, CODE UN/ EXX, D POP, H POP,
15|UNSDIV CALL, H PUSH, D PUSH, EXX, NEXT DECIMAL -->
```

```
 +--------Block     97--------
 0|( SNAP COMMAND )
 1|HEX CODE snap EXX,
 2|25 A MVI, PBSTAT OUT,
 3|H POP,
 4|D POP, E A MOV, PBAREADRL OUT, D A MOV, 40 ORI, PBAREADRH OUT,
 5|D POP, B POP,
 6|B INX, B INX, B INX, B SRLR, C RARR, B SRLR, C RARR,
 7|C A MOV, PBXWIDE OUT, A INR, A M MOV, H INX, E M MOV, H INX,
 8|L A MOV, PBLINADRL OUT, H A MOV, PBLINADRH OUT,
 9|50 A MVI, C SUB, PBXMOD OUT,
10|E A MOV, A DCR, PBYHIGH OUT, ( DO IT TO IT )
11|EXX, NEXT
12|: SNAP 0 ROT ROT RELABS SWAP DROP SWAP snap ;
13|DECIMAL -->
14|
15|
 +--------Block     98--------
 0|( 8 X 10 CHARACTER SET - ROTATED )
 1|HEX
 2|DATA CHRTBL
 3|0000 , 0000 , 0000 , 0000 , 0000 , 0000 , ( SPACE )
 4|E01F , F03F , 3030 , 3030 , F03F , E01F , ( 0 )
 5|0000 , 2030 , D03F , F03F , 0030 , 0000 , ( 1 )
 6|603E , 703F , 3033 , 3033 , F033 , E031 , ( 2 )
 7|6018 , 7038 , 3033 , 3033 , F03F , E01E , ( 3 )
 8|F003 , F003 , 0003 , 0003 , F03F , F03F , ( 4 )
 9|F01B , F03B , 3033 , 3033 , 303F , 001E , ( 5 )
10|E01F , F03F , 3033 , 3033 , 303F , 001E , ( 6 )
11|3000 , 3038 , 303E , B00F , F003 , F000 , ( 7 )
12|E01E , F03F , 3033 , 3033 , F03F , E01E , ( 8 )
13|E001 , F003 , 3033 , 3033 , F03F , E01F , ( 9 )
14|-->
15|
 +--------Block     99--------
 0|( MORE CHARS ) C03F , E03F , 700C , 700C , E03F , C03F , ( A )
 1|F03F , F03F , 3033 , 3033 , F03F , E01E , ( B )
 2|E01F , F03F , 3030 , 3030 , 7038 , 6018 , ( C )
 3|F03F , F03F , 3030 , 3030 , F03F , E01F , ( D )
 4|F03F , F03F , 3033 , 3033 , 3033 , 3030 , ( E )
 5|F03F , F03F , 3003 , 3003 , 3003 , 3000 , ( F )
 6|E01F , F03F , 3030 , 3036 , 303E , 201E , ( G )
 7|F03F , F03F , 0003 , 0003 , F03F , F03F , ( H )
 8|0000 , 3030 , F03F , F03F , 3030 , 0000 , ( I )
 9|001C , 003C , 0030 , 0030 , F03F , F01F , ( J )
10|F03F , F03F , 3003 , C00F , F03C , 7038 , ( K )
11|F03F , F03F , 0030 , 0030 , 0030 , 0030 , ( L )
12|F03F , 6060 , C0C1 , 0003 , 6060 , F03F , ( M )
13|F03F , F03F , C0C1 , 3003 , F03F , F03F , ( N )
14|E01F , F03F , 3033 , 3033 , F03F , E01F , ( O )
15|F03F , F03F , 3003 , 3003 , F003 , E001 , ( P ) -->
```

```
 +--------Block    100--------
 0|( CHARS ) E01F , F03F , 3020 , 3028 , F01F , E02F , ( Q )
 1|F03F , F03F , 3003 , 3007 , F03F , E03D , ( R )
 2|E019 , F03B , 3033 , 3033 , 703F , 601E , ( S )
 3|3000 , 3000 , F03F , F03F , 3000 , 3000 , ( T )
 4|F01F , F03F , 0030 , 0030 , F03F , F01F , ( U )
 5|F003 , F00F , 003E , 003E , F00F , F003 , ( V )
 6|F03F , 000C , 8007 , 8007 , 000C , F03F , ( W )
 7|7038 , F03C , C00F , C00F , F03C , 7038 , ( X )
 8|7000 , F000 , C03F , C03F , F000 , 7000 , ( Y )
 9|303C , 303E , 3037 , 3033 , F031 , F030 , ( Z )
10|CC? IFTRUE ROMIT IFEND -->
11|
12|
13|
14|
15|
 +--------Block    101--------
 0|( NEW CHARACTER DRAW ROUTINE )
 1|( IN HL=Y DE=X BC=EXPAND/MAGIC A= CHAR TO DISPLAY )
 2|HEX
 3|SUBR drawchar B PUSH, H PUSH, D PUSH, 20 SUI, 0< >, IF,
 4|0F SUI, 0B CPI, CY~, IF, 7 SUI, THEN, THEN,
 5|A L MOV, 0 H MVI, H DAD, H DAD, L E MOV, H D MOV,
 6|H DAD, D DAD, CHRTBL D LXI, D DAD, H PUSH, Y POPX,
 7|D POP, H POP, H PUSH, D PUSH, relabs CALL,
 8|602 D LXI, write CALL, D POP, H POP, H A MOV, 7 ADI,
 9|A H MOV, B POP, RET,
10|
11|( TERSE INTERFACE - X Y COLOR/MAGIC CHAR cpost --- NEW X Y )
12|CODE cpost EXX, B POP, C A MOV, B POP, H POP, D POP,
13|X PUSHX, Y PUSHX, drawchar CALL, Y POPX, X POPX,
14|D PUSH, H PUSH, B PUSH, EXX, NEXT
15|DECIMAL -->
 +--------Block    102--------
 0|( NORMAL BCD ADDITION )
 1|CODE BCD+! EXX, H POP, D POP,
 2|M A MOV, E ADD, DAA, A M MOV,
 3|H INX, M A MOV, D ADC, DAA, A M MOV,
 4|H INX, M A MOV, 0 ACI, DAA, A M MOV,
 5|EXX, NEXT
 6|DECIMAL -->
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block    103--------
 0|( VGS                          CPOST , SPOST )
 1|: CPOST ( post an ascii-character on the screen ; see options )
 2|  ( in= x , y , opt+ex/mag , ascii-char )
 3|        ( WRTSYS 0< > for pattern board 0= for software write )
 4|  ( out- character on screen )
 5|  cpost DROP DROP DROP ;
 6|
 7|: SPOST ( post an ascii-string on the screen ; see options )
 8|  ( in= x , y , opt+ex/mag , addr , count )
 9|       ( i.e. 0 0 28 A" STRING" COUNT SPOST )
10|        ( WRTSYS 0< > for pattern board 0= for software write )
11|        ( cannot be used in immediat mode )
12|  ( out- character on screen )
13|  OVER + SWAP DO I B@ cpost LOOP DROP DROP DROP ;
14|-->
15|
 +--------Block    104--------
 0|( DISPLAY 6 DIGIT BCD NUMBER -- X Y OPT NUMADDR DISPBCD6 )
 1|HEX SUBR digit 0F ANI, 0=, IF, D ORA, 0< >, IF, 0F0 A MVI, THEN,
 2|ELSE, 0 D MVI, THEN, 30 ADI, EXX, drawchar CALL, EXX, RET,
 3|HEX
 4|
 5|F= DGTL
 6|CODE DISPBCD6 <ASSEMBLE H POP, M A MOV, H INX, M ORA,
 7|H INX, M ORA, A D MOV, 3 E MVI,
 8|EXX, B POP, H POP, D POP, X PUSHX, Y PUSHX, EXX,
 9|LABEL DGTL M A MOV, RRC, RRC, RRC, RRC, digit CALL,
10|M A MOV, digit CALL, H DCX, E DCR, DGTL JRNZ,
11|Y POPX, X POPX, NEXT ASSEMBLE>
12|DECIMAL -->
13|
14|
15|
 +--------Block    105--------
 0|( n-processor MUSPCU, this is starting load block ) HEX
 1|( old TERSE music still works and runs on this MUSCPU or )
 2|( 2MUSCPU, multiple processors reload IY for music vector, )
 3|( and the variable SOUNDBOX to the port 1 past the NOISE port, )
 4|( ie. sounds are in 10-17, set SOUNDBOX to 18H. )
 5|0 BV= MUSICFLAG ( turns off all processors for GAMEOVER )
 6|18 C= CHIP1 58 C= CHIP2 ( high port of soundbox for Any chips )
 7|( EMUSIC uses CHIP1, E2MUSIC uses CHIP2, ... add EMUSIC's )
 8|6E C= PANPORT1 6C C= PANPORT2 ( left-low port of stereo pair )
 9|0 BV= THUMPCOUNTER ( special space missions experiment episode )
10|( TASK EQUATES FOR VECTOR OFFSETS ) HEX
11|( : (( ) ( ; ) ( : )) ) ( ; ) ( for relative RAM order )
12|-->
13|
14|
15|
```

```
 +--------Block    106--------
 0|( MUSIC EQUATES FOR VECTOR OFFSETS ) HEX
 1|{{  0 SC= BEGMUSRAM ( first byte of music-vector )
 2|SC= MUSPC SC= MUSPCH NC= MUSPCL ( music program counter )
 3|NC= STARTPC SC= STARTPCH NC= STARTPCL ( startover address )
 4|NC= SOUNDBOX ( highest port of I/O chips sound ports )
 5|NC= PANPORT# ( bottom and left port of stereo output )
 6|NC= MOVALUE ( currant value )
 7|NC= VIBTRACKER ( vibrato convience tracker for games ) }}
 8|{{  NC= MULTIPLE NC= PRIORITY }} ( for repeated and important )
 9|{{  NC= RAMPFLAG ( /|/|/ vs. /\/\/ ) NC= RAMBLEFLAG ( on/off )
10|NC= HIGHLIM NC= LOWLIM NC= STEP ( pertaining to MO walk )
11|NC= RAMBLETIMER ( master oscillator timer )
12|NC= TIMEBASE ( reload rambletimer value ) }}
13|NC= LIMCOUNTER ( MO limit counter )
14|-->
15|
 +--------Block    107--------
 0|( MUSIC VARIABLES & IY EQUATES FOR OFFSETS ) HEX
 1|{{  NC= STOPTB ( stopvalue for timebase-mover )
 2|NC= TBSTEP NC= TBTB NC= TBTIMER ( tbmover's ss,tb,timer ) }}
 3|{{  NC= NOSTOP ( noisemover's sv,ss,timer,tb,tracker )
 4|NC= NOSTEP NC= NOTIMER NC= NOTIMEBASE NC= NOVALUE }}
 5|{{  NC= STOPSTEPS ( MO's stepmover etc. )
 6|NC= BIGOFASTEP NC= STEPTIMEBASE NC= STEPTIMER }}
 7|{{  NC= STOPLOWLIM ( lowlim's mover's ram, stopvalue )
 8|NC= LOWSTEP ( ss or stepsize )
 9|NC= LOW# ( # of limits to hit before moving )
10|NC= LOWCOUNTER ( counting low# down ) }}
11|{{  NC= STOPHIGHLIM ( highmover's ram )
12|NC= HIGHSTEP NC= HIGH# NC= HIGHCOUNTER }}
13|-->
14|
15|
 +--------Block    108--------
 0|( STEREO EFFECTS RAM AND VOLUME CRES.-DECRES. RAM ) HEX
 1|( total pan volumes either channel, FFH,  64 STEPS BETWEEN )
 2|( load lowest port in PANPORT#, this is left side, 1+ right )
 3|( watch step starting direction for left-right action )
 4|{{  NC= LEFTPAN ( tracker )
 5|NC= PANSTEP ( step size )
 6|  ( timebase for updating )
 7|NC= PANTIMEBASE NC= PANTIMER
 8|  ( count # of limits to achieve )
 9|NC= PANCOUNTER }}
10|{{  NC= VOLHIGHLIM NC= VOLLOWLIM NC= VOLSTEP
11|NC= VOLTIMEBASE NC= VOLTIMER
12|NC= MOTRACKER ( 16 volumes taken from 0 )
13|-->
14|
15|
```

```
 +--------Block    109--------
 0|( MUSIC VARIABLES FOR COMPUTER MUSIC GENERATOR AND SYNCER ) HEX
 1|{{ NC= SYNCMO NC= STARTMC ( special bvarbs for THUMPING ) }}
 2|{{ NC= NOTETIMER ( note timer )
 3|NC= COMPDURATION ( computer music note duration )
 4|NC= COMPSTEP ( step = { 1,0,-1 } )
 5|NC= COMPTIMER ( for COMPDURATION moving )
 6|NC= ATRACKER NC= BTRACKER NC= CTRACKER NC= MOTRACKER
 7|NC= NOTECOUNTER ( for key changes ) }}
 8|( trackers of indecies to NOTABLE and MOTABLE )
 9|NC= MST ( MUSIC-STATE-TRANSITION jump around variable )
10|NC= ENDMUSRAM ( last byte of ram )
11|80 C= COMPTB ( ** an equate to reload timer ** )
12|DUP BARRAY MUSIC-BARRAY-1
13|BARRAY MUSIC-BARRAY-2
14|{ : MB1 } 0 MUSIC-BARRAY-1 { ; }
15|{ : MB2 } 0 MUSIC-BARRAY-2 { ; } -->
 +--------Block    110--------
 0|( MUSIC PROCESSOR COMANDS ) HEX ( data,PORT )
 1|{ : MASTER } 10 B, B, { ; } { : ATONE } 11 B, B, { ; }
 2|{ : BTONE } 12 B, B, { ; } { : CTONE } 13 B, B, { ; }
 3|{ : VIBS } 14 B, B, { ; }        { : ABVOLS } 16 B, B, { ; }
 4|{ : MCVOLS } 15 B, B, { ; }     { : NOISE } 17 B, B, { ; }
 5|( range,disp.,port ) { : RDRNDNTE } 0 B, B, B, B, { ; }
 6|( range,port ) { : RRNDNTE } 0 B, B, 0 B, B, { ; }
 7|( port ) { : RNDNTE } 0 B, B, 0 B, FF B, { ; }
 8|{ : DURATION } 1 B, B, { ; }    { : PLAY } 3 B, { ; }
 9|( address to cont. at ) { : LDPCC } 2 B, , { ; }
10|( time,#A ) { : ANOTE } ATONE DURATION { ; }
11|( time,#B ) { : BNOTE } BTONE DURATION { ; }
12|( time,#C ) { : CNOTE } CTONE DURATION { ; }
13|( #A,#B,#C ) { : TONES } CTONE BTONE ATONE { ; }
14|( time,#A,#B,#C ) { : NOTES } TONES DURATION { ; }
15|-->
 +--------Block    111--------
 0|( MUSIC PROCESSOR COMANDS cont. ) HEX
 1|{ : QUIET } 4 B, { ; } ( does an emusic )
 2|( time,step,low,high ) { : RAMBLE } 5 B, B, B, B, B, { ; }
 3|( time,step,low,high ) { : RAMP } 6 B, B, B, B, B, { ; }
 4|( computer music generator, stepsize {1,0,-1}, duration ---- )
 5|      { : GENMUSIC } 7 B, B, B, { ; }
 6|( : RERAMBLE } 8 B, { ; } ( restart ramble )
 7|( : STOPRAMBLE } 9 B, { ; }
 8|( : COUNTLIMIT } 0A B, B, { ; }
 9|( Format for following : timebase,stepsize,stepvalue ---- )
10|{ : MOVESTEP } 0B B, B, B, B, { ; }
11|( waitbetween,--,rv ) { : MOVELOWLIM } 0C B, B, B, B, { ; }
12|( holdbetween,--,rv ) { : MOVEHIGHLIM } 0D B, B, B, B, { ; }
13|( timebase ) { : MOVETB } 0E B, B, B, B, { ; }
14|( NOISE,tb,ss,rv ) { : MOVENOISE } 0F B, B, B, B, B, { ; }
15|-->
```

```
 +--------Block    112--------
 0|( MUSIC PROCESSOR COMANDS cont., STEREO STUFF and ABCRND ) HEX
 1|( try - timebase, stepsize, leftvolume, ---- )
 2|HEX     { : MOVESOUND } 18 B, B, B, B, { ; }
 3|( also notice that stepvol is pos. for left-->right )
 4|( for limited movement, use the following : # of limits ---- )
 5|        { : COUNTPANS } 19 B, B, { ; }
 6|( volume moving is ind. of stereo )
 7|( ABvols,MCvols,tb,ss,ll,hl ---- )
 8|        { : MOVEVOLS } 1A B, B, B, B, B, MCVOLS ABVOLS { ; }
 9|( special opcode to reload MC for fade out, STARTING MC ---- )
10|        { : HITMO } 1B B, B, { ; }
11|-->
12|
13|
14|
15|
 +--------Block    113--------
 0|( NOTE CONSTANTS ) HEX
 1|FD C= #G0  EE C= #GS0  E1 C= #A0  D4 C= #AS0  C8 C= #B0
 2|BD C= #C1  B2 C= #CS1  A8 C= #D1  9F C= #DS1  96 C= #E1
 3|8D C= #F1  85 C= #FS1  7E C= #G1  77 C= #GS1  70 C= #A1
 4|6A C= #AS1  64 C= #B1  5E C= #C2  59 C= #CS2  54 C= #D2
 5|4F C= #DS2  4A C= #E2  46 C= #F2  42 C= #FS2  3E C= #G2
 6|3B C= #GS2  37 C= #A2  34 C= #AS2  31 C= #B2  2E C= #C3
 7|2C C= #CS3  29 C= #D3  27 C= #DS3  25 C= #E3  22 C= #F3
 8|20 C= #FS3  1F C= #G3  1D C= #GS3  1B C= #A3  1A C= #AS3
 9|18 C= #B3  17 C= #C4  15 C= #CS4  14 C= #D4  13 C= #DS4
10|12 C= #E4  11 C= #F4  10 C= #FS4  0F C= #G4  0E C= #GS4
11|0D C= #A4  0B C= #C5  0A C= #CS5  09 C= #DS5 08 C= #F5
12|07 C= #G5  06 C= #A5  05 C= #C6  04 C= #DS6  03 C= #G6
13|02 C= #C7  01 C= #G7  00 C= #G8
14|BTABLE MOTABLE 23 B, 22 B, 20 B, 1E B, 1C B, 1A B, 18 B, 17 B,
15| 16 B, 15 B, 14 B, 13 B, 12 B, 11 B, 0D B, 0B B, -->
 +--------Block    114--------
 0|( SIN BTABLE FOR LEFT-RIGHT PAN VOLTAGES ) DECIMAL
 1|( : ^ )  ( STORE BYTES ON STACK IN RAM AS PATTERN )
 2|  { -1 >R BEGIN DUP -1 = IF DROP 1 ELSE >R 0 THEN END
 3|       BEGIN R> DUP -1 = IF DROP 1 ELSE } B, { 0 THEN END ; }
 4| -1 CONSTANT ~ ( MARK START OF PATTERN )
 5|BTABLE sin-table
 6|( 00-10 ) ~ 255 255 255 255 254 253 252 251 250 248 247 ^
 7|( 11-21 ) ~ 245 243 241 239 237 234 231 229 226 223 220 ^
 8|( 22-32 ) ~ 216 213 209 206 202 198 194 190 186 181 177 ^
 9|( 33-43 ) ~ 172 167 162 157 152 147 142 137 132 127 121 ^
10|( 44-54 ) ~ 115 109 104  98  92  86  80  74  68  62  56 ^
11|( 55-63 ) ~  50  44  38  31  25  19  13   6   0                ^
12|SUBR sin ( pass 1, 0<= < <= 63, in E ) 0 D MVI,
13|  0 sin-table H LXI, D DAD, M A MOV, RET,
14|
15|-->
```

```
 +--------Block    115--------
 0|( HELPING SUBR's for MUSCPU *** NOTICE *** ) HEX
 1|( The MUSPC rides in HL for the coarse of the MUSCPU )
 2|( EACH MUSCPU LOADS ITS STARTING RAM IN IY )
 3|SUBR PCJUMP ( reload MUSPC )
 4| M E MOV, H INX, M D MOV, XCHG, ( leave in HL )
 5| L MUSPC Y STX, H MUSPC 1+ Y STX, ( store )   RET,
 6|SUBR portout ( pass value in A, port in C )
 7| A E MOV, 17 A MVI, C CMP, ( all ports are 10-17 )
 8| 0>=, IF, ( check for bad values )
 9| 8 SUI, ( bottom ) C CMP, 0<, IF, ( oked )
10| 18 A MVI, C SUB, SOUNDBOX Y SUBX, NEG, A C MOV, E OUTP,
11| THEN, THEN, A XRA, RET,
12|SUBR babs ( byte absolute value )
13| 7 A BIT, 0<>, IF, NEG, 7 A RES, THEN, RET,
14|CODE BABS H POP, L A MOV, babs CALL, A L MOV, H PUSH, NEXT
15|-->
 +--------Block    116--------
 0|( HELPING SUBR's cont. ) HEX
 1|SUBR LIMITCOUNT ( detect Music-State-transition if completed )
 2| LIMCOUNTER Y A LDX, A ORA, 0<>, IF,
 3| A DCR, A LIMCOUNTER Y STX, 0=, IF, ( done )
 4| A RAMBLEFLAG Y STX, ( stop ramble ) 1 MST Y MVIX,
 5| THEN, THEN, RET,
 6|SUBR PANOUTS ( pass location in E )
 7| PANPORT# Y C LDX, E B MOV, ( save ) sin CALL,
 8| A OUTP, 3F A MVI, ( 64 steps ) B SUB, A E MOV,
 9| sin CALL, ( enter table from bottom ) C INR, A OUTP,
10| RET,
11|-->
12|
13|
14|
15|
 +--------Block    117--------
 0|( MUSIC PROCESSOR- emusic )
 1|DATA ENDMUS PLAY
 2|
 3|SUBR emusic ( ** each EMUSIC passes vector addr in DE' )
 4| MUSPC H LXI, D DAD, ENDMUS B LXI, C M MOV, H INX, B M MOV,
 5| A XRA, 6 H LXI, D DAD, ( skip MUSPC,STARTPC,SOUNDBOX,PANPORT# )
 6| ENDMUSRAM BEGMUSRAM - 6 - B MVI,
 7| BEGIN, A M MOV, H INX, B DCR, 0=, END,
 8| SOUNDBOX H LXI, D DAD, M C MOV, 8 B MVI,
 9| BEGIN, C DCR, A OUTP, B DCR, 0=, END,
10| EXX, RET,
11|-->
12|
13|
14|
15|
```

```
 +--------Block    118--------
 0|( OPCODE SUBR's, 0-4 )
 1|SUBR RANDOMNOTES H PUSH, ( save PC from RND )
 2| 0 D MVI, M E MOV, D PUSH, H INX, M E MOV, D PUSH, H INX,
 3| M E MOV, random CALL, D POP, ( disp. ) D DAD, ( returns in HL )
 4| B POP, L A MOV, portout CALL, H POP, 3 D LXI, D DAD, ( MUSPC )
 5| A XRA, RET,
 6|SUBR LOADTIMER M A MOV, A NOTETIMER Y STX, H INX, A XRA,
 7| A COMPDURATION Y STX, A INR, RET,
 8|SUBR CONTJUMP M E MOV, H INX, M D MOV, XCHG, A XRA, RET,
 9|SUBR QUITJUMP ( H DCX, 3 in A ) RET,
10|SUBR QUITYET? ( QUIET ) MULTIPLE Y DCRX,
11| 0< >, IF, STARTPC Y L LDX, STARTPC 1+ Y H LDX, A XRA,
12| ELSE, Y PUSHX, EXX, D POP, emusic CALL, 1 ORI, THEN, RET,
13|-->
14|
15|
 +--------Block    119--------
 0|( OPCODE SUBR's,  5-6, HL= MUSPC )
 1|FORWARD RAMBLESTORES
 2|SUBR RAMBLIN' A XRA, ( turn off ramp flag )
 3| LABEL RAMBLESTORES A RAMPFLAG Y STX,
 4| M A MOV, H INX, A HIGHLIM Y STX,
 5| M A MOV, H INX, A LOWLIM Y STX,
 6| M A MOV, H INX, A STEP Y STX,
 7| M A MOV, H INX, A RAMBLETIMER Y STX,
 8| A TIMEBASE Y STX, 1 A MVI, A RAMBLEFLAG Y STX, A DCR, RET,
 9|SUBR RAMPIN' 1 A MVI, RAMBLESTORES JMP,
10|-->
11|
12|
13|
14|
15|
 +--------Block    120--------
 0|( OPCODE SUBR's, 8-0B, 10H )
 1|SUBR MASTART ( MASTER, 10H ) SOUNDBOX Y A LDX, 8 SUI, A C MOV,
 2| M A MOV, H INX, A MOVALUE Y STX,
 3| A OUTP, A XRA, RET,
 4|SUBR RAMBLE-ON 1 A MVI, A RAMBLEFLAG Y STX, A XRA, RET,
 5|SUBR RAMBLE-OFF A XRA, A RAMBLEFLAG Y STX, RET,
 6|SUBR LIMITRAMBLE ( set up LIMCOUNTER )
 7| 1 A MVI, A RAMBLEFLAG Y STX, M A MOV, H INX,
 8| A LIMCOUNTER Y STX, A XRA, RET,
 9|SUBR STEPMOVIN' M A MOV, H INX, A STOPSTEPS Y STX,
10| M A MOV, H INX, A BIGOFASTEP Y STX, M A MOV, H INX,
11| A STEPTIMEBASE Y STX, A STEPTIMER Y STX, A XRA, RET,
12|-->
13|
14|
15|
```

```
 +--------Block    121--------
 0|( OPCODES 0C-0F )
 1|SUBR LOWMOVIN' M A MOV, H INX, A STOPLOWLIM Y STX,
 2| M A MOV, H INX, A LOWSTEP Y STX, M A MOV, H INX, A LOW# Y STX,
 3| A LOWCOUNTER Y STX, A XRA, RET,
 4|SUBR HIGHMOVIN' M A MOV, H INX, A STOPHIGHLIM Y STX,
 5| M A MOV, H INX, A HIGHSTEP Y STX, M A MOV, H INX,
 6| A HIGH# Y STX, A HIGHCOUNTER Y STX, A XRA, RET,
 7|SUBR TBMOVIN' M A MOV, H INX, A STOPTB Y STX, M A MOV, H INX,
 8| A TBSTEP Y STX, M A MOV, H INX, A TBTB Y STX, A TBTIMER Y STX,
 9| A XRA, RET,
10|SUBR NOMOVIN' M A MOV, H INX, A NOSTOP Y STX, M A MOV, H INX,
11| A NOSTEP Y STX, M A MOV, H INX, A NOTIMER Y STX,
12| A NOTIMEBASE Y STX, SOUNDBOX Y C LDX, C DCR,
13| M A MOV, H INX, A NOVALUE Y STX, A OUTP, A XRA, RET,
14|-->
15|
 +--------Block    122--------
 0|( OPCODES 11-16 I/O PORT OUTPUTS and PAN COUNTING, 1AH ) HEX
 1|SUBR OPPORT ( 11H-14H, 16-17H )
 2| RRC, A C MOV, M A MOV, H INX, portout JMP,
 3|SUBR MCMOVIN' ( 15H )
 4| RRC, A C MOV, M A MOV, H INX, A MCTRACKER Y STX, portout JMP,
 5|SUBR NOISEPORT ( 17H )
 6| RRC, A C MOV, M A MOV, H INX, A NOVALUE Y STX, portout JMP,
 7|SUBR SOUNDMOVIN' ( 18H )  M E MOV, H INX, E LEFTPAN Y STX,
 8| H PUSH, PANOUTS CALL, H POP, ( init )
 9| M A MOV, H INX, A PANSTEP Y STX,
10| M A MOV, H INX, A PANTIMEBASE Y STX,
11| A PANTIMER Y STX, FF PANCOUNTER Y MVIX, A XRA, RET,
12|SUBR PANLIMITCOUNTIN' ( 19 )
13| M A MOV, H INX, A PANCOUNTER Y STX,
14| PANTIMEBASE Y A LDX, A PANTIMER Y STX, A XRA, RET,
15|-->
 +--------Block    123--------
 0|( STEREO OPCODE 1A, THUMPER 1B, MUSIC GENERATOR 07H ) HEX
 1|SUBR VOLMOVIN' ( 1AH )
 2| M A MOV, H INX, A VOLHIGHLIM Y STX,
 3| M A MOV, H INX, A VOLOWLIM Y STX,
 4| M A MOV, H INX, A VOLSTEP Y STX,
 5| M A MOV, H INX, A VOLTIMEBASE Y STX,
 6| 1 VOLTIMER Y MVIX, A XRA, RET,
 7|SUBR MOHITTIN' ( 1B )
 8| 1 SYNCMO Y MVIX, ( turn on THUMPER-sync ) 3 A MVI, THUMPCOUNTER
 9| STA, M A MOV, H INX, A STARTMO Y STX, A XRA, RET,
10|SUBR MUSICIN' ( 07 )
11| M A MOV, H INX, A COMPDURATION Y STX, 1 NOTETIMER Y MVIX,
12| M A MOV, H INX, A COMPSTEP Y STX, COMPTB COMPTIMER Y MVIX,
13| 4 ATRACKER Y MVIX, 5 BTRACKER Y MVIX, 6 CTRACKER Y MVIX,
14| 8 MOTRACKER Y MVIX, A XRA, RET, ( zero )
15|-->
```

```
 +--------Block    124--------
 0|( OPCODE ADDRESS TABLE and FORWARDS ) HEX
 1|{ : FW ) FORWARD ( ; )          1B C= #-OF-OPCODES
 2|FW process FW endprocess FW MUSEND
 3|TABLE OPADDRESSES
 4|       RANDOMNOTES , LOADTIMER , CONTJUMP , QUITJUMP ,
 5|       QUITYET? , RAMBLIN' , RAMPIN' , MUSICIN' ,
 6|       RAMBLE-ON , RAMBLE-OFF , LIMITRAMBLE , STEPMOVIN' ,
 7|       LOWMOVIN' , HIGHMOVIN' , TBMOVIN' , NOMOVIN' ,
 8|       MASTART , 3 0 << OPPORT , >>
 9|       OPPORT , MCMOVIN' , OPPORT , NOISEPORT ,
10|       SOUNDMOVIN' , PANLIMITCOUNTIN' , VOLMOVIN' , MOHITTIN' ,
11|-->
12|
13|
14|
15|
 +--------Block    125--------
 0|( COMPMUSIC's +-disp., 15MOD, NOTABLE and THUMPLOCATION ) HEX
 1|BTABLE THUMPLOCATION ( where to locate sound in stereo image )
 2| 3F B, 2A B, 15 B, 00 B,
 3|BTABLE NOTABLE ( 3 octave range )
 4| #G0 B, #A0 B, #B0 B, #C1 B, #D1 B, #E1 B, #FS1 B,
 5| #G1 B, #A1 B, #B1 B, #C2 B, #D2 B, #E2 B, #FS2 B,
 6| #G2 B, #A2 B, #B2 B, #C3 B, #D3 B, #E3 B, #FS3 B,
 7|SUBR +-disp. ( change A to a + or - 3-bit # )
 8| 0 A BIT, 0<>, IF, ( neg ) F8 ORI, ELSE, 7 ANI, THEN, RET,
 9|SUBR 15MOD ( base 21decimal ) 15 CPI, 0>=, IF, 7 SUI,
10| ( only adjust note down 1 octave ) THEN, A ORA, 0<, IF,
11| ( adjust up 1 octave ) 7 ADI, THEN, RET,
12|SUBR UP-AN-OUT ( pass index in A ) EXX,
13| 0 NOTABLE D LXI, 0 H MVI, A L MOV,
14| D DAD, C INR, M A MOV, A OUTP, ( outp note ) EXX, RET,
15|CC? IFTRUE TIMOR IFEND -->
 +--------Block    126--------
 0|( STEREO STUFF, LIMITCOUNTING )
 1|( PANLIMITS- achieving limits of volumes per channel )
 2|SUBR PANLIMIT ( A&E= LEFTVOL, D=tb, H=counter, L=stepvol )
 3|  H INR, 0<>, IF, ( wasn't FF ) H DCR, H DCR, ( counted )
 4|   0=, IF, ( counted down )
 5|    1 MST Y MVIY, ( detect state transition ) 0 D MVI,
 6|   THEN, ELSE, H DCR, ( back to FF )
 7|  THEN, A XRA, L SUB, ( change step sign ) A L MOV,
 8| RET, ( zero or nonzero )
 9|
10|
11|-->
12|
13|
14|
15|
```

```
 +--------Block    127--------
 0|( ** MUSCPU **                          **STEREO** ) HEX
 1|SUBR MUSCPU ( runs in interrupt )
 2| A XRA, MST Y CMPX, 0< >, IF, RET, THEN, ( no background )
 3| NOTETIMER Y CMPX, 0< >, IF, NOTETIMER Y DCRX, 0=, IF,
 4| COMPDURATION Y CMPX, 0< >, IF, 2 A MVI, NOTECOUNTER Y DCRX,
 5| ELSE, A INR, THEN,
 6| A MST Y STX, THEN, THEN, ( * PANNER * )
 7|-->
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    128--------
 0|( ** MUSCPU **                          **STEREO** ) HEX
 1| LEFTPAN Y E LDX, ( setups for PANNER )
 2| PANTIMER Y CMPX, ( all vectoring routines follow )
 3| 0< >, IF, ( timer on ) PANTIMER Y DCRX, 0=, IF, ( expired )
 4| PANTIMEBASE Y D LDX, PANCOUNTER Y H LDX, PANSTEP Y L LDX,
 5|( A&E=leftvol, D=timer-reload, value H=counter, L=stepvol )
 6| E A MOV, L ADD, ( newpan ) A E MOV, A ORA,
 7| 0<, IF, ( low limit ) 0 E MVI, PANLIMIT CALL,
 8|( ~ screen 220, PANLIMIT zaps D, so a zero timer comes in )
 9| ELSE, 3F CPI, >=, IF, ( 64 ) 3F E MVI, PANLIMIT CALL,
10| THEN, THEN, L PANSTEP Y STX, D PANTIMER Y STX,
11| E LEFTPAN Y STX, ( PANOUTS creams E ) H PANCOUNTER Y STX,
12| THEN, THEN, PANOUTS CALL, ( pass location in E )
13|-->
14|
15|
 +--------Block    129--------
 0|( MUSCPU                              NOTETIMER,MOSYNC )
 1|A XRA, RAMBLEFLAG Y CMPX,  ( * RAMBLER * )
 2| 0< >, IF, RAMBLETIMER Y DCRX, 0=, IF,
 3| STEP Y L LDX, ( setup ) SOUNDBOX Y A LDX, 8 SUI, A C MOV,
 4| MOVALUE Y A LDX, L ADD, ( step ) A MOVALUE Y STX, ( upit )
 5| A OUTP, A D MOV, A XRA, SYNCMO Y CMPX, 0< >, IF, EXX,
 6|( * special opcode for thumping sound * ) STARTMC Y A LDX,
 7| A MCTRACKER Y STX, TIMEBASE Y A LDX, ( keep durations close )
 8| 0F0 ANI, RRC, RRC, RRC, RRC, 8 ADI, ^ VOLTIMEBASE Y STX,
 9| THUMPCOUNTER H LXI, M DCR, M C MOV, 0=, IF, A M MVI, THEN,
10| 0 B MVI, 0 THUMPLOCATION H LXI, C DAD, ( stomp loc. )
11| M E MOV, E LEFTPAN Y STX, ( lost PANNER pointer ) PANOUTS CALL,
12| EXX, THEN,
13|-->
14|
15|
```

```
 +--------Block    130--------
 0|( MUSCPU cont.              MORAMBLE, LOWMOVER, HIGHMOVER )
 1|( HIGHMOVER )
 2| LOWLIM Y A LDX, A DCR, D CMP, <, IF,
 3| HIGH# Y A LDX, A ORA, 0< >, IF,
 4| HIGHCOUNTER Y DCRX, 0=, IF, A C MOV,
 5| HIGHLIM Y A LDX, HIGHSTEP Y ADDX,
 6| STOPHIGHLIM Y CMPX, =, IF, 0 HIGH# Y MVIX, THEN,
 7| A HIGHLIM Y STX, C HIGHCOUNTER Y STX, THEN, THEN,
 8| LIMITCOUNT CALL, RAMPFLAG Y A LDX, 3 CPI, 0=, IF,
 9| ( just came from LOWMOVER  ) 1 A MVI, A RAMPFLAG Y STX, ELSE,
10| A ORA, 0< >, IF, ( RAMP )
11| RAMPFLAG Y INRX, ( tell LOWMOVER donothing )
12| HIGHLIM Y A LDX, L SUB, A MOVALUE Y STX, ELSE, ( change sign )
13| L SUB, A STEP Y STX, THEN, THEN, THEN,
14|-->
15|
 +--------Block    131--------
 0|( MUSCPU cont.,                  MORAMBLE cont., STEPMOVER )
 1| HIGHLIM Y A LDX, D CMP, >=, IF, ( at limit )
 2|( LOWMOVER )
 3| LOW# Y A LDX, A ORA, 0< >, IF, ( not dead already )
 4| LOWCOUNTER Y DCRX, 0=, IF, A C MOV,
 5| LOWLIM Y A LDX, LOWSTEP Y ADDX,
 6| STOPLOWLIM Y CMPX, =, IF, 0 LOW# Y MVIX, THEN,
 7| A LOWLIM Y STX, C LOWCOUNTER Y STX, THEN, THEN,
 8| LIMITCOUNT CALL, RAMPFLAG Y A LDX,
 9| 2 CPI, 0=, IF, RAMPFLAG Y DCRX, ELSE,
10| A ORA, 0< >, IF, ( ramp ) 3 A MVI, A RAMPFLAG Y STX,
11| ( tell HIGHMOVER to donutting )
12| LOWLIM Y A LDX, L SUB, A MOVALUE Y STX, ELSE, L SUB,
13| A STEP Y STX, THEN, THEN, THEN,
14| TIMEBASE Y A LDX, A RAMBLETIMER Y STX, THEN, THEN,
15|-->
 +--------Block    132--------
 0|( MUSCPU                        VOLUME MOVING )
 1|VOLTIMER Y A LDX, A ORA, 0< >, IF, A DCR, 0=, IF,
 2|( D=MCtracker, E=MOset, H=limitcheck, L=stepsize )
 3| MCTRACKER Y D LDX, D A MOV, 0F0 ANI, A E MOV, D A MOV,
 4| E SUB, VOLSTEP Y L LDX, L ADD, ( update tracker ) A D MOV,
 5| VOLHIGHLIM Y H LDX, H CMP, 0>=, IF, H D MOV, ( check top )
 6| A XRA, L SUB, A L MOV, ( change dir. )
 7| ELSE, A DCR, VOLOWLIM Y H LDX, H CMP, 0<, IF,
 8| H D MOV, ( low end ) A XRA, L SUB, A L MOV, ( change )
 9| THEN, THEN,
10| L VOLSTEP Y STX, D ^ MOV, ( Cvols ) RRC, RRC, RRC, RRC,
11| D ADD, ( ^Cvols to SCvols ) SOUNDBOX Y C LDX, C DCR, C DCR,
12| A OUTP, C DCR, D ^ MOV, E ORA, ( mesetting )
13| A MCTRACKER Y STX, ^ OUTP,
14| VOLTIMEBASE Y A LDX, THEN, A VOLTIMER Y STX, THEN,
15|-->
```

```
 +--------Block    133--------
 0|( MUSCPU                          STEPMOVER, COMPDURMOVER )
 1| STEPTIMER Y A LDX, A ORA, 0<>, IF, ( work on it )
 2| STEPTIMER Y DCRX, 0=, IF,
 3| STEP Y A LDX, A E MOV, ( save for sign ) babs CALL, ( abs )
 4| BIGOFASTEP Y ADDX, ( positive value )
 5| STOPSTEPS Y CMPX, ( positive ) 0<>, IF, ( not done )
 6| STEPTIMEBASE Y A LDX, A STEPTIMER Y STX, THEN, 7 E BIT,
 7| 0<>, IF, NEG, THEN, A STEP Y STX, ( store ) THEN, THEN,
 8|( COMP-DURATION MOVER ) COMPDURATION Y A LDX, A ORA, 0<>, IF,
 9| COMPTIMER Y DCRX, 0=, IF, COMPTB COMPTIMER Y MVIX, ( equate )
10| COMPSTEP Y ADDX, A DCR, 0<, IF, ( 1=>CARRY, 81=>NC )
11| 0 COMPDURATION Y MVIX, ( both stop )
12| 1 MST Y MVIX, ( tell background we're there ) ELSE,
13| A INR, A COMPDURATION Y STX, THEN, THEN, THEN,
14|-->
15|
 +--------Block    134--------
 0|( MUSPCU cont.,                    TBMOVER, NOMOVER )
 1|( TBMOVER timebase ) TBSTEP Y A LDX, A ORA, 0<>, IF,
 2| ( not done ) TBTIMER Y DCRX, 0=, IF,
 3| TIMEBASE Y ADDX, A TIMEBASE Y STX, STOPTB Y SUBX,
 4| 0=, IF, A TBSTEP Y STX, THEN,
 5| TBTB Y A LDX, A TBTIMER Y STX, THEN, THEN,
 6|( NOMOVER noise mover ) NOTIMER Y A LDX, A ORA, 0<>, IF,
 7| NOTIMER Y DCRX, 0=, IF, ( update )
 8| SOUNDBOX Y C LDX, C DCR, ( output port )
 9| NOVALUE Y A LDX, NOSTEP Y ADDX, A NOVALUE Y STX, A OUTP,
10| NOSTOP Y CMPX, ( timer to stop? )
11| 0<>, IF, ( not done, reload )
12| NOTIMEBASE Y A LDX, A NOTIMER Y STX, THEN, THEN, THEN,
13| RET,
14|-->
15|
 +--------Block    135--------
 0|( ** MUSIC INTERRUPTER **          COMPUTER MUSIC ) HEX
 1|SUBR MUSINTERP ( music interrupter, pre-load IY )
 2|<ASSEMBLE MST Y A LDX, A ORA, ( check overrun flag )
 3| 0=, IF, RET, THEN, A DCR, 0=, IF,  ( AHA!! state transition )
 4| process JMP, THEN, ( AHHA HA!! MST=2 => COMPUTER MUSIC )
 5| COMPDURATION Y A LDX, A NOTETIMER Y STX, ( compmusic duration )
 6| random CALL, ( use 3 reg. melody trackers, 1 for MO tracker )
 7| NOTECOUNTER Y A LDX, A ORA, 0<, IF, ( new key ) EXX,
 8| LDAR, ( gen new wait # ) 1 E MVI, 7 D MVI, ( C = RAN )
 9| BEGIN, RAR, CY, IF, E SLAR, THEN, D DCR, 0=, END, E SLAR,
10| ( 2^2 to 2^8 ) E NOTECOUNTER Y STX, EXX, 1 L MVI, ( inc MO )
11| ELSE, 0 L MVI, ( leave MO ) THEN,
12| L A MOV, ( do anyhow for setup )
13| MOTRACKER Y ADDX, 07 ANI, ( 0-15 ) A MOTRACKER Y STX,
14|-->
15|
```

```
 +--------Block    136--------
 0|( MUSCPU cont.,                    RANDOM NOTES )
 1| EXX, 0 MOTABLE H LXI, A E MOV, 0 D MVI, D DAD, M A MOV,
 2| A MOVALUE Y STX, A B MOV, ( save from soundbox )
 3| SOUNDBOX Y A LDX, 8 SUI, A C MOV, 20 CPI, <, IF, ( low chip )
 4| B OUTP, C D MOV, 40 ADI, A C MOV, B OUTP, ( MO ) D C MOV, THEN,
 5| EXX, H A MOV, ( use this # for disp. to index ) +-disp. CALL,
 6| ( +7 to -7 ) ATRACKER Y ADDX, 15MOD CALL, ( index 0-15H )
 7| A ATRACKER Y STX, UP-AN-OUT CALL,
 8| E A MOV, ( next note ) +-disp. CALL, ( +7 to -6 disp. )
 9| BTRACKER Y ADDX, 15MOD CALL, ( index 0-15H )
10| A BTRACKER Y STX, UP-AN-OUT CALL, ( get from NOTE TABLE )
11| D A MOV, ( next disp. ) +-disp. CALL, ( +7 to -6 )
12| CTRACKER Y ADDX, 15MOD CALL, ( index 0-15H )
13| A CTRACKER Y STX, UP-AN-OUT CALL,
14| ( done with notes ) MUSEND JMP,
15|-->
 +--------Block    137--------
 0|( MUSCPU cont.,                    PROCESS the score, )
 1|LABEL process  MUSPC Y L LDX, MUSPC 1+ Y H LDX,
 2| BEGIN, ( MUSPC in HL until done )
 3| M A MOV, H INX, #-OF-OPCODES 1+ CPI, ( bad opcode check )
 4| <, IF, EXX, ( swap to keep MUSPC around )
 5| endprocess H LXI, H PUSH, ( ret to end of process )
 6| 0 OPADDRESSES H LXI,  ( get address of opcode verb )
 7| RLC, ( words ) A E MOV, 0 D MVI, D DAD,
 8| M E MOV, H INX, M D MOV, D PUSH, ( RET to routine )
 9| EXX, ( put MUSPC in HL ) RET,
10| ELSE, 1 ORI, ( quit ) THEN,
11|LABEL endprocess A ORA, 0<>, END, ( opverbs return non-0 or 0 )
12| L MUSPC Y STX, H MUSPC 1+ Y STX,
13|LABEL MUSEND 0 MST Y MVIX, ( let interrupts run ) RET,
14|ASSEMBLE>
15| -->
 +--------Block    138--------
 0|( MUSIC PROCESSOR-                 MUSCPUS PUT TOGETHER ) HEX
 1|SUBR MUSCPUS
 2| MUSICFLAG LDA, A ORA, 0<>, IF, Y PUSHX, ( F4 A MVI, 0 OUT, )
 3| 0 MUSIC-BARRAY-1 Y LXIX, MUSCPU CALL, ( A2 A MVI, 0 OUT, )
 4| 0 MUSIC-BARRAY-2 Y LXIX, MUSCPU CALL, ( 53 A MVI, 0 OUT, )
 5|Y POPX, THEN, RET,
 6|SUBR busaround ( back-music-ground )
 7| MUSICFLAG LDA, A ORA, 0<>, IF, Y PUSHX, ( F4 A MVI, 4 OUT, )
 8| 0 MUSIC-BARRAY-1 Y LXIX, MUSINTERP CALL, ( A2 A MVI, 4 OUT, )
 9| 0 MUSIC-BARRAY-2 Y LXIX, MUSINTERP CALL, ( 53 A MVI, 4 OUT, )
10|Y POPX, THEN, RET,
11|CODE BMS ( code level back-music-ground ) B PUSH,
12| busaround CALL, B POP, NEXT
13|
14|-->
15|
```

```
 +--------Block    139--------
 0|( MUSIC PROCESSOR- ALL xmusics NEED AN IY LOAD ) HEX
 1|
 2|SUBR loadpc L MUSPC Y STX, L STARTPC Y STX,
 3| H MUSPC 1+ Y STX, H STARTPC 1+ Y STX, RET,
 4|-->
 5|
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    140--------
 0|( MUSCPU SUBROUTINE CALLS )
 1|( * SET SCORE IN HL, RAM IN IY, MULTIPLE IN E if req. * )
 2|SUBR bmusic
 3| PRIORITY Y A LDX, A ORA, 0=, IF, 1 MST Y MVIX,
 4| A NOTETIMER Y STX, A INR, A MULTIPLE Y STX, loadpc JMP,
 5| ( leave MST=1 for BMS ) THEN, RET,
 6|SUBR pmusic 1 MST Y MVIX, Y PUSHX, EXX, D POP, emusic CALL,
 7| 1 A MVI, A MST Y STX, A MULTIPLE Y STX, A PRIORITY Y STX,
 8|  loadpc JMP,
 9|SUBR mmusic PRIORITY Y A LDX, A ORA, 0=, IF, 1 MST Y MVIX,
10| A NOTETIMER Y STX, E MULTIPLE Y STX, loadpc JMP, THEN, RET,
11|SUBR mpmusic 1 MST Y MVIX, Y PUSHX, EXX, D POP, emusic CALL,
12| 1 A MVI, A MST Y STX, A PRIORITY Y STX, E MULTIPLE Y STX,
13|  loadpc JMP,
14|-->
15|
 +--------Block    141--------
 0|( MUSIC PROCESSOR- EMUSIC, BMUSIC, ... )
 1|CODE EMUSIC EXX, 0 MUSIC-BARRAY-1 D LXI, ( pass to emusic )
 2| MST H LXI, D DAD, 1 M MVI, ( make non-zero )
 3| SOUNDBOX H LXI, D DAD, CHIP1 M MVI,
 4| PANPORT* H LXI, D DAD, PANPORT1 M MVI,
 5| emusic CALL, ( musicoverun flag is zeroed last ) NEXT
 6|( *** ALWAYS CALL EMUSIC AS AN INIT IN PROGRAM *** )
 7|CODE BMUSIC H POP, Y PUSHX,
 8| 0 MUSIC-BARRAY-1 Y LXIX, bmusic CALL, Y POPX, NEXT
 9|CODE PMUSIC H POP, Y PUSHX,
10| 0 MUSIC-BARRAY-1 Y LXIX, pmusic CALL, Y POPX, NEXT
11|CODE MMUSIC H POP, D POP, Y PUSHX,
12| 0 MUSIC-BARRAY-1 Y LXIX, mmusic CALL, Y POPX, NEXT
13|CODE MPMUSIC H POP, D POP, Y PUSHX,
14| 0 MUSIC-BARRAY-1 Y LXIX, mpmusic CALL, Y POPX, NEXT
15|-->
```

```
 +--------Block    142--------
 0|( MUSIC PROCESSOR- E2MUSIC, B2MUSIC, ... )
 1|CODE E2MUSIC EXX, 0 MUSIC-BARRAY-2 D LXI, ( pass to emusic )
 2| MST H LXI, D DAD, 1 M MVI, ( make non-zero )
 3| SOUNDBOX H LXI, D DAD, CHIP2 M MVI,
 4| PANPORT# H LXI, D DAD, PANPORT2 M MVI,
 5| emusic CALL, ( musicoverun flag is zeroed last ) NEXT
 6|( ***  ALWAYS CALL E2MUSIC AS AN INIT IN PROGRAM  *** )
 7|CODE B2MUSIC H POP, Y PUSHX,
 8| 0 MUSIC-BARRAY-2 Y LXIX, bmusic CALL, Y POPX, NEXT
 9|CODE P2MUSIC H POP, Y PUSHX,
10| 0 MUSIC-BARRAY-2 Y LXIX, pmusic CALL, Y POPX, NEXT
11|CODE M2MUSIC H POP, D POP, Y PUSHX,
12| 0 MUSIC-BARRAY-2 Y LXIX, mmusic CALL, Y POPX, NEXT
13|CODE MP2MUSIC H POP, D POP, Y PUSHX,
14| 0 MUSIC-BARRAY-2 Y LXIX, mpmusic CALL, Y POPX, NEXT
15|-->
 +--------Block    143--------
 0|( JAYS VIDEO GAME GOODIES ) : CL SCRERASE ;
 1|DECIMAL { : BINARY } 2 BASE ! { ; }
 2|{ : QUAD } 4 BASE ! { ; }
 3|HEX
 4|CC? IFTRUE : GRAPHICS MAP 0 FB OUTP ; OTHERWISE : GRAPHICS ;
 5|IFEND
 6|: TEXOUT  10 * 8 + SWAP ROT 4 * + 100 * + ROT FFFC AND 40 *
 7| ROT 100 * ROT ; : NOW COUNT SPOST ;
 8|: XY 100 * SWAP 40 * SWAP ;
 9|DECIMAL
10|: SETOUT 64 * + HORIZONTAL ;
11|CODE DOIT H POP, PCHL, NEXT
12|: INIT GRAPHICS 1 8 OUTP 204 10 OUTP 43 0 SETOUT ;
13|-->
14|
15|
 +--------Block    144--------
 0|( QUEUE - VECTOR MANIPULATION ROUTINES )
 1|( THE QUEUE IS MAINTAINED AS A DOUBLE LINKED CIRCULAR LIST )
 2|( FIELD NAMES FOR QUEUE ENTRIES: )
 3|0 C= PQS ( STATUS ) 1 C= PQFL 2 C= PQFH ( FORWARD LINK )
 4|3 C= PQBL 4 C= PQBH ( BACKWARD LINK )
 5|5 C= PQRL 6 C= PQRH ( ROUTINE ) 7 C= PQTB ( TIME BASE )
 6|8 C= VXZW  ( EXCLUSION ZONE WIDTH ) 9 C= VAUXS ( AUX STATUS )
 7|10 C= VATMR ( ANIMATION TIMER )
 8|11 C= VTLL 12 C= VTLH ( TIME LIMIT )
 9|13 C= VXL 14 C= VXH 15 C= VDXL 16 C= VDXH ( X AND DX )
10|17 C= VDDXL 18 C= VDDXH ( DDX )
11|19 C= VYL 20 C= VYH 21 C= VDYL 22 C= VDYH ( Y AND DY )
12|23 C= VDDYL 24 C= VDDYH ( DDY )
13|25 C= VSAL 26 C= VSAH ( SCREEN ADDR ) 27 C= VMAGIC ( MAGIC )
14|28 C= VXPAND ( EXPANDER ) 29 C= VPATL 30 C= VPATH ( PAT ADDR )
15|-->
```

```
 +--------Block    145--------
 0|( VECTOR FIELD EQUATES CONTINUED )
 1|31 C= VFVPL 32 C= VFVPH ( FORMATION VECTOR POINTERS )
 2|33 C= VPCL 34 C= VPCH ( ANIMATION PROGRAM COUNTER )
 3|35 C= VSPL 36 C= VSPH ( ANIMATION STACK POINTER )
 4|37 C= VPTBL 38 C= VPTBH ( ANIMATION PATTERN TABLE )
 5|39 C= VIRL 40 C= VIRH ( INTERCEPT CHECK ROUTINE )
 6|41 C= VINTER ( INTERCEPT CODE ) 41 C= VRACK ( RACK CODE )
 7|42 C= VFNLPL 43 C= VFNLPH ( FINAL ANIMATION PATTERN )
 8|44 C= VSHFTA ( MAGIC REG USED IN LAST WRITE )
 9|45 C= VIDENT ( IDENTITY CODE - WHAT I AM )
10|46 C= VFXBL 47 C= VFXBH ( FORMATION X BIAS )
11|48 C= VFYBL 49 C= VFYBH ( FORMATION Y BIAS )
12|50 C= VASTKS ( ANIMATION STACK AREA START )
13|-->
14|
15|
 +--------Block    146--------
 0|( STATUS BIT EQUATES )
 1|7 C= PQSRH      ( RUN/HALT )
 2|6 C= PQSDW      ( DONT WRITE )
 3|5 C= PQSDE      ( DONT ERASE )
 4|4 C= PQSDF      ( DONT FREE )
 5|3 C= PQSDS      ( DONT SCREEN SYNCHONIZE )
 6|2 C= PQSUFP     ( USE FINAL PATTERN ON HALT )
 7|1 C= PQSNMT     ( NO MASTER TIME LIMIT )
 8|0 C= PQSFRZ     ( OFFSTAGE FREEZE )
 9|( AUXILLARY STATUS BITS )
10|7 C= ASFLOK     ( FORMATION MEMBER IS LOCKED IN )
11|( EQUATES FOR VECTOR HEAD )
12|0 C= QFL 1 C= QFH 2 C= QBL 3 C= QBH
13|-->
14|
15|
 +--------Block    147--------
 0|( VGS                    VWRITE )
 1|SUBR vwrite ( Write from a vector structure )
 2|  ( in- IX=vmagic of proper vector ; vector equates set )
 3|        ( WRTSYS 0< ) for pattern board 0= for software write )
 4|  ( out- pattern on screen ; scradr and shift saved for VERASE )
 5|  VXPAND X B LDX, VMAGIC X C LDX, VXH X D LDX, VXL X E LDX,
 6|  VPATH X H LDX, VPATL X L LDX, H PUSH, XTIY,
 7|  VYH X H LDX, VYL X L LDX,
 8|offalabs CALL,  ( calculates magic add. )
 9|  H VSAH X STX, L VSAL X STX,  ( set scradr for erase )
10| writep CALL,  ( write it )
11|    C VSHFTA X STX, ( save shift for erase ) Y POPX, RET,
12|-->
13|
14|
15|
```

```
 +--------Block    148--------
 0|( VGS                    VERASE )
 1|SUBR verase  ( does pattern board erase from vector IX )
 2|  ( in- IX=vmagic of proper vector ; vector equates set ;
 3|        scradr and shift saved from VWRITER )
 4|        ( WRTSYS 0<> for pattern board 0= for software write )
 5|  ( out- erased pattern from screen )
 6|  VXPAND X B LDX, VSHFTA X C LDX, VPATH X H LDX, VPATL X L LDX,
 7|  H PUSH, XTIY,
 8|  VSAH X H LDX, VSAL X L LDX,
 9|  writep CALL, Y POPX, RET,
10|DECIMAL -->
11|
12|
13|
14|
15|
 +--------Block    149--------
 0|( GLOBAL GAME RAM AREA START )
 1|HEX RAMBASE 300 + C= FIRSTRAMADDR FIRSTRAMADDR VPTR ! DECIMAL
 2|0 V= FBCOUNTER 3 BA= P1SCR 3 BA= P2SCR 0 V= MISSION
 3|0 V= P1ACT 0 V= P2ACT
 4|0 V= SKILLFACTOR 0 V= PLAYERUP
 5|0 V= NPLAYERS 0 V= OTHERFBCTR 0 V= OTHERSKILLF
 6|
 7|4 BARRAY vqhead : NILVQ 0 0 vqhead ! 0 2 vqhead ! ;
 8|
 9|FIRSTRAMADDR 40 + C= 1STCLRADDR
10|LASTRAMADDR 1STCLRADDR - 1+ C= CLRSIZE
11|DECIMAL -->
12|
13|
14|
15|
 +--------Block    150--------
 0|( STORAGE ALLOCATOR GOODIES )
 1|1STCLRADDR VPTR !
 2|24 C= NODECOUNT 64 C= NODESIZE NODECOUNT NODESIZE * C= POOLSIZE
 3|POOLSIZE BARRAY MEMPOOL 0 MEMPOOL VARIABLE FREELIST
 4|: INITFREELIST NODECOUNT 1 DO NODESIZE I * MEMPOOL
 5|  NODESIZE I 1 - * MEMPOOL 1+ ! LOOP ( THREAD THRU POINTERS )
 6|0 NODECOUNT 1 - NODESIZE * MEMPOOL 1+ ! 0 MEMPOOL FREELIST ! ;
 7|( GET A NODE FROM ASM LANGUAGE - RETURN: HL=NODE, 0 IF CAN'T )
 8|SUBR getnode FREELIST LHLD, L A MOV, H ORA, ( CHECK FOR NIL )
 9|   0<>, IF, H INX, M E MOV, H INX, M D MOV, H DCX, ( FREE=NXT )
10|H DCX, FREELIST SHED, THEN, RET,
11|CODE GETNODE DI, getnode CALL, H PUSH, EI, NEXT ( TERSE ENTRY )
12|( RELEASE NODE - HL=BLOCK TO FREE )
13|SUBR freenode FREELIST LHLD, FREELIST SHLD, ( LINK IN AT HEAD )
14|H INX, E M MOV, D INX, D M MOV, RET,
15|CODE FREENODE H POP, freenode CALL, NEXT -->
```

```
 +--------Block    151--------
 0|( ADD NODE TO QUEUE ROUTINE )
 1|SUBR ADDQ ( HL = NEW, IY = HEAD )
 2|QFL Y E LDX, QFH Y D LDX, E A MOV, D ORA, 0<>, IF,
 3|QBL Y C LDX, QBH Y B LDX, H PUSH, H INX,
 4|E M MOV, H INX, D M MOV, H INX, C M MOV, H INX, B M MOV,
 5|XCHG, D POP, H INX, H INX, H INX, E M MOV, H INX, D M MOV,
 6|E A MOV, B INX, B STAX, D A MOV, B INX, B STAX,
 7|ELSE, L E MOV, H D MOV, H INX, E M MOV, H INX, D M MOV,
 8|H INX, E M MOV, H INX, D M MOV, E QBL Y STX, D QBH Y STX,
 9|THEN, E QFL Y STX, D QFH Y STX, RET,
10|CODE ADDTQ DI, EXX, H POP, XTIY, ADDQ CALL, Y POPX, EXX, EI,
11|NEXT -->
12|
13|
14|
15|
 +--------Block    152--------
 0|( DELETE FROM QUEUE )
 1|SUBR delq ( IY=HEAD, IX=NODE TO DELETE )
 2|QFL Y E LDX, QFH Y D LDX, QBL Y L LDX, QBH Y H LDX, ( HEADER )
 3|A XRA, D DSBC, 0=, IF, ( IF I AM THE ONLY GUY LEFT )
 4|A QFL Y STX, A QFH Y STX, A QBL Y STX, A QBH Y STX, ( NIL OUT )
 5|ELSE, PQFL X E LDX, PQFH X D LDX, PQBL X C LDX, PQBH X B LDX,
 6|C L MOV, B H MOV, H INX, E M MOV, H INX, D M MOV, ( F[P[N]]=F )
 7|PQBL H LXI, D DAD, C M MOV, H INX, B M MOV, ( B[F[N]]=B[N] )
 8|X PUSHX, H POP, XCHG, H PUSH, ( DE=NODE ADDR, TOP=NODE FORW )
 9|QFL Y L LDX, QFH Y H LDX, A ANA, D DSBC, H POP,
10|0=, IF, L QFL Y STX, H QFH Y STX, ( SET HEAD TO F[N] )
11|ELSE, QBL Y L LDX, QBH Y H LDX, A ANA, D DSBC, 0=, IF,
12|C QBL Y STX, B QBH Y STX, ( SET TAIL ) THEN, THEN, THEN, RET,
13|CODE DELQ EXX, H POP, XTIY, X PUSHX, H PUSH, X POPX, delq CALL,
14|X POPX, Y POPX, EXX, NEXT
15|-->
 +--------Block    153--------
 0|( ADVANCE TO NEXT NODE ON QUEUE )
 1|SUBR nextq ( IY = HEAD )
 2|QFL Y L LDX, QFH Y H LDX, H A MOV, L ORA, RZ,
 3|H INX, M E MOV, H INX, M D MOV, ( DE=FORW[FORW[HEAD]] )
 4|H DCX, H DCX, L QBL Y STX, H QBH Y STX, ( BACK[HEAD]=NODE )
 5|E QFL Y STX, D QFH Y STX, ( FORW[HEAD]=FORW[NODE] )
 6|RET,
 7|CODE NEXTQ XTIY, nextq CALL, Y POPX, H PUSH, NEXT
 8|-->
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block    154--------
 0|( INCREMENT TIME BASES - C = TIME BASE, IY = Q HEAD )
 1|SUBR INCTB
 2|QFL Y L LDX, QFH Y H LDX, H A MOV, L ORA, ( Q FORW, NIL CHECK )
 3|RZ, L E MOV, H D MOV, BEGIN, ( QUIT IF NIL, ELSE REMEM FIRST )
 4|D PUSH, PQTB D LXI, D DAD, M A MOV, C ADD, A M MOV, ( UPDATE )
 5|A ANA, D DSBC, D POP, ( AND RETURN PTR TO NORMAL )
 6|H INX, M A MOV, H INX, M H MOV, A L MOV, ( HL=FORW[NODE] )
 7|E XRA, A B MOV, H A MOV, D XRA, B ORA, 0=, ( ONCE AROUND )
 8|END, RET,
 9|-->
10|
11|
12|
13|
14|
15|
 +--------Block    155--------
 0|( NEW, IMPROVED, HOTROD INTERRUPT SYSTEM ) DECIMAL
 1|( THESE PARAMETERS TUNE THE BACKGROUND TIME SLICING )
 2|0 V= BGWINDOW ( # OF LINES FOR BACKGROUND TIME SLICE )
 3|0 V= BGTLMT ( BACKGROUND MINUMUM SERVICE FREQUENCY )
 4|HEX CC? IFTRUE 62 C= IPNT OTHERWISE HERE 0F + FFF0 AND DP !
 5|DATA KPNT 0 , 0 , 0 , KPNT 2 + C= IPNT IFEND
 6|1 V= BACKGROUNDRUNNING 0 V= LOCKOUTCOUNTER 0 V= BGTIMER
 7|0 V= LPYC 0 V= LPFLAG IPNT 2 + C= BGINTVEC
 8|0 V= TIMER0 0 V= TIMER1 0 V= TIMER2 0 V= TIMER3
 9|( LIGHT PEN INTERRUPT ROUTINE )
10|SUBR LPINT PSW PUSH, VERAF IN, LPYC STA, 8 A MVI, LPFLAG STA,
11|INMOD OUT, PSW POP, EI, RET,
12|( ROUTINE TO RETURN Y ADDRESS SCREEN IS AT )
13|SUBR GETSYC A XRA, LPFLAG STA, 18 A MVI, INMOD OUT, BEGIN,
14|LPFLAG LDA, A ANA, 0< >, END, LPYC LDA, RET, -->
15|
 +--------Block    156--------
 0|( RESUME BACKGROUND - END INTERRUPT )
 1|DECIMAL F= ENDINT
 2|SUBR RESUMEBACKGROUND <ASSEMBLE
 3|DI, BGWINDOW LDA, C ADD, A C MOV,
 4|112 SUI, 52 CPI, ENDINT JRC, C A MOV, INLIN OUT,
 5|BGINTVEC A MVI, INFBK OUT,
 6|LABEL ENDINT Y POPX, X POPX, H POP, D POP, B POP, PSW POP,
 7|EXX, EXAF, H POP, D POP, B POP, 1 A MVI, BACKGROUNDRUNNING STA,
 8|BGTLMT LDA, BGTIMER STA,
 9|PSW POP, EI, RET, ASSEMBLE>
10|-->
11|
12|
13|
14|
15|
```

```
 +--------Block    157--------
 0|( TRY TO RUN SOMETHING IN FOREGROUND )
 1|F= RETPT F= TRYAGAIN
 2|HEX SUBR TRYFOREGROUND <ASSEMBLE
 3|A XRA, BACKGROUNDRUNNING STA, EI,
 4|LABEL TRYAGAIN GETSYC CALL, A C MOV, ( C=CURRENT LINE )
 5|LOCKOUTCOUNTER LDA, A ANA, RESUMEBACKGROUND JNZ, ( IF LOCKOUT )
 6|BGTIMER LDA, A ANA, RESUMEBACKGROUND JZ, ( OR TIMER COUNTDOWN )
 7|0 vqhead Y LXIX, QFL Y L LDX, QFH Y H LDX, H A MOV, L ORA,
 8|RESUMEBACKGROUND JZ, H PUSH, X POPX, PQTB X A LDX, A ANA,
 9|RESUMEBACKGROUND JZ, PQSDS PQS X BITX, 0=, IF,
10|VYH X A LDX, C SUB, 0<, IF, CMA, A INR, THEN,
11|VXZW X CMPX, RESUMEBACKGROUND JC, THEN, DI, nextq CALL,
12|EI, TRYAGAIN H LXI, H PUSH, PQRL X L LDX, PQRH X H LDX, PCHL,
13|ASSEMBLE>
14|-->
15|
 +--------Block    158--------
 0|( BACKGROUND END INTERRUPT )
 1|HEX SUBR BGENDI PSW PUSH, B PUSH, D PUSH, H PUSH, EXX, EXAF,
 2|PSW PUSH, B PUSH, D PUSH, H PUSH, X PUSHX, Y PUSHX,
 3|80 A MVI, INLIN OUT, IPNT A MVI, INFBK OUT, TRYFOREGROUND JMP,
 4|SUBR TIMINT ( TIMER INTERRUPT ROUTINE )
 5|PSW PUSH, B PUSH, D PUSH, H PUSH, EXX, EXAF, PSW PUSH, B PUSH,
 6|D PUSH, H PUSH, X PUSHX, Y PUSHX,
 7|TIMER0 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER0 SHLD, THEN,
 8|TIMER1 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER1 SHLD, THEN,
 9|TIMER2 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER2 SHLD, THEN,
10|TIMER3 LHLD, H A MOV, L ORA, 0<>, IF, H DCX, TIMER3 SHLD, THEN,
11|BGTIMER LDA, A ANA, 0<>, IF, A DCR, BGTIMER STA, THEN,
12|1 C MVI, 0 vqhead Y LXIX, INCTB CALL, MUSCPUS CALL,
13|BACKGROUNDRUNNING LDA, A ANA, TRYFOREGROUND JNZ,
14|Y POPX, X POPX, H POP, D POP, B POP, PSW POP,
15|EXX, EXAF, H POP, D POP, B POP, PSW POP, EI, RET, -->
 +--------Block    159--------
 0|( INTERRUPT START ROUTINE ) HEX
 1|CODE INTSTART DI, IPNT { SWAB } A MVI, STAI, IPNT A MVI,
 2|INFBK OUT, 1 A MVI, BACKGROUNDRUNNING STA, 8 A MVI, INMOD OUT,
 3|80 A MVI, INLIN OUT,
 4|IM2, EI, NEXT
 5|-->
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block    160--------
 0|( ROUTINE TO DELETE VECTOR IF STATUS SO INDICATES )
 1|SUBR KILLOFF PQSRH PQS X BITX, 0=, IF, DI, 0 vqhead Y LXIX,
 2|delq CALL, PQSDF PQS X BITX, 0 PQS X MVIX,
 3|0=, IF, X PUSHX, H POP, freenode CALL,
 4|THEN, EI, THEN, RET,
 5|-->
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    161--------
 0|( MACROS TO GENERATE ANIMATION OPCODES ) DECIMAL
 1|{ : SETP } 0 B, , { ; }
 2|{ : SETM } 2 B, B, { ; }
 3|{ : SETR } 4 B, , { ; }
 4|{ : SWAIT } 6 B, B, { ; }
 5|{ : ACALL } 8 B, , { ; }
 6|{ : AJMP } 10 B, , { ; }
 7|{ : SETDC } 12 B, SWAP , , { ; }
 8|{ : SETDDC } 14 B, SWAP , , { ; }
 9|{ : ARET } 16 B, { ; }
10|{ : AHALT } 18 B, { ; }
11|{ : SETI } 20 B, , { ; }
12|{ : SETXC } 22 B, , { ; }
13|{ : SETYC } 24 B, , { ; }
14|{ : DISPL } 26 B, SWAP B, B, { ; }
15|-->
 +--------Block    162--------
 0|( MORE ANIMATION MACRO STUFF )
 1|{ : AREPEAT } 28 B, B, HERE { ; }
 2|{ : ALOOP } 30 B, , { ; }
 3|{ : SETS } 32 B, B, B, { ; }
 4|{ : PATI } 34 B, B, { ; }
 5|{ : ASMCALL } 36 B, , { ; }
 6|{ : SETPT } 38 B, , { ; }
 7|{ : SETFR } 40 B, , { ; }
 8|{ : SETXFR } 42 B, B, { ; }
 9|{ : RANDOMDC } 44 B, SWAP B, , { ; }
10|{ : SETXB } 46 B, , { ; }
11|{ : SETYB } 48 B, , { ; }
12|{ : SETXP } 50 B, B, { ; }
13|{ : FOREVER } WHILE { ; }
14|{ : EVERFOR } 10 B, , { ; }
15|-->
```

```
 +--------Block    163--------
 0|( ANIMATION INTERPRETER ROUTINES )
 1|SUBR RASETP M E MOV, H INX, M D MOV, H INX, E VPATL X STX,
 2|D VPATH X STX, RET,
 3|SUBR RASETM M A MOV, H INX, A VMAGIC X STX, RET,
 4|SUBR RASETR M E MOV, H INX, M D MOV, H INX, E PQRL X STX,
 5|D PQRH X STX, RET,
 6|SUBR RAWAIT M A MOV, H INX, A VATMR X STX, L VPCL X STX,
 7|H VPCH X STX, H POP, RET,
 8|SUBR RACALL M C MOV, H INX, M B MOV, H INX, XCHG,
 9|VSPL X L LDX, VSPH X H LDX, E M MOV, H INX, D M MOV, H INX,
10|L VSPL X STX, H VSPH X STX, C L MOV, B H MOV, RET,
11|SUBR RARET VSPL X L LDX, VSPH X H LDX, H DCX, M D MOV, H DCX,
12|M E MOV, L VSPL X STX, H VSPH X STX, XCHG, RET,
13|SUBR RAHALT PQSRH PQS X RESX, H POP, RET,
14|SUBR RASETXP M A MOV, H INX, A VXPAND X STX, RET,
15|-->
 +--------Block    164--------
 0|( MORE ANIMATION INTERPRETER ROUTINES )
 1|SUBR RASETI M E MOV, H INX, M D MOV, H INX, E VIRL X STX,
 2|D VIRH X STX, RET,
 3|SUBR RASETXC M E MOV, H INX, M D MOV, H INX, E VXL X STX,
 4|D VXH X STX, RET,
 5|SUBR RASETYC M E MOV, H INX, M D MOV, H INX, E VYL X STX,
 6|D VYH X STX, RET,
 7|SUBR RAJMP M E MOV, H INX, M D MOV, XCHG, RET,
 8|SUBR RASETDC M E MOV, H INX, M D MOV, H INX,
 9|E VDXL X STX, D VDXH X STX,
10|M E MOV, H INX, M D MOV, H INX,
11|E VDYL X STX, D VDYH X STX, RET,
12|SUBR RASETDDC M E MOV, H INX, M D MOV, H INX, E VDDXL X STX,
13|D VDDXH X STX, M E MOV, H INX, M D MOV, H INX, E VDDYL X STX,
14|D VDDYH X STX, RET,
15|-->
 +--------Block    165--------
 0|( YET MORE ANIMATION INTERPRETER ROUTINES )
 1|SUBR RASETREP M A MOV, H INX, VSPL X E LDX, VSPH X D LDX,
 2|D STAX, D INX, E VSPL X STX, D VSPH X STX, RET,
 3|SUBR RALOOP M E MOV, H INX, M D MOV, H INX, VSPL X C LDX,
 4|VSPH X B LDX, B DCX, B LDAX, A DCR, 0< >, IF, B STAX, XCHG,
 5|ELSE, C VSPL X STX, B VSPH X STX, THEN, RET,
 6|SUBR RASETS M C MOV, H INX, M B MOV, H INX, PQS X A LDX,
 7|C XRA, B ANA, C ORA, A PQS X STX, RET,
 8|HEX
 9|SUBR RADIST M A MOV, H INX, XCHG, RRC, RRC, A B MOV, CO ANI,
10|A C MOV, B A MOV, 7 A BIT, 0< >, IF, CO ORI, ELSE, 3F ANI,
11|THEN, A B MOV, VXL X L LDX, VXH X H LDX, B DAD, L VXL X STX,
12|H VXH X STX, D LDAX, D INX, A B MOV, C C MVI,
13|VYL X L LDX, VYH X H LDX, B DAD, L VYL X STX, H VYH X STX,
14|XCHG, RET,
15|DECIMAL -->
```

```
  +--------Block    166--------
 0|( THE ABSOLUTELY LAST SCREEN OF ANIMATION INTERPRETER STUFF )
 1|SUBR RAPATI M C MOV, H INX, 0 B MVI, XCHG,
 2|VPTBL X L LDX, VPTBH X H LDX, B DAD, M C MOV, H INX, M B MOV,
 3|C VPATL X STX, B VPATH X STX, XCHG, RET,
 4|SUBR RASMCALL M E MOV, H INX, M D MOV, H INX, D PUSH, RET,
 5|SUBR RASETPT M E MOV, H INX, M D MOV, H INX, E VPTBL X STX,
 6|D VPTBH X STX, RET,
 7|SUBR RASETFP M E MOV, H INX, M D MOV, H INX, E VFNLPL X STX,
 8|D VFNLPH X STX, RET,
 9|SUBR RASETXZW M A MOV, H INX, A VXZW X STX, RET,
10|SUBR RARANDOMDO M C MOV, H INX, M E MOV, H INX, M D MOV, H INX,
11|LDAR, C ANA, RNZ, XCHG, RET,
12|SUBR RASETXB M E MOV, H INX, M D MOV, H INX, E VFXBL X STX,
13|D VFXBH X STX, RET,
14|SUBR RASETYB M E MOV, H INX, M D MOV, H INX, E VFYBL X STX,
15|D VFYBH X STX, RET, -->
  +--------Block    167--------
 0|( JUMP TABLE FOR INTERPRETER ROUTINES )
 1|DATA AJTBL RASETP , RASETM , RASETR , RAWAIT ,
 2|RACALL , RAJMP , RASETDC , RASETDDC , RARET ,
 3|RAHALT , RASETI , RASETXC , RASETYC , RADISP ,
 4|RASETREP , RALOOP , RASETS , RAPATI , RASMCALL ,
 5|RASETPT , RASETFP , RASETXZW , RARANDOMDO ,
 6|RASETXB , RASETYB , RASETXP ,
 7|-->
 8|
 9|
10|
11|
12|
13|
14|
15|
  +--------Block    168--------
 0|( ANIMATION UPDATOR ROUTINE )
 1|F= ANIRET
 2|SUBR ainter VATMR X A LDX, A ANA, RNZ, ( QUIT IF NOT NEEDED )
 3|VPCL X L LDX, VPCH X H LDX,
 4|LABEL ANIRET
 5|ANIRET D LXI, D PUSH, M C MOV, H INX, 0 B MVI, XCHG,
 6|AJTBL H LXI, B DAD, M C MOV, H INX, M B MOV, XCHG,
 7|B PUSH, RET,
 8|SUBR aup [illegible] CALL, PQSRH PQS X BITX, 0=, IF,
 9|PQSUFP PQS M BITX, 0<>, IF, VFNLPL X L LDX, VFNLPH X H LDX,
10|L VPATL X STX, H VPATH X STX, THEN, THEN, RET,
11|-->
12|
13|
14|
15|
```

```
  +--------Block    169--------
 0|( DECREMENT ANIMATION TIMERS, COMPUTE VECTORING TIME )
 1|F= TBVD F= TBOK F= TBQUIT
 2|SUBR TBCALC <ASSEMBLE DI,
 3|PQTB X C LDX, VATMR X B LDX, B A MOV, C SUB,
 4|0>=, IF, A VATMR X STX, 0 PQTB X MVIX,
 5|ELSE, C A MOV, B SUB, A PQTB X STX, 0 VATMR X MVIX, B C MOV,
 6|THEN, PQSNMT PQS X BITX, TBQUIT JRNZ, ( QUIT IF NO MASTER )
 7|0 B MVI, VTLL X L LDX, VTLH X H LDX, A ANA, B DSBC,
 8|TBVD JRZ, TBOK JP,
 9|LABEL TBVD L A MOV, C ADD, A C MOV, A XRA, A H MOV, A L MOV,
10|A VATMR X STX, A PQTB X STX, PQSRH PQS X RESX,
11|LABEL TBOK L VTLL X STX, H VTLH X STX,
12|LABEL TBQUIT EI, RET, ASSEMBLE>
13|-->
14|
15|
  +--------Block    170--------
 0|( TIME BASED VECTOR UPDATE - IX=VECTOR ADDR, IY=QUEUE ENTRY )
 1|( THIS VERSION VECTORS LINEARLY WITH LIMIT CHECKING )
 2|HEX F= .VLP1 F= .VLP2 F= NUD
 3|SUBR VECTLC <ASSEMBLE
 4|C A MOV, A ANA, RZ, ( DONT IF ZERO VECTORING WANTED )
 5|( NOW UPDATE COORDINATES )
 6|VXL X L LDX, VXH X H LDX, VDXL X E LDX, VDXH X D LDX, C B MOV,
 7|LABEL .VLP1 D DAD, .VLP1 DJNZ, H A MOV, 50 CPI, NUD JRNC,
 8|L VXL X STX, H VXH X STX,
 9|VYL X L LDX, VYH X H LDX, VDYL X E LDX, VDYH X D LDX, C B MOV,
10|LABEL .VLP2 D DAD, .VLP2 DJNZ, H A MOV, 0BA CPI, NUD JRNC,
11|L VYL X STX, H VYH X STX, 28 VXZW X MVIX, RET,
12|LABEL NUD PQSRH PQS X RESX, PQSDW PQS X SETX, RET, ASSEMBLE>
13|DECIMAL -->
14|
15|
  +--------Block    171--------
 0|( INITIALIZE INTERRUPT VERBS )
 1|CC? NOT IFTRUE
 2| LPINT IPNT 2 - U! TIMINT IPNT U! BGENDI BGINTVEC U!
 3|: FIREUP INTSTART ; OTHERWISE
 4|: FIREUP LPINT IPNT 2 - U! TIMINT IPNT U! BGENDI BGINTVEC U!
 5|INTSTART ; IFEND
 6|: START DI INIT INITFREELIST NILVQ LOCKOUTCOUNTER ZERO
 7|BGTIMER ZERO LPTLAG ZERO 33 BGWINDOW ! 2 BGTLMT ! FIREUP ;
 8|
 9|( ROUTINE TO VWRITE WITH INTERCEPT CHECKING )
10|SUBR VIWRITE INTST IN, VWRITE CALL, INTST IN,
11|A ANA, 0<>, IF, VIDL X L LDX, VIDH X H LDX,
12|H A MOV, L ORA, 0<>, IF, PCHL, THEN, THEN, RET,
13|DECIMAL
14|-->
15|
```

```
 +--------Block    172--------
 0|( SUBROUTINE TO UPDATE PATTERN USING XOR )
 1|SUBR XAWRITE TBCALC CALL,
 2|VECTLC CALL, ( UPDATE VECTOR )
 3|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
 4|PQSDE PQS X RESX,
 5| THEN, aup CALL, PQSDW PQS X BITX, 0=,
 6|IF, vwrite CALL, ELSE, PQSDW PQS X RESX, PQSDE PQS X SETX,
 7|THEN, KILLOFF JMP,
 8|( SUBROUTINE TO XAWRITE WITH INTERCEPT CHECKING )
 9|SUBR XIWRITE TBCALC CALL,
10|VECTLC  CALL, ( UPDATE DA VECTOR )
11|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
12|PQSDE PQS X RESX, THEN, aup CALL, PQSDW PQS X BITX, 0=,
13|IF, VIWRITE CALL, ELSE, PQSDW PQS X RESX, PQSDE PQS X SETX,
14|THEN, KILLOFF JMP,
15|-->
 +--------Block    173--------
 0|( SUBROUTINE TO VECTOR USING SECOND DERIVITIVE )
 1|F= VUPX F= VUPY
 2|SUBR VECTDD <ASSEMBLE PQSFRZ PQS X BITX, 0< >, IF,
 3|PQSDW PQS X SETX, RET, THEN, C A MOV, A ANA, RZ, PSW PUSH,
 4|VXL X L LDX, VXH X H LDX, VDXL X E LDX, VDXH X D LDX,
 5|VDDXL X C LDX, VDDXH X B LDX,
 6|LABEL VUPX XCHG, B DAD, XCHG, D DAD, A DCR, VUPX JRNZ,
 7|E VDXL X STX, D VDXH X STX, L VXL X STX, H VXH X STX,
 8|H A MOV, 80 CPI, CY~, IF, PQSFRZ PQS X SETX, PQSDW PQS X SETX,
 9|THEN, PSW POP, VYL X L LDX, VYH X H LDX, VDYL X E LDX,
10|VDYH X D LDX, VDDYL X C LDX, VDDYH X B LDX,
11|LABEL VUPY XCHG, B DAD, XCHG, D DAD, A DCR, VUPY JRNZ,
12|E VDYL X STX, D VDYH X STX, L VYL X STX, H VYH X STX,
13|H A MOV, 182 CPI, RC, PQSFRZ PQS X SETX, PQSDW PQS X SETX, RET,
14|ASSEMBLE> -->
15|
 +--------Block    174--------
 0|( SUBROUTINE TO UPDATE PATTERN USING XOR AND 2ND DERV VECTOR )
 1|SUBR XADDWRITE TBCALC CALL,
 2|VECTDD CALL,
 3|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
 4|PQSDE PQS X RESX,
 5| THEN, aup CALL,
 6|PQSDW PQS X BITX, 0=, IF, vwrite CALL, ELSE,
 7|PQSDE PQS X SETX, PQSDW PQS X RESX, THEN, KILLOFF JMP,
 8|-->
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block    175--------
 0|( UPDATE VECTOR FROM JOYSTICK ) HEX 11 C= JOYSTICK
 1|F= JYLP1 F= JYLP2 F= JXLP1 F= JXLP2 F= JXCK
 2|SUBR JOYUPD <ASSEMBLE C A MOV, A ANA, RZ, JOYSTICK IN, 18 ANI,
 3|18 CPI, JXCK JZ, VYL X L LDX, VYH X H LDX, VDYL X E LDX,
 4|VDYH X D LDX, C B MOV, JOYSTICK IN, 10 ANI, 0<>, IF,
 5|LABEL JYLP1 A ANA, D DSBC, JYLP1 DJNZ, H A MOV, VDDYL X CMPX,
 6|CY, IF, VDDYL X H LDX, 0 L MVI, THEN, ELSE,
 7|LABEL JYLP2 D DAD, JYLP2 DJNZ, H A MOV, VDDYH X CMPX, CY~, IF,
 8|VDDYH X H LDX, 0 L MVI, THEN, THEN, L VYL X STX, H VYH X STX,
 9|LABEL JXCK JOYSTICK IN, 6 ANI, 6 CPI, RZ, VXL X L LDX,
10|VXH X H LDX, VDXL X E LDX, VDXH X D LDX, C B MOV, JOYSTICK IN,
11|4 ANI, 0=, IF, LABEL JXLP1 A ANA, D DSBC, JXLP1 DJNZ, H A MOV,
12|VDDXL X CMPX, CY, IF, VDDXL X H LDX, 0 L MVI, THEN, ELSE,
13|LABEL JXLP2 D DAD, JXLP2 DJNZ, H A MOV, VDDXH X CMPX, CY~, IF,
14|VDDXH X H LDX, 0 L MVI, THEN, THEN, L VXL X STX, H VXH X STX,
15|RET, ASSEMBLE> DECIMAL -->
 +--------Block    176--------
 0|( SUBROUTINE TO UPDATE PATTERN FROM JOYSTICK )
 1|SUBR JOYWRITE TBCALC CALL,
 2|JOYUPD CALL, ( UPDATE FROM JOYSTICK )
 3|38 VXZW X MVIX,
 4|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
 5|PQSDE PQS X RESX,
 6| THEN, aup CALL, PQSDW PQS X BITX, 0=,
 7|IF, VIWRITE CALL,
 8|ELSE, PQSDE PQS X SETX, PQSDW PQS X RESX, THEN, KILLOFF JMP,
 9|-->
10|
11|
12|
13|
14|
15|
 +--------Block    177--------
 0|( COMPUTE DELTA FOR 1 COORDINATE )
 1|( FIRST A NEGATION SUBROUTINE )
 2|SUBR CMPHL H A MOV, CMA, A H MOV, L A MOV, CMA, A L MOV, H INX,
 3|RET,
 4|( IN: HL=TARGET, DE=TIME, BC=START )
 5|SUBR CDELTA B PUSH, A ANA, B DSBC, CY~, IF, UNSDIV CALL,
 6|ELSE, CMPHL CALL, UNSDIV CALL, CMPHL CALL, XCHG, CMPHL CALL,
 7|XCHG, THEN, B POP, B DAD, RET,
 8|-->
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block    178--------
 0|( CLEAR VECTOR ) F= INIZL
 1|SUBR CLRVEC <ASSEMBLE X PUSHX, H POP, 64 B MVI, A XRA,
 2|LABEL INIZL A M MOV, H INX, INIZL DJNZ, RET, ASSEMBLE>
 3|( RESET ANIMATION STUFF )
 4|SUBR CRASHA DI, L VPCL X STX, H VPCH X STX,
 5|X PUSHX, H POP, VASTKS D LXI, D DAD, L VSPL X STX,
 6|H VSPH X STX, 0 VATMR X MVIX, EI, RET,
 7|DECIMAL -->
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    179--------
 0|( SUBROUTINE TO PUT VECTOR ON PROCESS Q )
 1|SUBR STARTVEC DI, Y PUSHX,
 2|X PUSHX, H POP, 0 vqhead Y LXIX, ADDQ CALL,
 3|Y POPX, EI, RET,
 4|( SUBROUTINE TO INITIALIZE A STANDARD XOR WRITE )
 5|HEX
 6|SUBR SETSTDW 8 Y L LDX, 9 Y H LDX, CRASHA CALL,
 7|6 Y L LDX, 7 Y H LDX, L VTLL X STX, H VTLH X STX, L VRACK X STX,
 8| 5 Y A LDX, A VIDENT X STX, ( SET ID BYTE WITH H.O. STATUS )
 9|4 Y A LDX, A PQS X STX, 20 VMAGIC X MVIX,
10|30 VXZW X MVIX, XAWRITE H LXI,
11|L PQRL X STX, H PQRH X STX, RET,
12|DECIMAL -->
13|
14|
15|
 +--------Block    180--------
 0|( XVMOVE COMMAND - MOVE AN EXISTING VECTOR )
 1|( VRACK X1 Y1 X2 Y2 ALIST TIME STATUS VECTOR VMOVE )
 2|CODE XVMOVE X PUSHX, H POP, Y PUSHX, D POP, EXX,
 3|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX, CLRVEC CALL,
 4|16 Y C LDX, 17 Y B LDX, 12 Y L LDX, 13 Y H LDX,
 5|6 Y E LDX, 7 Y D LDX, D PUSH, CDELTA CALL,
 6|L VXL X STX, H VXH X STX, E VDXL X STX, D VDXH X STX,
 7|D POP, 14 Y C LDX, 15 Y B LDX, 10 Y L LDX, 11 Y H LDX,
 8|CDELTA CALL, L VYL X STX, H VYH X STX, E VDYL X STX,
 9|D VDYH X STX, SETSTDW CALL, 18 Y A LDX, A VRACK X STX,
10|STARTVEC CALL, UNFRAME 18 H LXI, SP DAD, SPHL,
11|EXX, D PUSH, Y POPX, H PUSH, X POPX, NEXT
12|: VMOVE GETNODE DUP 0 <> IF XVMOVE ELSE XDI ." FUBAR" THEN ;
13|DECIMAL -->
14|
15|
```

```
 +--------Block    181--------
 0|( XSTART COMMAND - START AN EXISTING VECTOR )
 1|( ALIST TIME STATUS VECTOR VMOVE )
 2|CODE XVSTART X PUSHX, H POP, Y PUSHX, D POP, EXX,
 3|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX, CLRVEC CALL,
 4|SETSTDW CALL, STARTVEC CALL,
 5|UNFRAME 8 H LXI, SP DAD, SPHL,
 6|EXX, D PUSH, Y POPX, H PUSH, X POPX, NEXT
 7|: VSTART GETNODE DUP 0 <> IF XVSTART ELSE XDI ." FUBAR" THEN ;
 8|DECIMAL -->
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    182--------
 0|( START A VECTOR WITH JUST INITIAL X AND Y ) DECIMAL
 1|( X Y ANIMATION TIME STATUS XYVSTART )
 2|SUBR XYVSTART DI, getnode CALL, EI, H PUSH,
 3|FRAME 2 Y L LDX, 3 Y H LDX, H PUSH, X POPX,
 4|CLRVEC CALL, 6 Y C LDX, C VRACK X STX,
 5|12 Y L LDX, 13 Y H LDX, L VXL X STX, H VXH X STX,
 6|10 Y L LDX, 11 Y H LDX, L VYL X STX, H VYH X STX,
 7|SETSTDW CALL, STARTVEC CALL,
 8|UNFRAME 12 H LXI, SP DAD, SPHL, RET,
 9|( CLUDGE TO CALL AS A VERB - VERY BIZARRE BUT IT SHOULD WORK )
10|SUBR CLUDGEIT H PUSH, D PUSH, B PUSH, EXX, H PUSH, D PUSH,
11|X PUSHX, D POP, Y PUSHX, H POP, EXX, XYVSTART JMP,
12|( X Y ANIMATION TIME STATUS XYVECTOR )
13|CODE XYVECTOR D POP, H POP, EXX, B POP, D POP, H POP,
14|CLUDGEIT CALL, EXX, D PUSH, X POPX, H PUSH, Y POPX, NEXT
15|-->
 +--------Block    183--------
 0|( CHECK FOR INTERCEPT WITH VECTOR )
 1|F= NOINT
 2|SUBR CHECKVEC <ASSEMBLE
 3|PQSRH PQS Y BITX, NOINT JZ, ( IF DEAD ALREADY )
 4|PQSDE PQS Y BITX, NOINT JNZ, ( OR IF NOT WRITTEN )
 5|VPATL X L LDX, VPATH X H LDX, M C MOV, H INX, M E MOV,
 6|VPATL Y L LDX, VPATH Y H LDX, M B MOV, H INX, M D MOV,
 7|VYH X A LDX, VYH Y SUBX, 0>=, IF, D CMP,
 8|NOINT JNC, ELSE, C ADD,  NOINT JM, THEN,
 9|VXH X A LDX, VXH Y SUBX,
10|0>=, IF, B CMP,
11|NOINT JNC, ELSE, C ADD,
12|NOINT JM, THEN, 1 A MVI, A ANA, RET,
13|LABEL NOINT A XRA, RET, ASSEMBLE>
14|-->
15|
```

```
 +--------Block    184--------
 0|( CHECK GROUP OF VECTORS FOR INTERCEPT )
 1|( IX = ME, C=MASK SPECIFING SUBSET TO EXAMINE )
 2|( RETURNS NZ AND IY=CULPRIT IF FOUND, ELSE Z )
 3|F= ICKL F= NOTEND F= NOTHIM
 4|SUBR CHECKALL <ASSEMBLE
 5|0 vqhead LHLD, ( HL=NEXT FELLA AFTER ME )
 6|LABEL ICKL X PUSHX, D POP, ( DE=ME )
 7|H A MOV, D CMP, NOTEND JRNZ, L A MOV, E CMP, RZ,
 8|LABEL NOTEND H PUSH, Y POPX,
 9|VIDENT Y A LDX, C ANA, NOTHIM JRZ, ( SELECTED BY MASK? )
10|B PUSH, CHECKVEC CALL, B POP,
11|RNZ, ( KICKOUT IF HE IS IT )
12|LABEL NOTHIM PQFL Y L LDX, PQFH Y H LDX, ICKL JMPR,
13|ASSEMBLE>
14|DECIMAL
15|-->
 +--------Block    185--------
 0|( NUMBER PATTERNS , 5 X 7 ORDERED 0-9 )
 1|DATA NUMPATS BINARY
 2|01111100 B, 10000010 B, 10000010 B, 10000010 B, 01111100 B,
 3|00000000 B, 10000100 B, 11111110 B, 10000000 B, 00000000 B,
 4|11100100 B, 10010010 B, 10010010 B, 10010010 B, 10001100 B,
 5|01000100 B, 10000010 B, 10010010 B, 10010010 B, 01101100 B,
 6|00110000 B, 00101000 B, 00100100 B, 11111110 B, 00100000 B,
 7|01001110 B, 10001010 B, 10001010 B, 10001010 B, 01110010 B,
 8|01111000 B, 10010100 B, 10010010 B, 10010010 B, 01000000 B,
 9|00000010 B, 11100010 B, 00010010 B, 00001010 B, 00000110 B,
10|01101100 B, 10010010 B, 10010010 B, 10010010 B, 01101100 B,
11|00001100 B, 10010010 B, 10010010 B, 01010010 B, 00111100 B,
12|DECIMAL -->
13|
14|
15|
 +--------Block    186--------
 0|( ROUTINE TO DISPLAY A BCD NUMBER 3 DIGITS LONG FROM VECTOR )
 1|HEX F= SHOWNUM F= NUMLP
 2|SUBR DISPBCD3 <ASSEMBLE
 3|VXL X E LDX, VXH X D LDX, VYL X L LDX, VYH X H LDX,
 4|428 B LXI, relabs CALL, XCHG, C A MOV,
 5|MAGIC OUT, 2 A MOV, XPAND OUT,
 6|VPATL X L LDX, VPATH X H LDX, H INX, M A MOV, 0F ANI,
 7|SHOWNUM CNZ, H DCX, M A MOV, RRC, RRC, RRC, RRC,
 8|SHOWNUM CALL, M A MOV,
 9|LABEL SHOWNUM H PUSH, F ANI, A C MOV, RLC, RLC, C ADD, A C MOV,
10|0 B MVI, NUMPATS H LXI, B DAD, XCHG, 5 B MVI,
11|LABEL NUMLP D LDAX, A M MOV, H INX, A M MOV, 0 XRA, H INX,
12|A M MOV, H INX, A M MOV, D INX,
13|L A MOV, 4D ADI, A L MOV, 0 A MVI, H ADC, A H MOV,
14|NUMLP DJNZ, 20 D LXI, B DAD, XCHG, H POP, RET, ASSEMBLE>
15|DECIMAL -->
```

```
 +--------Block    187--------
 0|( INTERRUPT WRITE NUMBER ROUTINE )
 1|SUBR NUMWRITE
 2|TBCALC CALL,
 3|aup CALL,
 4|PQSDW PQS X BITX, 0=, IF, DISPBCD3 CALL, THEN,
 5|KILLOFF JMP,
 6|-->
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    188--------
 0|( BASE STATION )
 1|DECIMAL DATA FIREBASE 5 B, 13 B,
 2|QUAD 0222 B, 2222 B, 2000 B, 0000 B, 0 B,
 3|2111 B, 1111 B, 2220 B, 0000 B, 0 B,
 4|0222 B, 2222 B, 2000 B, 0000 B, 0 B,
 5|0000 B, 1110 B, 0022 B, 2200 B, 0 B,
 6|0000 B, 0111 B, 0002 B, 2220 B, 0 B,
 7|1111 B, 1111 B, 1102 B, 2222 B, 0 B,
 8|0002 B, 2222 B, 2222 B, 2222 B, 1000 B,
 9|1111 B, 1111 B, 1102 B, 2222 B, 0 B,
10|0000 B, 0111 B, 0002 B, 2220 B, 0 B,
11|0000 B, 1110 B, 0022 B, 2200 B, 0 B,
12|0222 B, 2222 B, 2000 B, 0000 B, 0 B,
13|2111 B, 1111 B, 2220 B, 0000 B, 0 B,
14|0222 B, 2222 B, 2000 B, 0000 B, 0 B, -->
15|
 +--------Block    189--------
 0|( SMALL BASE ) DECIMAL DATA SMALBASE 4 B, 11 B, QUAD
 1|0222 B, 2220 B, 0000 B, 0 B,
 2|2222 B, 2200 B, 0000 B, 0 B,
 3|0011 B, 0000 B, 0000 B, 0 B,
 4|0111 B, 1000 B, 2200 B, 0 B,
 5|1111 B, 1110 B, 0220 B, 0 B,
 6|0002 B, 2222 B, 2222 B, 2000 B,
 7|1111 B, 1110 B, 0220 B, 0 B,
 8|0111 B, 1000 B, 2200 B, 0 B,
 9|0011 B, 0000 B, 0000 B, 0 B,
10|2222 B, 2200 B, 0000 B, 0 B,
11|0222 B, 2220 B, 0000 B, 0 B,
12|DECIMAL -->
13|
14|
15|
```

```
 +--------Block    190--------
 0|( GORF ) DECIMAL DATA GORF 6 B, 15 B, QUAD
 1|2000 B, 3330 B, 0000 B, 0000 B, 0010 B, 0 B,
 2|2003 B, 3333 B, 3300 B, 0000 B, 0100 B, 0 B,
 3|2223 B, 3333 B, 3333 B, 3000 B, 1000 B, 0 B,
 4|0003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0 B,
 5|0033 B, 3331 B, 1111 B, 3333 B, 3300 B, 0 B,
 6|0031 B, 1132 B, 2111 B, 3333 B, 3330 B, 0 B,
 7|0032 B, 1233 B, 3333 B, 3333 B, 3333 B, 0 B,
 8|0031 B, 1133 B, 3333 B, 3333 B, 3333 B, 0 B,
 9|0032 B, 1233 B, 3333 B, 3333 B, 3333 B, 0 B,
10|0031 B, 1131 B, 1111 B, 3333 B, 3330 B, 0 B,
11|0033 B, 3332 B, 2111 B, 3333 B, 3300 B, 0 B,
12|0003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0 B,
13|2223 B, 3333 B, 3333 B, 3000 B, 1000 B, 0 B,
14|2003 B, 3333 B, 3300 B, 0000 B, 0100 B, 0 B,
15|2000 B, 3330 B, 0000 B, 0000 B, 0010 B, 0 B, -->
 +--------Block    191--------
 0|( GORFB ) DECIMAL DATA GORFB 6 B, 15 B, QUAD
 1|0000 B, 3330 B, 0000 B, 0000 B, 2000 B, 0000 B,
 2|0003 B, 3333 B, 3300 B, 0000 B, 0200 B, 0000 B,
 3|1113 B, 3333 B, 3333 B, 3000 B, 2000 B, 0000 B,
 4|1003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0000 B,
 5|1033 B, 3331 B, 1222 B, 3333 B, 3300 B, 0000 B,
 6|0033 B, 2332 B, 2222 B, 3333 B, 3330 B, 0000 B,
 7|0033 B, 2333 B, 3333 B, 3333 B, 3333 B, 0000 B,
 8|0033 B, 2333 B, 3333 B, 3333 B, 3333 B, 0000 B,
 9|0033 B, 2333 B, 3333 B, 3333 B, 3333 B, 0000 B,
10|0033 B, 2331 B, 1222 B, 3333 B, 3330 B, 0000 B,
11|1033 B, 3332 B, 2222 B, 3333 B, 3300 B, 0000 B,
12|1003 B, 3333 B, 3333 B, 3333 B, 0000 B, 0000 B,
13|1113 B, 3333 B, 3333 B, 3000 B, 2000 B, 0000 B,
14|0003 B, 3333 B, 3300 B, 0000 B, 0200 B, 0000 B,
15|0000 B, 3330 B, 0000 B, 0000 B, 2000 B, 0000 B, DECIMAL -->
 +--------Block    192--------
 0|( FIRE BASE EXPLOSION PATTERN )
 1|DATA FBEXP1 4 B, 12 B, QUAD
 2|0000 B, 3000 B, 0010 B, 0000 B,
 3|3000 B, 3303 B, 0000 B, 0000 B,
 4|0333 B, 3333 B, 0000 B, 0000 B,
 5|0033 B, 3333 B, 3330 B, 0000 B,
 6|0033 B, 3133 B, 1333 B, 0000 B,
 7|3333 B, 1111 B, 1330 B, 0000 B,
 8|3331 B, 1111 B, 1330 B, 0000 B,
 9|0333 B, 3311 B, 1330 B, 0000 B,
10|0033 B, 0333 B, 0300 B, 0000 B,
11|1030 B, 0033 B, 0001 B, 0000 B,
12|0000 B, 0033 B, 0000 B, 0000 B,
13|0000 B, 0033 B, 0000 B, 0000 B,
14|DECIMAL -->
15|
```

```
  +--------Block    193--------
 0|( ANOTHER FIREBASE EXPLOSION PATTERN )
 1|DATA FBEXP2 5 B, 17 B, QUAD
 2|0001 B, 0000 B, 0000 B, 0000 B, 0000 B,
 3|0000 B, 1000 B, 0000 B, 0000 B, 0000 B,
 4|0000 B, 1110 B, 0000 B, 0000 B, 0000 B,
 5|0000 B, 0110 B, 0010 B, 0000 B, 0000 B,
 6|0000 B, 0111 B, 1110 B, 0000 B, 3000 B,
 7|0000 B, 0131 B, 1110 B, 0110 B, 0000 B,
 8|0000 B, 1133 B, 3111 B, 1111 B, 0000 B,
 9|0000 B, 1133 B, 3333 B, 3311 B, 0000 B,
10|3001 B, 1333 B, 3333 B, 3310 B, 0000 B,
11|0111 B, 1333 B, 3333 B, 3110 B, 0000 B,
12|0111 B, 3333 B, 3333 B, 3110 B, 0000 B,
13|0011 B, 3333 B, 3333 B, 3311 B, 0000 B,
14|0011 B, 3311 B, 3331 B, 3310 B, 0000 B,
15|-->
  +--------Block    194--------
 0|( CONTINUATION OF FBEXP2, PHASOR AND NULPAT )
 1|0000 B, 1111 B, 1111 B, 1110 B, 0000 B,
 2|0011 B, 0110 B, 0110 B, 1011 B, 1000 B,
 3|0010 B, 0000 B, 0000 B, 1000 B, 1000 B,
 4|1100 B, 0000 B, 0000 B, 1000 B, 0000 B,
 5|DECIMAL DATA FBURST 4 B, 1 B, QUAD 1111 B, 1111 B, 1111 B, 0 B,
 6|DECIMAL DATA NULPAT 1 B, 1 B, 0 B,
 7|DECIMAL -->
 8|
 9|
10|
11|
12|
13|
14|
15|
  +--------Block    195--------
 0|( FBEXP3 )
 1|DATA FBEXP3 6 B, 23 B, QUAD
 2|0000 B, 0000 B, 2000 B, 0000 B, 0000 B, 0000 B,
 3|0000 B, 0002 B, 1200 B, 0000 B, 0000 B, 0000 B,
 4|0000 B, 2221 B, 1120 B, 0000 B, 0000 B, 0000 B,
 5|0002 B, 1111 B, 1112 B, 0022 B, 0000 B, 0000 B,
 6|0023 B, 3133 B, 3311 B, 2212 B, 2000 B, 0000 B,
 7|0211 B, 3333 B, 3331 B, 1131 B, 1200 B, 0000 B,
 8|0021 B, 1333 B, 3333 B, 3331 B, 1131 B, 0000 B,
 9|0002 B, 0333 B, 3333 B, 3333 B, 1112 B, 0000 B,
10|0000 B, 2333 B, 3333 B, 3333 B, 3111 B, 0000 B,
11|0002 B, 1133 B, 3333 B, 3333 B, 3331 B, 1200 B,
12|0021 B, 1333 B, 3333 B, 3333 B, 3311 B, 2000 B,
13|0211 B, 3333 B, 3333 B, 3333 B, 3112 B, 0000 B,
14|2211 B, 3333 B, 3333 B, 3333 B, 1120 B, 0000 B,
15|-->
```

```
 +--------Block    196--------
 0|( CONTINUED FBEXP3 )
 1|0211 B, 1133 B, 3333 B, 3311 B, 1120 B, 0000 B,
 2|0021 B, 3333 B, 3333 B, 3331 B, 2200 B, 0000 B,
 3|0002 B, 3323 B, 3333 B, 3332 B, 0000 B, 0000 B,
 4|0000 B, 3231 B, 3333 B, 3332 B, 0000 B, 0000 B,
 5|0000 B, 2133 B, 3333 B, 3332 B, 2200 B, 0000 B,
 6|0002 B, 1113 B, 3133 B, 3331 B, 1120 B, 0000 B,
 7|0000 B, 2111 B, 1123 B, 1311 B, 1200 B, 0000 B,
 8|0000 B, 0211 B, 1202 B, 1111 B, 2000 B, 0000 B,
 9|0000 B, 0021 B, 2000 B, 2112 B, 0000 B, 0000 B,
10|0000 B, 0002 B, 0000 B, 0220 B, 0000 B, 0000 B,
11|-->
12|
13|
14|
15|
 +--------Block    197--------
 0|( FBEXP4 )
 1|DECIMAL DATA FBEXP4 6 B, 23 B, QUAD
 2|2000 B, 0000 B, 0020 B, 0000 B, 0000 B, 2000 B,
 3|0200 B, 0000 B, 0200 B, 0000 B, 0002 B, 0000 B,
 4|0020 B, 0000 B, 2000 B, 0000 B, 0020 B, 0000 B,
 5|0002 B, 0000 B, 2000 B, 0000 B, 0200 B, 0000 B,
 6|0000 B, 2202 B, 2201 B, 1000 B, 2000 B, 0000 B,
 7|0000 B, 0223 B, 1332 B, 1102 B, 0000 B, 0000 B,
 8|0000 B, 2233 B, 3333 B, 2110 B, 0002 B, 0000 B,
 9|0000 B, 0231 B, 3333 B, 3310 B, 0020 B, 0000 B,
10|0000 B, 0223 B, 1333 B, 3111 B, 2200 B, 0000 B,
11|0000 B, 0023 B, 1333 B, 3120 B, 0000 B, 0000 B,
12|0000 B, 2223 B, 1333 B, 3122 B, 0000 B, 0000 B,
13|0002 B, 2331 B, 3333 B, 3312 B, 0000 B, 0000 B,
14|0022 B, 3133 B, 3333 B, 3220 B, 0000 B, 0000 B,
15|-->
 +--------Block    198--------
 0|( FBEXP4 CONTINUED )
 1|0223 B, 3113 B, 3333 B, 1200 B, 0000 B, 0000 B,
 2|0002 B, 2311 B, 3331 B, 1220 B, 0000 B, 0000 B,
 3|0002 B, 2233 B, 3331 B, 3322 B, 0000 B, 0000 B,
 4|0020 B, 0232 B, 2233 B, 3320 B, 0000 B, 0000 B,
 5|0200 B, 0220 B, 0223 B, 3222 B, 0000 B, 0000 B,
 6|2000 B, 0000 B, 0022 B, 2302 B, 2000 B, 0000 B,
 7|0000 B, 0000 B, 0000 B, 0000 B, 0200 B, 0000 B,
 8|0000 B, 0000 B, 0000 B, 0000 B, 0020 B, 0000 B,
 9|0000 B, 0000 B, 0000 B, 0000 B, 0002 B, 0000 B,
10|0000 B, 0000 B, 0000 B, 0000 B, 0000 B, 2000 B,
11|DECIMAL -->
12|
13|
14|
15|
```

```
  +--------Block    199--------
 0|( FIREBASE EXPLOSION 5 )
 1|DATA FBEXP5 6 B, 23 B, QUAD
 2|~ 0000 0000 0010 0003 0000 0000 ^
 3|~ 0000 0300 1001 1000 0300 0000 ^
 4|~ 0000 2030 0200 0030 0000 0000 ^
 5|~ 0000 0000 0001 0000 0000 0000 ^
 6|~ 0300 2100 2000 3000 0200 0000 ^
 7|~ 0030 0200 0002 0000 0033 3000 ^
 8|~ 0300 0020 0220 0030 0300 0000 ^
 9|~ 0000 0000 0300 0000 0003 0000 ^
10|~ 0033 3002 2200 0110 0011 1000 ^
11|~ 0003 3001 2000 1100 0122 0000 ^
12|~ 0033 2201 2201 1010 1022 1000 ^
13|~ 3300 0221 2011 1100 1110 0000 ^
14|~ 3330 0022 2001 1110 1100 0000 ^
15|~ 0030 2000 0222 2100 0000 1000 ^ -->
  +--------Block    200--------
 0|( FBEXP5 CONTINUED )
 1|~ 0033 1110 0020 1020 1100 0000 ^
 2|~ 0300 0110 0200 1000 0100 0000 ^
 3|~ 0001 0003 0030 0010 0030 0000 ^
 4|~ 0000 0100 0000 0000 1000 0000 ^
 5|~ 0200 0000 0010 0000 0000 0000 ^
 6|~ 0001 0030 0000 0030 0010 0000 ^
 7|~ 0000 0000 0020 0000 0000 0000 ^
 8|~ 0000 0010 0000 0100 3000 0000 ^
 9|~ 0000 0000 0102 0030 0000 0000 ^
10|DECIMAL -->
11|
12|
13|
14|
15|
  +--------Block    201--------
 0|( FIRE BASE EXPLOSION 6 )
 1|DATA FBEXP6 6 B, 23 B, QUAD
 2|~ 0000 0000 3000 0000 0000 0000 ^
 3|~ 0000 0020 0000 0300 0000 0000 ^
 4|~ 0000 0000 0300 0000 0010 0000 ^
 5|~ 0001 0000 0000 0200 0000 0000 ^
 6|~ 0000 0000 0000 0000 0000 0000 ^
 7|~ 0002 0010 0001 0001 0200 0000 ^
 8|~ 0002 0000 1000 0000 0000 0100 ^
 9|~ 0000 0300 0010 0000 0100 3000 ^
10|~ 0100 3000 0000 0000 0000 0000 ^
11|~ 0001 0002 0010 0001 0300 0000 ^
12|~ 3000 0000 0020 0000 0003 0000 ^
13|~ 0020 0200 0000 0000 0010 0000 ^
14|~ 0000 0001 0200 0000 0000 0000 ^
15|~ 2030 1000 0100 0300 0010 0002 ^ -->
```

```
 +--------Block    202--------
 0|( FIRE BASE EXPLOSION 6 CONTINUED )
 1|~ 0001 0000 0200 0003 0000 1000 ^
 2|~ 1000 0010 0003 0100 0020 0000 ^
 3|~ 0000 0000 1000 0000 0000 0000 ^
 4|~ 0030 0300 0000 0000 0200 0000 ^
 5|~ 0000 0000 0000 0000 0000 0000 ^
 6|~ 0000 3010 0000 1000 0200 0000 ^
 7|~ 0000 0100 0200 0003 0000 0000 ^
 8|~ 0000 0003 0010 0000 0000 0000 ^
 9|~ 0000 0001 0020 1000 0000 0000 ^
10|DECIMAL -->
11|
12|
13|
14|
15|
 +--------Block    203--------
 0|( ALIEN EXPLOSION PATTERN )
 1|DATA EXPLOSION1 3 B, 11 B, QUAD
 2|0000 B, 0000 B, 0000 B, 0000 B, 0000 B, 0000 B,
 3|0000 B, 0010 B, 0000 B, 0100 B, 3000 B, 0000 B,
 4|0033 B, 3330 B, 0000 B, 0003 B, 1300 B, 0000 B,
 5|0031 B, 1130 B, 0000 B, 0033 B, 1330 B, 0000 B,
 6|0103 B, 3000 B, 0000 B, 0000 B, 0010 B, 0000 B,
 7|0000 B, 0000 B, 0000 B,
 8|DECIMAL DATA EXPLOSION2 3 B, 11 B, QUAD
 9|1001 B, 0001 B, 0000 B, 0100 B, 1010 B, 0000 B,
10|0011 B, 1100 B, 0000 B, 0111 B, 3111 B, 0000 B,
11|0013 B, 3310 B, 0000 B, 1011 B, 3110 B, 0000 B,
12|0113 B, 1100 B, 0000 B, 0011 B, 0111 B, 0000 B,
13|0101 B, 0110 B, 0000 B, 0000 B, 0010 B, 0000 B,
14|1000 B, 1001 B, 0000 B,
15|DECIMAL -->
 +--------Block    204--------
 0|( MORE ALIEN EXPLOSIONS )
 1|DATA EXPLOSION3 3 B, 12 B, QUAD
 2|0000 B, 0101 B, 0000 B,
 3|1000 B, 0000 B, 0000 B,
 4|0010 B, 0000 B, 0000 B,
 5|0000 B, 3000 B, 0000 B,
 6|0003 B, 3301 B, 0000 B,
 7|0033 B, 2301 B, 0000 B,
 8|0003 B, 2210 B, 0000 B,
 9|0003 B, 3330 B, 0000 B,
10|0100 B, 3002 B, 1000 B,
11|0000 B, 0000 B, 1000 B,
12|0001 B, 0000 B, 0000 B,
13|0000 B, 0010 B, 0000 B,
14|DECIMAL -->
15|
```

```
 +--------Block    205--------
 0|( EXPLOSION PATTERNS ) DECIMAL
 1|DATA EXPLOSION4 3 B, 11 B, QUAD
 2|3001 B, 0020 B, 0000 B, 3001 B, 0030 B, 0000 B,
 3|0300 B, 0300 B, 0000 B, 0001 B, 1003 B, 0000 B,
 4|0111 B, 1100 B, 0000 B, 2110 B, 1102 B, 0000 B,
 5|0101 B, 1000 B, 0000 B, 0111 B, 1101 B, 0000 B,
 6|2001 B, 0000 B, 0000 B, 3001 B, 0220 B, 0000 B,
 7|0020 B, 0101 B, 0000 B, DECIMAL
 8|DATA EXPLOSION5 3 B, 11 B, QUAD
 9|0000 B, 0010 B, 0000 B, 0200 B, 0000 B, 0000 B,
10|0000 B, 0300 B, 0000 B, 0000 B, 0000 B, 0000 B,
11|2000 B, 0010 B, 0000 B, 0030 B, 0012 B, 0000 B,
12|0000 B, 1000 B, 0000 B, 0100 B, 0000 B, 0000 B,
13|0020 B, 2030 B, 0000 B, 3000 B, 0000 B, 0000 B,
14|0002 B, 0001 B, 0000 B, DECIMAL
15|-->
 +--------Block    206--------
 0|( INVADERS- PLAYER SHOOTING SOUND, ID ) HEX
 1|DATA IDSCORE
 2| 88 ABVOLS 18 MCVOLS
 3| A0 87 15 TONES
 4| 50 1 -2 2 MOVENOISE
 5| 1 -2 3F ( far right ) MOVESOUND
 6| 4A MASTER 1 4 6A 8 RAMBLE 3 COUNTLIMITS
 7| PLAY
 8|QUIET
 9|-->
10|
11|
12|
13|
14|
15|
 +--------Block    207--------
 0|( INVADERS- PLAYER EXPLOSION, 1G -- AND PZIP PZ ) HEX
 1|DATA 1GSCORE
 2| 2 1 20 MOVESOUND
 3| 53 66 75 TONES
 4| 1F MCVOLS 0FF ABVOLS
 5| 4 5 1 01F MOVENOISE
 6| 3 MASTER 2A -1 3 2 RAMBLE 2 COUNTLIMITS
 7| PLAY FF 1F 0C -1 0 0F MOVEVOLS
 8| 1 2 10 MOVESOUND
 9| 4 COUNTLIMITS RERAMBLE
10|PLAY QUIET
11|DATA PZSCORE AGG ATT AGGG TONES 1 -4 3F MOVESOUND 10 MASTER
12|1 4 A0 10 RAMBLE 4 COUNTLIMITS 10 1 4 70 MOVENOISE
13|88 ABVOLS 00 MCVOLS 0F VIDS PLAY QUIET
14|-->
15|
```

```
 +--------Block    208--------
 0|( SPACE MISSIONS ZPIP SOUND - ZP ) HEX
 1|DATA ZPSCORE
 2| #G3 #F3 #CS3 TONES
 3| 2 1 0 MOVESOUND
 4| 0 1 4 30 MOVENOISE
 5| 60 MASTER 1 -4 60 30 RAMBLE 1 COUNTLIMITS
 6| 88 ABVOLS 18 MCVOLS PLAY
 7|2B MASTER 1 -4 2F 5 RAMBLE 1 COUNTLIMITS 1E 1 -4 6 MOVENOISE
 8|PLAY QUIET
 9|( SPACE MISSIONS BMUSIC BLOCK )
10|: 1G 1GSCORE PMUSIC & MS 1GSCORE P2MUSIC ;
11|: PZ PZSCORE BMUSIC ;
12|: ZP ZPSCORE BMUSIC ;
13|DECIMAL -->
14|
15|
 +--------Block    209--------
 0|( DRAW FIREBASES ON SCREEN )
 1|HEX TABLE FBADDRESSES 200 , 1000 , 1E00 , 2C00 , 3A00 ,
 2|: REDRAWFB FBCOUNTER @ IF
 3|FBCOUNTER @ 0 DO
 4|100 I FBADDRESSES @ SMALBASE 20 WRITEP LOOP THEN ;
 5|DECIMAL -->
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    210--------
 0|( GAME VARIABLES AND CONSTANTS )
 1|TIMER0 C= WTIMER TIMER1 C= BOMBTIMER TIMER2 C= ATTACKTIMER
 2|TIMER3 C= RACKTIMER 0 V= FIREACTION
 3|0 V= RACKDONE 0 V= INVADERSLEFT 0 V= PLAYERHIT 0 V= ENDOFFRAME
 4|0 V= GAMEOVER 0 V= PHASINTR 0 V= FIRETRACK
 5|0 V= SCRPTR 0 V= FV1 2 V= MAXBOMBS 4 ARRAY BVLIST
 6|0 V= FBANIM 0 V= RSTBL 0 V= REINIT ( MUST REINITIALIZE )
 7|10 C= NBOMBS 6 C= BOMBASIZE NBOMBS BOMBASIZE * C= BATOTAL
 8|BATOTAL BA= BOMBARRAY
 9|HEX 40 BA= FBVECTOR
10|0 V= PINTERFLAG 0 V= PINTERN 0 V= PINTERX 0 V= PINTERY
11|10 C= COUNTH 10 C= SWIPE 1 C= SWFIRE 0 C= PLYPRINTX 0F C= FFXLL
12|0 C= INIVLL 4000 C= INIVUL -200 C= BOMBDX 700 C= BOMBDY
13|200 C= INIBMY
14|DECIMAL
15|-->
```

```
 +--------Block    211--------
 0|( INITIALIZE GAME SCREEN ) HEX
 1|: INITSCREEN DI GRAPHICS 1 CONCM OUTP INITFREELIST
 2|NILVQ FIREUP @ 4000 4000 FILL CB VERBL OUTP D 0 SETOUT
 3|128 5C C A 1 BOX ;
 4|: UPDATESCORE PLAYERUP @ IF 0 P2SCR BCD+! 4900 9600 408 0 P2SCR
 5|ELSE 0 P1SCR BCD+! 4900 0 408 0 P1SCR THEN DI DISPBCD6 EI ;
 6|: INI1PG 4500 C 408 0 P1SCR DISPBCD6
 7|4D00 0300 408 A" ONE" COUNT SPOST
 8|4D00 4800 408 A" MISSION" COUNT SPOST
 9|49C0 5E00 428 MISSION @ CPOST REDRAWFB ;
10|: INI2PG INI1PG 4900 9600 408 0 P2SCR DISPBCD6
11|4D00 A100 408 A" TWO" COUNT SPOST ;
12|: DRAWMISSIONSCREEN INITSCREEN NPLAYERS @ IF INI2PG ELSE INI1PG
13|THEN START ;
14|DECIMAL -->
15|
 +--------Block    212--------
 0|( RACK UPDATOR )
 1|HEX 0 V= INVADERNUM 0 V= MASTERY 0 V= DMASTERY
 2|0 V= MASTERX 0 V= DMASTERX 0 V= INVLL 0 V= INVUL
 3|0 V= LEFTINVN 0 V= RIGHTINVN
 4|0 V= BUMPMASTERROUTINE 0 V= NORMLP1 0 V= NORMLP2
 5|8 BARRAY ALIVEBITS
 6|8 BARRAY RACKBITS DATA BITMASK 1 B, 2 B, 4 B, 8 B, 10 B, 20 B,
 7|40 B, 80 B, 80 ARRAY ANIMSTATE
 8|8 BARRAY RACKEXPS 0 V= INVPATAB
 9|: RESETRACK 0 INVADERNUM B! 0 MASTERY ! 0 MASTERX !
10|INIDMY DMASTERY !
11|FE00 DMASTERX ! INIVLL INVLL ! INIVUL INVUL ! ;
12|-->
13|
14|
15|
 +--------Block    213--------
 0|( SUBROUTINES TO CALCULATE DISPLACEMENTS FOR RACK MEMBER ) HEX
 1|SUBR CALCINVX 7 ANI, RLC, RLC, A H MOV, 0 L MVI, RET,
 2|SUBR CALCINVY 38 ANI, RLC, A H MOV, 0 L MVI, RET,
 3|( CHECK FOR SCREEN EDGE, NEGATE DELTA AND BUMP X IF AT IT )
 4|SUBR FLIPCHECK H A MOV, B CMP, RNZ, L A MOV, C CMP, RNZ,
 5|D A MOV, CMA, A D MOV, E A MOV, CMA, A E MOV, D INX, A XRA,
 6|RET,
 7|( INDEX RACK BITS AND ALIVE BITS )
 8|SUBR XRACKBITS C A MOV, 7 ANI, A E MOV, 0 D MVI,
 9|BITMASK H LXI, D DAD, M B MOV, C A MOV, RRC, RRC,
10|RRC, 7 ANI, A F MOV, A RACKBITS H LXI, D DAD,
11|M A MOV, B ANA, RET,
12|SUBR XALIVEBITS C A MOV, 7 ANI, A E MOV, 0 D MVI,
13|BITMASK H LXI, D DAD, M B MOV, C A MOV, RRC, RRC, RRC, 7 ANI,
14|A E MOV, A ALIVEBITS H LXI, D DAD, M A MOV, B ANA, RET,
15|-->
```

```
 +--------Block    214--------
 0|( WAIT AND ANIMATION TRACKING TABLE ROUTINES ) HEX
 1|: WAIT WTIMER !  BEGIN WTIMER @ 0= END ;
 2|SUBR NOTEANIM INVADERNUM LDA, RLC,
 3|RLC, A E MOV, 0 D MVI, 0 ANIMSTATE H LXI, D DAD, XCHG,
 4|MASTERY LHLD, DMASTERY LBCD, 7 B BIT, 0=, IF, B DAD, THEN,
 5|XCHG, DI, E M MOV, H INX, D M MOV, H INX, MASTERX LDED,
 6|E M MOV, H INX, D M MOV, EI, RET,
 7|SUBR GETASTATE C A MOV, RLC, RLC, A E MOV, 0 D MVI,
 8|0 ANIMSTATE H LXI, D DAD, M E MOV, H INX, M D MOV, H INX,
 9|H PUSH, C A MOV, CALCINVY CALL, D DAD, D POP, H PUSH, D PUSH,
10|1 H BIT, NORMLP1 LHLD, 0< >, IF, 10 D LXI, D DAD, THEN,
11|C A MOV, 7 ANI, RLC, A E MOV, 0 D MVI, D DAD,
12|M E MOV, H INX, M D MOV, D PUSH, Y POPX,
13|C A MOV, H POP, M E MOV, H INX, M D MOV, CALCINVX CALL, D DAD,
14|XCHG, H POP, RET,
15|-->
 +--------Block    215--------
 0|( RECOMPUTE LIMITS ) HEX
 1|F= LLFL F= LLFND F= ULFL F= ULFND
 2|SUBR RELMT <ASSEMBLE INVADERSLEFT LDA, A ANA,
 3|RZ, INIVLL H LXI, 0 ALIVEBITS D LXI, 800 B LXI,
 4|LABEL LLFL D LDAX, A ANA, LLFND JRNZ, H A MOV, 10 SUI,
 5|A H MOV, D INX, C A MOV, 8 ADI, A C MOV, LLFL DJNZ, RET,
 6|LABEL LLFND INVLL SHLD, 7 ALIVEBITS D LXI, INIVUL H LXI,
 7|C A MOV, LEFTINVN STA, 38 C MVI,
 8|LABEL ULFL D LDAX, A ANA, ULFND JRNZ, H A MOV, 10 ADI,
 9|A H MOV, D DCX, C A MOV, 8 SUI, A C MOV, ULFL JMPR,
10|LABEL ULFND INVUL SHLD, C A MOV, RIGHTINVN STA, RET, ASSEMBLE>
11|DECIMAL -->
12|
13|
14|
15|
 +--------Block    216--------
 0|( SUBR TO STEP MASTER COORDS ONE TICK AND LIMIT CHECK ) HEX
 1|( ROUTINE TO WRITE ONE INVADER, IF POSSIBLE )
 2|F= INVFIND F= STEPMASTER F= NONERD
 3|SUBR WRITEINVADER <ASSEMBLE
 4|INVADERNUM LDA, A C MOV, XALIVEBITS CALL, NONERD JRZ,
 5|DI, XRACKBITS CALL, INVFIND JNZ, NOTEANIM CALL, EI,
 6|LABEL NONERD INVADERNUM LDA, A INR, 3F ANI, INVADERNUM STA,
 7|WRITEINVADER JRNZ,
 8|LABEL STEPMASTER BUYMASTERROUTINE LHLD, PCHL,
 9|-->
10|
11|
12|
13|
14|
15|
```

```
 +--------Block    217--------
 0|( WE FOUND AN INVADER - WRITE HIM )
 1|LABEL INVFIND INVADERNUM LDA, CALCINVY CALL, MASTERY LDED,
 2|D DAD, XCHG, INVADERNUM LDA, 7 ANI, RLC, A C MOV,
 3|0 B MVI, INVPATAB LHLD, B DAD, M C MOV, H INX, M B MOV,
 4|B PUSH, XTIY, 20 B LXI, 1 D BIT, 0<>, IF, 1 Y H LDX, H DCR,
 5|0 L MVI, D DAD, XCHG, A0 C MVI, THEN, B PUSH,
 6|INVADERNUM LDA,
 7|CALCINVX CALL, MASTERX LBCD, B DAD, XCHG, B POP, relabs CALL,
 8|DI, writep CALL, NOTEANIM CALL, Y POPX, INVADERNUM LDA,
 9|A INR, 3F ANI, INVADERNUM STA, STEPMASTER JZ, RET,
10|ASSEMBLE>
11|CODE WRTINV RACKTIMER LDA, A ANA, 0=, IF, B PUSH, X PUSHX,
12|WRITEINVADER CALL, X POPX, B POP, THEN, NEXT
13|DECIMAL -->
14|
15|
 +--------Block    218--------
 0|( REWRITE A RACK MEMBER USING NORMAL PATTERNS )
 1|HEX
 2|( USED FOR GAME INITIALIZATION )
 3|F= TOGGLEMEMBER
 4|SUBR REWRITEMEMBER <ASSEMBLE XRACKBITS CALL, RZ,
 5|LABEL TOGGLEMEMBER
 6|GETASTATE CALL,
 7|20 B LXI, relabs CALL, DI, writep CALL, EI, RET,
 8|ASSEMBLE>
 9|CODE REWRITER H POP, B PUSH, Y PUSHX, L C MOV,
10|REWRITEMEMBER CALL, Y POPX, B POP, NEXT
11|DECIMAL -->
12|
13|
14|
15|
 +--------Block    219--------
 0|( REENTER RACK ) HEX
 1|SUBR PLOTREENTRY
 2|VRACK X C LDX, GETASTATE CALL,
 3|0 B MVI, H VYH X STX,
 4|28 VXZW X MVIX,
 5|XCHG, VXL X A LDX, 0C0 ANI, A E MOV,
 6|VXH X D LDX, A ANA, D DSBC,
 7|0<>, IF, H A MOV, A ANA, 0<, IF, -40 H LXI, ELSE, 40 H LXI,
 8|THEN, D DAD, L VXL X STX, H VXH X STX, B INR, THEN,
 9|B A MOV, 6 ANA, 0=, IF,
10|Y PUSHX, M POP, L VPATL X STX, H VPATH X STX,
11|PQSRH PQS X MOVX, VRACK X C LDX, XRACKBITS CALL,
12|B A MOV, M ORA, A M MOV, 20 VMAGIC X MVIX,
13|THEN, RET,
14|-->
15|
```

```
 +--------Block    220--------
 0|( INTERRUPT ROUTINE TO REENTER A GALAXIAN ) DECIMAL
 1|F= ROGER
 2|SUBR RENTGAL <ASSEMBLE
 3|TBCALC CALL, B PUSH,
 4|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE, PQSDE PQS X RESX,
 5|THEN, aup CALL, B POP,
 6|PQSRH PQS X BITX, 0< >, IF,
 7|LABEL ROGER B PUSH, PLOTREENTRY CALL, B POP,
 8|PQSRH PQS X BITX, 0< >, IF, C DCR, ROGER JRNZ, THEN,
 9|THEN, PQSDW PQS X BITX, 0=, IF, vwrite CALL, ELSE,
10|PQSRH PQS X BITX, 0< >, IF, PQSDW PQS X RESX,
11|PQSDE PQS X SETX, THEN, THEN,
12|KILLOFF JMP, ASSEMBLE>
13|-->
14|
15|
 +--------Block    221--------
 0|( CHECK FOR INTERCEPT WITH RACK MEMBER )
 1|( RETURNS NZ, C=INVADERNUM IF DETECTED, ELSE Z )
 2|F= NORKI
 3|HEX SUBR RACKCHECK <ASSEMBLE
 4|VXL X L LDX, VXH X H LDX, H INR, H INR,
 5|MASTERX LDED, A ANA, D DSBC, H A MOV, A ANA, NORKI JM,
 6|RRC, RRC, 7 ANI, A C MOV, VYL X L LDX, VYH X H LDX,
 7|MASTERY LDED, A ANA, D DSBC, H A MOV, 2 ADI, RRC,
 8|38 ANI, C ORA, A C MOV, XRACKBITS CALL, RET,
 9|LABEL NORKI A XRA, RET,
10|ASSEMBLE>
11|DECIMAL -->
12|
13|
14|
15|
 +--------Block    222--------
 0|( ANIMATION LIST AND ROUTINE TO EXPLODE THE FIREBASE )
 1|HEX DATA FBEXPSUB 0 0 SETDC
 2|4 6 DISPL FBEXP1 SETP 08 SWAIT A0 SETM 08 SWAIT
 3|20 SETM -4 -6 DISPL FBEXP4 SETP 08 SWAIT A0 SETM 08 SWAIT
 4|20 SETM 1 2 DISPL FBEXP2 SETP 08 SWAIT A0 SETM 08 SWAIT
 5|20 SETM -1 -2 DISPL FBEXP3 SETP 8 SWAIT A0 SETM 8 SWAIT 20 SETM
 6|ARET
 7|DATA FBEXP FBEXPSUB ACALL FBEXP5 SETP 8 SWAIT A0 SETM 8 SWAIT
 8|20 SETM FBEXP6 SETP 8 SWAIT A0 SETM 8 SWAIT NULPAT SETP
 9|1 SWAIT AWAIT
10|( ROUTINE TO EFFECT THE EXPLOSION )
11|SUBR EXPLODEFB PLAYERHIT LDA, A ANA, 0=, IF,
12|1 A MVI, PLAYERHIT STA, FBEXP H LXI,
13|CRASHA CALL, XWRITE H LXI, L PORL X STX, H PORH X STX, THEN,
14|RET, DECIMAL -->
15|
```

```
  +--------Block    223--------
 0|( SCORIN ) HEX TABLE ASTBL 60 , 60 , 80 , 100 , 300 , 200 ,
 1|500 , 500 ,
 2|DECIMAL DATA EXPISUB EXPLOSION1 SETP 5 SWAIT EXPLOSION2 SETP
 3|5 SWAIT EXPLOSION3 SETP 5 SWAIT EXPLOSION4 SETP 5 SWAIT
 4|EXPLOSION5 SETP 5 SWAIT NULPAT SETP ARET
 5|DATA EXPINV EXPISUB ACALL AHALT
 6|DATA EXPNS EXPISUB ACALL NUMWRITE SETR 1 SWAIT ARET
 7|DATA EXPNF 1 SWAIT HEX BF 40 SETS 1C SWAIT BF 0 SETS AHALT
 8|DECIMAL -->
 9|
10|
11|
12|
13|
14|
15|
  +--------Block    224--------
 0|( MORE SCORING GOODIES )
 1|DATA SCR60 EXPNS ACALL 0 ASTBL SETP EXPNF AJMP
 2|DATA SCR80 EXPNS ACALL 2 ASTBL SETP EXPNF AJMP
 3|DATA SCR100 EXPNS ACALL 3 ASTBL SETP EXPNF AJMP
 4|DATA SCR300 EXPNS ACALL 4 ASTBL SETP EXPNF AJMP
 5|DATA SCR200 EXPNS ACALL 5 ASTBL SETP EXPNF AJMP
 6|DATA SCR500 EXPNS ACALL 6 ASTBL SETP EXPNF AJMP
 7|HEX DATA EXPTHEGORF FBEXPSUB ACALL FBEXP5 SETP 40 SWAIT
 8|A0 SETM 40 SWAIT 20 SETM FBEXP6 SETP 40 SWAIT A0 SETM 40 SWAIT
 9|20 SETM NULPAT SETP NUMWRITE SETR 1 SWAIT 7 ASTBL SETP
10|EXPNF AJMP
11|DATA ATTACKEXPTBL SCR60 , SCR60 , SCR80 , SCR100 , SCR300 ,
12|SCR200 , SCR500 , EXPTHEGORF , DECIMAL -->
13|
14|
15|
  +--------Block    225--------
 0|( BACKGROUND PHASOR INTERCEPT PROCESSING ROUTINES )
 1|( ROUTINE TO EXPLODE AN INVADER )
 2|HEX SUBR PINTERPROC A DCR, 0=, IF, PINTERN LDA, 7 ANI, RLC,
 3|A C MOV, 0 B MVI, ATTACKEXPTBL H LXI, B DAD, M C MOV, H INX,
 4|M B MOV, PINTERX LDED, PINTERY LHLD,
 5|ELSE, PINTERN LDA, A C MOV,
 6|DI, XRACKBITS CALL, M XRA, A M MOV, XALIVEBITS CALL, M XRA,
 7|A M MOV, EI, B PUSH, TOGGLEMEMBER CALL, B POP, GETASTATE CALL,
 8|EXPINV B LXI, THEN, D PUSH, H PUSH, B PUSH,
 9|PINTERN LBCD, B PUSH, 0 C BIT, 0=, IF, INVADERSLEFT H LXI,
10|M DCR, THEN, 0A8 B LXI, B PUSH, XYVSTATE JMP,
11|( ROUTINE TO CHECK FOR INTERCEPT )
12|SUBR PINTERCHECK PINTERFLAG LDA, A ANA, RZ, PINTERPROC CALL,
13|RELMT CALL,
14|PINTERFLAG LDED, PINTERN LHLD, A XRA, PINTERFLAG STA, A INR,
15|RET, DECIMAL -->
```

```
 +--------Block    226--------
 0|( ROUTINE TO CALL FROM SCAN LOOP )
 1|CODE PIFCHECK X PUSHX, Y PUSHX, EXX, PINTERCHECK CALL,
 2|Y POPX, X POPX, 0< >, IF, H PUSH, D PUSH, 1 H LXI, ELSE,
 3|0 H LXI, THEN, H PUSH, EXX, NEXT
 4|: PHASORINTERCEPTCHECK PIFCHECK IF 1 = IF 7 AND ASTBL ZP
 5|ELSE 7 AND 2 * RSTBL @ + PZ THEN @
 6|UPDATESCORE THEN ;
 7|DECIMAL -->
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    227--------
 0|( ANIMATION SUBR TO INITIALIZE THE FIRE BASE )
 1|( NEEDS INTERCEPT AND LIMITS SET BEFORE CALL )
 2|HEX DATA PLAYERANIM JOYWRITE SETR 1000 SETYC 0600 SETXC
 3|20 100 SETDC
 4|FOREVER FIREBASE SETP 78 SWAIT EVERFOR
 5|( ROUTINE TO ACTIVATE THE FIREBASE )
 6|: ACTFB FBANIM @ @ 0B2 0 FBVECTOR XVSTART ;
 7|DECIMAL -->
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    228--------
 0|( CHECK FOR PLAYER HIT )
 1|HEX
 2|: PWAIT WTIMER ! BEGIN BMS PHASORINTERCEPTCHECK WTIMER @ 0=
 3|END ;
 4|: PLAYERHITCHECK PLAYERHIT @ IF 1G PHASORINTERCEPTCHECK
 5|96 PWAIT EMUSIC E2MUSIC FBCOUNTER @ 0= IF
 6|1 GAMEOVER ! 1 ENDOFFRAME ! ELSE FBCOUNTER @ 1 - DUP
 7|FBCOUNTER ! EDATPROCESSES @ 100 SWAP SMALBASE 20 WRITEP
 8|PLAYERHIT ZERO PWAIT @ 001F
 9|40 ATTACKTIMER ! ACTFB THEN
10|ELSE INVADERSLEFT @ 0= IF 30 PWAIT 1 ENDOFFRAME !
11|THEN THEN ;
12|DECIMAL
13|-->
14|
15|
```

```
 +--------Block    229--------
 0|( COMMON INITIALIZATION GOODIES )
 1|CODE NULCODE NEXT
 2|: REPAINTRACK 64 0 DO I REWRITER LOOP ;
 3|HEX
 4|: INITMISSIONRAM XDI GRAPHICS
 5|0 1STCLRADDR CLRSIZE FILL
 6|1 MUSICFLAG B! EMUSIC E2MUSIC
 7|1 WRTSYS B!
 8|' NULCODE REINIT ! NULRET FIREACTION ! ;
 9|: STARTGAME 0 P1SCR ZERO 1 P1SCR ZERO 0 P2SCR ZERO 1 P2SCR ZERO
10|3 FBCOUNTER ! PLAYERUP ZERO NPLAYERS ZERO
11|SKILLFACTOR ZERO ;
12|DECIMAL -->
13|
14|
15|
 +--------Block    230--------
 0|( SPECIAL ROUTINE TO MOVE PHASOR BLAST )
 1|( SUBROUTINE TO XAWRITE WITH INTERCEPT CHECKING )
 2|HEX F= PHL
 3|SUBR PHWRITE <ASSEMBLE PQTB X C LDX, 0 PQTB X MVIX,
 4|LABEL PHL B PUSH, VXH X A LDX, A INR, A VXH X STX,
 5|48 CPI, CY~, IF, PQSRH PQS X RESX, THEN,
 6|PQSDE PQS X BITX, 0=, IF, verase CALL, ELSE,
 7|PQSDE PQS X RESX, THEN, PQSRH PQS X BITX, 0< >, IF,
 8|VIWRITE CALL, THEN,
 9|B POP, PQSRH PQS X BITX, 0< >, IF, C DCR, PHL JRNZ,
10|THEN, KILLOFF JMP,
11|ASSEMBLE>
12|DECIMAL -->
13|
14|
15|
 +--------Block    231--------
 0|( START OR RESTART THE PHASOR MOVING )
 1|SUBR DOFIREACT C M MOV, FIREACTION LHLD, PCHL,
 2|HEX SUBR SETPXY
 3|0 FBVECTOR Y LXIX, VXL Y L LDX, VXH Y H LDX, 500 D LXI,
 4|D DAD, L VXL X STX, H VXH X STX, VYL Y L LDX, VYH Y H LDX,
 5|700 D LXI, D DAD, L VYL X STX, H VYH X STX, 60 VATMR X MVIX,
 6|RET,
 7|SUBR SHOOTPHASOR DOFIREACT CALL,
 8|CLRVEC CALL, SETPXY CALL, 0BB PQS X MVIX,
 9|20 VMAGIC X MVIX, 00 VXZH X MVIX, PHWRITE H LXI,
10|L PQRL X STX, H PQRH X STX, PHASINTR LHLD, L VIRL X STX,
11|H VIRH X STX,
12|FBURST H LXI, L VPATL X STX, H VPATH X STX, 1 VDXH X MVIX,
13|STARTVEC CALL, RET,
14|SUBR RESHOOTPHASOR DOFIREACT CALL, SETPXY CALL, RET,
15|DECIMAL -->
```

```
 +--------Block    232--------
 0|( CHECK FIRE SWITCH )
 1|F= FIREBUT
 2|SUBR FIRESWCK <ASSEMBLE
 3|FIRETRACK H LXI, JOYSTICK IN, A C MOV, M XRA, SWFIRE ANI,
 4|RZ, C A MOV, SWFIRE ANI, FIREBUT JRZ, C M MOV, RET,
 5|LABEL FIREBUT PV1 LIXD, DI, PQSRH PQS X BITX, 0<>, IF,
 6|( KICKOUT IF PHASOR EXPLOSION IN PROGRESS )
 7|PQSFRZ PQS X BITX, RNZ, RESHOOTPHASOR CALL,
 8|ELSE, PQS X A LDX, A ANA, RNZ, SHOOTPHASOR CALL, THEN,
 9|EI, IDSCORE H LXI, 0 MUSIC-BARRAY-1 Y LXIX, bmusic JMP,
10|ASSEMBLE>
11|HEX SUBR FSLITE A XRA, PV1 LIXD, PQSRH PQS X BITX, 0=, IF,
12|A INR, THEN, 26 OUT, RET,
13|CODE FIRECHECK X PUSHX, Y PUSHX, EXX, FSLITE CALL,
14|FIRESWCK CALL, EI, EXX, Y POPX, X POPX, NEXT
15|DECIMAL -->
 +--------Block    233--------
 0|( AWAIT THE ARRIVAL OF THE VERTICAL INTERVAL )
 1|HEX
 2|F= WVIL
 3|CODE WVI <ASSEMBLE
 4|DI, 11 A MVI, INMOD OUT,
 5|VERAF IN, A E MOV,
 6|LABEL WVIL VERAF IN, E CMP, WVIL JZ, 0D0 CPI, WVIL JC,
 7|0E0 CPI, WVIL JNC,
 8|8 A MVI, INMOD OUT,
 9|NEXT ASSEMBLE>
10|DECIMAL -->
11|
12|
13|
14|
15|
 +--------Block    234--------
 0|( NEW COLOR ROUTINES )
 1|HEX 8 BA= COLTBL 0 V= TARGETCT
 2|CODE MAKECOLS EXX, B POP, 0 COLTBL H LXI, TARGETCT LDED,
 3|8 B MVI, BEGIN, D LDAX, F8 ANI, C ORA, A M MOV, H INX,
 4|D INX, LOOP, EXX, NEXT
 5|CODE APPROACHL EXX, B POP, 0 COLTBL H LXI, 0 E MVI, 8 B MVI,
 6|BEGIN, M A MOV, 7 ANI, C CMP, 0<>, IF,
 7|C A MOV, 8 ANA, 0=, IF, M DCR, ELSE, M INR, THEN, E INR,
 8|THEN, H INX, LOOP, 0 H LXI, E A MOV, 0 ANA, 0=, IF, H INX,
 9|THEN, H PUSH, EXX, NEXT
10|CODE APPROACHC EXX, 0 COLTBL H LXI, TARGETCT LDED, 800 B LXI,
11|BEGIN, D LDAX, M CMP, 0<>, IF, CY, IF, M DCR, ELSE, M INR,
12|THEN, C INR, THEN, H INX, D INX, LOOP,
13|0 H LXI, C A MOV, 7 ANA, 0=, IF, H INX, THEN, H PUSH, EXX, NEXT
14|DECIMAL -->
15|
```

```
  +--------Block    235--------
 0|( FADE UP/DOWN ROUTINES )
 1|0 V= CWTMR
 2|: DC 0 COLTBL WVI COLOR EI ;
 3|: CWAIT CWTMR @ PWAIT ; : SCT CWTMR ! ;
 4|: STC TARGETCT ' ;
 5|: SC STC 8 0 DO I TARGETCT @ + B@ I COLTBL B! LOOP DC ;
 6|: FDC STC SCT 7 FLOOD CWAIT 7 MAKECOLS DC CWAIT
 7|BEGIN APPROACHC DC CWAIT END ;
 8|: FUC STC SCT 0 FLOOD CWAIT 0 MAKECOLS DC CWAIT
 9|BEGIN APPROACHC DC CWAIT END ;
10|: FDB SCT BEGIN 0 APPROACHL DC CWAIT END 0 WVI FLOOD EI ;
11|: FUW SCT BEGIN 7 APPROACHL DC CWAIT END 7 WVI FLOOD EI ;
12|DECIMAL -->
13|
14|
15|
  +--------Block    236--------
 0|( FORCE FIELD DRAWER ) DECIMAL
 1|TIMER2 C= FFTIMER 0 V= DDXC
 2|192 BARRAY FIELDADR 0 V= DXC 0 V= XC 0 V= FFLAG 0 V= FFBIAS
 3|: INITFF DDXC ! 0 DXC ! 25600 XC ! 0 96 DO
 4|DXC @ DDXC @ + DUP DXC !
 5|XC @ + DUP XC ! 256 / DUP I FIELDADR B! 191 I - FIELDADR B!
 6|-1 +LOOP 0 FFLAG ! ;
 7|HEX F= FFLOOP SUBR FIELDRAW <ASSEMBLE
 8|FFBIAS LHLD, C0 B MVI, H PUSH, 0 FIELDADR H LXI,
 9|LABEL FFLOOP M A MOV, A ANA, XTHL, 0< >, IF, A C MOV,
10|3 ANI, 20 ORI, MAGIC OUT, C A MOV, XCHG, RRC, RRC, 3F ANI,
11|A L MOV, 0 H MVI, D DAD, FF M MVI, H INX, 0 M MVI, XCHG, THEN,
12|50 D LXI, D DAD, XTHL, H INX, FFLOOP DJNZ, H POP, RET,
13|ASSEMBLE>
14|CODE DRAWFIELD EXX, DI, FIELDRAW CALL, EI, EXX, NEXT
15|DECIMAL -->
  +--------Block    237--------
 0|( MORE FORCE FIELD GOODIES )
 1|DECIMAL
 2|: DRAWFF FFLAG @ 0 = IF FFTIMER @ 0 = IF 1 DI FFLAG !
 3|DRAWFIELD EI THEN THEN ;
 4|: ERASEFF FFLAG @ IF 0 DI FFLAG ! DRAWFIELD EI THEN ;
 5|DECIMAL -->
 6|
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
```

```
 +--------Block    238--------
 0|( RADIAL LINE GENERATOR )
 1|0 C= LTMR
 2|1 C= LDXL
 3|2 C= LDXH
 4|3 C= LXL
 5|4 C= LXH
 6|5 C= LDYL
 7|6 C= LDYH
 8|7 C= LYL
 9|8 C= LYH
10|9 C= LXRL
11|10 C= LXRH
12|11 C= LYR
13|12 C= LRTMR
14|13 C= LPXV
15|-->
 +--------Block    239--------
 0|( RADIAL EFFECT VARIABLES )
 1|14 C= LMSIZ 32 C= LACOUNT
 2|LMSIZ LACOUNT * C= LASIZ
 3|LASIZ BA= LARRAY
 4|0 V= LINIT ( INITIALIZE LINE ROUTINE )
 5|0 V= LQUAD ( QUADRANT COUNTER ) 0 V= LRADIAL ( ANGLE )
 6|0 V= XB 0 V= YB ( BIASES )
 7|0 V= XS 0 V= YS 0 V= XF 0 V= YF ( LINE ENDPOINTS )
 8|0 V= SFBX 0 V= SFFX 0 V= SFBY 0 V= SFFY ( SCALE FACTORS )
 9|0 V= XLP 0 V= XLM 0 V= YLP 0 V= YLM ( LIMIT FACTORS )
10|-->
11|
12|
13|
14|
15|
 +--------Block    240--------
 0|( NEAT SUBROUTINES )
 1|SUBR SINE A E MOV, 0 D MVI, 0 sin-table H LXI, D DAD,
 2|M E MOV, RET,
 3|SUBR COSINE A E MOV, 63 A MVI, E SUB, A E MOV, 0 D MVI,
 4|0 sin-table H LXI, D DAD, M E MOV, RET,
 5|( UNSIGNED MULTIPLY )
 6|F= MNOSW F= MMPL F= NOADD
 7|SUBR UMPY <ASSEMBLE H A MOV, A ANA, MNOSW JRZ, XCHG,
 8|LABEL MNOSW L A MOV, 0 H LXI,
 9|LABEL MMPL DAD, NOADD JRNC, D DAD,
10|LABEL NOADD XCHG, H DAD, XCHG, A ANA, MMPL JRNZ, RET,
11|ASSEMBLE>
12|SUBR COMPL H A MOV, CMA, A H MOV, L A MOV, CMA, A L MOV,
13|H INX, RET,
14|-->
15|
```

```
 +--------Block    241--------
 0|( SUBR TO WRITE NEXT PIXEL IN A LINE )
 1|HEX F= TMRZ F= RTZ
 2|SUBR STEPLINE <ASSEMBLE
 3|LTMR X A LDX, A ANA, TMRZ JRZ, A DCR, A LTMR X STX,
 4|LDXL X E LDX, LDXH X D LDX, LXL X L LDX, LXH X H LDX, D DAD,
 5|L LXL X STX, H LXH X STX, XCHG,
 6|LDYL X C LDX, LDYH X B LDX, LYL X L LDX, LYH X H LDX, B DAD,
 7|L LYL X STX, H LYH X STX,
 8|20 C MVI, relabel CALL, C A MOV, DI, MAGIC OUT, 0C0 M MVI, EI,
 9|RET, LABEL TMRZ LRTMR X A LDX, A ANA, RTZ JRZ,
10|A LTMR X STX, 0 LRTMR X MVIX,
11|LXRL X L LDX, LXRH X H LDX, L LXL X STX, H LXH X STX,
12|LYR X A LDX, A LYH X STX, 0 LYL X MVIX, RET,
13|LABEL RTZ LINIT LHLD, PCHL,
14|ASSEMBLE>
15|DECIMAL -->
 +--------Block    242--------
 0|( OTHER NEAT VERBS )
 1|: LSTART LINIT ! LASIZ 0 DO 0 I LARRAY B! LOOP ;
 2|F= UPAL
 3|CODE UPDATEALL <ASSEMBLE X PUSHX, Y PUSHX, EXX, LACOUNT B LXI,
 4|0 LARRAY X LXIX,
 5|LABEL UPAL B PUSH, STEPLINE CALL,
 6|LMSIZ D LXI, D DADX,
 7|B POP, B DCX, C A MOV, B ORA, UPAL JRNZ,
 8|EXX, Y POPX, X POPX, NEXT ASSEMBLE>
 9|-->
10|
11|
12|
13|
14|
15|
 +--------Block    243--------
 0|( GENERATE A LINE )
 1|HEX F= DOY F= OKX
 2|SUBR GENLINE <ASSEMBLE random CALL, L A MOV, 3 ANI, LQUAD STA,
 3|D A MOV, 3F ANI, LRADIAL STA,
 4|( X START )
 5|SINE CALL, SFBX LHLD, UMPY CALL, H A MOV, XS STA,
 6|( X END )
 7|LRADIAL LDA, SINE CALL, SFFX LHLD, UMPY CALL,
 8|H A MOV, XF STA,
 9|( Y START )
10|LRADIAL LDA, COSINE CALL, SFBY LHLD, UMPY CALL,
11|H A MOV, YS STA,
12|( Y END )
13|LRADIAL LDA, COSINE CALL, SFFY LHLD, UMPY CALL,
14|H A MOV, YF STA,
15|-->
```

```
 +--------Block    244--------
 0|( LINE GENERATOR - CLIP CHECK )
 1|LQUAD LDA, 2 ANI, 0=, IF, XLM LDA, ELSE, XLP LDA, THEN,
 2|A C MOV, XF LDA, C CMP, DOY JRC, ( JUMP IF OK )
 3|B PUSH, YS LDED, YF LDA, E SUB, A E MOV,
 4|XF LDA, C SUB, A L MOV, 0 H MVI, H D MOV, UMPY CALL,
 5|XS LDED, XF LDA, E SUB, A E MOV, 0 D MVI, UNSDIV CALL,
 6|YF LDA, E SUB, YF STA, B POP, C A MOV, XF STA,
 7|( Y STUFF )
 8|LABEL DOY LQUAD LDA, A INR, 3 ANI, 2 CPI, CY, IF, YLM LDA,
 9|ELSE, YLP LDA, THEN, A C MOV, YF LDA, C CMP, OKX JRC,
10|B PUSH, XS LDED, XF LDA, E SUB, A E MOV,
11|YF LDA, C SUB, A L MOV, 0 H MVI, UMPY CALL,
12|YS LDED, YF LDA, E SUB, A E MOV, 0 D MVI, UNSDIV CALL,
13|XF LDA, E SUB, XF STA, B POP, C A MOV, YF STA,
14|-->
15|
 +--------Block    245--------
 0|( LINE GENERATOR - SET DELTAS )
 1|LABEL OKX XS LDED, XF LDA, E SUB, A C MOV, YS LDED, YF LDA,
 2|E SUB, A B MOV, C CMP, CY, IF, ( X IS LARGER )
 3|C LRTMR X STX, C LTMR X STX, 40 LDXL X MVIX, 0 LDXH X MVIX,
 4|B H MOV, 0 L MVI, C E MOV, L D MOV, UNSDIV CALL,
 5|E LDYL X STX, D LDYH X STX,
 6|ELSE, ( Y IS LARGER )
 7|B LRTMR X STX, B LTMR X STX, 0 LDYL X MVIX,
 8|1 LDYH X MVIX, C A MOV, RRC, RRC, A H MOV, 0C0 ANI, A L MOV,
 9|H A MOV, 3F ANI, A H MOV, B E MOV, 0 D MVI, UNSDIV CALL,
10|E LDXL X STX, D LDXH X STX,
11|THEN,
12|-->
13|
14|
15|
 +--------Block    246--------
 0|( ADJUST DELTAS TO QUADRANT, AND BIAS TO EFFECT CENTER )
 1|XS LDA, RRC, RRC, A D MOV, 0C0 ANI, A E MOV, D A MOV,
 2|3F ANI, A D MOV, XB LHLD,
 3|LQUAD LDA,
 4|2 ANI, 0< >, IF, A ANA, D DSBC, XCHG, LDXL X L LDX,
 5|LDXH X H LDX, COMHL CALL, L LDXL X STX, H LDXH X STX,
 6|XCHG, ELSE, D DAD,
 7|THEN, L LXL X STX, H LXH X STX, L LXRL X STX, H LXRH X STX,
 8|YS LDA, A D MOV, ^ E MVI, YB LHLD,
 9|LQUAD LDA, A INR, 3 ANI, 2 CPI, CY~, IF, A ANA, D DSBC,
10|XCHG, LDYL X L LDX, LDYH X H LDX, COMHL CALL,
11|L LDYL X STX, H LDYH X STX, XCHG,
12|ELSE, D DAD, THEN,
13|0 LYL X MVIX, H LYH X STX, H LYR X STX,
14|RET, ASSEMBLE> .NCFS
15|DECIMAL -->
```

```
 +--------Block    247--------
 0|( SET CENTER OF LINE EFFECT )
 1|: SETLXY 2DUP YB ! XB !
 2|256 / DUP YLP ! 192 SWAP - YLM !
 3|64 / DUP DUP 255 > IF DROP 255 THEN XLP !
 4|292 SWAP - DUP 255 > IF DROP 255 THEN XLM ! ;
 5|: SETSF SFFY ! SFFX ! SFBY ! SFBX ! ;
 6|DECIMAL -->
 7|
 8|
 9|
10|
11|
12|
13|
14|
15|
 +--------Block    248--------
 0|( CHECK FOR INTERCEPT WITH ANY OF THE ATTACKERS )
 1|F= CNH
 2|SUBR CKATRS <ASSEMBLE PINTERFLAG LDA, A ANA, CNH JRNZ,
 3|1 C MVI, CHECKALL CALL, CNH JRZ,
 4|PQSRH PQS Y RESX, PQSDW PQS Y SETX,
 5|VYL Y L LDX, VYH Y H LDX, PINTERY SHLD,
 6|VXL Y L LDX, VXH Y H LDX, PINTERX SHLD,
 7|VRACK Y A LDX,
 8|A A BIT, 0= IF, A C MOV, XALIVEBITS CALL, M XRA,
 9|A M MOV, THEN, 1 A MVI, PINTERFLAG STA, A ANA,
10|RET,
11|LABEL CNH A XRA, RET,
12|ASSEMBLE>
13|DECIMAL ;S
14|
15|
```