```
// Main Maxx Steele robot internal ROM
// located at $E000-$FFFF
// comments by R. wind 9-14-2002, updated 3-15-2006
// maxxbot@yahoo.com

// Zero page is RAM from $0000 to $00FF
// Warm start characters in RAM from $0100 to $0103
// Stack extends down in RAM from $01FF
// Program RAM from $0200 to $03FE, 2 bytes per command
// Music RAM is from $0400 to 04FF, 2 bytes per note
// Speech phrase RAM is at $0500
// IO is memory mapped:
//   $1000
//   $1200 display shift register
//   $1400 speech
//   $1600
//   $1C00 motors?
//   $1E00
// Cartridge ROM can start at $2000 to $B000 in $1000 increments
// Internal ROM is from $E000 to $FFFF

00000000:       28      (
00000001:       63      c
00000002:       2920    )
00000004:       3139    19
00000006:       38      8
00000007:       34      4
00000008:       204342  CB
0000000B:       53      S
0000000C:       20546F  TO
0000000F:       7973    YS

00000011:       6C7600  JMP ($0076)     NMI jumps to here

00000014:       48      PHA             initial NMI vector points here
00000015:       A900    LDA #$00
00000017:       8D001E  STA $1E00
0000001A:       68      PLA
0000001B:       40      RTI

0000001C:       8200    table gets copied to RAM @ $0072
0000001E:       0080
00000020:       14E0    at $0076 initially NMI vector
00000022:       C8FD    at $0078 initially IRQ vector
00000024:       CBE0    at $007A initially immediate mode vector

00000026:       D8D8    at $007C initially LO byte              F1D8 music
00000028:       D8DF                                            F1D8 music
0000002A:       94CF    IRQ jump table                          F1D8 music
0000002C:       DF82                                            FDDF nothing
                                                                EE94 timers
0000002E:       F1F1    at $0084 initially HO byte              EECF
00000030:       F1FD                                            FDDF nothing
00000032:       EEEE    IRQ jump table                          F482 speech
00000034:       FDF4

00000036:       32EE    at $008C initially
00000038:       0010    at $008E initially      cartridge ROM vector
0000003A:       98E5    at $0090 initially
0000003C:       D8F3    at $0092 initially      sound
0000003E:       CEF8    at $0094 initially      games vector
00000040:       FF
```

```
Reset
00000041:       78      SEI             RESET jumps to here
00000042:       A200    LDX #$00
00000044:       8E001E  STX $1E00       initialize memory mapped IO
00000047:       8E0012  STX $1200
0000004A:       D8      CLD
0000004B:       CA      DEX
0000004C:       9A      TXS             init stack pointer to $FF
0000004D:       A011    LDY #$11
0000004F:       88      DEY
00000050:       301D    BMI $1D         to $E06F (warm start)
00000052:       B900E0  LDA $E000,Y
00000055:       D90001  CMP $0100,Y     check RAM for copyright match to ROM
00000058:       F0F5    BEQ $F5         loop back to $E04F

here if no match to RAM
(cold start)
0000005A:       A000    LDY #$00        to here if mismatch on copyright
0000005C:       8A      TXA             accum = $FF
0000005D:       E8      INX             X=0
0000005E:       9400    STY $00,X       set $0000 to $00FF to 0 - zero-out zeropage
00000060:       9D0005  STA $0500,X     set $0500 to $05FF to $FF phrase table
00000063:       CA      DEX
00000064:       D0F8    BNE $F8         loop back to $E05E
00000066:       2055EF  JSR $EF55       zero $0400 to $04ff music table, also $3B
and $39
00000069:       2082E5  JSR $E582       init program pointers and flags (cold start)
0000006C:       207BE5  JSR $E57B       set a RAM byte to $FF (again)

here if 1st 16 bytes of RAM at $0100 match 1st bytes of ROM at $E000
(warm start)
0000006F:       A210    LDX #$10        to here if copyright in RAM OK
00000071:       BD00E0  LDA $E000,X
00000074:       9D0001  STA $0100,X     copy copyright from ROM to RAM
00000077:       CA      DEX
00000078:       10F7    BPL $F7         loop back to $E071
0000007A:       8667    STX $67         set $67 to $FF
0000007C:       866A    STX $6A         set $6A to $FF
0000007E:       A9F8    LDA #$F8
00000080:       8D0012  STA $1200
00000083:       A224    LDX #$24
00000085:       BD1CE0  LDA $E01C,X     copy tables to RAM (address vectors)
00000088:       9572    STA $72,X       at $72 to $96
0000008A:       CA      DEX
0000008B:       10F8    BPL $F8         to $E085
0000008D:       20EEEE  JSR $EEEE       init timer bytes
00000090:       A005    LDY #$05        start search for cartridge ROM
00000092:       18      CLC
00000093:       A58F    LDA $8F         initially = $10
00000095:       6910    ADC #$10        point to next 4kbyte boundary
00000097:       858F    STA $8F         check for ROMs from $2000 thru $B000
00000099:       C9C0    CMP #$C0
0000009B:       F019    BEQ $19         to $E0B6 if no valid ROM @ < $C000
0000009D:       B18E    LDA ($8E),Y
0000009F:       D9FEDF  CMP $DFFE,Y     compare ROM copyright to bytes 2 thru 5
000000A2:       D0EC    BNE $EC         to $E08E next segment if mismatch
000000A4:       88      DEY
000000A5:       C001    CPY #$01
000000A7:       D0F4    BNE $F4         loop to $E09D until 4 bytes of copyright
compared
000000A9:       B18E    LDA ($8E),Y     here if copyright matched
```

```
000000AB:       AA      TAX
000000AC:       88      DEY
000000AD:       B18E    LDA ($8E),Y
000000AF:       858E    STA $8E         use bytes 0 and 1 of cart as jump vector
000000B1:       868F    STX $8F
000000B3:       6C8E00  JMP ($008E)     jump through cartridge ROM vector


(no ROM cartridge or back from cartridge)
000000B6:       A9E6    LDA #$E6        to here if no cartridge ROM detected or
                                        from cartridge after table setup        "9"
000000B8:       204FED  JSR $ED4F       shift to $1200 display
000000BB:       A930    LDA #$30
000000BD:       207BED  JSR $ED7B       send to $1200
000000C0:       8506    STA $06
000000C2:       58      CLI
000000C3:       A210    LDX #$10        if bit 1 of $03 = 0,
000000C5:       2075F4  JSR $F475       say "Hello, I'm Maxx Steele."
000000C8:       20AFED  JSR $EDAF       check for stalled motors
000000CB:       58      CLI
000000CC:       A980    LDA #$80
000000CE:       8575    STA $75
000000D0:       0A      ASL A           A = mode 0 = immediate

000000D1:       850D    STA $0D         set mode $0D = A
000000D3:       AA      TAX
000000D4:       E8      INX             Immediate mode tune 1
000000D5:       2001EF  JSR $EF01       get pointers for tune
000000D8:       A200    LDX #$00
000000DA:       8600    STX $00
000000DC:       865B    STX $5B         zero speech flag
000000DE:       860E    STX $0E
000000E0:       8601    STX $01         zero these loc's
000000E2:       8673    STX $73
000000E4:       8674    STX $74
000000E6:       863A    STX $3A
000000E8:       CA      DEX
000000E9:       9A      TXS             reset stack pointer to $FF
000000EA:       2005E9  JSR $E905
000000ED:       A50D    LDA $0D         0 to 5 mode
000000EF:       F063    BEQ $63         to $E154 immediate mode if $0D = 0
000000F1:       18      CLC
000000F2:       6905    ADC #$05
000000F4:       AA      TAX             "LErn" "Prog" "run" "PLAy"
000000F5:       2084F6  JSR $F684       serial out to $1200 display
000000F8:       A60D    LDX $0D         mode
000000FA:       CA      DEX
000000FB:       D025    BNE $25         to $E122 past learn mode if $0D <> 1

(learn mode)
000000FD:       20ECE3  JSR $E3EC       here if $0D = 1 (learn mode)
00000100:       20A4E6  JSR $E6A4
00000103:       E010    CPX #$10
00000105:       D003    BNE $03         to $E10A if X <> #$10
00000107:       4C76EF  JMP $EF76       leave learn mode?
0000010A:       A52B    LDA $2B
0000010C:       D0F2    BNE $F2         to $E100
0000010E:       A50A    LDA $0A
00000110:       2901    AND #$01
00000112:       8507    STA $07
00000114:       2074E5  JSR $E574       init $0F,$10 and set $200 = $FF
00000117:       204EF6  JSR $F64E
```

```
0000011A:       A21E    LDX #$1E        if bit 1 of $03 = 0,    (SPon)
0000011C:       2075F4  JSR $F475       say "Please teach me."
0000011F:       4C61E1  JMP $E161       to learn mode

(not learn mode)
00000122:       CA      DEX
00000123:       D023    BNE $23         to $E148 past program mode if $0D <> 2
00000125:       20ECE3  JSR $E3EC       here if $0D = 2 (program mode)
00000128:       20A4E6  JSR $E6A4
0000012B:       E010    CPX #$10
0000012D:       D003    BNE $03         to $E132
0000012F:       4CE0EF  JMP $EFE0
00000132:       E012    CPX #$12
00000134:       D003    BNE $03         to $E139
00000136:       4C78F3  JMP $F378
00000139:       A52B    LDA $2B
0000013B:       D0EB    BNE $EB         to $E128
0000013D:       2074E5  JSR $E574
00000140:       A21F    LDX #$1F        if bit 1 of $03 = 0,    (SPon)
00000142:       2075F4  JSR $F475       say "Please program me."
00000145:       4C46E3  JMP $E346       to program mode

(not program mode)
00000148:       CA      DEX
00000149:       D006    BNE $06         to $E151(game mode)if $0D <> 3

(execute)
0000014B:       2063EF  JSR $EF63       here if $0D = 3 execute
                                        wait for loc $2B to be zero
0000014E:       4C34E4  JMP $E434       mode 3 (execute) function

(game mode)
00000151:       6C9400  JMP ($0094)     initially to $F8CE  games


(immediate mode)
00000154:       2063EF  JSR $EF63       here if immediate mode
                                        wait for loc $2B to be zero
00000157:       A220    LDX #$20
00000159:       2075F4  JSR $F475       if bit 1 of $03 = 0, say "I'm ready."
0000015C:       A9E0    LDA #$E0        "L"
0000015E:       204FED  JSR $ED4F       shift to $1200 display

(learn mode)
00000161:       A900    LDA #$00        here if learn mode
00000163:       850E    STA $0E
00000165:       2017E6  JSR $E617       get keypad key
00000168:       A515    LDA $15         keypad character?
0000016A:       A60D    LDX $0D         mode
0000016C:       F00D    BEQ $0D         to $E17B  if immediate mode
0000016E:       C90E    CMP #$0E        CLEAR key pressed?
00000170:       D00D    BNE $0D         to $E17F if $15 <> $0E
00000172:       20A9E3  JSR $E3A9       yes, do clear function
00000175:       204EF6  JSR $F64E       program step number to display
00000178:       4C61E1  JMP $E161       back to start of learn mode

here if immediate mode (keypad char in A)
0000017B:       C90C    CMP #$0C        WAIT key?
0000017D:       F02B    BEQ $2B         to $E1AA if $15 = #$0C
here if A <> $0C
0000017F:       A60E    LDX $0E
00000181:       D039    BNE $39         to $E1BC        if $0E <> 0
00000183:       C92C    CMP #$2C
```

```
00000185:       D004    BNE $04         to $E189 if $15 = #$2C
00000187:       A91C    LDA #$1C
00000189:       D029    BNE $29         to $E1B4 always
0000018B:       C929    CMP #$29
0000018D:       10D2    BPL $D2         to $E161 if >= $29
0000018F:       C920    CMP #$20
00000191:       3006    BMI $06         to $E199 if < $20
00000193:       290F    AND #$0F
00000195:       0910    ORA #$10
00000197:       D01B    BNE $1B         to $E1B4 always
00000199:       C910    CMP #$10
0000019B:       101F    BPL $1F         to $E1BC if >= $10
0000019D:       C90F    CMP #$0F
0000019F:       D005    BNE $05         to $E1A6 if <> $0F
= $0F
000001A1:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000001A4:       30BB    BMI $BB         to $E161
000001A6:       C90D    CMP #$0D
000001A8:       3005    BMI $05         to $E1AF if < $0D
>= $0D
000001AA:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control? stop motors?
000001AD:       30B2    BMI $B2         to $E161
< $0D
000001AF:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000001B2:       A515    LDA $15         key

000001B4:       8511    STA $11         opcode
000001B6:       20CEE6  JSR $E6CE
000001B9:       4C61E1  JMP $E161       back to start of learn mode

000001BC:       A50E    LDA $0E
000001BE:       0980    ORA #$80
000001C0:       8511    STA $11         opcode
000001C2:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000001C5:       20ECE3  JSR $E3EC
000001C8:       A60E    LDX $0E
000001CA:       CA      DEX
000001CB:       D01E    BNE $1E         to $E1EB if $0E <> 1
here if ($0E) = 1
000001CD:       2095E5  JSR $E595
000001D0:       B00B    BCS $0B         to $E1DD
000001D2:       A515    LDA $15
000001D4:       C90A    CMP #$0A
000001D6:       10A7    BPL $A7         back to $E17F if $15 >= #$0A
here if keycode <= $0A
000001D8:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control? stop motors?
000001DB:       30F0    BMI $F0         to $E1CD
000001DD:       A613    LDX $13         operand?
000001DF:       2001EF  JSR $EF01       get pointers for tune # in operand

000001E2:       20CBE2  JSR $E2CB
000001E5:       A50D    LDA $0D         mode
000001E7:       F054    BEQ $54         to $E23D if immediate mode
000001E9:       D0CE    BNE $CE         to $E1B9 if not immediate mode

000001EB:       CA      DEX
000001EC:       F003    BEQ $03         to $E1F1 if ($0E) = 2
000001EE:       CA      DEX
000001EF:       D016    BNE $16         to $E207
here if ($0E) = 3
```

```
000001F1:       2095E5  JSR $E595
000001F4:       B00B    BCS $0B         to $E201
000001F6:       A515    LDA $15         keycode?
000001F8:       C90A    CMP #$0A
000001FA:       10DA    BPL $DA         back to $E1D6 if $15 >= #$0A
A = motor command keycode
000001FC:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control? stop motors?
000001FF:       30F0    BMI $F0         back to $E1F1
00000201:       204BF4  JSR $F44B
00000204:       4CE2E1  JMP $E1E2       loop back

00000207:       A50D    LDA $0D
00000209:       D032    BNE $32         to $E23D if not immediate mode
0000020B:       CA      DEX             here if immediate mode
0000020C:       D07B    BNE $7B         to $E289
here if  ($0E) = 4
0000020E:       200DE6  JSR $E60D       wait for key < $20
00000211:       C910    CMP #$10
00000213:       10C1    BPL $C1         to $E1D6 if key >= #$10
00000215:       C907    CMP #$07
00000217:       3005    BMI $05         to $E21E if key < 7
00000219:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control? stop motors?
0000021C:       30F0    BMI $F0         to $E20E
0000021E:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
00000221:       A515    LDA $15         key code
00000223:       0A      ASL A           x2                                  ($15)
00000224:       18      CLC                                                  00  SEr
00000225:       6913    ADC #$13        +$13                                 01 Pdon
00000227:       AA      TAX                                                  02 Edof
00000228:       A415    LDY $15                                              03 UCon
0000022A:       D014    BNE $14         to $E240 if key <> 0                 04 Ebon
Y=0 key = SEr, PAr                                                           05 Spon
0000022C:       A502    LDA $02                                              06 manual
powerdown
0000022E:       4980    EOR #$80        invert bit 7 of $02
00000230:       8502    STA $02

00000232:       2980    AND #$80
00000234:       F001    BEQ $01         to $E237
00000236:       E8      INX
00000237:       2084F6  JSR $F684       send bytes to $1200 display
0000023A:       20ECE3  JSR $E3EC       wait for one of two ram locs to change
0000023D:       4C5CE1  JMP $E15C       to learn mode
Y=1 key = Pdon, Pdof
00000240:       88      DEY
00000241:       D009    BNE $09         to $E24C
00000243:       A503    LDA $03
00000245:       4980    EOR #$80        invert bit 7 of $03
00000247:       8503    STA $03
00000249:       4C32E2  JMP $E232
Y=2 key = Edon, Edof
0000024C:       88      DEY
0000024D:       D00A    BNE $0A         to $E259
0000024F:       A503    LDA $03
00000251:       4940    EOR #$40        toggle bit 4 of $03
00000253:       8503    STA $03
00000255:       0A      ASL A           x2
00000256:       4C32E2  JMP $E232

y=3 key = UCon, UCof
00000259:       88      DEY
0000025A:       D013    BNE $13         to $E26F
```

```
0000025C:       A503    LDA $03
0000025E:       4901    EOR #$01        toggle bit 0 of $03
00000260:       8503    STA $03
00000262:       4A      LSR A
00000263:       9001    BCC $01         to $E266
00000265:       E8      INX
00000266:       2084F6  JSR $F684       serial out to $1200 display
00000269:       2005E9  JSR $E905
0000026C:       4C3AE2  JMP $E23A
Y=4 key = Ebon, Ebof
0000026F:       88      DEY
00000270:       D00B    BNE $0B         to $E27D
00000272:       A502    LDA $02
00000274:       4902    EOR #$02        toggle bit 1 of $02     (Ebon, Ebof)
00000276:       8502    STA $02
00000278:       2902    AND #$02
0000027A:       4C34E2  JMP $E234
Y=5 key = SPon, SPof
0000027D:       88      DEY
0000027E:       D014    BNE $14         to $E294
00000280:       A503    LDA $03
00000282:       4902    EOR #$02        toggle bit 1 of $03     (SPon, SPof)
00000284:       8503    STA $03
00000286:       4C78E2  JMP $E278
here if ($0E) > 4
00000289:       CA      DEX
0000028A:       D013    BNE $13         to $E29F
here if ($0E) = 5
0000028C:       A204    LDX #$04        "notE"
0000028E:       2084F6  JSR $F684       serial out to $1200 display
00000291:       4C68EF  JMP $EF68
Y=6 key = manual power down
00000294:       A207    LDX #$07
00000296:       2001EF  JSR $EF01       get pointers for power-down tune
00000299:       2063EF  JSR $EF63       wait for loc $2B to be zero
0000029C:       4C78EE  JMP $EE78       power down
here if ($0E) > 5
0000029F:       4C61F7  JMP $F761



000002A2:       A40D    LDY $0D         mode
000002A4:       88      DEY
000002A5:       D01D    BNE $1D         to $E2C4 if mode $0D <> 1  = RTS
000002A7:       A507    LDA $07         here if learn mode
000002A9:       850A    STA $0A
000002AB:       A900    LDA #$00
000002AD:       8513    STA $13         operand
000002AF:       F01F    BEQ $1F         to $E2D0
000002B1:       A40D    LDY $0D
000002B3:       88      DEY
000002B4:       D00E    BNE $0E         to $E2C4 if $0D <> 1, RTS
000002B6:       A07E    LDY #$7E        here if learn mode
000002B8:       A511    LDA $11         opcode
000002BA:       290F    AND #$0F
000002BC:       D10F    CMP ($0F),Y     start at ($0F,10),$7E
000002BE:       F005    BEQ $05         to $E2C5
000002C0:       88      DEY             search downward for match to opcode LO nybl
000002C1:       88      DEY
000002C2:       D0F8    BNE $F8         loop back to $E2BC
000002C4:       60      RTS

000002C5:       C8      INY
```

```
000002C6:       A514    LDA $14         store ($14) in operand byte?
000002C8:       910F    STA ($0F),Y
000002CA:       60      RTS


move program up, add step to beginning ($0F,10) in learn mode
000002CB:       A40D    LDY $0D
000002CD:       88      DEY
000002CE:       D0F4    BNE $F4         to $E2C4 if $0D <> 1, RTS
000002D0:       A502    LDA $02         here if learn mode
000002D2:       0940    ORA #$40        set bit 6 of $02
000002D4:       8502    STA $02
000002D6:       A50F    LDA $0F
000002D8:       8524    STA $24         copy $0F,$10 to $24,$25
000002DA:       A510    LDA $10
000002DC:       8525    STA $25
000002DE:       A07E    LDY #$7E
000002E0:       A903    LDA #$03        check for memory full
000002E2:       C525    CMP $25         compare $24,$25 to #$037E
000002E4:       D019    BNE $19         to $E2FF if memory not full
000002E6:       A97E    LDA #$7E
000002E8:       C524    CMP $24
000002EA:       D013    BNE $13         to $E2FF if memory not full
000002EC:       A21D    LDX #$1D        here if equal to $037E
000002EE:       200FF4  JSR $F40F       say "Sorry, my circuits are full"
000002F1:       A20C    LDX #$0C        "FULL" ($037F is maximum program memory)
000002F3:       2084F6  JSR $F684       serial out to $1200 display
000002F6:       207EF4  JSR $F47E
000002F9:       20ECE3  JSR $E3EC
000002FC:       6C7A00  JMP ($007A)     to $E0CB initially  immediate mode
to here if memory not full
000002FF:       E624    INC $24         point to bytes in pairs
00000301:       E624    INC $24
00000303:       D002    BNE $02         to $E307
00000305:       E625    INC $25
00000307:       B124    LDA ($24),Y
00000309:       C9FF    CMP #$FF        end of program?
0000030B:       D0D3    BNE $D3         to $E230 if not end
here if (#24,25) points to end of program
0000030D:       A07E    LDY #$7E
0000030F:       B124    LDA ($24),Y     move whole program up 2 bytes
00000311:       A080    LDY #$80
00000313:       9124    STA ($24),Y
00000315:       88      DEY
00000316:       B124    LDA ($24),Y
00000318:       A081    LDY #$81
0000031A:       9124    STA ($24),Y
0000031C:       A524    LDA $24
0000031E:       D002    BNE $02         to $E322
00000320:       C625    DEC $25
00000322:       C624    DEC $24         decrement $24,$25 by 2
00000324:       C624    DEC $24
00000326:       A50F    LDA $0F
00000328:       C524    CMP $24
0000032A:       D0E1    BNE $E1         to $E30D if not equal
0000032C:       A510    LDA $10         compare $24,$25 to $0F,$10
0000032E:       C525    CMP $25
00000330:       D0DB    BNE $DB         to $E30D if not done moving
here after moving whole program up 2 bytes
00000332:       A511    LDA $11         opcode
00000334:       88      DEY
00000335:       910F    STA ($0F),Y     put $11 and $13 in program at start
00000337:       C8      INY
```

```
00000338:       A513    LDA $13         operand
0000033A:       910F    STA ($0F),Y
0000033C:       A900    LDA #$00
0000033E:       850E    STA $0E
00000340:       201AE4  JSR $E41A       increment $0F,$10 by 2
00000343:       4C4EF6  JMP $F64E       program step number to display


(program mode)
00000346:       204EF6  JSR $F64E       program step number to display
00000349:       A900    LDA #$00
0000034B:       850E    STA $0E
0000034D:       200DE6  JSR $E60D       wait for key < $20
00000350:       A615    LDX $15
00000352:       E00E    CPX #$0E
00000354:       D005    BNE $05         to $E35B if $15 <> #$0E
here if Clear key
00000356:       20A9E3  JSR $E3A9       here if $15 = #$0E = Clear
00000359:       10EB    BPL $EB         to $E346
here if not Clear key
0000035B:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
0000035E:       A615    LDX $15         keycode
00000360:       E00F    CPX #$0F        enter key?
00000362:       D006    BNE $06         to $E36A if $15 <> #$0F
here if Enter key
00000364:       2009E4  JSR $E409       execute program if $15 = #$0F
00000367:       4C46E3  JMP $E346       loop to top
here if not Enter key
0000036A:       E013    CPX #$13        MOTION/BACKSTEP key?
0000036C:       D006    BNE $06         to $E374 if $15 <> #$13
here if Motion/Back Step key
0000036E:       20F9E3  JSR $E3F9       $15 = #$13
00000371:       4C46E3  JMP $E346       loop to top
here if not Motion/Back Step key
00000374:       BDB5E6  LDA $E6B5,X     get opcode for keycode
00000377:       2940    AND #$40
00000379:       D004    BNE $04         to $E37F don't save X if bit 6 set
0000037B:       8611    STX $11         opcode
0000037D:       F00B    BEQ $0B         to $E38A
0000037F:       A50E    LDA $0E
00000381:       F0C3    BEQ $C3         to $E346
00000383:       0980    ORA #$80
00000385:       8511    STA $11         opcode
00000387:       20ECE3  JSR $E3EC       delay
0000038A:       2095E5  JSR $E595
0000038D:       B00E    BCS $0E         to $E39D
0000038F:       A615    LDX $15
00000391:       BDB5E6  LDA $E6B5,X     get opcode for keycode?
00000394:       2940    AND #$40
00000396:       D0C3    BNE $C3         to $E35B
00000398:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control?
0000039B:       30A9    BMI $A9         to $E346
0000039D:       A502    LDA $02
0000039F:       29BF    AND #$BF        clear bit 6 of $02
000003A1:       8502    STA $02
000003A3:       20D6E2  JSR $E2D6       check if out of program space
000003A6:       4C49E3  JMP $E349       loop back

000003A9:       A500    LDA $00         called if $15 = #$0E = clear key
                                        in program mode
000003AB:       0940    ORA #$40        set bit 6 of $00
000003AD:       8500    STA $00
```

```
000003AF:       A900    LDA #$00
000003B1:       850E    STA $0E
000003B3:       A9F4    LDA #$F4        init timer $28 to #$F4
000003B5:       8528    STA $28         gets decremented in IRQ routine
000003B7:       2055E5  JSR $E555       decrement $0F,$10 program pointer by 2, beg
000003BA:       B005    BCS $05         to $E3C1
000003BC:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000003BF:       302B    BMI $2B         to $E3EC
000003C1:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000003C4:       A50F    LDA $0F
000003C6:       8524    STA $24         copy $0F,$10  to $24,$25
000003C8:       A510    LDA $10
000003CA:       8525    STA $25
000003CC:       E624    INC $24
000003CE:       E624    INC $24         increment $24,$25 by 2
000003D0:       D002    BNE $02
000003D2:       E625    INC $25
000003D4:       A081    LDY #$81
000003D6:       B124    LDA ($24),Y
000003D8:       A07F    LDY #$7F        shift program down 2 bytes
000003DA:       9124    STA ($24),Y
000003DC:       C8      INY
000003DD:       B124    LDA ($24),Y
000003DF:       A07E    LDY #$7E
000003E1:       9124    STA ($24),Y
000003E3:       C9FF    CMP #$FF        loop if not end of program
000003E5:       D0E5    BNE $E5         to $E3CC
000003E7:       A20A    LDX #$0A        "CLr"
000003E9:       2084F6  JSR $F684       serial out to $1200 display

delay
000003EC:       A9F4    LDA #$F4        init timer $2A
000003EE:       852A    STA $2A         gets decremented in IRQ routine
000003F0:       2475    BIT $75         wait for one of two locs to change
000003F2:       1004    BPL $04         to $E3F8 = RTS
000003F4:       A52A    LDA $2A         gets decremented in IRQ routine
000003F6:       D0F8    BNE $F8         to $E3F0
000003F8:       60      RTS

000003F9:       2055E5  JSR $E555       called if keypad key $15 = #$13
in program mode, decrement $0F,$10 program pointer by 2, beg
000003FC:       9006    BCC $06         to $E404
000003FE:       2019E5  JSR $E519       get opcode, operand to $11,$13
00000401:       206CF6  JSR $F66C       execute one program command
00000404:       A933    LDA #$33
00000406:       4C00E6  JMP $E600       wait for $75 = A

(execute program)
00000409:       2019E5  JSR $E519       get opcode, operand to $11,$13
0000040C:       206CF6  JSR $F66C       execute one grogram command
0000040F:       A92F    LDA #$2F
00000411:       2000E6  JSR $E600       wait for $75 = A
00000414:       A511    LDA $11         opcode
00000416:       C9FF    CMP #$FF        if end of program...
00000418:       F008    BEQ $08         to $E422 = RTS
0000041A:       E60F    INC $0F         increment $0F,$10 by 2
0000041C:       E60F    INC $0F         program counter
0000041E:       D002    BNE $02
00000420:       E610    INC $10
00000422:       60      RTS
```

```
(clear program)
00000423:       2082E5  JSR $E582       initialize $0F,$10 pointer to $0180 = beg
00000426:       207BE5  JSR $E57B       put #$FF at $0180
00000429:       A20B    LDX #$0B        "CLrP"
0000042B:       2084F6  JSR $F684       serial out to $1200 display
0000042E:       20ECE3  JSR $E3EC       delay
00000431:       4C4EF6  JMP $F64E       program step number to display

(execute program)
00000434:       A901    LDA #$01        here if $0D = 3 (execute)
00000436:       8517    STA $17
00000438:       A502    LDA $02
0000043A:       4A      LSR A
0000043B:       B023    BCS $23         to $E460 if LO bit of $02 set
0000043D:       38      SEC             else set it in A
0000043E:       2A      ROL A
0000043F:       A8      TAY
00000440:       2902    AND #$02
00000442:       F004    BEQ $04         to $E448 if bit 1 of $02 = 0
00000444:       98      TYA
00000445:       29BF    AND #$BF        else zero bit 6 of $02
00000447:       A8      TAY
00000448:       8402    STY $02
0000044A:       2402    BIT $02
0000044C:       500F    BVC $0F         to $E45D if V bit of $02 = 0, set prog
pointer to $0180
0000044E:       A50A    LDA $0A
00000450:       2901    AND #$01
00000452:       4901    EOR #$01
00000454:       8507    STA $07
00000456:       2055E5  JSR $E555       decrement $0F,$10 program pointer by 2, beg
at $0180
00000459:       B005    BCS $05         to $E460 execute w/o init of program pointer
0000045B:       9023    BCC $23         to $E480

0000045D:       2082E5  JSR $E582       initialize $0F,$10 pointer to $0180 = beg
00000460:       204EF6  JSR $F64E       program step number to display
00000463:       2019E5  JSR $E519       get opcode, operand to $11,$13 from
($0F,10), $80,81
00000466:       C90C    CMP #$0C
00000468:       F008    BEQ $08         to $E472 if opcode = #$0C = delay
0000046A:       C9FF    CMP #$FF
0000046C:       F004    BEQ $04         to $E472 if opcode = #$FF = end
0000046E:       A502    LDA $02
00000470:       3003    BMI $03         to $E475 if bit 7 of $02 set
00000472:       2004E5  JSR $E504
00000475:       A511    LDA $11         opcode
00000477:       C9FF    CMP #$FF
00000479:       D011    BNE $11         to $E48C if $11 <> #$FF
opcode = $FF = program end
0000047B:       A20E    LDX #$0E        here if end of prog  "End"
0000047D:       2084F6  JSR $F684       serial out to $1200 display
00000480:       2004E5  JSR $E504
00000483:       A502    LDA $02
00000485:       4941    EOR #$41        invert bits 6 and 0 of $02
00000487:       8502    STA $02         exit program execution
00000489:       6C7A00  JMP ($007A)     to $E0CB immediate mode
here if not end of program
0000048C:       2025E5  JSR $E525
0000048F:       20A4E6  JSR $E6A4
00000492:       E00C    CPX #$0C
00000494:       D014    BNE $14         to $E4AA if X <> #$0C
00000496:       206CF6  JSR $F66C       execute one program command
```

```
00000499:       200DE6  JSR $E60D       wait for key < $20
0000049C:       C90F    CMP #$0F
0000049E:       F005    BEQ $05         to $E4A5 if key = $0F = ENTER?
000004A0:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control?
000004A3:       30F4    BMI $F4         to $E499
000004A5:       A92F    LDA #$2F
000004A7:       2000E6  JSR $E600       wait for $75 = A
000004AA:       A511    LDA $11         opcode
000004AC:       C90C    CMP #$0C        delay command
000004AE:       D01C    BNE $1C         to $E4CC if not delay opcode
opcode = $0C = delay
000004B0:       A513    LDA $13         operand  duration of delay
000004B2:       F043    BEQ $43         to $E4F7 if zero
000004B4:       8527    STA $27         set 1 second timer to operand value
000004B6:       A527    LDA $27         gets decremented to zero in IRQ routine
000004B8:       F03D    BEQ $3D         to $E4F7 if $27 = 0
000004BA:       2426    BIT $26         wait for bit 7 of $26 to set
                                        $26 gets decremented in IRQ routine
000004BC:       10FC    BPL $FC         to $E4BA
000004BE:       2426    BIT $26         wait for bit 7 of $26 to zero
000004C0:       30FC    BMI $FC         to $E4BE
000004C2:       A527    LDA $27         gets decremented in IRQ routine
000004C4:       8513    STA $13         operand?
000004C6:       206CF6  JSR $F66C
000004C9:       4C8FE4  JMP $E48F
here if not delay
000004CC:       AA      TAX             here if not delay command
000004CD:       300F    BMI $0F         to $E4DE if opcode bit 7 set
000004CF:       293F    AND #$3F
000004D1:       20FEE9  JSR $E9FE
000004D4:       A507    LDA $07
000004D6:       850A    STA $0A
000004D8:       A501    LDA $01
000004DA:       101B    BPL $1B         to $E4F7 if bit 7 of $01 = 0
000004DC:       30B1    BMI $B1         to $E48F if bit 7 of $01 = 1
000004DE:       290F    AND #$0F        here if opcode > $7F
000004E0:       C901    CMP #$01
000004E2:       D00C    BNE $0C         to $E4F0 if <> $81
opcode = $81 = play tune
000004E4:       A52B    LDA $2B         here of opcode $81 = play tune
000004E6:       D0A7    BNE $A7         to $E48F if $2B <> 0
000004E8:       A613    LDX $13         operand = tune #
000004EA:       2001EF  JSR $EF01       get pointers for tune # in operand
000004ED:       4CF7E4  JMP $E4F7
here if opcode not $81, but >$7F
000004F0:       A55B    LDA $5B         here if opcode > $7F, and <> $81
000004F2:       D09B    BNE $9B         to $E492 if $5B <> 0
000004F4:       204BF4  JSR $F44B
000004F7:       2402    BIT $02
000004F9:       5003    BVC $03         to $E4FE if V bit of $02 = 0
000004FB:       4C56E4  JMP $E456
000004FE:       201AE4  JSR $E41A       increment $0F,$10 by 2 program counter
00000501:       4C60E4  JMP $E460

opcode = $0C = delay?
00000504:       20AFED  JSR $EDAF       check for stalled motors
00000507:       A9FF    LDA #$FF
00000509:       20FEE9  JSR $E9FE
0000050C:       A508    LDA $08
0000050E:       290C    AND #$0C
00000510:       0505    ORA $05
00000512:       055B    ORA $5B
00000514:       052B    ORA $2B
```

```
00000516:       D0EC    BNE $EC         loop back to $E504
00000518:       60      RTS

get opcode, operand from ($0F,10), $80,81 to $11,13
00000519:       A081    LDY #$81        get opcode, operand to $11,$13
0000051B:       B10F    LDA ($0F),Y
0000051D:       8513    STA $13         operand
0000051F:       88      DEY
00000520:       B10F    LDA ($0F),Y
00000522:       8511    STA $11         opcode
00000524:       60      RTS


00000525:       2402    BIT $02
00000527:       5058    BVC $58         to $E581 (RTS) if V bit of $02 = 0 (RTS)
00000529:       A611    LDX $11         opcode
0000052B:       E00B    CPX #$0B        home?
0000052D:       1009    BPL $09         to $E536 if opcode >= #$0B
opcode < $0B
0000052F:       BD4DE5  LDA $E54D,X     here if < $0B
00000532:       E008    CPX #$08        change octave?
00000534:       1003    BPL $03         to $E539

00000536:       8511    STA $11         here if <$08 OR >$0A

00000538:       60      RTS
00000539:       D00C    BNE $0C         to $E547 change octave bit?
opcode = 8 = change octave?
0000053B:       A513    LDA $13         here if = $08
0000053D:       F007    BEQ $07         to $E546 = RTS, skip if operand = 0
0000053F:       38      SEC
00000540:       A955    LDA #$55
00000542:       E513    SBC $13
00000544:       8513    STA $13         operand
00000546:       60      RTS

opcode = 9 or $A
00000547:       A513    LDA $13         operand
00000549:       4901    EOR #$01        invert octave bit?
0000054B:       10F7    BPL $F7         to $E542

table ref'd at $E52F
0000054D:       0002
0000054F:       0103
00000551:       0504
00000553:       0706


decrement $0F,$10 program pointer by 2, beg at $0180
00000555:       A901    LDA #1
00000556:       C510    CMP $10
00000558:       D00D    BNE $0D         to $E567 = RTS if bit 0 of $10 = 1
0000055A:       A50F    LDA $0F
0000055D:       C980    CMP #$80
0000055F:       D007    BNE $07         to $E568
00000561:       A20F    LDX #$0F        "bEg"
00000563:       2084F6  JSR $F684       serial out to $1200 display
00000566:       18      CLC
00000567:       60      RTS
00000568:       A50F    LDA $0F
0000056A:       D002    BNE $02
0000056C:       C610    DEC $10         decrement $0F,$10 by 2
0000056E:       C60F    DEC $0F
```

```
00000570:       C60F    DEC $0F
00000572:       38      SEC
00000573:       60      RTS

00000574:       2403    BIT $03
00000576:       7009    BVS $09         to $E581 RTS if V bit of $03 set (RTS)
00000578:       2082E5  JSR $E582       initialize $0F,$10 pointer to $0180 = beg
0000057B:       A9FF    LDA #$FF
0000057D:       A080    LDY #$80        put #$FF at $0180+$80
0000057F:       910F    STA ($0F),Y
00000581:       60      RTS



initialize $0F,$10 pointer to $0180 = beg
00000582:       A980    LDA #$80
00000584:       850F    STA $0F
00000586:       A901    LDA #$01        set $0F,$10 to $0180 = beg
00000588:       8510    STA $10
0000058A:       A502    LDA $02
0000058C:       29BF    AND #$BF        clear bit 6 of $02
0000058E:       8502    STA $02
00000590:       A900    LDA #$00
00000592:       8507    STA $07         $07 = 0
00000594:       60      RTS



00000595:       6C9000  JMP ($0090)     initially to $E598

00000598:       2068F6  JSR $F668
0000059B:       0600    ASL $00
0000059D:       38      SEC
0000059E:       6600    ROR $00
000005A0:       A900    LDA #$00
000005A2:       8513    STA $13         operand?
000005A4:       200DE6  JSR $E60D       wait for key < $20
000005A7:       C90F    CMP #$0F
000005A9:       F035    BEQ $35         to $E5E0 if key = $0F
000005AB:       C90A    CMP #$0A
000005AD:       102B    BPL $2B         to $E5DA if key >= $0A
000005AF:       0613    ASL $13
000005B1:       B027    BCS $27         to $E5DA
000005B3:       6513    ADC $13
000005B5:       B023    BCS $23         to $E5DA
000005B7:       0613    ASL $13
000005B9:       B01F    BCS $1F         to $E5DA
000005BB:       0613    ASL $13
000005BD:       B01B    BCS $1B         to $E5DA
000005BF:       6513    ADC $13
000005C1:       B017    BCS $17         to $E5DA
000005C3:       A612    LDX $12         0 TO $0D
000005C5:       DDECE5  CMP $E5EC,X
000005C8:       F002    BEQ $02         to $E5CC
000005CA:       B00E    BCS $0E         to $E5DA
000005CC:       8513    STA $13         operand?
000005CE:       0600    ASL $00
000005D0:       4600    LSR $00
000005D2:       206CF6  JSR $F66C       execute one program command
000005D5:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000005D8:       30CA    BMI $CA         to $E5A4
```

```
000005DA:       A900    LDA #$00
000005DC:       8513    STA $13         operand?
000005DE:       18      CLC
000005DF:       60      RTS
key = $0F or $0A
000005E0:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000005E3:       A92F    LDA #$2F
000005E5:       2000E6  JSR $E600       wait for $75 = A
000005E8:       A513    LDA $13         operand?
000005EA:       38      SEC
000005EB:       60      RTS

000005EC:       63      $E5EC,X
000005ED:       FF
000005EE:       FF
000005EF:       63
000005F0:       63
000005F1:       63
000005F2:       63
000005F3:       63
000005F4:       54
000005F5:       0101
000005F7:       FF
000005F8:       FF
000005F9:       08


000005FA:       8C200F
000005FD:       0963
000005FF:       FF

00000600:       A080    LDY #$80
00000602:       2475    BIT $75         wait for H0 bit of $75 to zero
00000604:       30FC    BMI $FC         to $E602
00000606:       C575    CMP $75
00000608:       8475    STY $75
0000060A:       D0F6    BNE $F6         to $E602
0000060C:       60      RTS

wait for key < $20
0000060D:       2017E6  JSR $E617       get keypad key
00000610:       A515    LDA $15         key pressed?
00000612:       C920    CMP #$20
00000614:       10F7    BPL $F7         loop back to $E60D if $15 >= #$20
00000616:       60      RTS


get keypad key
00000617:       20A4E6  JSR $E6A4
0000061A:       E080    CPX #$80
0000061C:       F0F9    BEQ $F9         loop to $E617 if X = #$80
0000061E:       E020    CPX #$20
00000620:       3001    BMI $01         to $E623 if X < #$20
00000622:       60      RTS
00000623:       E015    CPX #$15
00000625:       D01E    BNE $1E         to $E645 if X <> #$15
00000627:       A50D    LDA $0D
00000629:       C902    CMP #$02
0000062B:       F004    BEQ $04         to $E631 if program mode
0000062D:       C901    CMP #$01
0000062F:       D014    BNE $14         to $E645 if not learn mode
00000631:       2400    BIT $00         here if learn or program mode
```

```
00000633:       5007    BVC $07         to $E63C
00000635:       A528    LDA $28         gets decremented in IRQ routine
00000637:       F003    BEQ $03         to $E63C if timer $28 = 0
00000639:       2023E4  JSR $E423
0000063C:       A500    LDA $00
0000063E:       29BF    AND #$BF        clear bit 6 of $00
00000640:       8500    STA $00
00000642:       4C17E6  JMP $E617       loop back to top


00000645:       BDB5E6  LDA $E6B5,X     here if not learn or program mode
                                        commands table?
00000648:       3055    BMI $55         to $E69F if table entry > $7F
0000064A:       C940    CMP #$40
0000064C:       30D4    BMI $D4         to $E622 = RTS if A < #$40
0000064E:       2907    AND #$07
00000650:       AA      TAX
00000651:       A508    LDA $08
00000653:       290C    AND #$0C
00000655:       0505    ORA $05
00000657:       D0C9    BNE $C9         to $E622 = RTS
00000659:       E006    CPX #$06
0000065B:       D00E    BNE $0E
0000065D:       A50D    LDA $0D         mode
0000065F:       D033    BNE $33         to $E694 if not immediate mode
00000661:       A50E    LDA $0E         here if immediate mode
00000663:       C906    CMP #$06
00000665:       D022    BNE $22         to $E689
00000667:       A204    LDX #$04
00000669:       D01E    BNE $1E         to $E689
0000066B:       E003    CPX #$03
0000066D:       D00A    BNE $0A         to $E67F
0000066F:       A50E    LDA $0E
00000671:       C903    CMP #$03
00000673:       D014    BNE $14         to $E689
00000675:       A202    LDX #$02
00000677:       D010    BNE $10         to $E689
00000679:       E001    CPX #$01
0000067B:       D00C    BNE $0C         to $E691
0000067D:       A50E    LDA $0E
0000067F:       C901    CMP #$01
00000681:       D006    BNE $06         to $E689
00000683:       A50D    LDA $0D
00000685:       D002    BNE $02         to $E689 if not immediate mode
00000687:       A205    LDX #$05
00000689:       8A      TXA
0000068A:       F00B    BEQ $0B         to $E697
0000068C:       CA      DEX             "notE"
0000068D:       2084F6  JSR $F684       serial out to $1200 display
00000690:       E8      INX
00000691:       860E    STX $0E
00000693:       60      RTS

here if not immediate mode
00000694:       4C40EF  JMP $EF40       set Y=$01, A=$6F motor control?
00000697:       A9E0    LDA #$E0        "7"
00000699:       204FED  JSR $ED4F       shift to $1200 LO bit, send HO 1st
0000069C:       4C91E6  JMP $E691
0000069F:       2907    AND #$07
000006A1:       4CD1E0  JMP $E0D1

000006A4:       20AFED  JSR $EDAF       check for stalled motors
000006A7:       2059E9  JSR $E959
```

```
000006AA:       A980    LDA #$80
000006AC:       A675    LDX $75
000006AE:       3004    BMI $04         to $E6B4 = RTS
000006B0:       8575    STA $75
000006B2:       8615    STX $15         keycode?
000006B4:       60      RTS


000006B5:       00010203        $19 bytes       Commands for each keycode
000006B9:       04050607        ref'd at E374
000006BD:       08090A0B
000006C1:       0C0D0E0F
000006C5:       41464313
000006C9:       84828183
000006CD:       80


000006CE:       AA      TAX
000006CF:       D01E    BNE $1E
000006D1:       A505    LDA $05
000006D3:       290F    AND #$0F
000006D5:       D077    BNE $77
000006D7:       A507    LDA $07
000006D9:       4A      LSR A
000006DA:       900A    BCC $0A
000006DC:       A505    LDA $05
000006DE:       0902    ORA #$02
000006E0:       8505    STA $05
000006E2:       A9E5    LDA #$E5
000006E4:       D039    BNE $39
000006E6:       4605    LSR $05
000006E8:       38      SEC
000006E9:       2605    ROL $05
000006EB:       A9D5    LDA #$D5
000006ED:       D030    BNE $30
000006EF:       A505    LDA $05
000006F1:       CA      DEX
000006F2:       D00A    BNE $0A
000006F4:       290F    AND #$0F
000006F6:       D056    BNE $56
000006F8:       209DE9  JSR $E99D
000006FB:       4CA2E2  JMP $E2A2
000006FE:       CA      DEX
000006FF:       D00A    BNE $0A
00000701:       290F    AND #$0F
00000703:       D049    BNE $49
00000705:       20BEE9  JSR $E9BE
00000708:       4CA2E2  JMP $E2A2
0000070B:       CA      DEX
0000070C:       D025    BNE $25
0000070E:       290F    AND #$0F
00000710:       D03C    BNE $3C
00000712:       A507    LDA $07
00000714:       4A      LSR A
00000715:       9012    BCC $12
00000717:       A505    LDA $05
00000719:       0908    ORA #$08        set bit 3 of $05
0000071B:       8505    STA $05
0000071D:       A9E4    LDA #$E4
0000071F:       20B3E9  JSR $E9B3
00000722:       A9FF    LDA #$FF        init timer $28 to #$FF
```

```
00000724:       8528    STA $28         gets decremented in IRQ routine
00000726:       4CA2E2  JMP $E2A2
00000729:       A505    LDA $05
0000072B:       0904    ORA #$04        set bit 2 of $05
0000072D:       8505    STA $05
0000072F:       A9D4    LDA #$D4
00000731:       D0EC    BNE $EC
00000733:       CA      DEX
00000734:       D019    BNE $19
00000736:       2930    AND #$30
00000738:       D014    BNE $14
0000073A:       2409    BIT $09
0000073C:       7010    BVS $10
0000073E:       A505    LDA $05
00000740:       0910    ORA #$10        set bit 4 of $05
00000742:       8505    STA $05
00000744:       A507    LDA $07
00000746:       29EF    AND #$EF        zero bit 4 of $07
00000748:       8507    STA $07
0000074A:       A912    LDA #$12
0000074C:       D046    BNE $46
0000074E:       60      RTS
0000074F:       CA      DEX
00000750:       D018    BNE $18
00000752:       2930    AND #$30
00000754:       D0F8    BNE $F8         to $E74E = RTS
00000756:       2409    BIT $09
00000758:       30F4    BMI $F4
0000075A:       A505    LDA $05
0000075C:       0920    ORA #$20        set bit 5 of $05
0000075E:       8505    STA $05
00000760:       A507    LDA $07
00000762:       0910    ORA #$10        set bit 4 of $07
00000764:       8507    STA $07
00000766:       A922    LDA #$22
00000768:       D02A    BNE $2A
0000076A:       A503    LDA $03
0000076C:       4A      LSR A
0000076D:       A505    LDA $05
0000076F:       CA      DEX
00000770:       D028    BNE $28
00000772:       29F0    AND #$F0
00000774:       9002    BCC $02
00000776:       29C0    AND #$C0
00000778:       D0D4    BNE $D4
0000077A:       2409    BIT $09
0000077C:       70D0    BVS $D0
0000077E:       A505    LDA $05
00000780:       B002    BCS $02
00000782:       0960    ORA #$60
00000784:       0940    ORA #$40
00000786:       8505    STA $05
00000788:       A507    LDA $07
0000078A:       29BF    AND #$BF
0000078C:       B002    BCS $02
0000078E:       0910    ORA #$10
00000790:       8507    STA $07
00000792:       A911    LDA #$11
00000794:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000797:       4CA2E2  JMP $E2A2
0000079A:       CA      DEX
0000079B:       D024    BNE $24
0000079D:       29F0    AND #$F0
```

```
0000079F:       9002    BCC $02
000007A1:       29C0    AND #$C0
000007A3:       D0A9    BNE $A9
000007A5:       2409    BIT $09
000007A7:       30A5    BMI $A5
000007A9:       A505    LDA $05
000007AB:       B002    BCS $02
000007AD:       0990    ORA #$90
000007AF:       0980    ORA #$80
000007B1:       8505    STA $05
000007B3:       A507    LDA $07
000007B5:       0940    ORA #$40
000007B7:       B002    BCS $02
000007B9:       29EF    AND #$EF
000007BB:       8507    STA $07
000007BD:       A921    LDA #$21
000007BF:       D0D3    BNE $D3
000007C1:       CA      DEX
000007C2:       D015    BNE $15
000007C4:       A9B1    LDA #$B1
000007C6:       207BED  JSR $ED7B       send to $1200
000007C9:       8508    STA $08
000007CB:       290C    AND #$0C
000007CD:       D0D4    BNE $D4
000007CF:       A508    LDA $08
000007D1:       0908    ORA #$08        set bit 3 of $08
000007D3:       8508    STA $08
000007D5:       A920    LDA #$20
000007D7:       D0BB    BNE $BB
000007D9:       CA      DEX
000007DA:       D030    BNE $30
000007DC:       A9B1    LDA #$B1
000007DE:       207BED  JSR $ED7B       send to $1200
000007E1:       8508    STA $08
000007E3:       290C    AND #$0C
000007E5:       D0BC    BNE $BC
000007E7:       A508    LDA $08
000007E9:       0904    ORA #$04        set bit 2 of $08
000007EB:       8508    STA $08
000007ED:       A506    LDA $06
000007EF:       4904    EOR #$04
000007F1:       8506    STA $06
000007F3:       2904    AND #$04
000007F5:       F009    BEQ $09
000007F7:       A950    LDA #$50
000007F9:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
000007FC:       A900    LDA #$00
000007FE:       F007    BEQ $07
00000800:       A970    LDA #$70
00000802:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000805:       A901    LDA #$01
00000807:       8513    STA $13         operand
00000809:       4CCBE2  JMP $E2CB

0000080C:       CA      DEX
0000080D:       D00E    BNE $0E
0000080F:       2471    BIT $71
00000811:       5005    BVC $05
00000813:       20CAEB  JSR $EBCA
00000816:       F0E4    BEQ $E4
00000818:       20CEEB  JSR $EBCE
0000081B:       D0E8    BNE $E8         to $E805
0000081D:       CA      DEX
```

```
0000081E:       D009    BNE $09         to $E829
00000820:       20DCEB  JSR $EBDC
00000823:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
00000826:       4CA2E2  JMP $E2A2

00000829:       CA      DEX
0000082A:       D007    BNE $07         to $E833
0000082C:       A9FF    LDA #$FF        256d seconds
0000082E:       8527    STA $27         gets decremented in IRQ routine
00000830:       4CA2E2  JMP $E2A2

00000833:       CA      DEX
00000834:       CA      DEX
00000835:       CA      DEX
00000836:       CA      DEX
00000837:       D010    BNE $10         to $E849
00000839:       290C    AND #$0C
0000083B:       D01F    BNE $1F         to $E85C = RTS
0000083D:       A505    LDA $05
0000083F:       2903    AND #$03
00000841:       F019    BEQ $19
00000843:       203DE9  JSR $E93D
00000846:       4CB1E2  JMP $E2B1

00000849:       CA      DEX
0000084A:       D007    BNE $07         to $E853
0000084C:       290F    AND #$0F
0000084E:       C905    CMP #$05
00000850:       F0F1    BEQ $F1         to $E843
00000852:       60      RTS

00000853:       CA      DEX
00000854:       D007    BNE $07         to $E85DE
00000856:       290F    AND #$0F
00000858:       C90A    CMP #$0A
0000085A:       F0E7    BEQ $E7
0000085C:       60      RTS

0000085D:       CA      DEX
0000085E:       D00B    BNE $0B         to $E86B
00000860:       2903    AND #$03
00000862:       D006    BNE $06         to $E86A RTS
00000864:       A505    LDA $05
00000866:       290C    AND #$0C
00000868:       D0D9    BNE $D9
0000086A:       60      RTS

0000086B:       CA      DEX
0000086C:       D029    BNE $29
0000086E:       2910    AND #$10
00000870:       F0F8    BEQ $F8
00000872:       A503    LDA $03
00000874:       4A      LSR A
00000875:       B006    BCS $06
00000877:       A505    LDA $05
00000879:       29C0    AND #$C0
0000087B:       D0ED    BNE $ED
0000087D:       A50B    LDA $0B
0000087F:       29CF    AND #$CF
00000881:       850B    STA $0B
00000883:       A902    LDA #$02        -
00000885:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
```

```
00000888:       20E8E9  JSR $E9E8       delay $18 decrements of $2A
0000088B:       20D4EC  JSR $ECD4
0000088E:       A505    LDA $05
00000890:       29CF    AND #$CF        zero bits 4 and 5 of $05
00000892:       8505    STA $05
00000894:       4CB1E2  JMP $E2B1

00000897:       CA      DEX
00000898:       D005    BNE $05         to $E89F
0000089A:       2920    AND #$20
0000089C:       D0D4    BNE $D4
0000089E:       60      RTS

0000089F:       CA      DEX
000008A0:       D030    BNE $30         to $E8D2
000008A2:       2940    AND #$40
000008A4:       F0F8    BEQ $F8
000008A6:       A901    LDA #$01
000008A8:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
000008AB:       20E8E9  JSR $E9E8       delay $18 decrements of $2A
000008AE:       20E3EC  JSR $ECE3
000008B1:       A503    LDA $03
000008B3:       4A      LSR A
000008B4:       08      PHP
000008B5:       B003    BCS $03         to $E8BA
000008B7:       20D4EC  JSR $ECD4
000008BA:       A50B    LDA $0B
000008BC:       28      PLP
000008BD:       B002    BCS $02         to $E8C1
000008BF:       290F    AND #$0F
000008C1:       293F    AND #$3F
000008C3:       850B    STA $0B
000008C5:       A505    LDA $05
000008C7:       B002    BCS $02
000008C9:       290F    AND #$0F
000008CB:       293F    AND #$3F
000008CD:       8505    STA $05
000008CF:       4CB1E2  JMP $E2B1

000008D2:       CA      DEX
000008D3:       D004    BNE $04         to $E8D9
000008D5:       0A      ASL A
000008D6:       B0CE    BCS $CE
000008D8:       60      RTS
000008D9:       CA      DEX
000008DA:       D020    BNE $20
000008DC:       A508    LDA $08
000008DE:       2908    AND #$08
000008E0:       F0F6    BEQ $F6
000008E2:       A50C    LDA $0C
000008E4:       29F7    AND #$F7        zero bit 3 of $0C
000008E6:       850C    STA $0C
000008E8:       A900    LDA #$00
000008EA:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
000008ED:       20E8E9  JSR $E9E8       delay $18 decrements of $2A
000008F0:       20F2EC  JSR $ECF2
000008F3:       A508    LDA $08
000008F5:       29F7    AND #$F7        zero bit 3 of $08
000008F7:       8508    STA $08
000008F9:       4CB1E2  JMP $E2B1

000008FC:       A527    LDA $27         gets decremented in IRQ routine
000008FE:       49FF    EOR #$FF        at 1 second intervals
```

```
00000900:       8514    STA $14         one's complement of $27 to $14
00000902:       4CB1E2  JMP $E2B1

00000905:       A908    LDA #$08
00000907:       8D0012  STA $1200
0000090A:       A508    LDA $08
0000090C:       29F1    AND #$F1        zero bits 1,2,and 3 of $08
0000090E:       8508    STA $08
00000910:       20CAEB  JSR $EBCA
00000913:       8505    STA $05
00000915:       850B    STA $0B
00000917:       850C    STA $0C
00000919:       8517    STA $17
0000091B:       A918    LDA #$18
0000091D:       8D0012  STA $1200
00000920:       A503    LDA $03
00000922:       4A      LSR A
00000923:       A9F8    LDA #$F8
00000925:       B001    BCS $01         to $E928 if UCof
00000927:       0A      ASL A
00000928:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
0000092B:       20E8E9  JSR $E9E8       delay $18 decrements of $2A
0000092E:       20C5EC  JSR $ECC5
00000931:       20D4EC  JSR $ECD4
00000934:       20E3EC  JSR $ECE3
00000937:       20F2EC  JSR $ECF2
0000093A:       4C1BEC  JMP $EC1B       send #$C1 until bit 7 of $1000 set

0000093D:       A50B    LDA $0B
0000093F:       29F0    AND #$F0        zero bits 0,1,2, and 3 of $0B
00000941:       850B    STA $0B
00000943:       A903    LDA #$03
00000945:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000948:       20E8E9  JSR $E9E8       delay $18 decrements of $2A
0000094B:       20C5EC  JSR $ECC5
0000094E:       A505    LDA $05
00000950:       29F0    AND #$F0        zero bits 0,1,2, and 3 of $05
00000952:       8505    STA $05
00000954:       A900    LDA #$00
00000956:       8517    STA $17         $17 = 0
00000958:       60      RTS

00000959:       A50D    LDA $0D         mode
0000095B:       C902    CMP #$02
0000095D:       10F9    BPL $F9         to $E958 = RTS if $0D >= 2
0000095F:       A505    LDA $05         here if immediate or learn mode
00000961:       290F    AND #$0F
00000963:       F0F3    BEQ $F3         to $E958 = RTS if LO 4 bits of $05 <> 0
00000965:       A517    LDA $17
00000967:       D0EF    BNE $EF         to $E958 = RTS if $17 <> 0
00000969:       A528    LDA $28         gets decremented in IRQ routine
0000096B:       D0EB    BNE $EB         to $E958 = RTS if timer $28 <> 0
0000096D:       A9F4    LDA #$F4        init timer $28 to #$F4
0000096F:       8528    STA $28         gets decremented in IRQ routine
00000971:       E617    INC $17
00000973:       A505    LDA $05
00000975:       4A      LSR A
00000976:       9005    BCC $05         to $E87D
00000978:       A9D5    LDA #$D5
0000097A:       20B3E9  JSR $E9B3       shift out to $1200
0000097D:       A505    LDA $05
0000097F:       2902    AND #$02
00000981:       F005    BEQ $05         to $E988
```

```
00000983:       A9E5    LDA #$E5
00000985:       20B3E9  JSR $E9B3       shift out to $1200
00000988:       A505    LDA $05
0000098A:       2904    AND #$04
0000098C:       F005    BEQ $05         to $E993
0000098E:       A9D4    LDA #$D4
00000990:       20B3E9  JSR $E9B3       shift out to $1200
00000993:       A505    LDA $05
00000995:       2908    AND #$08
00000997:       F0BF    BEQ $BF
00000999:       A9E4    LDA #$E4
0000099B:       D016    BNE $16
0000099D:       A505    LDA $05
0000099F:       29F0    AND #$F0
000009A1:       0905    ORA #$05        zero bits 0,1,2,and 3
000009A3:       8505    STA $05         and set bits 2 and 0 of $05
000009A5:       A507    LDA $07
000009A7:       0902    ORA #$02
000009A9:       29FE    AND #$FE        set bit 1 and zero bit 4 of $07
000009AB:       8507    STA $07
000009AD:       A9D3    LDA #$D3
000009AF:       A2FF    LDX #$FF        init timer $28 to #$FF
000009B1:       8628    STX $28         gets decremented in IRQ routine

000009B3:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
000009B6:       A517    LDA $17
000009B8:       18      CLC
000009B9:       6903    ADC #$03
000009BB:       4C55ED  JMP $ED55       shift 8 bits to LO bit of $1200


000009BE:       A505    LDA $05
000009C0:       29FA    AND #$FA        clear bits 0 and 2 of $05
000009C2:       090A    ORA #$0A        set bits 1 and 3 of $05
000009C4:       8505    STA $05
000009C6:       A507    LDA $07
000009C8:       0903    ORA #$03        set bits 0 and 1 of $07
000009CA:       8507    STA $07
000009CC:       A9E3    LDA #$E3
000009CE:       D0DF    BNE $DF         back to $E9AF

000009D0:       20AFED  JSR $EDAF       check for stalled motors
000009D3:       A9B0    LDA #$B0
000009D5:       207BED  JSR $ED7B       send to $1200
000009D8:       8505    STA $05
000009DA:       D0F4    BNE $F4         back to $E9D0
000009DC:       A9B1    LDA #$B1
000009DE:       207BED  JSR $ED7B       send to $1200
000009E1:       8508    STA $08
000009E3:       290C    AND #$0C
000009E5:       D0E9    BNE $E9         loop back to $E9D0
000009E7:       60      RTS

000009E8:       A918    LDA #$18        store #$18 to $2A
000009EA:       852A    STA $2A         gets decremented in IRQ routine
000009EC:       A52A    LDA $2A         wait for $2A to zero
000009EE:       D0FC    BNE $FC
000009F0:       60      RTS

000009F1:       20AFED  JSR $EDAF       check for stalled motors
000009F4:       2C0010  BIT $1000
000009F7:       30F8    BMI $F8         loop back to $E9F1
000009F9:       A9C1    LDA #$C1
```

```
000009FB:       4C55ED  JMP $ED55       shift 8 bits to LO bit of $1200


000009FE:       8512    STA $12
00000A00:       2C0010  BIT $1000
00000A03:       3059    BMI $59         to $EA5E
00000A05:       A9C1    LDA #$C1
00000A07:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000A0A:       A930    LDA #$30
00000A0C:       207BED  JSR $ED7B       send to $1200
00000A0F:       8506    STA $06
00000A11:       A9B0    LDA #$B0
00000A13:       207BED  JSR $ED7B       send to $1200
00000A16:       8505    STA $05
00000A18:       29C0    AND #$C0
00000A1A:       D010    BNE $10
00000A1C:       20E3EC  JSR $ECE3
00000A1F:       A503    LDA $03
00000A21:       4A      LSR A
00000A22:       A50B    LDA $0B
00000A24:       B002    BCS $02
00000A26:       290F    AND #$0F
00000A28:       293F    AND #$3F
00000A2A:       850B    STA $0B
00000A2C:       A505    LDA $05
00000A2E:       2930    AND #$30
00000A30:       D009    BNE $09
00000A32:       20D4EC  JSR $ECD4
00000A35:       A50B    LDA $0B
00000A37:       29CF    AND #$CF        zero bits 4 and 5 of $0B
00000A39:       850B    STA $0B
00000A3B:       A505    LDA $05
00000A3D:       290F    AND #$0F
00000A3F:       D009    BNE $09
00000A41:       20C5EC  JSR $ECC5
00000A44:       A50B    LDA $0B
00000A46:       29F0    AND #$F0        zero bits 0 thru 3 of $0B
00000A48:       850B    STA $0B
00000A4A:       A9B1    LDA #$B1
00000A4C:       207BED  JSR $ED7B       send to $1200
00000A4F:       8508    STA $08
00000A51:       290C    AND #$0C
00000A53:       D009    BNE $09
00000A55:       20F2EC  JSR $ECF2
00000A58:       A50C    LDA $0C
00000A5A:       29F3    AND #$F3        zero bits 2 and 3 of $0C
00000A5C:       850C    STA $0C
00000A5E:       0601    ASL $01
00000A60:       4601    LSR $01
00000A62:       A612    LDX $12
00000A64:       1001    BPL $01
00000A66:       60      RTS
00000A67:       E009    CPX #$09
00000A69:       1004    BPL $04
00000A6B:       A513    LDA $13         operand
00000A6D:       F0F7    BEQ $F7         to $EA66 = RTS if operand = 0
00000A6F:       8A      TXA
00000A70:       D028    BNE $28
00000A72:       A505    LDA $05
00000A74:       290F    AND #$0F
00000A76:       D038    BNE $38
00000A78:       A507    LDA $07
00000A7A:       4A      LSR A
```

```
00000A7B:       B00E    BCS $0E
00000A7D:       4605    LSR $05
00000A7F:       38      SEC
00000A80:       2605    ROL $05
00000A82:       A9D5    LDA #$D5
00000A84:       20B3E9  JSR $E9B3
00000A87:       A995    LDA #$95
00000A89:       D03A    BNE $3A
00000A8B:       A505    LDA $05
00000A8D:       0902    ORA #$02        set bit 1 of $05
00000A8F:       8505    STA $05
00000A91:       A9E5    LDA #$E5
00000A93:       20B3E9  JSR $E9B3
00000A96:       A9A5    LDA #$A5
00000A98:       D02B    BNE $2B
00000A9A:       A505    LDA $05
00000A9C:       CA      DEX
00000A9D:       D014    BNE $14
00000A9F:       290F    AND #$0F
00000AA1:       D00D    BNE $0D
00000AA3:       A505    LDA $05
00000AA5:       0905    ORA #$05        set bits 0 and 2 of $05
00000AA7:       8505    STA $05
00000AA9:       209DE9  JSR $E99D
00000AAC:       A993    LDA #$93
00000AAE:       D015    BNE $15         to $EAC5
00000AB0:       4CBFEC  JMP $ECBF

00000AB3:       CA      DEX
00000AB4:       D017    BNE $17
00000AB6:       290F    AND #$0F
00000AB8:       D0F6    BNE $F6         to $EAB0
00000ABA:       A505    LDA $05
00000ABC:       090A    ORA #$0A        set bits 1 and 3 of $05
00000ABE:       8505    STA $05
00000AC0:       20BEE9  JSR $E9BE
00000AC3:       A9A3    LDA #$A3

00000AC5:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200 display
00000AC8:       A513    LDA $13         operand
00000ACA:       4C55ED  JMP $ED55       shift 8 bits to LO bit of $1200 display

00000ACD:       CA      DEX
00000ACE:       D027    BNE $27
00000AD0:       290F    AND #$0F
00000AD2:       D0DC    BNE $DC
00000AD4:       A507    LDA $07
00000AD6:       4A      LSR A
00000AD7:       B00F    BCS $0F         to $EAE8
00000AD9:       A505    LDA $05
00000ADB:       0904    ORA #$04        set bit 2 of $05
00000ADD:       8505    STA $05
00000ADF:       A9D4    LDA #$D4
00000AE1:       20B3E9  JSR $E9B3
00000AE4:       A994    LDA #$94
00000AE6:       D0DD    BNE $DD

00000AE8:       A505    LDA $05
00000AEA:       0908    ORA #$08        set bit 3 of $05
00000AEC:       8505    STA $05
00000AEE:       A9E4    LDA #$E4
00000AF0:       20B3E9  JSR $E9B3
00000AF3:       A9A4    LDA #$A4
```

```
00000AF5:       D0CE    BNE $CE
00000AF7:       CA      DEX
00000AF8:       D018    BNE $18
00000AFA:       2930    AND #$30
00000AFC:       D0B2    BNE $B2
00000AFE:       2409    BIT $09
00000B00:       702B    BVS $2B
00000B02:       A505    LDA $05
00000B04:       0910    ORA #$10        set bit 4 of $05
00000B06:       8505    STA $05
00000B08:       A507    LDA $07
00000B0A:       29EF    AND #$EF        zero bit 4 of $07
00000B0C:       8507    STA $07
00000B0E:       A992    LDA #$92
00000B10:       D0B3    BNE $B3
00000B12:       CA      DEX
00000B13:       D019    BNE $19
00000B15:       2930    AND #$30
00000B17:       D069    BNE $69
00000B19:       2409    BIT $09
00000B1B:       3010    BMI $10
00000B1D:       A505    LDA $05
00000B1F:       0920    ORA #$20        set bit 5 of $05
00000B21:       8505    STA $05
00000B23:       A507    LDA $07
00000B25:       0910    ORA #$10        set bit 4 of $07
00000B27:       8507    STA $07
00000B29:       A9A2    LDA #$A2
00000B2B:       D02B    BNE $2B
00000B2D:       60      RTS

00000B2E:       A503    LDA $03
00000B30:       4A      LSR A
00000B31:       A505    LDA $05
00000B33:       CA      DEX
00000B34:       D025    BNE $25
00000B36:       29F0    AND #$F0
00000B38:       9002    BCC $02
00000B3A:       29C0    AND #$C0
00000B3C:       D044    BNE $44
00000B3E:       2409    BIT $09
00000B40:       70EB    BVS $EB
00000B42:       A505    LDA $05
00000B44:       B002    BCS $02
00000B46:       0960    ORA #$60
00000B48:       0940    ORA #$40
00000B4A:       8505    STA $05
00000B4C:       A507    LDA $07
00000B4E:       29BF    AND #$BF        zero bit 6 of $07
00000B50:       B002    BCS $02         to $EB54
00000B52:       0910    ORA #$10        set bit 5 of $07
00000B54:       8507    STA $07
00000B56:       A991    LDA #$91

00000B58:       4CC5EA  JMP $EAC5
00000B5B:       CA      DEX
00000B5C:       D027    BNE $27         to $EB85
00000B5E:       29F0    AND #$F0
00000B60:       9002    BCC $02         to $EB64
00000B62:       29C0    AND #$C0
00000B64:       D01C    BNE $1C
00000B66:       2409    BIT $09
00000B68:       30C3    BMI $C3
```

```
00000B6A:       A505    LDA $05
00000B6C:       B002    BCS $02
00000B6E:       0990    ORA #$90
00000B70:       0980    ORA #$80
00000B72:       8505    STA $05
00000B74:       A507    LDA $07
00000B76:       0940    ORA #$40        set bit 6 of $07
00000B78:       B002    BCS $02         to $EB7C
00000B7A:       29EF    AND #$EF        zero bit 5 of $07
00000B7C:       8507    STA $07
00000B7E:       A9A1    LDA #$A1
00000B80:       D0D6    BNE $D6
00000B82:       4CBFEC  JMP $ECBF

00000B85:       A508    LDA $08
00000B87:       CA      DEX
00000B88:       D00E    BNE $0E         to $EB98
00000B8A:       290C    AND #$0C
00000B8C:       D0F4    BNE $F4         back to $EB82
00000B8E:       A508    LDA $08
00000B90:       0908    ORA #$08        set bit 3 of $08
00000B92:       8508    STA $08
00000B94:       A9A0    LDA #$A0
00000B96:       D0C0    BNE $C0         back to $EB58
00000B98:       CA      DEX
00000B99:       D028    BNE $28         to $EBC3
00000B9B:       290C    AND #$0C
00000B9D:       D0E3    BNE $E3         back to $EB82
00000B9F:       A508    LDA $08
00000BA1:       0904    ORA #$04        set bit 2 of $08
00000BA3:       8508    STA $08
00000BA5:       A513    LDA $13         operand
00000BA7:       F00D    BEQ $0D         to $EBB6 if $13 = 0
00000BA9:       A506    LDA $06
00000BAB:       0904    ORA #$04        set bit 2 and clear bit 4 of $06
00000BAD:       29F7    AND #$F7
00000BAF:       8506    STA $06
00000BB1:       A970    LDA #$70
00000BB3:       4C55ED  JMP $ED55       shift 8 bits to LO bit of $1200

00000BB6:       A506    LDA $06
00000BB8:       0908    ORA #$08        set bit 3 and clear bit 2 of $06
00000BBA:       29FB    AND #$FB
00000BBC:       8506    STA $06
00000BBE:       A950    LDA #$50
00000BC0:       4C55ED  JMP $ED55       shift 8 bits to LO bit of $1200


00000BC3:       CA      DEX
00000BC4:       D00D    BNE $0D
00000BC6:       A513    LDA $13         operand
00000BC8:       D004    BNE $04         to $EBCE if $13 <> 0

00000BCA:       A900    LDA #$00
00000BCC:       1002    BPL $02         to $EBD0
00000BCE:       A940    LDA #$40
00000BD0:       8571    STA $71
00000BD2:       60      RTS


00000BD3:       CA      DEX
00000BD4:       D053    BNE $53
00000BD6:       290C    AND #$0C
```

```
00000BD8:       0505    ORA $05
00000BDA:       D0A6    BNE $A6

00000BDC:       A940    LDA #$40
00000BDE:       8507    STA $07
00000BE0:       20CAEB  JSR $EBCA
00000BE3:       A960    LDA #$60
00000BE5:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000BE8:       A962    LDA #$62
00000BEA:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000BED:       20D0E9  JSR $E9D0
00000BF0:       A970    LDA #$70
00000BF2:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000BF5:       A961    LDA #$61
00000BF7:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000BFA:       20D0E9  JSR $E9D0
00000BFD:       A900    LDA #$00
00000BFF:       8518    STA $18
00000C01:       851E    STA $1E
00000C03:       8521    STA $21
00000C05:       8506    STA $06
00000C07:       851A    STA $1A
00000C09:       851D    STA $1D
00000C0B:       8520    STA $20
00000C0D:       8523    STA $23
00000C0F:       A9FF    LDA #$FF
00000C11:       851B    STA $1B
00000C13:       20E8E9  JSR $E9E8       delay $18 decrements of $2A
00000C16:       A9C0    LDA #$C0
00000C18:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000C1B:       2C0010  BIT $1000
00000C1E:       1001    BPL $01         send #$C1 until bit 7 of $1000 set
00000C20:       60      RTS

00000C21:       A9C1    LDA #$C1
00000C23:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000C26:       4C1BEC  JMP $EC1B       loop back



00000C29:       CA      DEX
00000C2A:       CA      DEX
00000C2B:       CA      DEX
00000C2C:       CA      DEX
00000C2D:       CA      DEX
00000C2E:       D021    BNE $21         to #EC51
X was 5
00000C30:       2001ED  JSR $ED01
00000C33:       A984    LDA #$84

00000C35:       207BED  JSR $ED7B       send to $1200
00000C38:       851A    STA $1A
00000C3A:       8514    STA $14
00000C3C:       A507    LDA $07
00000C3E:       4A      LSR A
00000C3F:       9008    BCC $08         to $EC49
00000C41:       38      SEC
00000C42:       A518    LDA $18
00000C44:       E514    SBC $14
00000C46:       8518    STA $18
00000C48:       60      RTS

00000C49:       18      CLC
```

```
00000C4A:       A518    LDA $18
00000C4C:       6514    ADC $14
00000C4E:       8518    STA $18
00000C50:       60      RTS

here if X was > 5
00000C51:       CA      DEX
00000C52:       D022    BNE $22         to #EC76
X was 6
00000C54:       2016ED  JSR $ED16
00000C57:       A982    LDA #$82
00000C59:       207BED  JSR $ED7B       send to $1200
00000C5C:       851D    STA $1D
00000C5E:       8514    STA $14
00000C60:       A507    LDA $07
00000C62:       2910    AND #$10
00000C64:       F008    BEQ $08         to $EC6E
00000C66:       38      SEC
00000C67:       A51B    LDA $1B
00000C69:       E514    SBC $14
00000C6B:       851B    STA $1B
00000C6D:       60      RTS

00000C6E:       18      CLC
00000C6F:       A51B    LDA $1B
00000C71:       6514    ADC $14
00000C73:       851B    STA $1B
00000C75:       60      RTS

here if X was > 6
00000C76:       CA      DEX
00000C77:       D020    BNE $20         to $EC99
X was 7
00000C79:       202CED  JSR $ED2C
00000C7C:       A981    LDA #$81

00000C7E:       207BED  JSR $ED7B       send to $1200
00000C81:       8520    STA $20
00000C83:       8514    STA $14
00000C85:       2407    BIT $07
00000C87:       5008    BVC $08
00000C89:       38      SEC
00000C8A:       A51E    LDA $1E
00000C8C:       E514    SBC $14
00000C8E:       851E    STA $1E
00000C90:       60      RTS

00000C91:       18      CLC
00000C92:       A51E    LDA $1E
00000C94:       6514    ADC $14
00000C96:       851E    STA $1E
00000C98:       60      RTS

here if X was > 7
00000C99:       CA      DEX
00000C9A:       D028    BNE $28         to $ECC4 = RTS
X was 8
00000C9C:       2040ED  JSR $ED40
00000C9F:       A980    LDA #$80
00000CA1:       207BED  JSR $ED7B       send to $1200
00000CA4:       C955    CMP #$55
00000CA6:       9006    BCC $06         to $ECAE
00000CA8:       38      SEC
```

```
00000CA9:       E955    SBC #$55
00000CAB:       4CA4EC  JMP $ECA4

00000CAE:       8514    STA $14
00000CB0:       8523    STA $23
00000CB2:       18      CLC
00000CB3:       6521    ADC $21
00000CB5:       C955    CMP #$55
00000CB7:       9003    BCC $03         to $ECBC
00000CB9:       38      SEC
00000CBA:       E955    SBC #$55

00000CBC:       8521    STA $21
00000CBE:       60      RTS



00000CBF:       0601    ASL $01
00000CC1:       38      SEC
00000CC2:       6601    ROR $01
00000CC4:       60      RTS

00000CC5:       2001ED  JSR $ED01
00000CC8:       A944    LDA #$44
00000CCA:       2035EC  JSR $EC35
00000CCD:       A900    LDA #$00
00000CCF:       851A    STA $1A         $19 = $1A = 0
00000CD1:       8519    STA $19
00000CD3:       60      RTS

00000CD4:       2016ED  JSR $ED16
00000CD7:       A942    LDA #$42
00000CD9:       2059EC  JSR $EC59
00000CDC:       A900    LDA #$00
00000CDE:       851D    STA $1D         $1C = $1D = 0
00000CE0:       851C    STA $1C
00000CE2:       60      RTS

00000CE3:       202CED  JSR $ED2C
00000CE6:       A941    LDA #$41
00000CE8:       207EEC  JSR $EC7E
00000CEB:       A900    LDA #$00
00000CED:       8520    STA $20         $1F = $20 = 0
00000CEF:       851F    STA $1F
00000CF1:       60      RTS

00000CF2:       2040ED  JSR $ED40
00000CF5:       A940    LDA #$40
00000CF7:       20A1EC  JSR $ECA1
00000CFA:       A900    LDA #$00
00000CFC:       8523    STA $23         $22 = $23 = 0
00000CFE:       8522    STA $22
00000D00:       60      RTS


00000D01:       A507    LDA $07
00000D03:       4A      LSR A
00000D04:       B008    BCS $08         to $ED0E
00000D06:       38      SEC
00000D07:       A518    LDA $18
00000D09:       E51A    SBC $1A
00000D0B:       4C13ED  JMP $ED13
```

```
00000D0E:       18      CLC
00000D0F:       A518    LDA $18
00000D11:       651A    ADC $1A         $18 = $18 + $1A

00000D13:       8518    STA $18
00000D15:       60      RTS


00000D16:       A507    LDA $07
00000D18:       2910    AND #$10
00000D1A:       D008    BNE $08         to $ED24
00000D1C:       38      SEC
00000D1D:       A51B    LDA $1B
00000D1F:       E51D    SBC $1D
00000D21:       4C29ED  JMP $ED29
00000D24:       18      CLC
00000D25:       A51B    LDA $1B
00000D27:       651D    ADC $1D         $1B = $1B + $1D
00000D29:       851B    STA $1B
00000D2B:       60      RTS


00000D2C:       2407    BIT $07         to $ED35
00000D2E:       7008    BVS $08         to $ED38
00000D30:       38      SEC
00000D31:       A51E    LDA $1E
00000D33:       E520    SBC $20
00000D35:       4C3DED  JMP $ED3D
00000D38:       18      CLC
00000D39:       A51E    LDA $1E
00000D3B:       6520    ADC $20         $1E = $1E + $20
00000D3D:       851E    STA $1E
00000D3F:       60      RTS

00000D40:       38      SEC
00000D41:       A521    LDA $21
00000D43:       E523    SBC $23         $21 = $21 - $23
00000D45:       8521    STA $21
00000D47:       60      RTS


00000D48:       A9E3    LDA #$E3
00000D4A:       204FED  JSR $ED4F       shift to $1200 LO bit, send HO data 1st
00000D4D:       A9F3    LDA #$F3
00000D4F:       855F    STA $5F         shift A to $1200 LO bit, send HO data 1st
00000D51:       A90D    LDA #$0D
00000D53:       D004    BNE $04         to $ED59
00000D55:       855F    STA $5F
00000D57:       A90E    LDA #$0E
00000D59:       855E    STA $5E
00000D5B:       A908    LDA #$08        shift 8 bits to LO bit of $1200
00000D5D:       8560    STA $60         bit counter
00000D5F:       2C0010  BIT $1000       wait for bit to zero
00000D62:       50FB    BVC $FB         loop back to $ED5F
00000D64:       A55E    LDA $5E
00000D66:       265F    ROL $5F         shift next HO bit of A to LO bit of $1200
00000D68:       2A      ROL A
00000D69:       8D0012  STA $1200
00000D6C:       2C0010  BIT $1000       wait for bit to zero
00000D6F:       70FB    BVS $FB         loop back to $ED6C
00000D71:       A9F8    LDA #$F8
00000D73:       8D0012  STA $1200
```

```
00000D76:       C660    DEC $60         loop if not all 8 bits transferred
00000D78:       D0E5    BNE $E5         to $ED5F
00000D7A:       60      RTS


00000D7B:       2055ED  JSR $ED55       shift 8 bits to LO bit of $1200
00000D7E:       A908    LDA #$08        8 loops
00000D80:       8560    STA $60
00000D82:       2C0010  BIT $1000       wait for bit to zero
00000D85:       50FB    BVC $FB         loop back to $ED82
00000D87:       A9F4    LDA #$F4
00000D89:       8D0012  STA $1200
00000D8C:       2C0010  BIT $1000       wait for bit to zero
00000D8F:       70FB    BVS $FB         loop back to $ED8C
00000D91:       AD0010  LDA $1000
00000D94:       4A      LSR A
00000D95:       265F    ROL $5F
00000D97:       A9F0    LDA #$F0
00000D99:       8D0012  STA $1200
00000D9C:       C660    DEC $60
00000D9E:       D0E2    BNE $E2         loop back to $ED82
00000DA0:       2C0010  BIT $1000       wait for bit to zero
00000DA3:       50FB    BVC $FB         loop back to $EDA0
00000DA5:       A9F8    LDA #$F8
00000DA7:       8D0012  STA $1200
00000DAA:       A55F    LDA $5F
00000DAC:       49FF    EOR #$FF
00000DAE:       60      RTS


check for stalled motors
00000DAF:       A529    LDA $29         timer gets decremented in IRQ routine
00000DB1:       F003    BEQ $03         to $EDB6 if $29 = 0
00000DB3:       4C2FEE  JMP $EE2F
00000DB6:       A9B1    LDA #$B1
00000DB8:       207BED  JSR $ED7B       send to $1200
00000DBB:       8508    STA $08
00000DBD:       A9B0    LDA #$B0
00000DBF:       207BED  JSR $ED7B       send to $1200
00000DC2:       8505    STA $05
00000DC4:       250B    AND $0B
00000DC6:       290F    AND #$0F
00000DC8:       F00B    BEQ $0B         to $EDD5
00000DCA:       A984    LDA #$84
00000DCC:       207BED  JSR $ED7B       send to $1200
00000DCF:       C519    CMP $19
00000DD1:       F049    BEQ $49         to $EE1C = stall
00000DD3:       8519    STA $19
00000DD5:       A50B    LDA $0B
00000DD7:       2505    AND $05
00000DD9:       2930    AND #$30
00000DDB:       F00B    BEQ $0B         to $EDE8
00000DDD:       A982    LDA #$82
00000DDF:       207BED  JSR $ED7B       send to $1200
00000DE2:       C51C    CMP $1C
00000DE4:       F036    BEQ $36         to $EE1C = stall
00000DE6:       851C    STA $1C
00000DE8:       A50B    LDA $0B
00000DEA:       2505    AND $05
00000DEC:       29C0    AND #$C0
00000DEE:       F00B    BEQ $0B         to $EDFB
00000DF0:       A981    LDA #$81
00000DF2:       207BED  JSR $ED7B       send to $1200
```

```
00000DF5:       C51F    CMP $1F
00000DF7:       F023    BEQ $23         to $EE1C = stall
00000DF9:       851F    STA $1F
00000DFB:       A50C    LDA $0C
00000DFD:       2508    AND $08
00000DFF:       2908    AND #$08
00000E01:       F00B    BEQ $0B         to $EE0E
00000E03:       A980    LDA #$80
00000E05:       207BED  JSR $ED7B       send to $1200
00000E08:       C522    CMP $22
00000E0A:       F010    BEQ $10         to $EE1C = stall
00000E0C:       8522    STA $22
00000E0E:       A505    LDA $05
00000E10:       850B    STA $0B
00000E12:       A508    LDA $08
00000E14:       850C    STA $0C
00000E16:       A902    LDA #$02        init timer $29 to 2
00000E18:       8529    STA $29         gets decremented in IRQ routine
00000E1A:       D013    BNE $13         always to $EE2F
00000E1C:       2005E9  JSR $E905
00000E1F:       A20D    LDX #$0D        "StAL"
00000E21:       2084F6  JSR $F684       serial out to $1200 display
00000E24:       E6A3    INC $A3
00000E26:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control? stop motors?
00000E29:       20ECE3  JSR $E3EC       delay
00000E2C:       6C7A00  JMP ($007A)     to $E0CB immediate mode




00000E2F:       6C8C00  JMP ($008C)     initially vectors to $EE32

00000E32:       2404    BIT $04
00000E34:       1025    BPL $25         to $EE5B past Reveille
00000E36:       2C0016  BIT $1600
00000E39:       7020    BVS $20         to $EE5B past Reveille
00000E3B:       A900    LDA #$00
00000E3D:       8504    STA $04
00000E3F:       A9EE    LDA #$EE        "0"
00000E41:       204FED  JSR $ED4F       shift to $1200 display
00000E44:       A50D    LDA $0D         mode
00000E46:       D00B    BNE $0B         to $EE53 (Reveille) if not immediate mode
00000E48:       A910    LDA #$10        here if immediate mode
00000E4A:       2403    BIT $03
00000E4C:       F005    BEQ $05         to $EE53 if bit 4 of $03 = 0 = Edof
00000E4E:       A903    LDA #$03        set mode 3 = execute
00000E50:       4CD1E0  JMP $E0D1

(Reveille)
00000E53:       A206    LDX #$06
00000E55:       2001EF  JSR $EF01       get pointers for Reveille tune
00000E58:       4C63EF  JMP $EF63       wait for loc $2B to be zero

00000E5B:       A50D    LDA $0D         mode
00000E5D:       C903    CMP #$03
00000E5F:       F028    BEQ $28         to $EE89 if mode 3 execute
00000E61:       C904    CMP #$04

00000E63:       F024    BEQ $24         to $EE89 if mode 4 game
00000E65:       A574    LDA $74
00000E67:       C903    CMP #$03
00000E69:       901E    BCC $1E         to $EE89
```

```
00000E6B:       A208    LDX #$08
00000E6D:       2001EF  JSR $EF01       get pointers for taps tune
00000E70:       A575    LDA $75
00000E72:       1012    BPL $12         to $EE86 if $75 >= 0
00000E74:       A52B    LDA $2B
00000E76:       D0F8    BNE $F8         loop to $EE70 if $2B <> 0
(power down)
00000E78:       78      SEI             disable interrupts
00000E79:       A902    LDA #$02
00000E7B:       8D001E  STA $1E00
00000E7E:       A9E5    LDA #$E5
00000E80:       204FED  JSR $ED4F       shift to $1200 display
00000E83:       4C83EE  JMP $EE83       tight loop to itself!



00000E86:       20EEEE  JSR $EEEE       init timer bytes
00000E89:       A596    LDA $96         here if mode 3 (execute) or 4 (game)
00000E8B:       D007    BNE $07         to $EE94 RTS
00000E8D:       C696    DEC $96
00000E8F:       A21A    LDX #$1A
00000E91:       4C0FF4  JMP $F40F       say "I need energy, please recharge me."
00000E94:       60      RTS


timer interrupt routine
IRQ 4
00000E95:       C626    DEC $26         entered through IRQ jump table
                                        when $56 = 4
00000E97:       D028    BNE $28         to $EEBF
00000E99:       A9F4    LDA #$F4        here if $26 = 0
00000E9B:       8526    STA $26         reset $26 to #$F4 on underflow
00000E9D:       A696    LDX $96
00000E9F:       F00A    BEQ $0A         to $EEAB
00000EA1:       CA      DEX
00000EA2:       2C0016  BIT $1600
00000EA5:       1002    BPL $02         to $EEA9
00000EA7:       A2FF    LDX #$FF
00000EA9:       8696    STX $96         reset $96 to #$FF
00000EAB:       2403    BIT $03
00000EAD:       3006    BMI $06         to $EEB5
00000EAF:       C673    DEC $73
00000EB1:       D002    BNE $02         to $EEB5
00000EB3:       E674    INC $74         increment $74 on underflows of $73
00000EB5:       A527    LDA $27         decrement $27, stop at 0
00000EB7:       F002    BEQ $02         to $EEBB
00000EB9:       C627    DEC $27
00000EBB:       A529    LDA $29         decrement $29, stop at 0
00000EBD:       F002    BEQ $02         to $EEC1
00000EBF:       C629    DEC $29
00000EC1:       A528    LDA $28         decrement $28, stop at 0
00000EC3:       F002    BEQ $02         to $EEC7
00000EC5:       C628    DEC $28
00000EC7:       A52A    LDA $2A         decrement $2A, stop at 0
00000EC9:       F002    BEQ $02         to $EECD
00000ECB:       C62A    DEC $2A
00000ECD:       4CE0FD  JMP $FDE0       return from IRQ


IRQ 5
00000ED0:       2471    BIT $71         entered through IRQ jump table
```

```
                                        when $56 = 5
00000ED2:       7004    BVS $04         to $EED8
00000ED4:       A900    LDA #$00
00000ED6:       F00A    BEQ $0A         to $EEE2
00000ED8:       C672    DEC $72
00000EDA:       300B    BMI $0B         to $EEE7 = RTS
00000EDC:       A982    LDA #$82
00000EDE:       8572    STA $72
00000EE0:       4571    EOR $71
00000EE2:       8D0010  STA $1000
00000EE5:       8571    STA $71
00000EE7:       4CE0FD  JMP $FDE0       return from IRQ


00000EEA:       A900    LDA #$00
00000EEC:       8555    STA $55



init timer bytes
00000EEE:       A203    LDX #$03
00000EF0:       A900    LDA #$00
00000EF2:       952B    STA $2B,X
00000EF4:       852F    STA $2F
00000EF6:       853A    STA $3A
00000EF8:       2078F2  JSR $F278
00000EFB:       CA      DEX
00000EFC:       10F2    BPL $F2         loop back to $EEF0
00000EFE:       60      RTS




00000EFF:       A207    LDX #$07        music table?
00000F01:       A900    LDA #$00
00000F03:       852B    STA $2B
00000F05:       852D    STA $2D
00000F07:       852F    STA $2F
00000F09:       8555    STA $55
00000F0B:       BDECFD  LDA $FDEC,X
00000F0E:       8537    STA $37         music selection table
00000F10:       BDF5FD  LDA $FDF5,X
00000F13:       8538    STA $38
00000F15:       A005    LDY #$05        copy 6 bytes to $31+
00000F17:       B137    LDA ($37),Y     music phrase pointers
00000F19:       993100  STA $0031,Y
00000F1C:       88      DEY
00000F1D:       10F8    BPL $F8         loop back to $EF17
00000F1F:       A001    LDY #$01
00000F21:       A204    LDX #$04
00000F23:       A980    LDA #$80
00000F25:       952B    STA $2B,X
00000F27:       942C    STY $2C,X
00000F29:       CA      DEX
00000F2A:       CA      DEX
00000F2B:       10F8    BPL $F8         loop back to $EF25
00000F2D:       60      RTS



motor control?
set Y=$3C, A=$0A motor control?
00000F2E:       A03C    LDY #$3C
```

```
00000F30:       D00A    BNE $0A         to $EF3C
set Y=$3A, A=$0A motor control?
00000F32:       A03A    LDY #$3A
00000F34:       D006    BNE $06         to $EF3C
set Y=$38, A=$0A motor control?
00000F36:       A038    LDY #$38
00000F38:       D002    BNE $02         to $EF3C
set Y=$36, A=$0A motor control?
00000F3A:       A036    LDY #$36
00000F3C:       A90A    LDA #$0A
00000F3E:       D004    BNE $04         to $EF44
set Y=$01, A=$6F motor control?
00000F40:       A001    LDY #$01
00000F42:       A96F    LDA #$6F

00000F44:       242B    BIT $2B
00000F46:       30E5    BMI $E5         to $EF2D = RTS
00000F48:       852C    STA $2C
00000F4A:       A940    LDA #$40
00000F4C:       052B    ORA $2B         set bit 6 of $2B
00000F4E:       852B    STA $2B
00000F50:       A200    LDX #$00
00000F52:       4C22F2  JMP $F222       motor control?



00000F55:       A900    LDA #$00        zero $0400 to $04FF, also $3B and $39
00000F57:       AA      TAX
00000F58:       9D0004  STA $0400,X     zero $0400 to $04FF (tune 0)
00000F5B:       E8      INX
00000F5C:       D0FA    BNE $FA         to $EF58
00000F5E:       863B    STX $3B         zero $3B as pointer to $0400+
00000F60:       8639    STX $39
00000F62:       60      RTS

00000F63:       A52B    LDA $2B         wait for loc $2B to be zero
00000F65:       D0FC    BNE $FC         loop back to $EF63
00000F67:       60      RTS



after "notE" to display
00000F68:       A980    LDA #$80        programming the music
00000F6A:       853A    STA $3A
00000F6C:       202FEE  JSR $EE2F
00000F6F:       A53A    LDA $3A
00000F71:       4A      LSR A
00000F72:       90F8    BCC $F8         to $EF6A
00000F74:       B03A    BCS $3A         to $EFB0
00000F76:       A200    LDX #$00        "Song"
00000F78:       2084F6  JSR $F684       shift bytes to $1200 display
00000F7B:       2063EF  JSR $EF63       wait for loc $2B to be zero
00000F7E:       2403    BIT $03
00000F80:       7007    BVS $07         to $EF89 if V bit of $03 is set
00000F82:       2055EF  JSR $EF55
00000F85:       A9FE    LDA #$FE
00000F87:       853B    STA $3B         set $3B to #$FE as pointer to $0400+
00000F89:       A9FF    LDA #$FF
00000F8B:       853A    STA $3A         set $3A to $FF
00000F8D:       853C    STA $3C         set $3C to $FF
00000F8F:       A980    LDA #$80
```

```
00000F91:       8575    STA $75
00000F93:       202FEE  JSR $EE2F
00000F96:       A53C    LDA $3C
00000F98:       C90E    CMP #$0E
00000F9A:       B0F7    BCS $F7         to $EF91 if $3C < #$0E
00000F9C:       20B8F0  JSR $F0B8       add note to music memory?
00000F9F:       A63B    LDX $3B         get pointer to $0400+ from $3B
00000FA1:       E0FC    CPX #$FC
00000FA3:       D003    BNE $03         to $EFA8 if $3B <> #$FC
00000FA5:       4CECE2  JMP $E2EC       here if $3B = #$FC
                                        say "Sorry, my circuits are full"
00000FA8:       202FEE  JSR $EE2F       here if music space not full
00000FAB:       A53A    LDA $3A
00000FAD:       4A      LSR A
00000FAE:       90EC    BCC $EC         to $EF9C if LO bit of $3A = 0
00000FB0:       A53C    LDA $3C
00000FB2:       2920    AND #$20
00000FB4:       F0FA    BEQ $FA         to $EFB0 loop if bit 5 of $3C = 0
00000FB6:       A53A    LDA $3A
00000FB8:       29C0    AND #$C0        zero bits 6 and 7 of $3A
00000FBA:       853A    STA $3A
00000FBC:       A53C    LDA $3C
00000FBE:       291F    AND #$1F
00000FC0:       C90D    CMP #$0D
00000FC2:       D00A    BNE $0A         to $EFCE
00000FC4:       A539    LDA $39
00000FC6:       490C    EOR #$0C        invert bits 6 and 7 of $39
00000FC8:       8539    STA $39
00000FCA:       243A    BIT $3A
00000FCC:       70CE    BVS $CE         to $EF9C
00000FCE:       243A    BIT $3A
00000FD0:       509A    BVC $9A         to $EF6C
00000FD2:       C917    CMP #$17
00000FD4:       D0C6    BNE $C6         to $EF9C
00000FD6:       A200    LDX #$00
00000FD8:       2001EF  JSR $EF01       get pointers for programmable tune
00000FDB:       2063EF  JSR $EF63       wait for loc $2B to be zero
00000FDE:       F0BC    BEQ $BC         always to $EF9C
00000FE0:       A200    LDX #$00        "Song"
00000FE2:       2084F6  JSR $F684       serial out to $1200
00000FE5:       2063EF  JSR $EF63       wait for loc $2B to be zero
00000FE8:       2403    BIT $03
00000FEA:       7003    BVS $03
00000FEC:       2055EF  JSR $EF55
00000FEF:       20D0F0  JSR $F0D0
00000FF2:       2085F2  JSR $F285
00000FF5:       C917    CMP #$17
00000FF7:       F02C    BEQ $2C         to $F025
00000FF9:       C913    CMP #$13
00000FFB:       F069    BEQ $69         to $F066
00000FFD:       C910    CMP #$10
00000FFF:       3005    BMI $05         to $F006
00001001:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control?
00001004:       30E9    BMI $E9         to $EFEF
00001006:       A63B    LDX $3B         get pointer to $0400+ from $3B
00001008:       C90C    CMP #$0C
0000100A:       3024    BMI $24         to $F030
0000100C:       F01E    BEQ $1E         to $F02C
0000100E:       C90F    CMP #$0F
00001010:       F065    BEQ $65         to $F077
00001012:       C90E    CMP #$0E
00001014:       F07D    BEQ $7D         to $F093
00001016:       A539    LDA $39
```

```
00001018:       490C    EOR #$0C        toggle bits 6 and 7 of $39
0000101A:       8539    STA $39
0000101C:       18      CLC
0000101D:       690D    ADC #$0D
0000101F:       A8      TAY
00001020:       203CEF  JSR $EF3C
00001023:       30CA    BMI $CA         to $EFEF
00001025:       A200    LDX #$00
00001027:       2001EF  JSR $EF01       get pointers for programmable tune
0000102A:       30C3    BMI $C3         to $EFEF
0000102C:       A900    LDA #$00
0000102E:       F005    BEQ $05         to $F035
00001030:       18      CLC
00001031:       6539    ADC $39
00001033:       690D    ADC #$0D
00001035:       E0FE    CPX #$FE
00001037:       D003    BNE $03         to $FO3C
00001039:       4CECE2  JMP $E2EC
0000103C:       9D0104  STA $0401,X     store A to programmable tune
0000103F:       A8      TAY
00001040:       F003    BEQ $03         to $F045
00001042:       203CEF  JSR $EF3C
00001045:       A901    LDA #$01
00001047:       A63B    LDX $3B         get pointer to $0400+ from $3B
00001049:       9D0004  STA $0400,X     store A to programmable tune
0000104C:       20D0F0  JSR $F0D0
0000104F:       2085F2  JSR $F285
00001052:       F0AD    BEQ $AD         to $FO01
00001054:       C909    CMP #$09
00001056:       10A9    BPL $A9         to $FO01
00001058:       AA      TAX             write to 0400+ (tune 0)
00001059:       BD5BF1  LDA $F15B,X     music note durations
0000105C:       A63B    LDX $3B         get pointer to $0400+ from $3B
0000105E:       9D0004  STA $0400,X     store A to programmable tune
00001061:       2032EF  JSR $EF32       set Y=$3A, A=$0A motor control?
00001064:       3089    BMI $89         to $EFEF
00001066:       A63B    LDX $3B         get pointer to $0400+ from $3B
00001068:       F085    BEQ $85         to $F0EF
0000106A:       CA      DEX
0000106B:       CA      DEX
0000106C:       BD0004  LDA $0400,X
0000106F:       852C    STA $2C
00001071:       BD0104  LDA $0401,X
00001074:       4C85F0  JMP $F085
00001077:       A63B    LDX $3B         get pointer to $0400+ from $3B
00001079:       BD0004  LDA $0400,X
0000107C:       F012    $12
0000107E:       852C    STA $2C
00001080:       BD0104  LDA $0401,X
00001083:       E8      INX
00001084:       E8      INX
00001085:       863B    STX $3B         update $3B as pointer to $0400+
00001087:       A280    LDX #$80
00001089:       862B    STX $2B
0000108B:       A200    LDX #$00
0000108D:       201CF2  JSR $F21C       music
00001090:       4CEFEF  JMP $EFEF
00001093:       A900    LDA #$00        end table at $0400+ with 0000
00001095:       A63B    LDX $3B         get pointer to $0400+ from $3B
00001097:       F0BD    BEQ $BD         to $F056
00001099:       CA      DEX
0000109A:       9D0004  STA $0400,X
0000109D:       CA      DEX
```

```
0000109E:       9D0004  STA $0400,X
000010A1:       863B    STX $3B         update $3B as pointer to $0400+
000010A3:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000010A6:       20D0F0  JSR $F0D0
000010A9:       2085F2  JSR $F285
000010AC:       C910    CMP #$10
000010AE:       F003    BEQ $03         to $F0B3
000010B0:       4CF5EF  JMP $EFF5
000010B3:       2055EF  JSR $EF55
000010B6:       F0D8    BEQ $D8         to $EF8F
000010B8:       48      PHA
000010B9:       A63B    LDX $3B         get pointer to $0400+ from $3B
000010BB:       A539    LDA $39         (music)
000010BD:       C513    CMP $13         operand (frequency)
000010BF:       D00A    BNE $0A         to $F0CB
000010C1:       BD0004  LDA $0400,X     opcode (duration)
000010C4:       2051F1  JSR $F151       get index of duration to X
000010C7:       E412    CPX $12         index of music note duration?
000010C9:       F003    BEQ $03         to $F0CE if match
000010CB:       20D0F0  JSR $F0D0
000010CE:       68      PLA
000010CF:       60      RTS

000010D0:       48      PHA
000010D1:       8A      TXA
000010D2:       48      PHA
000010D3:       98      TYA
000010D4:       48      PHA
000010D5:       A53B    LDA $3B         get pointer to $0400+ from $3B
000010D7:       4A      LSR A           /2          (music)
000010D8:       2039F7  JSR $F739
000010DB:       2048ED  JSR $ED48
000010DE:       A524    LDA $24
000010E0:       48      PHA
000010E1:       4A      LSR A
000010E2:       4A      LSR A           /16d
000010E3:       4A      LSR A
000010E4:       4A      LSR A
000010E5:       AA      TAX
000010E6:       18      CLC
000010E7:       A539    LDA $39
000010E9:       8513    STA $13         operand?
000010EB:       F002    BEQ $02         to $F0EF if $39 = $13 = #0
000010ED:       A901    LDA #$01        decimal point
000010EF:       1DBEF8  ORA $F8BE,X     table of hex digits
000010F2:       204FED  JSR $ED4F       shift to $1200 display
000010F5:       A63B    LDX $3B         get pointer to $0400+ from $3B
000010F7:       BD0004  LDA $0400,X     operand (duration)
000010FA:       1D0104  ORA $0401,X     opcode (frequency)
000010FD:       D004    BNE $04         to $F103 if not both = 0
000010FF:       A010    LDY #$10        here if end of music (both = 0)
00001101:       D010    BNE $10         to $F113
00001103:       A00A    LDY #$0A
00001105:       BD0104  LDA $0401,X
00001108:       F009    BEQ $09         to $F113 if frequency = 0
0000110A:       38      SEC
0000110B:       E90C    SBC #$0C        loop to make negative octave
0000110D:       10FB    BPL $FB         to $F10A if frequency - #$0C > 0
0000110F:       AA      TAX
00001110:       BC51F0  LDY $F051,X     1 octave table at $F145
00001113:       68      PLA
00001114:       290F    AND #$0F        LO nybl -> X
```

```
00001116:       AA      TAX
00001117:       98      TYA
00001118:       2901    AND #$01
0000111A:       1DBEF8  ORA $F8BE,X     table of hex digits
0000111D:       204FED  JSR $ED4F       shift to $1200 display
00001120:       98      TYA
00001121:       204FED  JSR $ED4F       shift to $1200 display
00001124:       A63B    LDX $3B         get pointer to $0400+ from $3B
00001126:       BD0004  LDA $0400,X     duration
00001129:       2051F1  JSR $F151       get index of duration to X
0000112C:       8612    STX $12         index of music note duration
0000112E:       BDBFF8  LDA $F8BF,X
00001131:       C9E6    CMP #$E6        "9"
00001133:       D002    BNE $02         to $F137
00001135:       A910    LDA #$10        "_"
00001137:       204FED  JSR $ED4F       shift to $1200 display
0000113A:       A9E2    LDA #$E2        "
0000113C:       204FED  JSR $ED4F       shift to $1200 display
0000113F:       68      PLA
00001140:       A8      TAY
00001141:       68      PLA
00001142:       AA      TAX
00001143:       68      PLA
00001144:       60      RTS

00001145:       3E9C9D7A                $F051,X
00001149:       7B9E8E8F                music-related
0000114B:       F6F7EEEF                12 notes of 1 octave?

00001151:       A208    LDX #8          pointer within duration table
00001153:       DD5BF1  CMP $F15B,X      X = 1-8
00001156:       9003    BCC $03         to $F15B = RTS
00001158:       CA      DEX
00001159:       D0F8    BNE $F8         to $F153
0000115B:       60      RTS



note duration table
0000115C:       E0      whole           $F15B,X
0000115D:       A8      dotted half
0000115E:       70      half            music note durations
0000115F:       54      dotted quarter
00001160:       38      quarter
00001161:       2A      dotted eighth
00001162:       1C      eighth
00001163:       0E      sixteenth



00001164:       E0FC    CPX #$E0        music IRQ related
00001166:       F00B    BEQ $0B         to $F173
00001168:       E8      INX
00001169:       E8      INX
0000116A:       9D0104  STA $0401,X
0000116D:       FE0004  INC $0400,X
00001170:       863B    STX $3B         update $3B as pointer to $0400+
00001172:       60      RTS
00001173:       68      PLA             pop return address and discard
00001174:       68      PLA
00001175:       A200    LDX #$00
00001177:       F07C    BEQ $7C         to $F0F5
00001179:       243A    BIT $3A
```

```
0000117B:       5018    BVC $18         to $F195 return from IRQ?
0000117D:       A53C    LDA $3C
0000117F:       291F    AND #$1F
00001181:       C90E    CMP #$0E
00001183:       B010    BCS $10         to $F195 = return from IRQ?
00001185:       A63B    LDX $3B         get pointer to $0400+ from $3B
00001187:       FE0004  INC $0400,X     learn-mode music?
0000118A:       D009    BNE $09         to $F195 if $0400,X = $FF (IRQ return?)
0000118C:       DE0004  DEC $0400,X
0000118F:       BD0104  LDA $0401,X
00001192:       2064F1  JSR $F164
00001195:       4CE0FD  JMP $FDE0       return from IRQ?

00001198:       8A      TXA
00001199:       D0FA    BNE $FA         to $F195
0000119B:       243A    BIT $3A
0000119D:       10F6    BPL $F6         to $F195
0000119F:       A475    LDY $75
000011A1:       30D6    BMI $D6         to $F179
000011A3:       843C    STY $3C
000011A5:       A980    LDA #$80
000011A7:       8575    STA $75
000011A9:       C02C    CPY #$2C
000011AB:       B0E8    BCS $E8         to $F195
000011AD:       0A      ASL A           x2
000011AE:       C020    CPY #$20
000011B0:       B013    BCS $13         to $F1C0 if Y < #$20
000011B2:       C00C    CPY #$0C

000011B4:       9009    BCC $09         to $F1BF if Y >= #$0C
000011B6:       F00D    BEQ $0D         to $F1C5 if Y = #$0C
000011B8:       463A    LSR $3A
000011BA:       38      SEC             set bit 0 of $3A
000011BB:       263A    ROL $3A
000011BD:       D036    BNE $36         to $F1F5
000011BF:       98      TYA

000011C0:       18      CLC             music-related
000011C1:       6539    ADC $39
000011C3:       690D    ADC #$0D
000011C5:       A8      TAY
000011C6:       A53A    LDA $3A
000011C8:       29C0    AND #$C0        clear bits 0 thru 5 of $3A
000011CA:       853A    STA $3A
000011CC:       0A      ASL A
000011CD:       1006    BPL $06         to $F1D5
000011CF:       A63B    LDX $3B         get pointer to $0400+ from $3B
000011D1:       98      TYA
000011D2:       2064F1  JSR $F164
000011D5:       98      TYA
000011D6:       4C07F2  JMP $F207       music lookup and output to $1C00



IRQ 0,1,2
music interrupt routine
000011D9:       A556    LDA $56         entered through first 3 entries
                                        of IRQ jump table - music, 3 voices
000011DB:       A8      TAY             Y = music voice # = 0, 1, or 2
000011DC:       0A      ASL A           X=Y*2= 0, 2, or 4
000011DD:       AA      TAX
000011DE:       B975F3  LDA $F375,Y     = 1, 2 or 4
```

```
000011E1:       2555    AND $55
000011E3:       D02A    BNE $2A         to $F20F
000011E5:       B52B    LDA $2B,X
000011E7:       F0AF    BEQ $AF         to $F298
000011E9:       D62C    DEC $2C,X       duration counter
000011EB:       D01F    BNE $1F         to $F20C
000011ED:       242B    BIT $2B
000011EF:       7004    BVS $04         to $F1F5
000011F1:       A131    LDA ($31,X)     music table pointer
000011F3:       D006    BNE $06         to $F1FB if not end of music
here if end of music voice
000011F5:       A900    LDA #$00
000011F7:       952B    STA $2B,X
000011F9:       F00C    BEQ $0C         to $F207  music lookup and output to $1C00
000011FB:       952C    STA $2C,X       duration to $2C,X
000011FD:       F631    INC $31,X       increment LO byte of music pointer
000011FF:       A131    LDA ($31,X)     frequency to A
00001201:       F631    INC $31,X       increment 2 byte music pointer
00001203:       D002    BNE $02         to $F207
00001205:       F632    INC $32,X

00001207:       A656    LDX $56
00001209:       201CF2  JSR $F21C       music lookup and output to $1C00
0000120C:       4CE0FD  JMP $FDE0       return from IRQ

0000120F:       D62C    DEC $2C,X
00001211:       D0F9    BNE $F9         to $F20C return from IRQ
00001213:       B95200  LDA $0052,Y     Y = 0, 1, or 2 = music voice #
00001216:       48      PHA             jump thru table at $52 and $4F
00001217:       B94F00  LDA $004F,Y
0000121A:       48      PHA
0000121B:       60      RTS



0000121C:       A8      TAY
0000121D:       8D0016  STA $1600
00001220:       F056    BEQ $56         to $F278

motor control?
00001222:       88      DEY
00001223:       98      TYA
00001224:       A000    LDY #$00
00001226:       38      SEC
00001227:       E90C    SBC #$0C        divide by 12? music octave?
00001229:       3003    BMI $03         to $F22E
0000122B:       C8      INY
0000122C:       10F8    BPL $F8         loop back to $F226
0000122E:       8438    STY $38         $38 is dividend
00001230:       A8      TAY             remainder-12 to Y
00001231:       B9C0F1  LDA $F1C0,Y     at $F2B4
00001234:       8537    STA $37
00001236:       B9B4F1  LDA $F1B4,Y     at $F2A8
00001239:       A438    LDY $38

0000123B:       4637    LSR $37
0000123D:       6A      ROR A
0000123E:       88      DEY
0000123F:       10FA    BPL $FA         loop back to $F23B
00001241:       4A      LSR A
00001242:       4A      LSR A           /16d
00001243:       4A      LSR A
00001244:       4A      LSR A
```

```
00001245:       1DC0F2  ORA $F2C0,X
00001248:       29EF    AND #$EF
0000124A:       8D001C  STA $1C00
0000124D:       207FF2  JSR $F27F       delay
00001250:       297F    AND #$7F        clear HO bit
00001252:       8D001C  STA $1C00
00001255:       207FF2  JSR $F27F       delay
00001258:       A537    LDA $37
0000125A:       8D001C  STA $1C00
0000125D:       207FF2  JSR $F27F       delay
00001260:       BDC0F2  LDA $F2C0,X
00001263:       8D001C  STA $1C00
00001266:       60      RTS



00001267:       A200    LDX #$00
00001269:       F006    BEQ $06         to $F271
0000126B:       A201    LDX #$01
0000126D:       D002    BNE $02         to $F271
0000126F:       A202    LDX #$02
00001271:       BD72F3  LDA $F372,X     masks to clear bits
00001274:       2555    AND $55
00001276:       8555    STA $55

00001278:       BDC0F2  LDA $F2C0,X
0000127B:       090F    ORA #$0F
0000127D:       D0E4    BNE $E4         to $F263 if LO nybl not zero



(delay)
0000127F:       2082F2  JSR $F282       to following do-nothing subroutine
00001282:       EA      NOP
00001283:       EA      NOP
00001284:       60      RTS



00001285:       202FEE  JSR $EE2F
00001288:       A575    LDA $75
0000128A:       30F9    BMI $F9         loop to $F285 if $75 < 0
0000128C:       A080    LDY #$80
0000128E:       8475    STY $75
00001290:       2920    AND #$20
00001292:       D0F1    BNE $F1         to $F285
00001294:       202FEE  JSR $EE2F
00001297:       A575    LDA $75
00001299:       30F9    BMI $F9         to $F294 if $75 < 0
0000129B:       2920    AND #$20
0000129D:       F0F5    BEQ $F5         to $F294 if $75 bit 5 set
0000129F:       A575    LDA $75
000012A1:       A080    LDY #$80
000012A3:       8475    STY $75
000012A5:       291F    AND #$1F
000012A7:       60      RTS

000012A8:       60      $F1B4,Y ?
000012A9:       A060
000012AB:       60      Y < 0
```

```
000012AC:       C060
000012AE:       60
000012AF:       A040
000012B1:       00
000012B2:       00
000012B3:       40

000012B4:       77      $F1C0,Y  ?
000012B5:       706A
000012B7:       64
000012B8:       5E5954
000012BB:       4F      Y < 0
000012BC:       4B
000012BD:       47
000012BE:       43
000012BF:       3F


000012C0:       90B0    $F2C0,X
000012C2:       D0F0


000012C4:       A001    LDY #$01        music IRQ related
000012C6:       A257    LDX #$57
000012C8:       A9F3    LDA #$F3
000012CA:       D01A    BNE $1A         to $F2E6
000012CC:       A000    LDY #$00
000012CE:       A225    LDX #$25
000012D0:       A9F3    LDA #$F3
000012D2:       D012    BNE $12         to $F2E6
000012D4:       A247    LDX #$47
000012D6:       A9F3    LDA #$F3
000012D8:       8443    STY $43
000012DA:       D008    BNE $08         to $F2E4
000012DC:       A901    LDA #$01
000012DE:       8549    STA $49
000012E0:       A206    LDX #$06
000012E2:       A9F3    LDA #$F3
000012E4:       A000    LDY #$00
000012E6:       48      PHA
000012E7:       B972F3  LDA $F372,Y     bit reset masks
000012EA:       2555    AND $55
000012EC:       8555    STA $55
000012EE:       964F    STX $4F,Y
000012F0:       68      PLA
000012F1:       995200  STA $0052,Y
000012F4:       98      TYA
000012F5:       0A      ASL A
000012F6:       AA      TAX
000012F7:       A901    LDA #$01
000012F9:       952C    STA $2C,X
000012FB:       B549    LDA $49,X
000012FD:       9544    STA $44,X
000012FF:       B975F3  LDA $F375,Y     bit set masks
00001302:       0555    ORA $55
00001304:       8555    STA $55
00001306:       60      RTS


00001307:       C644    DEC $44         entered thru table @ $4F,$52
                                        from music IRQ
00001309:       1008    BPL $08         to $F313
0000130B:       A902    LDA #$02        reset $44 to 2 on underflow
```

```
0000130D:       8544    STA $44
0000130F:       C63D    DEC $3D         decrement $3D, stop at = 1
00001311:       D002    BNE $02         to $F315
00001313:       E63D    INC $3D
00001315:       38      SEC
00001316:       A53D    LDA $3D
00001318:       E938    SBC #$38
0000131A:       F00D    BEQ $0D         to $F329 if $3D = #$38
0000131C:       38      SEC
0000131D:       A950    LDA #$50
0000131F:       E53D    SBC $3D
00001321:       4A      LSR A
00001322:       852C    STA $2C
00001324:       D01D    BNE $1D         to $F341


00001326:       2031F3  JSR $F331       entered thru table @ $4F,$52
00001329:       A656    LDX $56         from music IRQ
0000132B:       2071F2  JSR $F271
0000132E:       4CE0FD  JMP $FDE0       return from IRQ?

00001331:       B543    LDA $43,X
00001333:       952C    STA $2C,X
00001335:       18      CLC
00001336:       B53D    LDA $3D,X
00001338:       753E    ADC $3E,X
0000133A:       953D    STA $3D,X
0000133C:       D644    DEC $44,X
0000133E:       D001    BNE $01         skip RTS
00001340:       60      RTS
00001341:       68      PLA             dump return address
00001342:       68      PLA
00001343:       B53D    LDA $3D,X
00001345:       4C07F2  JMP $F207

00001348:       A543    LDA $43         entered thru table @ $4F,$52
0000134A:       852C    STA $2C         from music IRQ
0000134C:       200CFA  JSR $FA0C
0000134F:       A5A4    LDA $A4
00001351:       293F    AND #$3F
00001353:       A8      TAY
00001354:       C8      INY
00001355:       98      TYA
00001356:       D0ED    BNE $ED         to $F345 = JMP $F207
00001358:       2031F3  JSR $F331
0000135B:       B549    LDA $49,X
0000135D:       9544    STA $44,X
0000135F:       B53E    LDA $3E,X
00001361:       49FF    EOR #$FF
00001363:       953E    STA $3E,X
00001365:       F63E    INC $3E,X       negate $3E,X
00001367:       D0DA    BNE $DA         to $F343
00001369:       2031F3  JSR $F331
0000136C:       D64A    DEC $4A,X
0000136E:       D0EB    BNE $EB         to $F35B
00001370:       F0B7    BEQ $B7         to $F329

masks to clear bits with AND $F372,Y
00001372:       FEFDFB
masks to set bits with OR $F375,Y
00001375:       010204

00001378:       A910    LDA #$10
```

```
0000137A:       8511    STA $11         command opcode
0000137C:       2095E5  JSR $E595
0000137F:       B005    BCS $05         to $F386
00001381:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control?
00001384:       30F2    BMI $F2         to $F378
00001386:       A513    LDA $13         operand
00001388:       0A      ASL A
00001389:       0A      ASL A           times 16d
0000138A:       0A      ASL A
0000138B:       0A      ASL A
0000138C:       8558    STA $58         point 58,59 to 05x0 (voice phrase), x =
operand
0000138E:       A905    LDA #$05
00001390:       8559    STA $59
00001392:       A00F    LDY #$0F
00001394:       A9FF    LDA #$FF        set block to $FF
00001396:       9158    STA ($58),Y
00001398:       88      DEY
00001399:       10FB    BPL $FB         to $F392
0000139B:       A90E    LDA #$0E
0000139D:       8511    STA $11         command opcode

0000139F:       2095E5  JSR $E595
000013A2:       B006    BCS $06         to $F3AA
000013A4:       E00E    CPX #$0E
000013A6:       F01A    BEQ $1A         to $F3C2 if X = $0E (clear)
000013A8:       D0F5    BNE $F5         to $F3BD
000013AA:       A000    LDY #$00
000013AC:       9158    STA ($58),Y
000013AE:       AA      TAX
000013AF:       2053F4  JSR $F453
000013B2:       207EF4  JSR $F47E
000013B5:       E658    INC $58
000013B7:       A558    LDA $58
000013B9:       490F    EOR #$0F
000013BB:       290F    AND #$0F
000013BD:       D0E0    BNE $E0         loop back to $F39F
000013BF:       6C7A00  JMP ($007A)     initially to $EOCB immediate mode
here if X=$0E (clear)
000013C2:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
000013C5:       A558    LDA $58
000013C7:       290F    AND #$0F
000013C9:       F0D4    BEQ $D4
000013CB:       C658    DEC $58
000013CD:       A000    LDY #$00
000013CF:       A9FF    LDA #$FF
000013D1:       9158    STA ($58),Y
000013D3:       30CA    BMI $CA         loop back to $F39F



output a speech sound with index X
000013D5:       6C9200  JMP ($0092)     initially to $F3D8

000013D8:       E08C    CPX #$8C        max number of sounds
000013DA:       F029    BEQ $29         to $F405
000013DC:       A900    LDA #$00
000013DE:       2065F4  JSR $F465       clock 1 parallel nybl (LO nybl of A) to
speech chip
000013E1:       BD67F5  LDA $F567,X     2 tables of $8C (140d) items
000013E4:       2060F4  JSR $F460       clock 1 parallel nybl (HO nybl of A) to
speech chip
```

```
000013E7:       98      TYA             with 2 bytes from tables,
000013E8:       2065F4  JSR $F465       clock 1 parallel nybl (LO nybl of A) to
speech chip
000013EB:       BDDBF4  LDA $F4DB,X     to $1400 LO bits
000013EE:       2060F4  JSR $F460       clock 1 parallel nybl (HO nybl of A) to
speech chip
000013F1:       98      TYA
000013F2:       290F    AND #$0F        LO nybl
000013F4:       A080    LDY #$80
000013F6:       8457    STY $57
000013F8:       A8      TAY
000013F9:       8D0014  STA $1400       switch bit 7 of $1400 to 1 then 0
000013FC:       0980    ORA #$80
000013FE:       8D0014  STA $1400
00001401:       8C0014  STY $1400
00001404:       60      RTS

00001405:       A97A    LDA #$7A
00001407:       855A    STA $5A
00001409:       A940    LDA #$40
0000140B:       055B    ORA $5B
0000140D:       D02A    BNE $2A         to $F439 - STA $5B, RTS

0000140F:       A900    LDA #$00
00001411:       855B    STA $5B
00001413:       8A      TXA
00001414:       29F0    AND #$F0
00001416:       D00E    BNE $0E         to $F426 if phrase # > $0F
00001418:       8A      TXA
00001419:       0A      ASL A           x 16
0000141A:       0A      ASL A
0000141B:       0A      ASL A
0000141C:       0A      ASL A
0000141D:       8558    STA $58
0000141F:       A905    LDA #$05        point $58,$59 to $0500+?*$10 (speech phrases
in RAM)
00001421:       8559    STA $59
00001423:       4C37F4  JMP $F437       return with $5B = $80 speech flag
here if ROM phrase
00001426:       A9F3    LDA #$F3
00001428:       8558    STA $58         point $58,$59 to $F5F3 (speech phrases in
ROM)
0000142A:       A9F5    LDA #$F5
0000142C:       8559    STA $59
0000142E:       A000    LDY #$00        Y = 0
00001430:       8A      TXA
00001431:       38      SEC
00001432:       E910    SBC #$10        subtract $10 from phrase #
00001434:       AA      TAX
00001435:       D005    BNE $05         to $F43C if ROM phrase not #00
00001437:       A980    LDA #$80
00001439:       855B    STA $5B         return with $5B = $80 speech flag
0000143B:       60      RTS
here if ROM phrase not #00
0000143C:       B158    LDA ($58),Y
0000143E:       E658    INC $58
00001440:       D002    BNE $02         loop until $FF at end of phrase
00001442:       E659    INC $59
00001444:       C9FF    CMP #$FF        inc $58,58 to end of speech segment
00001446:       D0F4    BNE $F4         to $F43C
00001448:       CA      DEX
00001449:       10EA    BPL $EA         back to $F437 - return with $5B = $80
0000144B:       A613    LDX $13         operand
```

```
0000144D:       A511    LDA $11         command opcode
0000144F:       C982    CMP #$82
00001451:       D0BC    BNE $BC         loop back to $F40F if opcode not #$82

(speak)
00001453:       A900    LDA #$00        here if opcode = #$82
00001455:       855B    STA $5B
00001457:       20D5F3  JSR $F3D5       output a sound with index X
0000145A:       A920    LDA #$20
0000145C:       055B    ORA $5B         set bit 5 of $5B in A
0000145E:       D0D9    BNE $D9         always to $F439 - return with $5B bit 5 set

clock 1 parallel nybl (HO nybl of A) to speech chip
00001460:       A8      TAY
00001461:       4A      LSR A
00001462:       4A      LSR A           /16d
00001463:       4A      LSR A
00001464:       4A      LSR A
clock 1 parallel nybl (LO nybl of A) to speech chip
00001465:       290F    AND #$0F
00001467:       8D0014  STA $1400
0000146A:       0910    ORA #$10
0000146C:       8D0014  STA $1400       switch bit 4 of $1400 to 1 and back to 0
0000146F:       29EF    AND #$EF        to clock out the LO 4 bits in parallel
00001471:       8D0014  STA $1400
00001474:       60      RTS



if SPon, then speak phrase
00001475:       A902    LDA #$02
00001477:       2403    BIT $03
00001479:       D007    BNE $07         to $F482 = RTS if bit 1 of $03 set
0000147B:       200FF4  JSR $F40F       speech
0000147E:       A55B    LDA $5B         wait for $5B to zero
00001480:       D0FC    BNE $FC         to $F47E if not end of speech
00001482:       60      RTS



IRQ 7
speech IRQ routine
00001483:       2C0012  BIT $1200       entered through IRQ jump table
00001486:       102D    BPL $2D         when $56 = 7  (speech)
00001488:       2457    BIT $57
0000148A:       304C    BMI $4C         to $F4D8 return from IRQ
0000148C:       A920    LDA #$20
0000148E:       245B    BIT $5B
00001490:       500A    BVC $0A         to $F49C
00001492:       C65A    DEC $5A
00001494:       D042    BNE $42         to $F4D8 return from IRQ
00001496:       A55B    LDA $5B
00001498:       29BF    AND #$BF
0000149A:       703A    BVS $3A         to $F4D6
0000149C:       D033    BNE $33         to $F4D1
0000149E:       1038    BPL $38         to $F4D8 return from IRQ
000014A0:       A000    LDY #$00
000014A2:       B158    LDA ($58),Y
000014A4:       E658    INC $58
000014A6:       D002    BNE $02         increment $58,$59
000014A8:       E659    INC $59
000014AA:       C9FF    CMP #$FF
```

```
000014AC:       F023    BEQ $23         to $F4D1 if end of speech
here if not end of speech
000014AE:       AA      TAX
000014AF:       20D5F3  JSR $F3D5       output a sound with index X
000014B2:       4CD8F4  JMP $F4D8       return from IRQ


000014B5:       C65D    DEC $5D
000014B7:       D012    BNE $12         to $F4CB
000014B9:       A910    LDA #$10
000014BB:       855D    STA $5D         if $5D = 0, reset to $10
000014BD:       A55C    LDA $5C
000014BF:       8D0016  STA $1600
000014C2:       0A      ASL A
000014C3:       455C    EOR $5C
000014C5:       49FF    EOR #$FF
000014C7:       0A      ASL A           x4
000014C8:       0A      ASL A
000014C9:       265C    ROL $5C
000014CB:       A900    LDA #$00
000014CD:       8557    STA $57
000014CF:       1007    BPL $07         to $F4D8 return from IRQ

to here if end of speech
000014D1:       A900    LDA #$00
000014D3:       8D0016  STA $1600
000014D6:       855B    STA $5B         zero speech flag
000014D8:       4CE0FD  JMP $FDE0       return from IRQ


000014DB:       E939A0EF        140d speech code commands,      ZERO  $00 = $7,7,E,9
000014DF:       AE2CCA53        2nd byte                        ONE   $01 = $6,9,3,9
000014E3:       F759561D                                        TWO   $02 = $4,5,A,0
000014E7:       B5A15C46                                        THREE $03 = $6,9,E,F
000014EB:       9FD34800                                        FOUR  $04 = $0,7,A,E
000014EF:       BED9782F                                        FIVE  $05 = $6,F,2,C
000014F3:       8ABA0160                                        SIX   $06 = $6,F,C,A
000014F7:       B60935FC                                        SEVEN $07 = $7,0,5,3
000014FB:       F1EADBD1                                        EIGHT $08 = $7,0,F,7
000014FF:       DE2ECB73                                        NINE  $09 = $7,1,5,9
00001503:       2656E457
00001507:       B5FCC24B
0000150B:       A86EAB8C
0000150F:       B7B385C8
00001513:       042B65D4
00001517:       ACD256C4
0000151B:       EE008DB7
0000151F:       235AC3BC
00001523:       6BA76762
00001527:       36352EB3
0000152B:       5259F88D
0000152F:       EDB42497
00001533:       76F6CEE1
00001537:       7ADFC0F0
0000153B:       D8305423
0000153F:       BB59E177
00001543:       7BE90D3F
00001547:       380F0FCB
0000154B:       A940A4B5
0000154F:       367927D6
00001553:       993C9D00
00001557:       81FF2B92
0000155B:       35465383
```

```
0000155F:       800966F0
00001562:       673624AF

00001567:       77694569        140d speech code commands,
0000156B:       076F6F70        1st byte
0000156F:       70717273
00001573:       73747577
00001577:       75767640
0000157B:       53584A57
0000157F:       594B0B68
00001583:       065A4904
00001587:       6D446264
0000158B:       650E0A5D
0000158F:       620F497B
00001593:       5F4A464B
00001597:       670B7B4F
0000159B:       5C7A5678
0000159F:       4246596B
000015A3:       635E647D
000015A7:       5500526C
000015AB:       07060550
000015AF:       0D0E0263
000015B3:       475E5F42
000015B7:       5B6E5661
000015BB:       02437C00
000015BF:       015B7947
000015C3:       4D660764
000015C7:       5A48585E
000015CB:       0C514004
000015CF:       4C4C5344
000015D3:       55527D08
000015D7:       03437848
000015DB:       4E400809
000015DF:       054F576B
000015E3:       0A406C4E
000015E7:       66655054
000015EB:       6A7F6D60
000015EF:       41790C02

ROM phrases
000015F3:       418C44          Hello (pause) I                 $10
000015F6:       19546E8C        am Maxx Steele (pause)
000015FA:       FF
000015FB:       64294279        Please choose how tough         $11
000015FF:       FF
00001600:       64293A          Please choose game              $12
00001603:       FF
00001604:       3D63            Good play                       $13
00001606:       FF
00001607:       738A            Thank you                       $14
00001609:       FF
0000160A:       4A761C77        Is there any thing              $15
0000160E:       44263104        I can do for
00001612:       8A              you
00001613:       FF
00001614:       3D5649          Good morn ing                   $16
00001617:       FF
00001618:       4B4A7802        It is time to                   $17
0000161C:       3B7C            get up
0000161E:       FF
0000161F:       546E878B        Maxx Steele win zz              $18
00001623:       FF
00001624:       2C8C8A87        Congratulations (pause) you win $19
```

```
00001628:       FF
00001629:       4459348C        I need energy (pause)           $1A
0000162D:       64672755        please re charge me
00001631:       FF
00001632:       3A608C          Game over (pause)               $1B
00001635:       FF
00001636:       293502          Choose enter to                 $1C
00001639:       6315            play again
0000163B:       FF
0000163C:       6C8C57          Sorry (pause) my                $1D
0000163F:       2A1E39          circuits are full
00001642:       FF
00001643:       64              Please                          $1E
00001644:       7155            teach me
00001646:       FF
00001647:       64              Please                          $1F
00001648:       6555            program me
0000164A:       FF
0000164B:       4668            I'm ready                       $20
0000164D:       FF

program step number to display
0000164E:       A50F    LDA $0F         program counter LO byte
00001650:       38      SEC
00001651:       E980    SBC #$80
00001653:       8513    STA $13
00001655:       A510    LDA $10         program counter HO byte
00001657:       E901    SBC #$01        subtract beg location
00001659:       4A      LSR A           /2
0000165A:       6613    ROR $13         program step # in $13
0000165C:       2037F7  JSR $F737
0000165F:       2048ED  JSR $ED48       to display
00001662:       A200    LDX #$00
00001664:       8A      TXA
00001665:       4CEBF6  JMP $F6EB       should be $F6DB??????

00001668:       A2F0    LDX #$F0
0000166A:       D010    BNE $10         to $F67C



execute one program command
0000166C:       2037F7  JSR $F737       calc program step
0000166F:       A411    LDY $11         command opcode
00001671:       A20E    LDX #$0E
00001673:       C8      INY
00001674:       F00E    BEQ $0E         to $F684 if opcode = $FF, "End" to display,
X = $0E
00001676:       E8      INX
00001677:       C8      INY
00001678:       F00A    BEQ $0A         to $F684 if opcode = $FE, "beg" to display,
X = $0F
0000167A:       A2FF    LDX #$FF

0000167C:       A511    LDA $11         command opcode
0000167E:       C90B    CMP #$0B        HOME command?
00001680:       D022    BNE $22         to $F6A4 if not HOME command
to here if home command
00001682:       A210    LDX #$10        here if = $0B  "init" to display
00001684:       2048ED  JSR $ED48
00001687:       BD58F8  LDA $F858,X     first byte, set 1
0000168A:       204FED  JSR $ED4F       shift to $1200 display
0000168D:       BD38F8  LDA $F838,X     second byte, set 1
```

```
00001690:       204FED  JSR $ED4F       shift to $1200 display
00001693:       BD18F8  LDA $F818,X     third byte, set 1
00001696:       204FED  JSR $ED4F       shift to $1200 display
00001699:       BDF8F7  LDA $F7F8,X     fourth byte, set 1
0000169C:       204FED  JSR $ED4F       shift to $1200 display
0000169F:       A9E2    LDA #$E2
000016A1:       4C4FED  JMP $ED4F       shift to $1200 display

(here if opcode not $FE, $FF, OR $0B)
000016A4:       A511    LDA $11         command opcode
000016A6:       1005    BPL $05         to $F6AD if HO bit of $11 = zero
000016A8:       2907    AND #$07        here if bit 7 set in opcode
000016AA:       18      CLC
000016AB:       690C    ADC #$0C        $80-$87 become $0C-$13
000016AD:       8512    STA $12         table index
000016AF:       2048ED  JSR $ED48
000016B2:       A412    LDY $12
000016B4:       B978F8  LDA $F878,Y     first byte, set 2
000016B7:       204FED  JSR $ED4F       shift to $1200 display
000016BA:       B9AAF8  LDA $F8AA,Y     second byte, set 2
000016BD:       F02F    BEQ $2F         to $F6EE
000016BF:       B997F8  LDA $F897,Y     third byte, set 2
000016C2:       204FED  JSR $ED4F       shift to $1200 display
000016C5:       B9AAF8  LDA $F8AA,Y     fourth byte, set 2
000016C8:       3039    BMI $39         to $F703
000016CA:       B989F8  LDA $F889,Y     fifth byte, set 2
000016CD:       204FED  JSR $ED4F       shift to $1200 display
000016D0:       F048    BEQ $48         to $F71A
000016D2:       A2FF    LDX #$FF
000016D4:       3002    BMI $02         to $F6D8
000016D6:       A200    LDX #$00
000016D8:       2048ED  JSR $ED48
000016DB:       A525    LDA $25         DISPLAY ($25) IN HEX
000016DD:       4A      LSR A
000016DE:       4A      LSR A           /16d
000016DF:       4A      LSR A
000016E0:       4A      LSR A           HO nyble of $25 in A
000016E1:       D004    BNE $04         to $F6E7
000016E3:       E0FF    CPX #$FF
000016E5:       F004    BEQ $04         to $F6EB
000016E7:       AA      TAX
000016E8:       BDBEF8  LDA $F8BE,X     hex char table for display

000016EB:       204FED  JSR $ED4F       shift to $1200 display
000016EE:       E0F0    CPX #$F0
000016F0:       F037    BEQ $37         to $F729
here if X <> $F0
000016F2:       A525    LDA $25
000016F4:       290F    AND #$0F        LO nybl of $25 in A
000016F6:       D004    BNE $04         to $F6FC
000016F8:       E0FF    CPX #$FF
000016FA:       F004    BEQ $04         to $F700
000016FC:       AA      TAX
000016FD:       BDBEF8  LDA $F8BE,X     hex char table for display
00001700:       204FED  JSR $ED4F       shift to $1200 display
00001703:       E0F0    CPX #$F0
00001705:       F027    BEQ $27         to $F72E
00001707:       A524    LDA $24         DISPLAY ($24) IN HEX
00001709:       4A      LSR A
0000170A:       4A      LSR A           /16d
0000170B:       4A      LSR A
0000170C:       4A      LSR A           HO nyble of $24 in A
0000170D:       D004    BNE $04         to $F713
```

```
0000170F:       E0FF    CPX #$FF
00001711:       F004    BEQ $04         to $F717
00001713:       AA      TAX
00001714:       BDBEF8  LDA $F8BE,X     hex char table for display
00001717:       204FED  JSR $ED4F       shift to $1200 display
0000171A:       E0F0    CPX #$F0
0000171C:       F015    BEQ $15         to $F733
0000171E:       A524    LDA $24
00001720:       290F    AND #$0F        LO nyble of $24 in A
00001722:       AA      TAX
00001723:       BDBEF8  LDA $F8BE,X     hex char table for display
00001726:       4C9CF6  JMP $F69C
here if x = $F0
00001729:       A910    LDA #$10        "_"
0000172B:       204FED  JSR $ED4F       shift to $1200 display
0000172E:       A910    LDA #$10        "_"
00001730:       204FED  JSR $ED4F       shift to $1200 display
00001733:       A910    LDA #$10
00001735:       D0EF    BNE $EF         to $F726 = JMP $F69C

00001737:       A513    LDA $13         program step #
00001739:       8524    STA $24
0000173B:       0A      ASL A           *2
0000173C:       A900    LDA #$00
0000173E:       2A      ROL A
0000173F:       8525    STA $25         program byte # in $24,25
00001741:       F8      SED             decimal mode
00001742:       A007    LDY #$07
00001744:       0624    ASL $24
00001746:       9007    BCC $07         to $F74F
00001748:       7956F7  ADC $F756,Y
0000174B:       9002    BCC $02         to $F74F
0000174D:       E625    INC $25
0000174F:       88      DEY
00001750:       10F2    BPL $F2         to $F744
00001752:       8524    STA $24
00001754:       D8      CLD             exit decimal mode
00001755:       60      RTS

00001756:       00010307                decimal value bytes
0000175A:       15316326



0000175E:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control? stop motors?

00001761:       200DE6  JSR $E60D       wait for key < $20
00001764:       C910    CMP #$10
00001766:       3003    BMI $03         to $F76B if key < $10
00001768:       4C7FE1  JMP $E17F
here if key < $10
0000176B:       C908    CMP #$08
0000176D:       10EF    BPL $EF         to $F75E if key >= $08
here if key < 8
0000176F:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
00001772:       A2E8    LDX #$E8
00001774:       A9E0    LDA #$E0
00001776:       A415    LDY $15         key
00001778:       D005    BNE $05         to $F77F if key <> 0
here if key = 0
0000177A:       20D9F7  JSR $F7D9
0000177D:       10E2    BPL $E2         loop back to $F761
```

```
here if key <> 0
0000177F:       CA      DEX
00001780:       88      DEY
00001781:       F0F7    BEQ $F7         to $F77A if key = 1
00001783:       A9E1    LDA #$E1
00001785:       A2EA    LDX #$EA
00001787:       88      DEY
00001788:       D005    BNE $05         to $F78F if key > 2
here if key = 2 or 3
0000178A:       20D9F7  JSR $F7D9
0000178D:       102A    BPL $2A         to $F7B9
here if key > 2
0000178F:       CA      DEX
00001790:       88      DEY
00001791:       F0F7    BEQ $F7         to $F78A if key = 3
00001793:       88      DEY
00001794:       D007    BNE $07         to $F79D if key > 4
here if key = 4
00001796:       A9E0    LDA #$E0        "7"
00001798:       204FED  JSR $ED4F       shift to $1200 display
0000179B:       F0C4    BEQ $C4         to $F761
0000179D:       A9E1    LDA #$E1        "7."
0000179F:       88      DEY
000017A0:       F014    BEQ $14         to $F7B6 if key = 5
her if key > 5
000017A2:       88      DEY
000017A3:       D022    BNE $22         to $F7C7 if key > 6
here if key = 6
000017A5:       204FED  JSR $ED4F       shift to $1200 display
000017A8:       A504    LDA $04
000017AA:       4980    EOR #$80        invert bit 7 of $04
000017AC:       8504    STA $04
000017AE:       3004    BMI $04         to $F7B2
000017B0:       A9EE    LDA #$EE
000017B2:       D002    BNE $02         to $F7B6
000017B4:       A9EB    LDA #$EB        "9."
000017B6:       204FED  JSR $ED4F       shift to $1200 display
000017B9:       A575    LDA $75         wait until HO bit of $75 = 0
000017BB:       30FC    BMI $FC         to $F7B9
000017BD:       A980    LDA #$80
000017BF:       8575    STA $75
000017C1:       20ECE3  JSR $E3EC
000017C4:       4C96F7  JMP $F796
here if key > 6
000017C7:       A211    LDX #$11        "AEoF"
000017C9:       A503    LDA $03
000017CB:       4910    EOR #$10        invert bit 4 of $03
000017CD:       8503    STA $03         Toggle automatic execution bit
000017CF:       2910    AND #$10        0 = off
000017D1:       F001    BEQ $01         to $F7D4
000017D3:       E8      INX             "AEon"
000017D4:       2084F6  JSR $F684       serial out to $1200 display
000017D7:       F0E4    BEQ $E4         back to $F7BD

000017D9:       204FED  JSR $ED4F       shift to $1200 display
000017DC:       A903    LDA #$03
000017DE:       8516    STA $16
000017E0:       8A      TXA
000017E1:       204FED  JSR $ED4F       shift to $1200 display
000017E4:       A940    LDA #$40
000017E6:       A416    LDY $16
000017E8:       F003    BEQ $03         to $F7ED
000017EA:       C616    DEC $16
```

```
000017EC:       0A      ASL A
000017ED:       852A    STA $2A         gets decremented in IRQ routine
000017EF:       A52A    LDA $2A         wait for $2A or bit 8 of $75 to zero
000017F1:       F0ED    BEQ $ED         to $F7E0
000017F3:       2475    BIT $75
000017F5:       30F8    BMI $F8         to $F7EF
000017F7:       60      RTS
```

(segments in display for 32 different messages)			Index	Display
000017F8:	F69E1E1E	32d bytes, set 1, fourth	00	Song
000017FC:	9E9C2AF6		01	SPEE
00001800:	2A760ACE		02	SEnt
00001804:	1C1C7AF6		03	StAt
00001808:	1E8E2A0A		04	notE
0000180C:	0A2A8E8E		05	CLOC
00001810:	2A2A8E2A		06	LErn
00001814:	8E2A8E7A		07	Prog
			08	run
00001818:	C49E2AEE	32d bytes, set 1, third	09	PLAy
0000181C:	1EFC0AC6		0A	CLr
00001820:	38EE1C0A		0B	CLrP
00001824:	1CEE2A9E		0C	FULL
00001828:	203A3A9E		0D	StAL
0000182C:	EE3A3A3A		0E	End
00001830:	3A3A3A3A		0F	beg
00001834:	3A3A3AEE		10	init
			11	AEof
00001838:	C6CE9E1E	32d bytes, set 1, second	12	AEon
0000183C:	3A1C9E84		13	SEr
00001840:	0A1C9C1C		14	PAr
00001844:	7C1E9E3E		15	Pdon
00001848:	2A9E9EB6		16	PdoF
0000184C:	CE7A7A7A		17	EdoF
00001850:	7A9C9C3E		18	Edon
00001854:	3ECECE3E		19	UCon
			1A	UCoF
00001858:	B6B6B6B6	32d bytes, set 1, first	1B	Ebon
0000185C:	2A9C1CCE		1C	EboF
00001860:	00CE009C		1D	SPon
00001864:	8EB60000		1E	SPoF
00001868:	20EEEE00		1F	bAd
0000186C:	00CECE9E			
00001870:	9E7C7C9E			
00001874:	9EB6B600			

				Index	Display
Function					
00001878:	1C	17d bytes, set 2, first	L	00	L
Drive left					
00001879:	8E3E0A		F b r	01	F
Drive forward					
0000187C:	7C		U	02	b
Drive reverse					
0000187D:	7C		U	03	r
Drive right					
0000187E:	EEEE9C		A A C	04	Uu
Wrist up					
00001881:	9C		C	05	Ud
Wrist down					

```
00001882:       6E007A                                  H   d       06      Au      Arms
up
00001885:       B6B6                                    S S         07      Ad      Arms
down
00001887:       B6CE                                    S P         08      Cr      Claw
rotate
                                                                    09      Cc      Claw
close  or open
00001889:       00      14d bytes, set 2, fifth                     0A      HL      Lamp
on or off
0000188A:       00                                                  0B
0000188B:       00                                                  0C      d
delay
0000188C:       00                                                  0D      Sn      Song
number
0000188D:       00                                                  0E      S
Speech
0000188E:       00                                                  0F      SS
Speech?
0000188F:       00                                                  10      PS
Program speech
00001890:       00
00001891:       00
00001892:       3A                                      o
00001893:       00
00001894:       00
00001895:       00
00001896:       00

00001897:       00      19d bytes, set 2, third
00001898:       00
00001899:       00
0000189A:       00
0000189B:       38                                      u
0000189C:       7A                                      d
0000189D:       38                                      u
0000189E:       7A                                      d
0000189F:       0A                                      r
000018A0:       1A                                      c
000018A1:       1C                                      L
000018A2:       00
000018A3:       00
000018A4:       2A                                      n
000018A5:       00
000018A6:       B6B6                                    S S
000018A8:       00
000018A9:       00

000018AA:       80      20d bytes, set 2, second and fourth
000018AB:       00
000018AC:       00
000018AD:       80
000018AE:       80
000018AF:       80
000018B0:       80
000018B1:       80
000018B2:       80
000018B3:       40
000018B4:       40
000018B5:       00
000018B6:       00
000018B7:       40
000018B8:       00
```

```
000018B9:       80
000018BA:       80
000018BB:       40
000018BC:       80
000018BD:       00

000018BE:       FC60DAF2                music note frequency display #          0 1
2 3
000018C2:       66B6BEE0                16d bytes, ref'd at F12E                4 5
6 7
000018C6:       FEE6EE3E                                                        8 9
A b
000018CA:       9C7A9E8E                                                        C d
E F

000018CE:       20DCEB  JSR $EBDC       games
000018D1:       2063EF  JSR $EF63       wait for loc $2B to be zero
000018D4:       A212    LDX #$12
000018D6:       20EAF9  JSR $F9EA       say "Please choose game"
000018D9:       08      PHP
000018DA:       A920    LDA #$20
000018DC:       8513    STA $13         operand
000018DE:       A905    LDA #$05
000018E0:       20FEE9  JSR $E9FE
000018E3:       28      PLP
000018E4:       F003    BEQ $03         to $F8E9 game #1
000018E6:       4CA4FA  JMP $FAA4       to game #2

(game #1)
000018E9:       20DEF9  JSR $F9DE       zero $A6 thru $AD
000018EC:       20E8F9  JSR $F9E8       choose how tough
000018EF:       86A6    STX $A6
000018F1:       A214    LDX #$14
000018F3:       200FF4  JSR $F40F       say "Thank you."
000018F6:       2001FA  JSR $FA01
000018F9:       A902    LDA #$02
000018FB:       85AC    STA $AC
000018FD:       8517    STA $17
000018FF:       A903    LDA #$03
00001901:       85A9    STA $A9
00001903:       2004E5  JSR $E504
00001906:       A914    LDA #$14        $14 seconds delay
00001908:       8527    STA $27         gets decremented in IRQ routine
0000190A:       A220    LDX #$20
0000190C:       200FF4  JSR $F40F       say "I'm ready."
0000190F:       207EF4  JSR $F47E
00001912:       20CEEB  JSR $EBCE
00001915:       A6A6    LDX $A6
00001917:       BC9CFA  LDY $FA9C,X
0000191A:       20D4F2  JSR $F2D4
0000191D:       2C0012  BIT $1200
00001920:       700A    BVS $0A         to $F92C
00001922:       200CFA  JSR $FA0C
00001925:       A527    LDA $27         gets decremented in IRQ routine
00001927:       D0F4    BNE $F4         to $F91D if timer not zero
00001929:       4CC6F9  JMP $F9C6       game over
0000192C:       20CAEB  JSR $EBCA
0000192F:       2071F2  JSR $F271
00001932:       A6A6    LDX $A6
00001934:       BDA0FA  LDA $FAA0,X
00001937:       853D    STA $3D         gets decremented in IRQ routine
00001939:       BD90FA  LDA $FA90,X     seconds delay table
0000193C:       8527    STA $27         gets decremented in IRQ routine
```

```
0000193E:       20DCF2  JSR $F2DC
00001941:       201BFA  JSR $FA1B
00001944:       A527    LDA $27         gets decremented in IRQ routine
00001946:       D0F9    BNE $F9         to $F941
00001948:       A948    LDA #$48
0000194A:       853D    STA $3D         gets decremented in IRQ routine
0000194C:       A905    LDA #$05
0000194E:       8543    STA $43
00001950:       A9FF    LDA #$FF
00001952:       853E    STA $3E
00001954:       8549    STA $49
00001956:       20CCF2  JSR $F2CC
00001959:       20CEEB  JSR $EBCE
0000195C:       A6A6    LDX $A6
0000195E:       BD94FA  LDA $FA94,X     delay table
00001961:       8528    STA $28         gets decremented in IRQ routine
00001963:       2C0012  BIT $1200
00001966:       702E    BVS $2E         to $F996
00001968:       201BFA  JSR $FA1B
0000196B:       A528    LDA $28         gets decremented in IRQ routine
0000196D:       D0F4    BNE $F4         to $F961
0000196F:       2071F2  JSR $F271
00001972:       20CAEB  JSR $EBCA
00001975:       A207    LDX #$07
00001977:       2001EF  JSR $EF01       get pointers for power-down tune
0000197A:       20D0E9  JSR $E9D0
0000197D:       A5AB    LDA $AB
0000197F:       F003    BEQ $03         to $F984
00001981:       2042FA  JSR $FA42
00001984:       2004E5  JSR $E504
00001987:       E6A9    INC $A9
00001989:       A6A6    LDX $A6
0000198B:       F002    BEQ $02         to $F98F
0000198D:       C6A6    DEC $A6
0000198F:       C6AC    DEC $AC
00001991:       3033    BMI $33         to $F9C6
00001993:       4C06F9  JMP $F906
00001996:       2071F2  JSR $F271
00001999:       20CAEB  JSR $EBCA
0000199C:       A213    LDX #$13
0000199E:       200FF4  JSR $F40F       say "Good play."
000019A1:       A5A6    LDA $A6
000019A3:       F8      SED
000019A4:       38      SEC
000019A5:       65A7    ADC $A7
000019A7:       85A7    STA $A7
000019A9:       A5A8    LDA $A8
000019AB:       6900    ADC #$00
000019AD:       85A8    STA $A8
000019AF:       D8      CLD
000019B0:       2001FA  JSR $FA01
000019B3:       C6A9    DEC $A9
000019B5:       100C    BPL $0C         to $F9C3
000019B7:       A6A6    LDX $A6
000019B9:       E003    CPX #$03
000019BB:       B006    BCS $06         to $F9C3
000019BD:       A903    LDA #$03
000019BF:       85A9    STA $A9
000019C1:       E6A6    INC $A6
000019C3:       4C2CF9  JMP $F92C
000019C6:       2001FA  JSR $FA01
000019C9:       2071F2  JSR $F271
000019CC:       A21B    LDX #$1B
```

```
000019CE:       200FF4  JSR $F40F       say "Game over,"
000019D1:       2004E5  JSR $E504
000019D4:       2067FA  JSR $FA67       "Press enter" & wait for key
000019D7:       08      PHP             here if enter pressed
000019D8:       20DCEB  JSR $EBDC
000019DB:       4CDAF8  JMP $F8DA       restart game

000019DE:       A207    LDX #$07        zero $A6 thru $AD
000019E0:       A900    LDA #$00
000019E2:       95A6    STA $A6,X
000019E4:       CA      DEX
000019E5:       10FB    BPL $FB         to $F9E2
000019E7:       60      RTS

000019E8:       A211    LDX #$11
000019EA:       200FF4  JSR $F40F       say "Please choose how tough."
000019ED:       A911    LDA #$11
000019EF:       8511    STA $11         opcode?
000019F1:       2095E5  JSR $E595
000019F4:       9004    BCC $04         to $F9FA
000019F6:       C904    CMP #$04
000019F8:       9005    BCC $05         to $F9FF if A >= 4
000019FA:       2040EF  JSR $EF40       set Y=$01, A=$6F motor control?
000019FD:       30F2    BMI $F2         loop to $F9F1
000019FF:       AA      TAX
00001A00:       60      RTS

00001A01:       A5A7    LDA $A7
00001A03:       8524    STA $24
00001A05:       A5A8    LDA $A8         copy $A7,$A8 to $24,$25
00001A07:       8525    STA $25
00001A09:       4CD2F6  JMP $F6D2

00001A0C:       A5A5    LDA $A5
00001A0E:       45A4    EOR $A4
00001A10:       49FF    EOR #$FF
00001A12:       2A      ROL A
00001A13:       2A      ROL A
00001A14:       78      SEI
00001A15:       26A4    ROL $A4
00001A17:       26A5    ROL $A5
00001A19:       58      CLI
00001A1A:       60      RTS

00001A1B:       20AFED  JSR $EDAF       check for stalled motors
00001A1E:       A9FF    LDA #$FF
00001A20:       20FEE9  JSR $E9FE
00001A23:       A90F    LDA #$0F
00001A25:       2405    BIT $05
00001A27:       D0F1    BNE $F1         to $FA1A = RTS
00001A29:       A5AB    LDA $AB
00001A2B:       D015    BNE $15         to $FA42
00001A2D:       4607    LSR $07
00001A2F:       0607    ASL $07
00001A31:       200CFA  JSR $FA0C
00001A34:       A6A6    LDX $A6
00001A36:       3D98FA  AND $FA98,X
00001A39:       85AB    STA $AB
00001A3B:       297F    AND #$7F
00001A3D:       24AB    BIT $AB
00001A3F:       4C51FA  JMP $FA51

00001A42:       4607    LSR $07
```

```
00001A44:       38      SEC
00001A45:       2607    ROL $07
00001A47:       A5AB    LDA $AB
00001A49:       297F    AND #$7F
00001A4B:       A200    LDX #$00
00001A4D:       24AB    BIT $AB
00001A4F:       86AB    STX $AB
00001A51:       8513    STA $13

00001A53:       3004    BMI $04         to $FA59
00001A55:       A903    LDA #$03
00001A57:       1002    BPL $02         to $FA5B
00001A59:       A900    LDA #$00
00001A5B:       4CFEE9  JMP $E9FE

00001A5E:       A901    LDA #$01        set bit 0 of $27 and wait for it to zero
00001A60:       8527    STA $27         gets decremented in IRQ routine
00001A62:       A527    LDA $27         gets decremented in IRQ routine
00001A64:       D0FC    BNE $FC         1 second delay
00001A66:       60      RTS

00001A67:       A56F    LDA $6F
00001A69:       8570    STA $70
00001A6B:       A980    LDA #$80
00001A6D:       8575    STA $75
00001A6F:       A21C    LDX #$1C
00001A71:       200FF4  JSR $F40F       say "Choose enter to play again."
00001A74:       A910    LDA #$10        wait time for enter key = 16 seconds?
00001A76:       8527    STA $27         gets decremented in IRQ routine
00001A78:       A527    LDA $27         gets decremented in IRQ routine
00001A7A:       F011    BEQ $11         to $7A8D = timout to immediate mode
00001A7C:       A575    LDA $75         keyboard key?
00001A7E:       30F8    BMI $F8         to $7A78 if bit 7 of $75 set
00001A80:       C90F    CMP #$0F        enter key?
00001A82:       D009    BNE $09         to $7A8D if not enter key, imm mode
here if enter key pressed
00001A84:       202EEF  JSR $EF2E       set Y=$3C, A=$0A motor control?  all motors,
0-7 in A
00001A87:       A980    LDA #$80
00001A89:       8575    STA $75
00001A8B:       0A      ASL A
00001A8C:       60      RTS
00001A8D:       6C7A00  JMP ($007A)     to $E0CB immediate mode

00001A90:       04030201                delay values = seconds

00001A94:       FFC08040                delay values

00001A98:       81838387                mask to zero bits

00001A9C:       20181410

00001AA0:       01050913

(game #2)
00001AA4:       20DEF9  JSR $F9DE       zero $A6 thru $AD, play game #2
00001AA7:       8527    STA $27
00001AA9:       20E8F9  JSR $F9E8       "Choose how tough"
00001AAC:       8A      TXA
00001AAD:       0A      ASL A
00001AAE:       0A      ASL A           x4
00001AAF:       85A6    STA $A6
00001AB1:       200CFA  JSR $FA0C
```

```
00001AB4:       4A      LSR A
00001AB5:       0908    ORA #$08
00001AB7:       85A9    STA $A9
00001AB9:       206FF2  JSR $F26F
00001ABC:       20CAEB  JSR $EBCA
00001ABF:       A527    LDA $27         gets decremented in IRQ routine
00001AC1:       D021    BNE $21         to $FAE4
00001AC3:       20CEFB  JSR $FBCE
00001AC6:       E6A6    INC $A6
00001AC8:       A018    LDY #$18
00001ACA:       20D4F2  JSR $F2D4
00001ACD:       A903    LDA #$03
00001ACF:       2060FA  JSR $FA60
00001AD2:       2067F2  JSR $F267
00001AD5:       A268    LDX #$68
00001AD7:       2053F4  JSR $F453
00001ADA:       2034FC  JSR $FC34
00001ADD:       205EFA  JSR $FA5E       set bit 0 of $27 and wait for it to zero
00001AE0:       A919    LDA #$19
00001AE2:       8527    STA $27         gets decremented in IRQ routine
00001AE4:       A900    LDA #$00
00001AE6:       85AB    STA $AB
00001AE8:       20EDFB  JSR $FBED
00001AEB:       208CFB  JSR $FB8C
00001AEE:       9010    BCC $10         to $FB00
00001AF0:       200CFA  JSR $FA0C
00001AF3:       29C0    AND #$C0
00001AF5:       D006    BNE $06         to $FAFD
00001AF7:       2012FC  JSR $FC12
00001AFA:       18      CLC
00001AFB:       9003    BCC $03         to $FB00
00001AFD:       20D6FB  JSR $FBD6
00001B00:       20BFFB  JSR $FBBF       software delay loop
00001B03:       C6A9    DEC $A9
00001B05:       D0E4    BNE $E4         to $FAEB
00001B07:       206BF2  JSR $F26B
00001B0A:       200CFA  JSR $FA0C
00001B0D:       4A      LSR A
00001B0E:       0922    ORA #$22
00001B10:       85A9    STA $A9
00001B12:       290F    AND #$0F
00001B14:       85AA    STA $AA
00001B16:       A5AB    LDA $AB
00001B18:       D023    BNE $23         to $FB3D
00001B1A:       C6AA    DEC $AA
00001B1C:       F00E    BEQ $0E         to $FB2C
00001B1E:       208CFB  JSR $FB8C
00001B21:       905C    BCC $5C         to $FB7F
00001B23:       20D6FB  JSR $FBD6
00001B26:       2021FC  JSR $FC21
00001B29:       4C7FFB  JMP $FB7F
00001B2C:       200CFA  JSR $FA0C
00001B2F:       291F    AND #$1F
00001B31:       0904    ORA #$04
00001B33:       85AA    STA $AA
00001B35:       2000FC  JSR $FC00
00001B38:       20CEEB  JSR $EBCE
00001B3B:       C6AB    DEC $AB
00001B3D:       C6AA    DEC $AA
00001B3F:       D02B    BNE $2B         to $FB6C
00001B41:       206FF2  JSR $F26F
00001B44:       20CAEB  JSR $EBCA
00001B47:       24AB    BIT $AB
```

```
00001B49:       3009    BMI $09
00001B4B:       A5A6    LDA $A6
00001B4D:       0A      ASL A
00001B4E:       0A      ASL A
00001B4F:       0A      ASL A
00001B50:       6932    ADC #$32
00001B52:       8528    STA $28         gets decremented in IRQ routine
00001B54:       A900    LDA #$00
00001B56:       85AB    STA $AB
00001B58:       200CFA  JSR $FA0C
00001B5B:       291F    AND #$1F
00001B5D:       090F    ORA #$0F
00001B5F:       85AA    STA $AA
00001B61:       20B5FB  JSR $FBB5
00001B64:       D019    BNE $19
00001B66:       A996    LDA #$96
00001B68:       852A    STA $2A         gets decremented in IRQ routine
00001B6A:       D013    BNE $13
00001B6C:       20B1FB  JSR $FBB1
00001B6F:       D002    BNE $02
00001B71:       F0AB    BEQ $AB
00001B73:       200CFA  JSR $FA0C
00001B76:       24AB    BIT $AB
00001B78:       1005    BPL $05
00001B7A:       46AB    LSR $AB
00001B7C:       2025FC  JSR $FC25
00001B7F:       20BFFB  JSR $FBBF       software delay loop
00001B82:       C6A9    DEC $A9
00001B84:       D090    BNE $90         to $FB16
00001B86:       202FEE  JSR $EE2F
00001B89:       4CB1FA  JMP $FAB1

00001B8C:       A528    LDA $28         gets decremented in IRQ routine
00001B8E:       D015    BNE $15         to $FBA5 if $28 <> 0
00001B90:       20B5FB  JSR $FBB5
00001B93:       D012    BNE $12         to $FBA7
00001B95:       A52A    LDA $2A         gets decremented in IRQ routine
00001B97:       D00C    BNE $0C         to $FBA5 if $2A <> 0
00001B99:       A996    LDA #$96        reset $2A to #$96 = 150d
00001B9B:       852A    STA $2A         gets decremented in IRQ routine
00001B9D:       20B1FB  JSR $FBB1
00001BA0:       D004    BNE $04         to $FBA6
00001BA2:       2012FC  JSR $FC12
00001BA5:       18      CLC
00001BA6:       60      RTS
00001BA7:       A219    LDX #$19
00001BA9:       E42A    CPX $2A         gets decremented in IRQ routine
00001BAB:       B002    BCS $02         to $FBAF if $2A > #$19
00001BAD:       862A    STX $2A         gets decremented in IRQ routine
00001BAF:       18      CLC
00001BB0:       60      RTS

00001BB1:       A201    LDX #$01
00001BB3:       D002    BNE $02         to $FBB7
00001BB5:       A204    LDX #$04
00001BB7:       E466    CPX $66
00001BB9:       F002    BEQ $02         to $FBBD
00001BBB:       E469    CPX $69
00001BBD:       38      SEC
00001BBE:       60      RTS

00001BBF:       A5A6    LDA $A6
00001BC1:       0A      ASL A
```

```
00001BC2:       0A      ASL A           x8
00001BC3:       0A      ASL A
00001BC4:       AA      TAX
00001BC5:       A03C    LDY #$3C
00001BC7:       88      DEY             delay loop
00001BC8:       D0FD    BNE $FD         to $FBC7
00001BCA:       E8      INX
00001BCB:       D0F8    BNE $F8         to $FBC5
00001BCD:       60      RTS

00001BCE:       A5A6    LDA $A6
00001BD0:       2039F7  JSR $F739
00001BD3:       4CD2F6  JMP $F6D2

00001BD6:       A5A6    LDA $A6
00001BD8:       4A      LSR A
00001BD9:       18      CLC
00001BDA:       6928    ADC #$28
00001BDC:       853D    STA $3D         gets decremented in IRQ routine
00001BDE:       A906    LDA #$06
00001BE0:       8543    STA $43
00001BE2:       A910    LDA #$10
00001BE4:       8549    STA $49
00001BE6:       A9FF    LDA #$FF
00001BE8:       853E    STA $3E
00001BEA:       4CCCF2  JMP $F2CC
00001BED:       A001    LDY #$01
00001BEF:       843F    STY $3F
00001BF1:       8440    STY $40
00001BF3:       A908    LDA #$08
00001BF5:       854B    STA $4B
00001BF7:       A920    LDA #$20
00001BF9:       38      SEC
00001BFA:       E5A6    SBC $A6
00001BFC:       8545    STA $45
00001BFE:       D00F    BNE $0F
00001C00:       A920    LDA #$20
00001C02:       8541    STA $41
00001C04:       A90A    LDA #$0A
00001C06:       854D    STA $4D
00001C08:       A001    LDY #$01
00001C0A:       8447    STY $47
00001C0C:       C8      INY
00001C0D:       8442    STY $42
00001C0F:       4CC6F2  JMP $F2C6

00001C12:       203AEF  JSR $EF3A       set Y=$36, A=$0A  motor control?
00001C15:       F8      SED
00001C16:       38      SEC
00001C17:       A201    LDX #$01
00001C19:       B5A7    LDA $A7,X
00001C1B:       E901    SBC #$01
00001C1D:       9010    BCC $10
00001C1F:       B00C    BCS $0C
00001C21:       A201    LDX #$01
00001C23:       D002    BNE $02
00001C25:       A200    LDX #$00
00001C27:       A901    LDA #$01
00001C29:       18      CLC
00001C2A:       F8      SED
00001C2B:       75A7    ADC $A7,X
00001C2D:       95A7    STA $A7,X
00001C2F:       D8      CLD
```

```
00001C30:       C925    CMP #$25
00001C32:       F00B    BEQ $0B
00001C34:       A5A7    LDA $A7
00001C36:       8524    STA $24         copy $A7,$A8 to $24,$25
00001C38:       A5A8    LDA $A8
00001C3A:       8525    STA $25
00001C3C:       4CD6F6  JMP $F6D6
00001C3F:       205EFA  JSR $FA5E       set bit 0 of $27 and wait for it to zero
00001C42:       CA      DEX
00001C43:       08      PHP
00001C44:       20CAEB  JSR $EBCA
00001C47:       20EAEE  JSR $EEEA
00001C4A:       A218    LDX #$18        say "Maxx Steele wins."
00001C4C:       28      PLP
00001C4D:       D001    BNE $01         to $FC50
00001C4F:       E8      INX
00001C50:       200FF4  JSR $F40F       say "Congratulations, you win."
00001C53:       C6A6    DEC $A6
00001C55:       A902    LDA #$02        three loops
00001C57:       85A9    STA $A9
00001C59:       2034FC  JSR $FC34
00001C5C:       205EFA  JSR $FA5E       set bit 0 of $27 and wait for it to zero
00001C5F:       20CEFB  JSR $FBCE
00001C62:       205EFA  JSR $FA5E       set bit 0 of $27 and wait for it to zero
00001C65:       C6A9    DEC $A9
00001C67:       10F0    BPL $F0         to $FC59
00001C69:       2067FA  JSR $FA67       "Press enter to play again"
00001C6C:       4CA4FA  JMP $FAA4       back to start of game #2


00001C6F:       A000    LDY #$00        called from IRQ routine at $FDCD
00001C71:       B667    LDX $67,Y
00001C73:       E0FF    CPX #$FF
00001C75:       F014    BEQ $14         to $FC8B
00001C77:       E8      INX
00001C78:       E0C8    CPX #$C8
00001C7A:       D00D    BNE $0D         to $FC89
00001C7C:       A920    LDA #$20
00001C7E:       196600  ORA $0066,Y
00001C81:       2093FD  JSR $FD93
00001C84:       9666    STX $66,Y
00001C86:       9665    STX $65,Y
00001C88:       CA      DEX
00001C89:       9667    STX $67,Y
00001C8B:       88      DEY
00001C8C:       1004    BPL $04         to $FC92
00001C8E:       A003    LDY #$03
00001C90:       D0DF    BNE $DF         to $FC71
00001C92:       A463    LDY $63
00001C94:       D017    BNE $17         to $FCAD
00001C96:       2C0014  BIT $1400
00001C99:       100D    BPL $0D         to $FCA8
00001C9B:       A662    LDX $62
00001C9D:       E8      INX
00001C9E:       E002    CPX #$02
00001CA0:       D008    BNE $08         to $FCAA
00001CA2:       E663    INC $63
00001CA4:       A201    LDX #$01
00001CA6:       D002    BNE $02         to $FCAA
00001CA8:       A200    LDX #$00
00001CAA:       4CADFD  JMP $FDAD
00001CAD:       C004    CPY #$04
```

```
00001CAF:       F004    BEQ $04         to $FCB5
00001CB1:       C008    CPY #$08
00001CB3:       D03F    BNE $3F         to $FCF4
00001CB5:       AD0014  LDA $1400
00001CB8:       2A      ROL A
00001CB9:       2A      ROL A
00001CBA:       4564    EOR $64
00001CBC:       2901    AND #$01
00001CBE:       B8      CLV
00001CBF:       18      CLC
00001CC0:       697F    ADC #$7F
00001CC2:       A662    LDX $62
00001CC4:       D00F    BNE $0F         to $FCD5
00001CC6:       A97F    LDA #$7F
00001CC8:       2564    AND $64         clear HO bit of $64
00001CCA:       8564    STA $64
00001CCC:       5002    BVC $02
00001CCE:       E662    INC $62
00001CD0:       E662    INC $62
00001CD2:       4CAFFD  JMP $FDAF
00001CD5:       CA      DEX
00001CD6:       D00C    BNE $0C
00001CD8:       70F6    BVS $F6         to $FCD0
00001CDA:       A564    LDA $64
00001CDC:       300E    BMI $0E         to $FCE8
00001CDE:       0980    ORA #$80
00001CE0:       8564    STA $64
00001CE2:       30EE    BMI $EE         to $FCD0
00001CE4:       5006    BVC $06         to $FCEC
00001CE6:       E663    INC $63
00001CE8:       A200    LDX #$00
00001CEA:       F039    BEQ $39         to $FD25
00001CEC:       E6A1    INC $A1
00001CEE:       A200    LDX #$00
00001CF0:       8663    STX $63
00001CF2:       F031    BEQ $31         to $FD25
00001CF4:       A662    LDX $62
00001CF6:       E002    CPX #$02
00001CF8:       D0D6    BNE $D6         to $FCCE
00001CFA:       A200    LDX #$00
00001CFC:       18      CLC
00001CFD:       2C0014  BIT $1400
00001D00:       1001    BPL $01         to $FD03
00001D02:       38      SEC
00001D03:       2664    ROL $64
00001D05:       A463    LDY $63
00001D07:       C00A    CPY #$0A
00001D09:       F004    BEQ $04         to $FD0F
00001D0B:       E663    INC $63
00001D0D:       D016    BNE $16         to $FD25
00001D0F:       8663    STX $63
00001D11:       A464    LDY $64
00001D13:       C465    CPY $65
00001D15:       D004    BNE $04         to $FD1B
00001D17:       8667    STX $67
00001D19:       F006    BEQ $06         to $FD21
00001D1B:       C468    CPY $68
00001D1D:       D009    BNE $09         to $FD28
00001D1F:       866A    STX $6A
00001D21:       8674    STX $74
00001D23:       8673    STX $73
00001D25:       4CADFD  JMP $FDAD
00001D28:       98      TYA
```

```
00001D29:       29C0    AND #$C0
00001D2B:       18      CLC
00001D2C:       2A      ROL A
00001D2D:       2A      ROL A
00001D2E:       2A      ROL A
00001D2F:       8562    STA $62
00001D31:       98      TYA
00001D32:       A006    LDY #$06
00001D34:       4A      LSR A
00001D35:       9001    BCC $01         to $FD38
00001D37:       E8      INX
00001D38:       88      DEY
00001D39:       D0F9    BNE $F9         to $FD34
00001D3B:       8A      TXA
00001D3C:       A200    LDX #$00
00001D3E:       2903    AND #$03
00001D40:       C562    CMP $62
00001D42:       F005    BEQ $05         to $FD49
00001D44:       E6A1    INC $A1
00001D46:       4CADFD  JMP $FDAD
00001D49:       A920    LDA #$20
00001D4B:       2464    BIT $64
00001D4D:       F024    BEQ $24         to $FD73
00001D4F:       A564    LDA $64
00001D51:       291F    AND #$1F
00001D53:       C566    CMP $66
00001D55:       D009    BNE $09         to $FD60
00001D57:       8665    STX $65
00001D59:       8666    STX $66
00001D5B:       CA      DEX
00001D5C:       8667    STX $67
00001D5E:       D00B    BNE $0B         to $FD6B
00001D60:       C569    CMP $69
00001D62:       D00C    BNE $0C         to $FD70
00001D64:       8668    STX $68
00001D66:       8669    STX $69
00001D68:       CA      DEX
00001D69:       866A    STX $6A
00001D6B:       0920    ORA #$20
00001D6D:       2093FD  JSR $FD93
00001D70:       4CADFD  JMP $FDAD
00001D73:       A464    LDY $64
00001D75:       C0D9    CPY #$D9
00001D77:       F015    BEQ $15         to $FD8E
00001D79:       98      TYA
00001D7A:       293F    AND #$3F
00001D7C:       E466    CPX $66
00001D7E:       D008    BNE $08         to $FD88
00001D80:       8667    STX $67
00001D82:       8465    STY $65
00001D84:       8566    STA $66
00001D86:       F022    BEQ $22         to $FDAA
00001D88:       E469    CPX $69
00001D8A:       F018    BEQ $18         to $FD34
00001D8C:       E6A2    INC $A2
00001D8E:       8662    STX $62
00001D90:       6C7A00  JMP ($007A)     to $E0CB immediate mode


00001D93:       A66F    LDX $6F
00001D95:       18      CLC
00001D96:       69FF    ADC #$FF
00001D98:       956B    STA $6B,X
```

```
00001D9A:       CA      DEX
00001D9B:       1002    BPL $02
00001D9D:       A203    LDX #$03
00001D9F:       866F    STX $6F
00001DA1:       A200    LDX #$00
00001DA3:       60      RTS

00001DA4:       866A    STX $6A
00001DA6:       8468    STY $68
00001DA8:       8569    STA $69
00001DAA:       2093FD  JSR $FD93
00001DAD:       8662    STX $62
00001DAF:       2475    BIT $75
00001DB1:       1011    BPL $11         to $FDC4 = RTS
00001DB3:       A670    LDX $70
00001DB5:       E46F    CPX $6F
00001DB7:       F00B    BEQ $0B         to $FDC4 = RTS
00001DB9:       B46B    LDY $6B,X
00001DBB:       8475    STY $75
00001DBD:       CA      DEX
00001DBE:       1002    BPL $02         to $FDC2
00001DC0:       A203    LDX #$03
00001DC2:       8670    STX $70
00001DC4:       60      RTS



00001DC5:       6C7800  JMP ($0078)     IRQ jumps to here

00001DC8:       48      PHA             initial IRQ vectors to here
00001DC9:       8A      TXA
00001DCA:       48      PHA
00001DCB:       98      TYA
00001DCC:       48      PHA
00001DCD:       206FFC  JSR $FC6F       called on every interrupt
00001DD0:       A656    LDX $56         IRQ jump index
00001DD2:       CA      DEX             bump to next task
00001DD3:       1002    BPL $02         to $FDD7
00001DD5:       A207    LDX #$07        reset $56 to 7 at underflow
00001DD7:       8656    STX $56         distribute among 8 interrupt tasks
00001DD9:       B584    LDA $84,X       sequentially, one per interrupt
00001DDB:       48      PHA             jump thru IRQ jump table
00001DDC:       B57C    LDA $7C,X
00001DDE:       48      PHA
00001DDF:       60      RTS


00001DE0:       68      PLA             entered through IRQ jump table
00001DE1:       A8      TAY             when $56 = 3 or 6
00001DE2:       68      PLA             returns from interrupt
00001DE3:       AA      TAX
00001DE4:       68      PLA
00001DE5:       40      RTI


00001DE6:       FF
00001DE7:       FF
00001DE8:       FF
00001DE9:       FF
00001DEA:       FF
00001DEB:       FF
```

```
        music selection vector table:
00001DEC:       FE0422  LO byte
00001DEF:       A2FC14
00001DF2:       54A4B0

00001DF5:       FDFEFE  HI byte
00001DF8:       FEFEFF
00001DFB:       FFFFFF


00001DFE:       0004    tune 0 = Programmable tune @ $0400+
00001E00:       12FF    only 1 voice
00001E02:       12FF


00001E04:       0AFE    tune 1 = Immediate mode tune
00001E06:       16FE
00001E08:       12FF    2 voices used

00001E0A:       1635    all tunes:      1st byte = duration
00001E0C:       162C                    2nd  byte = tone
00001E0E:       162E    three separate voices
00001E10:       1631    only two used here
00001E12:       1738
00001E14:       0000

00001E16:       1631
00001E18:       1629
00001E1A:       162C
00001E1C:       1631
00001E1E:       1735
00001E20:       0000


00001E22:       28FE    tune 2 = Learn mode tune
00001E24:       3EFE    all three voices used
00001E26:       60FE

00001E28:       8438
00001E2A:       8435
00001E2C:       8433
00001E2E:       8431
00001E30:       5833
00001E32:       2C31
00001E34:       582E
00001E36:       2C31
00001E38:       842E
00001E3A:       842C
00001E3C:       0000

00001E3E:       8435
00001E40:       8431
00001E42:       B030
00001E44:       2C2C
00001E46:       2C29
00001E48:       1622
00001E4A:       1624
00001E4C:       1625
00001E4E:       1627
00001E50:       1629
00001E52:       6E2A
00001E54:       2C2E
```

```
00001E56:       2C2A
00001E58:       2C25
00001E5A:       2C2A
00001E5C:       8429
00001E5E:       0000

00001E60:       2C25
00001E62:       2C29
00001E64:       2C2C
00001E66:       0B29
00001E68:       0B31
00001E6A:       0B29
00001E6C:       0B31
00001E6E:       0B29
00001E70:       0B31
00001E72:       0B29
00001E74:       0B31
00001E76:       0B29
00001E78:       0B31
00001E7A:       0B29
00001E7C:       0B31
00001E7E:       2C24
00001E80:       2C27
00001E82:       582C
00001E84:       2C27
00001E86:       2C25
00001E88:       161E
00001E8A:       1620
00001E8C:       1622
00001E8E:       1624
00001E90:       1625
00001E92:       1627
00001E94:       2C22
00001E96:       2C24
00001E98:       2C2A
00001E9A:       2C25
00001E9C:       2C22
00001E9E:       B025
00001EA0:       0000


00001EA2:       A8FE    tune 3 = Program mode tune
00001EA4:       CAFE    all three voices used
00001EA6:       E6FE

00001EA8:       2C25
00001EAA:       2C2C
00001EAC:       2C2A
00001EAE:       2C31
00001EB0:       2C25
00001EB2:       2C2C
00001EB2:       2C2A
00001EB6:       2C31
00001EB8:       2C25
00001EBA:       2C2C
00001EBC:       2C2A
00001EBE:       2C31
00001EC0:       4200
00001EC2:       162F
00001EC4:       4200
00001EC6:       422F
00001EC8:       0000
```

```
00001ECA:       B000
00001ECC:       2C31
00001ECE:       2C38
00001ED0:       2C36
00001ED2:       2C3D
00001ED4:       2C31
00001ED6:       2C38
00001ED8:       2C36
00001EDA:       2C3D
00001EDC:       4200
00001EDE:       1638
00001EE0:       4200
00001EE2:       4238
00001EE4:       0000

00001EE6:       FF00
00001EE8:       6100
00001EEA:       2C19
00001EEC:       2C20
00001EEE:       2C1E
00001EF0:       2C25
00001EF2:       4200
00001EF4:       1625
00001EF6:       4200
00001EF8:       4225
00001EFA:       0000


00001EFC:       02FF    tune 4 = Execute mode tune
00001EFE:       12FF    1 voice used
00001F00:       12FF

00001F02:       1325
00001F04:       1431
00001F06:       142B
00001F08:       142C
00001F0A:       132F
00001F0C:       1431
00001F0E:       1426
00001F10:       1431
00001F12:       0000


00001F14:       1AFF    tune 5 = Game mode tune
00001F16:       32FF    all three voices used
00001F18:       40FF

00001F1A:       7225
00001F1C:       2627
00001F1E:       4C28
00001F20:       4C25
00001F22:       182F
00001F24:       182E
00001F26:       182D
00001F28:       182C
00001F2A:       182B
00001F2C:       182A
00001F2E:       9831
00001F30:       0000

00001F32:       FF00
00001F34:       7900
00001F36:       183B
```

```
00001F38:       1839
00001F3A:       1838
00001F3C:       983D
00001F3E:       0000

00001F40:       FF00
00001F42:       3100
00001F44:       1823
00001F46:       1822
00001F48:       1821
00001F4A:       1820
00001F4C:       181F
00001F4E:       181E
00001F50:       9825
00001F52:       0000


00001F54:       5AFF    tune 6 = Reveillee
00001F56:       12FF    1 voice used
00001F58:       12FF

00001F5A:       3814    quarter
00001F5C:       3819    quarter
00001F5E:       1C1D    eightth
00001F60:       1C19    eightth
00001F62:       3814    quarter
00001F64:       381D    quarter
00001F66:       3819    quarter
00001F68:       1C1D    eightth
00001F6A:       1C19    eightth
00001F6C:       3814    quarter
00001F6E:       381D    quarter
00001F70:       3819    quarter
00001F72:       1C1D    eightth
00001F74:       1C19    eightth
00001F76:       3814    quarter
00001F78:       3819    quarter
00001F7A:       701D
00001F7C:       3819    quarter
00001F7E:       3814    quarter
00001F80:       3819    quarter
00001F82:       1C1D    eightth
00001F84:       1C19    eightth
00001F86:       3814    quarter
00001F88:       381D    quarter
00001F8A:       3819    quarter
00001F8C:       1C1D    eightth
00001F8E:       1C19    eightth
00001F90:       3814    quarter
00001F92:       381D    quarter
00001F94:       3819    quarter
00001F96:       1C1D    eightth
00001F98:       1C19    eightth
00001F9A:       3114
00001F9C:       0700
00001F9E:       3814    quarter
00001FA0:       A819    dotted half
00001FA2:       0000


00001FA4:       AAFF    tune 7 = Power-down tune
00001FA6:       12FF    1 voice used
00001FA8:       12FF
```

```
00001FAA:       8F16
00001FAC:       FE0F
00001FAE:       0000


00001FB0:       B6FF    tune 8 = Taps
00001FB2:       12FF    1 voice used
00001FB4:       12FF

00001FB6:       7414
00001FB8:       0A00
00001FBA:       2A14    dotted eighth
00001FBC:       FF19    max duration
00001FBE:       F919    duration extension
00001FC0:       7E14    half+sixteenth
00001FC2:       2A19    dotted eighth
00001FC4:       FF1D    max duration
00001FC6:       F91D    duration extension
00001FC8:       5414    dotted quarter
00001FCA:       5419    dotted quarter
00001FCc:       A81D    dotted half
00001FCE:       5414    dotted quarter
00001FD0:       5419    dotted quarter
00001FD2:       A81D    dotted half
00001FD4:       5414    dotted quarter
00001FD6:       5419    dotted quarter
00001FD8:       FF1D    max duration
00001FDA:       F91D    duration extension
00001FDC:       7E19    half+sixteenth
00001FDE:       2A1D    dotted eighth
00001FE0:       A820    dotted half
00001FE2:       A820    dotted half
00001FE4:       A81D    dotted half
00001FE6:       A819    dotted half
00001FE8:       FF14    max duration
00001FEA:       EF14    duration extension
00001FEC:       0A00
00001FEE:       7414
00001FF0:       0A00
00001FF2:       2A14    dotted eighth
00001FF4:       FF19    max duration
00001FF6:       FE19    duration extension
00001FF8:       0000

00001FFA:       11E0    NMI jump address
00001FFC:       41E0    RESET jump address
00001FFE:       C5FD    INT jump address
```