```
    *** JUKE_BOX ***
    B = SONGNO to be started

    PUSH BC
    CALL PT_IX_TO_SxDATA (IX set)
    POP BC
    RET if song in progress
      byte 0 := SONGNO
      set NEXT_NOTE_PTR to 1st note in song
      CALL LOAD_NEXT_NOTE
      CALL UP_CH_DATA_PTRS
      RET


    *** PLAY_SONGS ***

    set A for CH1 OFF code
    set MSN C for CH1 attenuation
    set MSN B for CH1 frequency
    IX := (PTR_TO_S_CH_1)
    (i.e., pt IX to byte 0 data area of song for CH1)
    CALL TONE_OUT
    set A for CH2 OFF code
    set MSN C for CH2 attenuation
    set MSN B for CH2 frequency
    IX := (PTR_TO_S_CH_2)
    (i.e., pt IX to byte 0 data area of song for CH2)
    CALL TONE_OUT
    set A for CH3 OFF code
    set MSN C for CH3 attenuation
    set MSN B for CH3 frequency
    IX := (PTR_TO_S_CH_3)
    (i.e., pt IX to byte 0 data area of song for CH3)
    CALL TONE_OUT
    set A for CH0 OFF code
    set MSN C for CH0 attenuation
    IX := (PTR_TO_S_CH_0)
    (i.e., pt IX to byte 0 data area of song for CH0)
    IF area INACTIVE
      turn off CH0
    ELSE
      CALL UPATNCTRL (send current atn)
      set LSN A for current ctrl data
      IF current ctrl data diff from last
        reload SAVE_CTRL
        CALL UPATNCTRL (send new ctrl)
      ENDIF
    ENDIF
    RET


    *** SND_MANAGER ***

    CALL PT_IX_TO_SxDATA, song 01
    LOOP
      RET if end of song data areas
        CALL PROCESS_DATA_AREA
        pt IX to byte 0 next song data area
    REPEAT LOOP

    *** PROCESS_DATA_AREA ***
    IX = addr byte 0 of a song data area

    CALL AREA_SONG_IS
    RET if area INACTIVE
      IF SONGNO = 62
        JP SFX+7 (RET from SFX)
      ENDIF
        CALL ATN_SWEEP
        CALL FREQ_SWEEP
        IF note over
    EFXOVER:  PUSH CH0 ! SONGNO note just over
          CALL LOAD_NEXT_NOTE
          POP CH0 ! SONGNO note just over
          IF CH0 ! SONGNO newly loaded note not =
          CH0 ! SONGNO note just over
            CALL UP_CH_DATA_PTRS
          ENDIF
        ENDIF
        RET


    *** LOAD_NEXT_NOTE ***
    IX = addr byte 0 of a song data area
    byte 0 = CH0 (or 80) ! SONGNO
    (SFX = addr of a special effect note's routine)

    PUSH 0 0 ! SONGNO (CH6 not pushed)
    deactivate area (byte 0 = FF)
    A := (NEXT_NOTE_PTR) (header new ROM note)

    CASE header type of
    1)rest:
      PUSH header of new ROM note
      set NEXT_NOTE_PTR for next note in song
      (i.e., the note after this new note)
      set bytes in song data area:
        ATN := off
        NLEN := 5 bit rest duration
        FSTEP := 0 (i.e., no freq sweep)
        ASTEP := 0 (no atn sweep)
      JP MODB0
    2)end of song:
      IF end repeat
        POP BC (B := SONGNO)
        CALL JUKE_BOX to reload 1st note of this song
        RET (to PROCESS_DATA_AREA)
      ENDIF
        PUSH inactive code
        JP MODB0
    3)special effect:
      POP IY (IY := SONGNO)
      PUSH IY to put SONGNO back on stack
      PUSH header new ROM note
      set NEXT_NOTE_PTR, bytes 1&2, to SFX
      (address special effect routine)
      set DE to SFX
      HL := addr next note in song
      (i.e, the note after this new effect note)
      PUSH IY, POP AF (A := SONGNO)
      PUSH DE, POP IY (IY := SFX)
      DE := PASS1, PUSH DE
      JP (IY), i.e., "CALL (IY)", RET to PASS1
      (SFX saves SONGNO & addr next note)
      PASS1: IY := IY + 7
      DE := MODB0, PUSH DE
      JP (IY), i.e , "CALL (IY+7)", RET to MODB0
      (SFX+7 loads initial effect data)
    4)normal note:
      CASE note type
        0)NEXT_NOTE_PTR := adr song's nxt note
          move 3 bytes ROM note data to RAM
          FSTEP := 0 (no freq sweep)
          ASTEP := 0 (no atn sweep)
          JR MODB0
        1)NEXT_NOTE_PTR := adr song's nxt note
          move 5 bytes ROM note data to RAM
          ASTEP := 0 (no atn sweep)
        2)NEXT_NOTE_PTR := adr song's nxt note
          move 5 bytes ROM note data to RAM
          FSTEP := 0 (no freq sweep)
          JR MODB0
        3)NEXT_NOTE_PTR := adr song's nxt note
          move 7 bytes ROM note data to RAM
      ENDCASE
    ENDCASE

    MODB0: PUSH IX
    POP HL to point to byte 0
    POP AF (A := header new note)
    POP BC (B := SONGNO)
    RET if header is inactive (i.e., song is over)
      IF header is for a special effect
        B := 62, the SONGNO for all effect notes
      ENDIF
      byte 0 := CH0 (from header new note) ! SONGNO (from B)
      RET
```

FIGURE 1

SxDATA

Description: Storage area for the various timers and output data for song number x. The song data areas MUST be stored in a contiguous block of CPU RAM and the data area used by song number one MUST be the first data area in the block. Song data area storage is allocated according to addresses stored in LST_OF_SND_ADDRS, a table stored in CART ROM.

Byte 0 of each data area, in addition to giving CH# and song number, can indicate two special conditions:
- byte 0 = 0FFH: song(s) using this data area are inactive
- byte 0 = 00H: indicates end of song data areas

If SONGNO = 62, the address of a special sound effect routine is stored in bytes 1 and 2.

Length in bytes: 10
Location: CPU RAM
Beginning address: pointed to by a 16 bit entry in LST_OF_SND_ADDRS

Offset	Contents B7 B6 B5 B4 B3 B2 B1 B0	Description
0	CH# / SONGNO	B7 - B6: song channel number, 0 to 3 B5 - B0: song number, 1 to 61 SONGNO = 62, snd effect adr in next 2 bytes
1	the LSB of an address...	usually, the addr of the next note in song; if SONGNO = 0FEH, this is the LSB of the addr of the special sound.effect routine
2	the MSB of an address...	usually, the addr of the next note in song; if SONGNO = 0FEH, this is the MSB of the addr of the special sound effect routine
3	F2 F3 F4 F5 F6 F7 F8 F9	bottom 8 bits of 10 bit freq data
4	ATN:CTRL or ATN:0 0 F0 F1	if CH# = 0 (noise): ATN / 0 FB SHIFT if CH# = 1 - 3 (tone): MSN = 4 bit ATN, LSN = top 2 bits freq (0 0 F0 F1)
5	NLEN	determines duration of note: if freq swept, = # of steps in the sweep if not, NLEN * 16.7ms = duration of note
6	FPS / FPSV	freq sweep duration prescaler: FPS = prescaler reload value FPSV = temp FPS variable storage
7	FSTEP	freq sweep step size: 1 to 127, -1 to -128 if FSTEP = 0, freq is not to be swept

8	ASTEP / ALEN	ALEN = # steps in atnswp: 2 - 15 (0 => 16)
		ASTEP = step size: 1 to 7, -1 to -8
		if whole byte = 00, atn not to be swept

9	APS / APSV	atn duration prescaler: APS = prescaler reload value APSV = temp APS variable storage

DURATIONS:

```
fixed frequency =       NLEN * 16.7ms
frequency sweep =       [((NLEN - 1) * FPS) + initial FPSV] * 16.7ms
duration 1st step =     initial FPSV * 16.7ms
duration all others =   FPS
  FPS: 0 to 15 (0 => 16)
  FPSV: 0 to 15 (0 => 16)
  NLEN: 0 to 255 (0 => 256)
attenuation sweep =     [((ALEN - 1) * APS) + initial APSV] * 16.7ms
duration 1st step =     initial APSV * 16.7ms
duration all others =   APS
  APS: 0 to 15 (0 => 16)
  APSV: 0 to 15 (0 => 16)
  ALEN: 0 to 15 (0 => 16)
```

Note Header

FIGURE 2

Length in bytes: 1
Location: begins each block of 1 to 10 bytes of note data in CART ROM

Offset	Contents B7 B6 B5 B4 B3 B2 B1 B0	Description
		*** REST
0	CH# 1 duration	if B7 = 1, header describes a rest: B7 -B6 = channel number, 0 - 3 B4 - B0 = duration, 1 to 31
		or, if B5 = 0, header preceeds note data or is special indicator:
		*** NOTE
0	CH# 0 0 0 0 type	note data follows: B7 - B6 = channel number, 0 - 3 B1 - B0 = note type, 0 -3.
		or *** END OF SONG / REPEAT SONG
0	CH# 0 1 R 0 0 0	if B4 = 1, end of song on channel in B7-B6: if B3 = 1, repeat song forever if B3 = 0, don't repeat
		or *** SPECIAL EFFECT
0	CH# 0 0 0 1 0 0	this note is to be "played" by a special sound effect routine whose addr is contained in the following 2 bytes

REST DURATION = duration * 16.7ms
duration: 1 to 31

FIGURE 3

LST_OF_SND_ADDRS

Description: LST_OF_SND_ADDRS is a list of the starting addresses of each song's data area and note list. They are used by JUKE_BOX as source (note list) and destination (song data area) pointers. Each song's entries are stored as follows:

Byte 1: LSB of the address of the start of that song's note list
Byte 2: MSB of the address of the start of that song's note list
Byte 3: LSB of the address of the start of that song's data area
Byte 4: MSB of the address of the start of that song's data area

The beginning address of LST_OF_SND_ADDRS is stored in a dedicated CPU RAM 16 bit word, PTR_TO_LST_OF_SND_ADDRS (xxxxH), allowing the cartridge programmer to place LST_OF_SND_ADDRS wherever desired.

NOTE: In other data structures, six bits are allocated for the song number (SONGNO). However, song numbers 0, 62, and 63 are used as special indicators, leaving song numbers 1 - 61 available. Therefore, the first entry in LST_OF_SND_ADDRS is for song number 1.

Length in bytes: 4 * total number of songs
Location: CART ROM
Beginning address: pointed to by CPU RAM word @LST_OF_SND_ADDRS (xxxxH)

Offset	Contents B7 B6 B5 B4 B3 B2 B1 B0	Description
0	LSB	of starting adr of the note list for song number 1
1	MSB	of starting adr of the note list for song number 1
2	LSB	of starting adr of the data area for song number 1
3	MSB	of starting adr of the data area for song number 1
4	LSB	of starting adr of the note list for song number 2
etc...		
4 * n	MSB	of starting adr of the data area for song number n (n = total number of songs)

FIGURE 4

Rests

Length in bytes: 2
Location: CART ROM
Beginning address: pointed to by bytes 1&2 in that song's data area

Offset	Contents B7 B6 B5 B4 B3 B2 B1 B0	Description
0	CH# | 1 | duration	if B7 = 1, header describes a rest: B4 - B0 = duration, 1 to 30

REST DURATION = duration * 16.7ms
duration: 1 to 30

Note Type 0:
fixed frequency, fixed attenuation

Length in bytes: 4
Location: CART ROM
Beginning address: pointed to by bytes 1&2 in that song's CPU RAM data area

Offset	Contents B7 B6 B5 B4 B3 B2 B1 B0	Description
0	CH# | 0 0 0 0 0 0	header
1	F2 F3 F4 F5 F6 F7 F8 F9	least significant 8 bits of 10 bit frq data
2	ATN | 0 0 F0 F1	ATN = 4 bit atn data, LSN = top 2 bits freq
3	NLEN	NLEN * 16.7ms = duration of note

NOTE DURATION = NLEN * 16.7ms
NLEN: 1 to 255 (0 =) 256)

FIGURE 7

Note Type 3:
swept frequency, swept attenuation

Length in bytes: 8
Location: CART ROM
Beginning address: pointed to by bytes 1&2 in that song's CPU RAM data area

Offset	Contents B7 B6 B5 B4 B3 B2 B1 B0	Description
0	CH# | 0 | 0 | 0 | 0 | 1 | 1	header
1	F2 F3 F4 F5 F6 F7 F8 F9	least sig 8 bits of init 10 bit frq data
2	ATN | 0 0 F0 F1	ATN = init atn data, LSN = top 2 bits frq
3	NLEN	NLEN = number of steps in the sweep
4	FPS | FPSV	freq sweep duration prescaler: FPS = prescaler reload value FPSV = initial FPSV
5	FSTEP	freq sweep step size: 1 to 127, -1 to -128

6	ASTEP | ALEN	ALEN = # steps in atnswp: 2 - 15 (0 => 16)
		ASTEP = step size: 1 to 7, -1 to -8
		if whole byte = 00, atn not to be swept

7	APS | APSV	atn duration prescaler: APS = prescaler reload value APSV = initial APSV

NOTE DURATION = [((NLEN - 1) x FPS) + initial FPSV] x 16.7ms
duration 1st step = initial FPSV x 16.7ms
duration all others = FPS
  FPS: 0 to 15 (0 => 16)
  FPSV: 0 to 15 (0 => 16)
  NLEN: 0 to 255 (0 => 256)

ATN SWEEP DURATION = [((ALEN - 1) x APS) + initial APSV] x 16.7ms
duration 1st step = initial APSV x 16.7ms
duration all others = APS
  APS: 0 to 15 (0 => 16)
  APSV: 0 to 15 (0 => 16)
  ALEN: 0 to 15 (0 => 16)

FIGURE 8

Noise notes:
special case type 2 notes

Length in bytes: 5
Location: CART ROM
Beginning address: pointed to by bytes 18Z in that song's CPU RAM data area

Offset	Contents B7 B6 B5 B4 B3 B2 B1 B0	Description
0	0 0 0 0 0 0 1 0	header (CH0 = 0, indicates noise note)
1	ATN | 0 FB SHIFT	MSN = 4 bit noise ATN data (init if swept) LSN = noise control data (SHIFT = NF0 NF1)
2	NLEN	NLEN x 16.7ms = duration of note

3	ASTEP | ALEN	ALEN = # steps in atnswp: 2 - 15 (0 => 16)
		ASTEP = step size: 1 to 7, -1 to -8
		if whole byte = 00, atn not to be swept

4	APS | APSV	atn duration prescaler: APS = prescaler reload value APSV = temp APS variable storage

NOTE DURATION = NLEN x 16.7ms
  NLEN: 1 to 255 (0 => 256)

ATN SWEEP DURATION = [((ALEN - 1) x APS) + initial APSV] x 16.7ms
duration 1st step = initial APSV x 16.7ms
duration all others = APS
  APS: 0 to 15 (0 => 16)
  APSV: 0 to 15 (0 => 16)
  ALEN: 0 to 15 (0 => 16)

FIGURE 9

Dedicated cartridge RAM locations and Special Effect format

Length in bytes: 11
Location: CPU RAM
Beginning address: 7020H

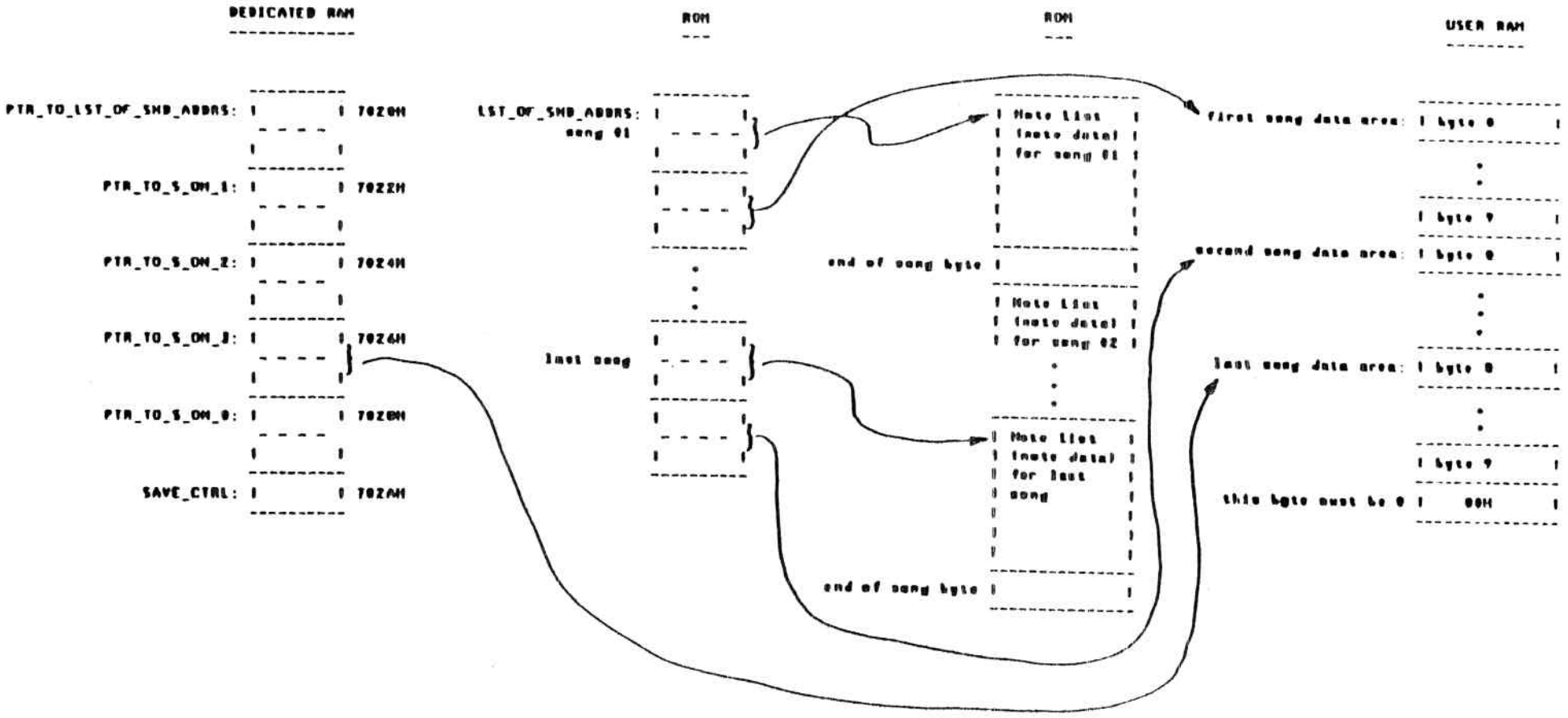
```
PTR_TO_LST_OF_SND_ADDRS   DS 2  ;pointer to start of LST_OF_SND_ADDRS

PTR_TO_S_ON_1   DS 2    ;pointer to data area of song to be played on CH# 1
PTR_TO_S_ON_2   DS 2    ;pointer to data area of song to be played on CH# 2
PTR_TO_S_ON_3   DS 2    ;pointer to data area of song to be played on CH# 3
PTR_TO_S_ON_0   DS 2    ;pointer to data area of song to be played on CH# 0

SAVE_CTRL       DS 1    ;LSN = last control data sent to noise generator
```

Special Effect format

All special effect routines should be written in the following format (SFX = the address of the effect routine, stored in ROM after the effect's header, IX is passed pointing to the song's data area):

```
SFX:    LD (SAVE_x_NNP),HL      ;save address of next note in song
        LD (SAVE_x_SONGNO),A    ;save song's SONGNO
        RET                     ;to LOAD_NEXT_NOTE
SFX47:  LD HL,SAVE_x_SONGNO     ;test for 1st pass through effect
        BIT 7,(HL)
        JR NZ,NOT_PASS_1
          SET 7,(HL)    ;to prevent further passes thru inits
          ...           ;initialize bytes within the data area here
          RET           ;to LOAD_NEXT_NOTE
NOT_PASS_1: ...         ;code for pass 2 or greater starts here,
        .               ;which algorithmically modifies freq, atn,
        .               ;or control data within song data area
        .               ;pointed to by IX
        RET (to PROCESS_DATA_AREA) if effect not over
          ;if here, effect is over, so restore SONGNO and addr nxt note in song
          LD HL,(SAVE_x_NNP)    ;HL := addr next note in song
          LD DE,SAVE_x_SONGNO   ;DE := addr saved song number
          CALL LEAVE_EFFECT     ;to restore them to bytes 0 - 2 in data area
          JP EFXOVER            ;in PROCESS_DATA_AREA to load song's next note
```

FIGURE 10


DEDICATED RAM
ROM
ROM
USER RAM
PTR_TO_LST_OF_SND_ADDRS:
78Z0H
PTR_TO_S_ON_1:
78Z2H
PTR_TO_S_ON_2:
78Z4H
PTR_TO_S_ON_3:
78Z6H
PTR_TO_S_ON_0:
78Z8H
SAVE_CTRL:
78ZAH
LST_OF_SND_ADDRS:
song 01
last song
end of song byte
Note list
(note data)
for song 01
Note list
(note data)
for song 02
Note list
(note data)
for last
song
end of song byte
first song data area:
byte 0
byte 9
second song data area:
byte 0
byte 9
last song data area:
byte 0
byte 9
this byte must be 0
00H