APPENDIX C

COLECOVISION SOUND USERS' MANUAL

Version 1.1
May 3,1982

also, half way down the page:
"...pointed to by CPU RAM word LST_OF_SND_ADDRS"
- should read -
"...pointed to by CPU RAM word PTR_TO_LST_OF_SND_ADDRS"

FIG 4 second line:
"Length in bytes: 2"
- should read -
"Length in bytes: 1"

also, about 6 lines down from that:
"B4 - B0= duration, 1 to 31"
- should read -
"B4 - B0= duration, 1 to 30"

*** NOTES ***

1) After reading through page 6 in the Users' Manual, it may be helpful to review the following summary of dedicated pointers and data structures (discussion refers to Figure 10, included as part of these notes):

DEDICATED RAM

Prior to calling any OS sound routines (except INIT_SOUND), the 11 CRAM locations 7020H through 702AH must be initialized to meaningful data. Ten of the locations are two byte pointers:

PTR_TO_LST_OF_SND_ADDRS:
7020-21H - Points to the start of a list of pointers in cartridge ROM, LST_OF_SND_ADDRS (see later). The OS sound routines know where the cartridge-dependent LST_OF_SND_ADDRS is stored through this pointer. It is shown pointing to the first byte in this ROM list (as it must).

PTR_TO_S_ON_1:
7022-23H - This and the following three pointers are used by OS sound routines to store the addresses of the four song data areas which currently contain the sound data to be modified/output to the four sound generators in the TI sound chip. This pointer stores the address for the song currently playing on tone generator #1.

-- EXAMPLE -- PTR_TO_S_ON_3 is shown pointing to the second song data area. I.e., data for the song currently playing on tone generator #3 happens to be stored in the second song data area. The second data area is used for purposes of illustration only: other songs may very well require that data for tone generator #3 be stored in a different song data area.

PTR_TO_S_ON_2:
7024-25H - As above, for tone generator #2.

PTR_TO_S_ON_3:
7026-27H - As above, for tone generator #3.

PTR_TO_S_ON_0:
7028-29H - As above, for tone generator #0 (the noise generator).

The final byte at 702AH, SAVE_CTRL, is used by the OS sound routines to store data necessary for smooth operation of the noise generator (see bottom of page 11 in the Users' Manual).

All 11 bytes should be initialised before the OS routines which operate upon them are called: this is done by calling INIT_SOUND and passing the appropriate cartridge-dependent information (see Users' Manual pages 5 and 13).

ROM

Cartridge ROM to be used by OS sound routines is divided into two sections: LST_OF_SND_ADDRS and the Note List.

LST_OF_SND_ADDRS:
A contiguous list of 4 bytes per each song (or special effect) used by the game. In each 4 byte section, the first 2 bytes are a pointer to the beginning of the song's note list (also in ROM). The second two bytes are a pointer to the song data area in RAM to be used by that song (review pages 2 and 3 in the Manual). Of course, another song may also use the same data area, since there can be (and usually are) more songs than there are data areas. NOTE, however, that Song #1 MUST be the first entry in LST_OF_SND_ADDRS for the OS routines to operate properly.

-- EXAMPLE -- The first two bytes in this list are a pointer to song #1's note list, as they MUST be. The second two bytes are a pointer to the song's data area in RAM, shown here pointing to the first song data area as also MUST be the case (see later). As can also be seen from the figure, the last song happens to use the second song data area.

In summary, there is a note list for every note list pointer in LST_OF_SND_ADDRS, but, since songs may share RAM data areas, there are almost always more data areas than there are data area pointers.

NOTE LIST:

The Note List is a contiguous block of ROM containing the data which comprise the notes of each song. The number of bytes per song of course varies with the length of the song. The first byte of the note list for a song is pointed to by a two byte entry in LST_OF_SND_ADDRS (see above). The last byte of each song's note list is a single byte end of song/repeat code (see page 2 and Figure 2 in the Users' Manual).

USER RAM

This is the area in CRAM that the cartridge programmer has chosen to hold the ten byte song data areas which contain sound and timing information to be processed by the OS sound routines. These data areas must be stored as contiguous blocks of ten bytes each. In all cases but one, the programmer may choose to "play" a song in any data area; however, song #1 MUST use the FIRST song data area for the OS routines to work properly. Also, the byte immediately following the last byte in the last data area MUST be zero (this code tells the OS routine SND_MANAGER to stop looking for more song data areas; see bottom of page 5, Users' Manual). This byte will automatically be set to zero by proper invocation of the INIT_SOUND routine before any other OS sound routines are called (see pages 5 and 13).

2) The ColecoVision OS entry point names of some of the sound routines and dedicated locations are different from their names given in the Sound Users' Manual. They are:

Entry Point	Sound Users' Manual
PLAY_IT	JUKE_BOX
SOUND_MAN	SND_MANAGER
SOUND_INIT	INIT_SOUND
NOTES	PTR_TO_LST_OF_SND_ADDRS

You should use the entry point names.

3) Page 22, last paragraph: It is mentioned that a special effect routine may want to call the OS sound routines FREQ_SWEEP and ATN_SWEEP to operate upon data within the effect's data area, which "require that data be ordered appropriately within a song data area". This means:

Whether or not the special effect uses FREQ_SWEEP or ATN_SWEEP: bytes 3 and 4 (see FIGURE 1) MUST contain the frequency and attenuation data as specified. This is because PLAY_SONGS (called every interrupt) will output

For FREQ_SWEEP used by itself - in addition to bytes 3 and 4, bytes 5, 6, and 7 must contain data as specified. Bytes 8 and 9 may be used for whatever (since FREQ_SWEEP doesn't look at them).

For ATN_SWEEP used by itself - in addition to bytes 3 and 4, bytes 5, 8, and 9 must contain data as specified. Bytes 6 and 7 may be used for whatever (since ATN_SWEEP doesn't look at them).

If both FREQ_SWEEP and ATN_SWEEP are used, all bytes in the data area must look as specified in FIGURE 1.

The Colecovision operating system includes routines which allow the cartridge programmer to store "songs" and simple sound effects in tabular form in cartridge ROM, and play them on request during the game. More complex, special sound effects can also be created and played within the same data structure and procedural format. This Sound Users' Manual describes the data structures expected by and the use of the operating system sound routines.

SUMMARY OF FEATURES

* six song note types: combinations of fixed or variable frequency and attenuation, "noise" (percussion) notes, and rests

* hierarchical structure of local data areas assigned to each sound channel, which allows the temporary "overwriting" of lower priority songs (i.e., lower priority songs are not truncated by higher priority songs or sound effects that use the same channel, but continue unheard until the higher priority songs finish)

* the ability to easily include a special sound effect (say, a cymbal crash) as part of a song composed primarily of musical tones

* both songs and special, independent sound effects (e.g., an explosion sound) can utilize the same data structures and output procedures

* sweep routines, which automatically create frequency or attenuation sweeps, simplify note data storage and can be used by special sound effects

* song end codes allow songs to be played once, or automatically restarted upon completion (repeat forever)

GENERAL DESCRIPTION

* Song data areas, SxDATA: RAM map mode of sound chip operation

The Colecovision operating system provides sound routines which output frequency, attenuation, and control data to the TI 76489 sound chip. Data to be sent to a particular sound generator channel is expected to be stored within a ten byte block of CPU RAM called a "song data area". A song data area, then, contains a RAM record of the current values "playing" on a sound channel.

Each song data area can also contain timing and descriptive information which allows for simple generation of musical notes. A "song" can be created by storing in CART ROM a list of note parameters which specify note duration, frequency, and attenuation. When a song is started, O/S routines are provided to load the data describing the first note of the song into a song data area. Each song data area is then processed at regular intervals by routines which modify and output the area's data to the sound chip. When a note is completed, the next note in the song is automatically loaded and the process continues.

O/S routines also exist which facilitate the creation of "special effects": sound routines written by the cartridge programmer which algorithmically generate data to be sent to the sound chip (as opposed to the table look-up, song approach).

RAM space for at least four song data areas must be reserved by the cartridge programmer: one each to describe the current status of the four sound chip channels. More than four song data areas will be required if the ability to "overwrite" lower priority songs is desired, and some songs may share the same data area (see the following discussion, "Hierarchy of song data areas: priority, truncation, overwriting"). The first byte in each song data area, byte 0 (its offset from the beginning of the data block = 0), contains the channel number upon which the song is to be played (0: noise generator, 1 to 3: tone generator) and the song's identification number (SONGNO: 1 to 61). A song data area is referred to throughout the rest of this manual as "SxDATA", where x = the song's SONGNO. For a detailed description of each byte in a song data area, see the following discussion, "NOTES", and refer to Figure 1.

* Note list storage and note headers

A note list is a sequential list of frequency and timer data stored in cartridge ROM that, when processed and output to the sound chip, create the notes that comprise a song. Each block of 1 to 8 bytes of data that describes a note in the list must begin with a one byte header which contains information (bit flags or values) that indicate (see Figure 2):

1) The number of the channel upon which to play the note
2) The note type (one of 4 combinations of fixed or swept frequency and attenuation, plus a rest).

The single byte header can also be used as an end-of-song marker, a repeat-song indicator, or an indicator that a "note" is to be determined algorithmically by a special sound effect routine (the starting address of which immediately follows).

A 16 bit pointer to the location of the header or the next note to be played (NEXT_NOTE_PTR) is maintained by the O/S routine SND_MANAGER in each song's data area (SxDATA, offsets 1 and 2).

* LST_OF_SND_ADDRS and PTR_TO_LST_OF_SND_ADDRS

The O/S routines expect the ten byte long song data areas to be stored contiguously in CPU RAM, starting with the data area used by song number one. The beginning addresses of each of these data areas, as well as the addresses of the headers of the first note in each song, are stored in a ROM table called LST_OF_SND_ADDRS (see Figure 3). The cartridge programmer may place this table wherever desired in ROM. The O/S routines know its starting address through a dedicated CPU RAM location, PTR_TO_LST_OF_SND_ADDRS, which must be loaded with the 16 bit address of the table by the cartridge program before calling any O/S routines which use it (see description of INIT_SOUND).

* Hierarchy of song data areas: truncation, priority, overwriting

The routine that does the processing of the note data stored in the song data areas, SND_MANAGER, and the routine that outputs the modified data to the sound chip, PLAY_SONGS, are designed to be called by the cartridge program every Video Display Processor (VDP) interrupt (every 16.7 ms). Starting with the data area for song number one, SND_MANAGER processes the appropriate timer and sweep counters and modifies the frequency and attenuation data accordingly. If the data area is assigned to a special effect, SND_MANAGER calls that effect. When a note is finished, SND_MANAGER, using the data area's next note pointer, moves data for the next note of the song into the area.

After the operations upon a data area have been performed, the sound chip channel number (CH#) stored in byte 0 of that data area is consulted and the appropriate "channel data area pointer" (PTR_TO_S_ON_x) is set to point to the beginning of the data area just processed (four of these pointers exist at dedicated 16 bit locations in CPU RAM, one for each of the sound generator channels). The following data areas are then processed in the same fashion, in order of occurence, until the end of data area code, 00, is reached. If a data area is inactive, i.e., if the song(s) that use it aren't playing at the moment, SND_MANAGER simply passes it over, doing no processing or channel data area pointer modification.

PLAY_SONGS, usually called immediately prior to SND_MANAGER, outputs data to the sound chip from the four song data areas pointed to by the channel data area pointers. Thus, a channel output priority is established on the basis of ordinal position within the data area block: the last data area processed that uses a given channel is the one that will be played on that channel. E.g.,

order of data area within data block containing all song data areas	songs that use this data area: SONGNO/CH# song is to be played on	
1st	1/CH#x	(remember: although other songs may use this data area also , song 1 MUST use it)
...		
5th	6/CH#2	
6th	3/CH#2	
7th	11/CH#2	
...		
10th	2/CH#3; 4/CH#3; 7/CH#3	

First, consider channel 2. Let's say that the only songs which use channel 2 are assigned three contiguous song data areas, 5th through 7th (grouping songs which use the same channel isn't necessary as far as the code is concerned, but it makes it simpler to think about). SND_MANAGER, as it makes its way through the song data areas in order, will process the 5th data area (which "belongs" to song 6) and set PTR_TO_S_ON_2 to the address of byte 0 in the 5th data area. Then, the 6th area will be processed, resulting in resetting PTR_TO_S_ON_2 to the 6th data area (song number 3). Likewise, the 7th data area will be processed, finally leaving PTR_TO_S_ON_2 pointing to the 7th data area (song number 11). The next time PLAY_SONGS is called, it will send to sound chip channel 2 frequency and attenuation data in the data area pointed to by PTR_TO_S_ON_2, namely, the data for song number 11.

Note that although only song 11 will be heard on channel 2 this pass through PLAY_SONGS, the timers and data for songs 6 and 3 were nonetheless modified, regardless of the existence of the higher priority song 11. That is, all songs "keep going", whether or not their data will be output during PLAY_SONGS. Songs 6 and 3 are said to have been "overwritten" by song 11. Should song 11 become inactive (end) before song 6 and/or song 3, then the highest priority of the remaining active songs (i.e., the last data area within the block of data areas to use a given channel) will be heard.

Thus, assigning several data areas to songs which use the same channel allows the creation of "background" songs which can be momentarily interrupted (overwritten) by a higher priority song or sound effect (e.g., an explosion, or a bonus song) and continue on after the overwriting song is over.

Now examine the 10th song data area. Again, let's say that the only songs that use channel 3 are shown here and that they all share the 10th data area. In this case, the programmer may have arranged things such that songs 2, 4, and 7 are never active simultaneously: i.e., there was no reason to assign three different data areas for songs which never overlap. If, however, this is not the case and, say, song 4 may be started before song 7 is finished, the O/S routines would stop song 7 in favor of song 4. That is, for songs that share the same data area, the most recent song started is the song heard, and interrupted songs do not continue: i.e., songs sharing the same data area truncate each other. In many cases this may be both acceptable and desirable, as it saves RAM space.

NOTE: The preceeding description states that a channel data area pointer is updated every time SND_MANAGER processes a song data area: this is actually not the case. To save processing time, a routine which updates all the pointers is called only when, after loading the next note in a song, SND_MANAGER detects that the data area's CH# or SONGNO has changed. This happens whenever the next note: 1) uses a different channel (see "Noise notes: special case Type 2 notes), 2) is a special effect note, or 3) is an end-of-song indicator. It is only necessary to update the channel data area pointers in these cases, and when a new song is started (in JUKE_BOX). See "Pseudo code versions of main routines".

* The four basic routines, briefly

The following four O/S routines are the only ones that need be called to create songs which use the six standard note types (more complete descriptions of each routine can be found in the "OPERATING SYSTEM ROUTINES" section):

INIT_SOUND: This routine should be called immediately after power on, before any sound processing can occur. It turns off the sound generators, initializes the CART RAM locations to be used as song data areas, sets up the four channel data area pointers, and initializes PTR_TO_LST_OF_SND_ADDRS.

INPUT: n
TYPE: 8 bit constant
PASSED: in B
DESCRIPTION: number of song data areas used by the game

INPUT: LST_OF_SND_ADDRS
TYPE: 16 bit address
PASSED: in HL
DESCRIPTION: LST_OF_SND_ADDRS is the base address of a list of the starting addresses of each song's data area and note list.

OUTPUT: 1) turns off all sound generators -
2) initializes PTR_TO_LST_OF_SND_ADDRS
3) writes the inactive code (0FFH) to byte 0 of the n song data areas
4) stores 00 at end of song data areas
5) sets the 4 channel song pointers to a dummy inactive area
6) sets SAVE_CTRL to 0FFH (see "Noise notes" discussion)

JUKE_BOX: JUKE_BOX is called to start a song. Using a song number passed in B, JUKE_BOX loads the data for the song's first note into the appropriate song data area, thereby truncating whatever song had been "playing" in that data area. (The address of the appropriate area is found by using the song number as an index into the LST_OF_SND_ADDRS table). It also formats the data area's header and sets up the next note pointer. If the song is a special sound effect, its next note pointer is set to the address of the special effect routine. The next time PLAY_SONGS is called, that song's first note will be played.

If JUKE_BOX is called with a song number of a song already in progress, it returns immediately (i.e., it doesn't restart the song).

INPUT: song number to be started
TYPE: 8 bit constant, 1 to 61
PASSED: in B

CALLS: PT_IX_TO_SxDATA, LOAD_NEXT_NOTE_PTR, UP_CH_DATA_PTRS

OUTPUT: 1) moves the song's first note data to the appropriate song data area
1) formats byte 0 header of the song's data area
2) points next note pointer in data area (bytes 1&2) to address of first note in song, or address of special sound effect routine

SND_MANAGER: SND_MANAGER should be called every VDP interrupt (every 16.7 ms). For each data area, SND_MANAGER processes the appropriate timer and sweep counters and modifies the frequency and attenuation data accordingly. If the data area is assigned to a special effect, SND_MANAGER calls that effect. When a note is finished, SND_MANAGER, using the data area's next note pointer, moves data for the next note of the song into the area. If SND_MANAGER reads a header byte (in CART ROM) that has bits 3&4 set, indicating repeat song, it will start the song again by reloading the first note in the song.

After the operations upon a data area have been performed, if necessary, the channel data area pointers (PTR_TO_S_ON_X) are updated. The following data

SND_MANAGER does not output the modified frequency and attenuation data. PLAY_SONGS is called just before SND_MANAGER to do this.

Special codes in byte 0 of the song data area indicate:

- 255: data area inactive, do no processing
- 62: a special effect is to be played; SND_MANAGER calls the effect routine
- 0: end of song data areas (SND_MANAGER processes data areas until it sees 0 in byte 0)

NOTE: Song number 1 MUST use the first area in the block of song data areas.

INPUT: none

CALLS: PT_IX_TO_SxDATA, PROCESS_DATA_AREA

OUTPUT: Calls routines which:

1) decrement song duration and sweep timers
2) modify swept frequency and attenuation values
3) call special effects routines where necessary
4) update the channel data area pointers if necessary
5) restart the song if indicated

PLAY_SONGS: PLAY_SONGS takes the frequency and attenuation data pointed to by the four channel data area pointers (PTR_TO_S_ON_x) and outputs it to the four sound chip generators.

INPUT: none

CALLS: TONE_OUT, UPATNCTRL

OUTPUT:

1) current freq and atn data is output to each tone generator, if song/effect on that channel is active; if song on that channel is inactive, that generator is turned off
2) noise generator is sent current atn data, and control data, if new
3) modifies SAVE_CTRL if necessary

These four routines would normally be called as follows:

power on inits done by O/S
cartridge program receives control:
LD B, # of song data areas used in the game
LD HL, address where LST_OF_SND_ADDRS is stored in ROM
CALL INIT_SOUND to initialize song data areas
whatever other power on inits you want to do
start game:
.
.
.
LD B, # of song you want to start
CALL JUKE_BOX to set up for start of song
.
.
VDP interrupt occurs:
CALL PLAY_SONGS to output data
CALL SND_MANAGER to process song data
whatever else you want to do during VDP interrupts
RETN to game