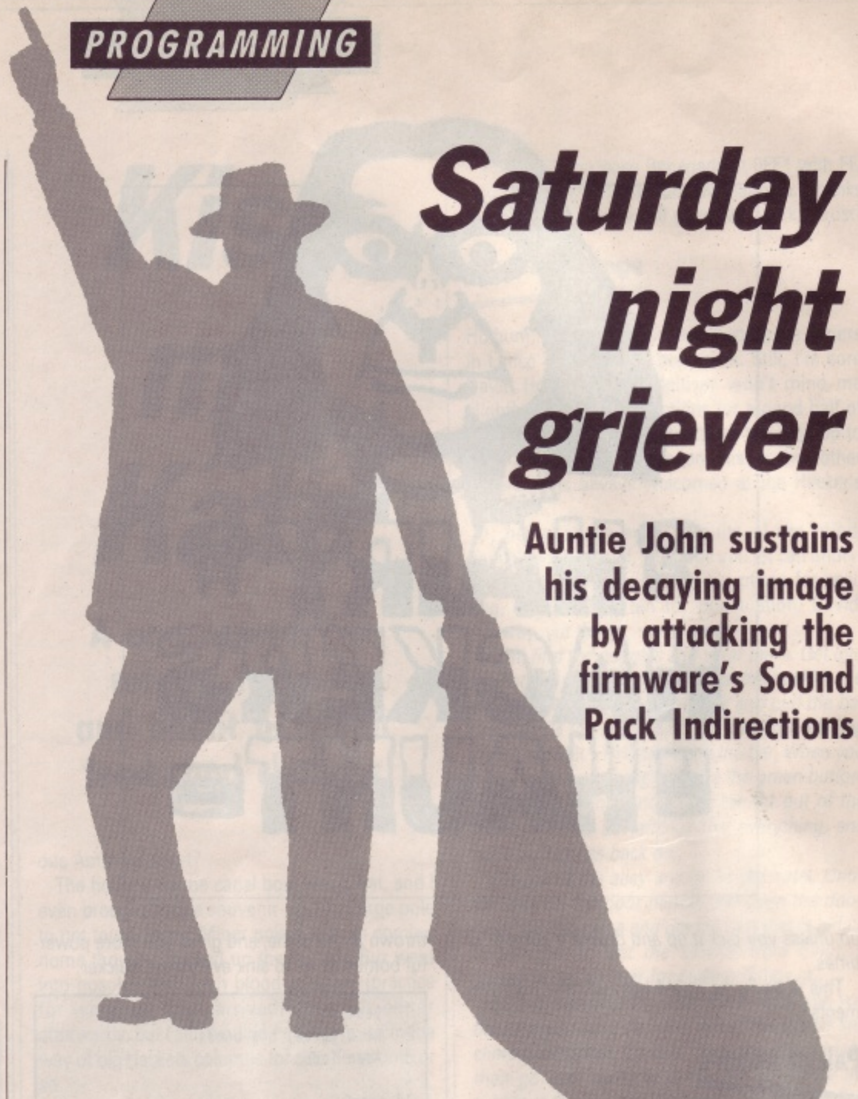
# Saturday night griever

### Auntie John sustains his decaying image by attacking the firmware's Sound Pack Indirections



**DRAT! I've just got a Read Error on my chocolate digestive biscuit. I knew I shouldn't have formatted it to hold 278k. It's all to do with the cheaper chocolate they are using these days. It was only a short time ago that you could safely dunk your biccies in hot tea and still get more than 300k of storage on them.**

Sigh. So here I am, freaking out to Pink Floyd and wishing that when I played my synthesiser it wouldn't sound like a cross between New Order and a car accident. Sound, it seems, is easy to make, but difficult to make well. If you think you have mastered the Basic sound commands we are going to have some fun, because this month it is the firmware's Sound Pack Indirections that get the special AJ treatment.

The sound chip used in the Amstrad CPC machines is that old faithful, the General Instruments AY-3-8912. It was originally designed for use with the CP1600 – a CPU that has long since passed into obscurity.

The AY-3-8912 chip has an integral eight bit I/O port that is used in the CPCs to scan the keyboard. It also has several volume envelopes, and the CPC uses software interrupts to allow the use of tone envelopes. You can program the sound chip directly if you want, but to avoid nasty hang-ups a firmware call must be used.

## Stop and start

Let's begin by looking at two of the simpler firmware calls – SOUND HOLD at &BCB6 and SOUND CONTINUE at &BCB9. These can be used to pause and continue any sounds being generated in mid-bleep. This can be useful if you are trying to write a Pause Game routine for your latest machine code game.

Listing I is the assembly code needed for a routine that you can easily add to your programs. If the Control key is pressed the routine pauses the sound and goes into a loop which is only left when the Copy key is pressed. Perhaps you could add some code to write "Game paused, press COPY to continue".

I must tell you this joke that Fiona's brother, Raymond, told me: Q. What do you call a Chinese woman with a food processor on her head? A. Blenda. Yes, well he thought it was funny and he drives a Fiesta, so he must have a good sense of humour...

The other firmware calls to set up envelopes and add the sounds to the sound queue are just like their Basic equivalents.

An important point to remember is that the data these routines need must not lie underneath a rom. I remember spending many long hours trying to debug a sound routine that refused to work simply because the data was not between addresses &4000 and &BFFF. Don't make the same mistake.

The first routine is SOUND QUEUE at &BCAA. This works in the same way as the SOUND keyword in Basic. You supply it with the address of the data in the HL register pair and it gets to work. But the data must be laid out in a specific order – one slightly different to Basic – and takes nine bytes of ram. The order of the data can be seen in Figure I.

The Channel and Rendezvous data byte works in exactly the same way as the Basic SOUND command, with channels numbered 1,2 and 4. A zero in the other data positions usually means use a default value.

The firmware call SOUND AMPL ENVELOPE at &BCBC sets up an amplitude envelope, which controls the volume of the sound produced. Just like SOUND QUEUE, it needs its data supplied in a block of memory pointed to by HL. For future reference it also needs an envelope number supplied in the A register. Fifteen such envelopes can be created.

An amplitude envelope contains one byte storing the number of three-byte sections that follow. There are a maximum of five sections. The bytes in a section control the step count, the step size and the pause time – again just like from Basic with ENV.

If the most significant bit (MSB) of the step count is set – that is, if 128 is added to the value – a hardware envelope is used. This is one generated solely by the sound chip itself, not the

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Byte 8 |
|---|---|---|---|---|---|---|---|---|
| Channels and rendezvous data. | Amplitude envelope number. (0 to 15) | Tone envelope number. | Tone period. Two bytes are needed to give the required range. (0 to 4095) | | Noise period. (0 to 31) | Initial amplitude of the volume. (0 to 15) | Two bytes to hold the duration. (0 to 32767) | |

*Figure I: The order of the data for SOUND QUEUE*

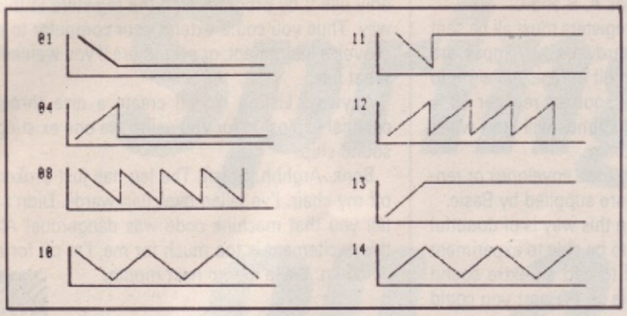


Figure II: Hardware envelopes

operating system. In this the step count holds the hardware envelope number (see Figure II) and the two bytes that follow hold its period.

SOUND TONE ENVELOPE at &BCBF similarly sets up a tone envelope – which controls how the pitch of a sound changes with time. This is equivalent to an ENT command from Basic. Again, the first byte in the data block contains the number of three-byte sections. The bytes in each section are the step count, the step size and the pause time – just like from within Basic.

As before, special cases are indicated by setting certain bits. If the MSB of the section count – the first byte in the data block – is set, the envelope will repeat itself.

Values of &F0 to &FF in the step count cause the section to be an absolute section – instead of changing the period by relative values, the period is changed immediately to the period held in the next two bytes.

Sometimes all this gets too much for me, so I chicken out and set up my envelopes from within a Basic loader program. This makes life a lot easier because the sounds can still be called from machine code.

Listing II sets up an envelope of each type, and then adds the sound to the Sound Queue. As the "flush queue" bit of the channel number is set, the sound happens immediately.

## Bookends

Green is back from his Trans Europe Juggling Exhibition, and claims to have dropped one in every major capital city. Personal friends of Green will not find this very difficult to believe. This guy takes juggling to the ultimate in spectator sports, especially when he gets drunk and tries to juggle a knife, a pair of scissors and a cheese grater. Hope the skin grows back soon.

Another friend of mine comes straight from the sixties. Allow me to introduce you to the politically aware Chris the Hippy, who likes nothing better than to mellow-out with some Bob

```
; Pause Game routine.
; Intended to be called from within
; the main loop of a game.

test_key   equ &bb1e
hold_sound equ &bcb6
cont_sound equ &bcb9

.pausegame
ld a,23
call test_key              ;If CTRL not pressed
ret z                      ;then return from routine.
call hold_sound            ;Pause sound. .loop
ld a,9
call test_key
jr z,loop                  ;Loop until COPY pressed.
call cont_sound            ;Allow sound to continue.
ret                        ;Return from routine.

end
```

Listing I

| Reg. | Function | Bits used |
|---|---|---|
| 0 | Cha A tone | Least Significant eight bits of period. |
| 1 | Cha A tone | The lower four bits contain tne higher order bits of the period. |
| 2 | Cha B tone | Least Significant eight bits of period. |
| 3 | Cha B tone | The lower four bits contain the higher order bits of the period. |
| 4 | Cha C tone | Least Significant eignt bits of period. |
| 5 | Cha C tone | The lower four bits contain the higher order bits of the period. |
| 6 | Noise period | A value from 0 to 31. |
| 7 | Control Reg. | Bit 7: must be zero<br>Bit 6: must be zero<br>Bit 5: Reset means Noise on Cha C<br>Bit 4: Reset means Noise on Cha B<br>Bit 3: Reset means Noise on Cha A<br>Bit 2: Reset means Tone on Cha C<br>Bit 1: Reset means Tone on Cha B<br>Bit 0: Reset means Tone on Cha A |
| 8 | Cha A vol. | The four LSB's contain the volume from 0-15. If the fifth bit is set the hardware envelope is used |
| 9 | Cha B vol. | The four LSB's contain the volume from 0-15. If the fifth bit is set the nardware envelope is used |
| 10 | Cha C vol. | The four LSB's contain the volume from 0-15. If the fifth bit is set the hardware envelope is used |
| 11<br>12 | Envelope<br>period | Least Significant byte of period.<br>Most significant byte of period. |
| 13 | Envelope No. | One of the seven different hardware envelopes (see Figure II). |

Figure III: The sound chip registers

```
; Sets up an envelope of each type,
; and then adds the sound to the Sound Queue.

sound_queue   equ &bcaa
ampl_envelope equ &bcbc
tone_envelope equ &bcbf

ld a,1
ld hl,envdata
call ampl_envelope       ;Set volume envelope
ld a,1
ld hl,entdata
call tone_envelope       ;Set tone envelope
ld hl,sound_data
call sound_queue         ;Create sound
ret                      ;Return to Basic

; Sound data
envdata    db 1,15,-1,20             ;Only one section.
entdata    db 130,10,10,1,10,-10,1   ;Two sections.
sound_data db 135,1,1,32,3,0,0,0,0

end
```

Listing II

Dylan and start revolutions in small countries. He also likes old people and children. He wants to run his own chain of fashion boutiques and be an air hostess.

## Direct assault

To program the sound chip directly, the call MC SOUND REGISTER at &BD34 must be used. The A register contains the sound chip register to use, and C contains the data.

Figure III details the registers on the AY-3-8912, not including the I/O registers. The three sound channels are referred to as A, B and C.

To create a sound the registers must all be sent their set up values. If hardware envelopes are used – by setting the fifth bit in registers eight to 10 – the sound starts as soon as register 13 is written to, otherwise the sounds will start when the control register lets them.

Notice that there are no tone envelopes or rendezvous facilities; these are supplied by Basic.

Using the sound chip in this way is of doubtful importance, but it's nice to be able to experiment with it. If you happened to add an extra sound chip to your computer via an I/O port you could only use it by programming the registers in this way. Thus you could extend your computer to a six-voice instrument, or even more if you wanted. What fun.

Anyway, Listing III will create a nice three-channel explosion for you using the one existing sound chip.

Bonk. Arghhh. Crash. The leg has just broken off my chair. I've fallen over backwards. Didn't I tell you that machine code was dangerous! All this excitement is too much for me, I'm off for a lie down. Wake me up next month.

```
;
; Explosion.
;

sound_register equ &bd34

org    ;(you choose where - anywhere - even under a rom!)

ld a,6
ld c,31
call sound_register     ;Set noise period.
ld a,7
ld c,7                  ;That's 00000111 in binary.
call sound_register     ;Enable noise on all channels.
ld a,8
ld c,16
call sound_register     ;Enable envelopes Ch. A
ld a,9
ld c,16
call sound_register     ;Enable envelopes Ch. B
ld a,10
ld c,16
call sound_register     ;Enable envelopes Ch. C
ld a,11
ld c,255
call sound_register     ;Set LSB of envelope period.
ld a,12
ld c,25
call sound_register     ;Set MSB of envelope period.
ld a,13
ld c,1
call sound_register     ;Choose envelope and cause
                        ;sound to start.
ret                     ;Back to Basic.

end
```

*Listing III*