

QPC Qsound notes

Concept section

AY-3 sound system

QPC2 has an implementation of the ABC-Electronik, Qsound sound generator. Which contained a AY-3-8910 sound generator chip, The AY-3_891x range was used in several 8-bit computers in the 1980's.

QPC2 implements this sound generator with two emulated AY-3-8910 chips that are numbered 0 and 1.

Each AY-3-8910 contains three sound channels. The three channels in chip 0 are defined as channels 1 to 3, and the three channels in chip 1 are defined as channels 4 to 6.

Channels 1 and 4 play in the left speaker, channels 3 and 6 play in the right speaker. And channels 2 and 5 play in both speakers.

There is a noise channel, which can be mixed into the three tone channels.

And a set of envelope shapes provide control of the volume of a generated tone (and/or noise).

playing notes

A sequence of musical notes may be sent to any of the six sound channels with the **PLAY** command containing a string of special characters.

Construction of the sound string

| Function | Values |
|------------------------------|--|
| Notes | C D E F G A H (H corresponds to B, HB to B flat) |
| Sharps | # |
| Flats | b |
| Rests | p (one length unit) |
| Change in octave | o0 o1 .. o7 (default o2) |
| Change in volume | v0 v1 .. v15 V16 switches to envelope control |
| Duration of note in 1/50 sec | 10 .. 1255 (default: 15) |
| Change of noise frequency | n0 n1 .. 31 (default n0) |
| Determine warp curve | w0 w1 .. w15 (default w0) |
| Change length of warp | x0 x1 .. x32767 (default is x0) |
| Synchronisation stop | s causes a sound channel to wait |
| Activate a waiting channel | r1 r2 .. r6 |

To play a sequence of notes, you first need to clear the interrupt driven sound queues with **SOUND_AY**, then set the notes to play with **PLAY**, then **RELEASE** the sound channel queue.

The following program will play a scale of notes from the left channel, then two seconds later, play another scale in the right channel.

```
100 sound1$="pv15o4sCDEFGAHo5CDEFGAHp"
110 sound2$="pv15o2r1CDEFGAHo3CDEFGAHp"
120 SOUND_AY : REMark clear queues
130 PLAY 1,sound1$ : REMark plays in left channel
140 PAUSE 25
150 RELEASE 1 : REMark start playing
160 REMark wait for sound channel 1 to stop
170 REPEAT loop
180 IF NOT(PLAYING(1)) THEN EXIT loop
190 END REPEAT loop
200 PAUSE 100 : REMark wait 2 seconds
```

210 PLAY 3,sound2\$: REMark plays in right channel
 220 PAUSE 25
 220 RELEASE 3

| Command | Function |
|----------|---|
| HOLD | pauses the playing of a sound channel |
| PLAY | sets a sound channel with a sequence of notes |
| PLAYING | tests if a sound channel is currently playing |
| RELEASE | starts, or resumes playing of sound channels |
| SOUND_AY | clears sound channel queues |

playing sound effects

Various sound effects can be obtained by writing directly to the registers of the AY-3 chips.

Each of the two emulated AY-3-8910's has a set of 14 read/write registers. The two I/O ports of the AY-3-8910 are not emulated.

The functions of the registers are as follows.

| Reg | Function | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---------------------|-----|---|---|----|----|----|----|----|----|
| 00 | channel A tone | LSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 01 | channel A | MSB | x | x | x | x | B | A | 9 | 8 |
| 02 | channel B tone | LSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 03 | channel B | MSB | x | x | x | x | B | A | 9 | 8 |
| 04 | channel C tone | LSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 05 | channel C | MSB | x | x | x | x | B | A | 9 | 8 |
| 06 | noise period | | x | x | x | x | 3 | 2 | 1 | 0 |
| 07 | release | | | | nC | nB | nA | sC | sB | sA |
| 08 | channel A amplitude | | x | x | x | w | 3 | 2 | 1 | 0 |
| 09 | channel B amplitude | | x | x | x | w | 3 | 2 | 1 | 0 |
| 10 | channel C amplitude | | x | x | x | w | 3 | 2 | 1 | 0 |
| 11 | envelope period | LSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 12 | envelope period | MSB | F | E | D | C | B | A | 9 | 8 |
| 13 | envelope curve | | x | x | x | x | w3 | w2 | w1 | w0 |

Where

X: bit not used
 nA .. nC: If bit is reset channel is emitting noise.
 sA .. sC: If bit is reset channel is emitting sound.
 w0 .. w3: envelope curve
 w: Bit activates envelope control.

Registers 00 .. 05

Defines the pitch of the channel. Two registers define a note. The main time is divided by 16. By counting down the 12-bit-counter the output frequency is generated. The larger the number in these registers the lower the pitch.

Register 06

Defines the noise frequency. This works like the pitch control but with 5 bits only. The digital noise generator can be switched to any of the three channels.

Register 07

Used to switch the various sound sources. It controls the switching on or off of the tone generators (one for each channel), the switching on or off of the noise generator(single output to any channel).

Each binary bit of the number in this register acts as one switch. A 0 being on, and a 1 being off.

Registers 08 .. 10






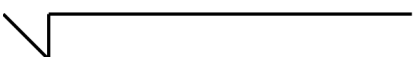

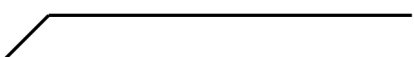
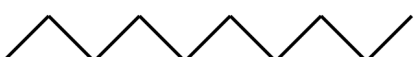

Used to control the volume of each channel. values from 0 to 15 gives 16 levels of volume control in logarithmic steps. 0 being low, 15 being full volume. A value of 16, hands control of the volume for that channel over to the envelope shaper.

Registers 11 and 12

Defines a 16-bit length of time for one envelope period.

Register 13

Used to control the envelope shape and pattern. It gives control of attack (rise to full volume), and decay (fall to zero). Either may be set to fast or slow in single or repeating patterns as follows.

| w3 | w2 | w1 | w0 | Envelope curve |
|----|----|----|----|--|
| 0 | 0 | x | x |  |
| 0 | 1 | x | x |  |
| 1 | 0 | 0 | 0 |  |
| 1 | 0 | 0 | 1 |  |
| 1 | 0 | 1 | 0 |  |
| 1 | 0 | 1 | 1 |  |
| 1 | 1 | 0 | 0 |  |
| 1 | 1 | 0 | 1 |  |
| 1 | 1 | 1 | 0 |  |
| 1 | 1 | 1 | 1 |  |
| | | | | Envelope period |

w3 sets Continue

w2 sets Attack

w1 sets Alternate

w0 sets Hold

The following command will produce 'ping' noise

LIST_AY 100,0,200,0,50,1,0,248,16,16,16,150,6,1

The command breaks down as

| Reg | Function | Value | Effect |
|--------|----------------------|-------|------------------|
| 0/1 | channel A tone | 100 | |
| 2/3 | channel B tone | 200 | |
| 4/5 | channel C tone | 306 | |
| 6 | noise period | 0 | |
| 7 | release | 248 | |
| 8/9/10 | channel A/B/C volume | 16 | envelope control |
| 11/12 | envelope period | 1686 | |
| 13 | envelope curve | 1 | |

This program turns your keyboard into a musical instrument. Pressing keys produce different tones to be played from the left speaker.

```
100 REPEAT loop
110 a=10*(CODE(INKEY$(-1))-12)
120 SOUND_AY 1,a,16 : FOR b=1 TO 4 : PAUSE 1
130 SOUND_AY 1,800,0
140 END REPEAT loop
```

| Command | Function |
|----------|--|
| ENVELOPE | sets the warp period and curve registers |
| LIST_AY | set all registers at once |
| PEEK_AY | reads the value of one of the registers |
| POKE_AY | set one of the registers |

Currently in QPC2 V5.02 there are problems in some of the AY-3, SBASIC commands. The following program will patch some of these problems.

```
100 REMark Qsound live patch
110 REMark For QPC2 V5.02
120 REMark SMSQ/E v3.38 only
130 :
140 def_block = PEEK_L(!!$164)
150 mc_entry = PEEK_L(def_block+2)
160 PRINT "Definition block at ";HEX$(def_block,32)
170 PRINT "    MC_ENTRY at ";HEX$(mc_entry,32)
180 PRINT
190 PRINT
210 :
220 IF 256*PEEK(mc_entry+$4BC)+PEEK(mc_entry+$4BD)<>$B26B THEN PRINT
    "error":STOP
230 POKE_W mc_entry+$4BC,$B22B
240 PRINT "RELEASE patch done."
250 :
260 IF 256*PEEK(mc_entry+$9FC)+PEEK(mc_entry+$9FD)<>$3233 THEN PRINT
    "error":STOP
270 POKE_W mc_entry+$9FC,$1233
280 PRINT "AY_CHIPS and AY_TYPE patch done."
290 :
300 IF 256*PEEK(mc_entry+$8B6)+PEEK(mc_entry+$8B7)<>$2F13 THEN PRINT
    "error":STOP
310 POKE_W mc_entry+$8B6,$2F0B
320 PRINT "SOUND_AY patch done."
330 :
340 PRINT "Live patch applied."
```

To take effect, this program will need to run every time that QPC2 is started.

devices section

QSOUND*cTtFfPpVv_buffer* AY-3 sound generator Transmit only

- c chip number (1 or 2)
- [t] chip type (0 or 1) default 0
- [f] chip clock frequency (0 to 32767) default 17744
- [p] speaker configuration (0 to 6) default 1
- [v] master volume (0 to 255) default 255

buffer size - default 10

default: **qsound1T0F17744P1V255_10**
(chip 1, type 0, frequency 1.7744MHz, speaker configuration 1, master volume maximum, 10 byte buffer)

example: **OPEN#3,qsound**
OPEN#4,qsound2F20000V128
OPEN#4,qsound1F1500V250_50

qsound

programmable sound device

The QSOUND device can be used to directly set the emulated AY-3 sound generators registers, and manage some of the default settings of the emulated AY-3 chips.

The QSOUND device is output only, you cannot read anything from it. The following will open #3 to the second emulated AY-3 chip (chip 1), and set it's registers 0 to 13 with the values 1 to 14.

```
OPEN#3,qsound2  
BPUT#3;1,2,3,4,5,6,7,8,9,10,11,12,13,14
```

This does the same as the command **LIST_AY 1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14**

QSOUND can be supplied some modifiers to change some of the settings of the AY-3 emulation.

```
QSOUNDcTfFpVv_buffer
```

where-

T defines the type of emulation {default 0}
0 = AY
1 = YM

F defines the clock frequency of the AY-3 chips divided by 100 {default 17744}
0 to 32767

P defines the configuration of the speakers {default 1}
0 = mono
1 = ABC
2 = ACB
3 = BAC
4 = BCA
5 = CAB
6 = CBA

where the letters signify the Left, Both, and Right speakers
so ABC means that sound channel A will be in the left speaker, sound channel B will be in both speakers, and sound channel C will be in the right speaker.

V defines the master volume {default 255}
0 to 255

Note that changing any of the above modifiers will effect both emulated AY-3 chips.

warning: Currently the V modifier does not work

Keyword section

AY_CHIPS programmable sound generator

AY_CHIPS is a function to return the number of AY-3 chips that are emulated by QPC2

syntax: **AY_CHIPS**

example: i. **PRINT AY_CHIPS**
 ii. **chip_count = AY_CHIPS**

note: For more information on the AY-3 sound system, see the QPC Concepts document.

warning: This command is currently broken. It returns 518, where it should be 2
 See the QPC Concepts document for a patch program.

AY_TYPE programmable sound generator

AY_TYPE is a function to return the type of the AY-3 chips that are emulated by QPC2

syntax: **AY_TYPE**

example: i. **PRINT AY_TYPE**
 ii. **chip_type = AY_TYPE**

note: For more information on the AY-3 sound system, see the QPC Concepts document.

warning: This command is currently broken. It returns 1, where it should be 0
 See the QPC Concepts document for a patch program.

BELL **EXPLODE**

SHOOT programmable sound generator

BELL, **EXPLODE**, and **SHOOT** uses the AY-3 sound system to produce descriptive sound effects.

syntax: **BELL**
 EXPLODE
 SHOOT

example: i. **BELL**
 ii. **EXPLODE**
 iii. **SHOOT**

note: For more information on the AY-3 sound system, see the QPC Concepts document.

ENVELOPE programmable sound generator

ENVELOPE will set the envelope register, and the envelope period registers.

Shape is one of the 10 available envelope shapes where 0 to 3 are the same first shape, and 4 to 7 are the same second shape.

Period is defined as the chip clock frequency (usually 1.774MHz) divided by 256. The time of one cycle of the resultant frequency is a single period value.

In the default case $1.7734\text{MHz} / 256 = 6927\text{Hz}$ and the cycle time is $1 / 6927 = 144.3\mu\text{S}$

syntax: *shape* := *numeric_expression* {0 to 15}
 period := *numeric_expression* {0 to 4095}

ENVELOPE *shape* , *period*

example: **ENVELOPE 11 , 2500** {2500 x 144.3uS = 0.36 seconds}

note: For more information on the AY-3 sound system, see the QPC Concepts document.

warning: **ENVELOPE only works on AY-3 chip 0, and not on AY-3 chip 1**

HOLD programmable sound generator

HOLD will pause all, or a designated interrupt sound list. **HOLD** without a parameter, or a value of zero will pause all of the interrupt sound lists.

SOUND_AY will be needed to clear the sound lists.

syntax: *sound_list* := *numeric_expression* {0 to 6}

HOLD [*sound_list*]

example: i. **HOLD**
 ii. **HOLD 0**
 iii. **HOLD 2**
 iv. **HOLD int4**

note: Currently **HOLD** stops with an error 'invalid channel ID' if you try to hold a sound list that is currently not in use.

For more information on the AY-3 sound system, see the QPC Concepts document.

LIST_AY programmable sound generator

LIST_AY will set the values of the designated AY-3 chips registers. There are two AY-3 chips emulated in QPC, designated as 0, and 1. If no chip parameter is supplied, then chip 0 will be default one used.

Fourteen registers may be set by this command.

| Register | Usage | | Valid values |
|----------|---------------------|-----|--------------|
| 0 | channel A tone | LSB | {0 to 255} |
| 1 | channel A | MSB | {0 to 15} |
| 2 | channel B tone | LSB | {0 to 255} |
| 3 | channel B | MSB | {0 to 15} |
| 4 | channel C tone | LSB | {0 to 255} |
| 5 | channel C | MSB | {0 to 15} |
| 6 | noise period | | {0 to 15} |
| 7 | release | | {0 to 255} |
| 8 | channel A amplitude | | {0 to 31} |
| 9 | channel B amplitude | | {0 to 31} |
| 10 | channel C amplitude | | {0 to 31} |
| 11 | envelope period | LSB | {0 to 255} |
| 12 | envelope period | MSB | {0 to 255} |
| 13 | envelope curve | | {0 to 15} |

syntax: *ay_chip* := *numeric_expression* {0 or 1}
rx := *numeric_expression* {0 to 255}

LIST_AY [*ay_chip*,] *r0, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r13*

example: i. **LIST_AY 100, 5, 50, 3, 0, 0, 6, 0, 15, 15, 15, 0, 0, 10**
ii. **LIST_AY 1, 100, 5, 50, 3, 0, 0, 6, 0, 15, 15, 15, 0, 0, 10** {as above but chip 1}
iii. **LIST_AY 0, 0, 0, 0, 0, 0, 15, 55, 16, 0, 0, 160, 15, 0** {gunshot}

note: For more information on the AY-3 sound system, see the QPC Concepts document.

warning: Currently **LIST_AY** does not work correctly if the chip number is supplied. So you can only set the default chip 0.

The following procedure **SET_AY**, will do the same as **LIST_AY**
All parameters must be supplied.

```
1000 DEFine PROCEDURE SET_AY chip,r0,r1,r2,r3,r4,r5,r6,r7,r8,r9,  
      r10,r11,r12,r13)  
1010 REMark Replacement LISY_AY command  
1020 POKE_AY chip,0,r0  
1030 POKE_AY chip,1,r1  
1040 POKE_AY chip,2,r2  
1050 POKE_AY chip,3,r3  
1060 POKE_AY chip,4,r4  
1070 POKE_AY chip,5,r5  
1080 POKE_AY chip,6,r6  
1090 POKE_AY chip,7,r7  
1100 POKE_AY chip,8,r8  
1110 POKE_AY chip,9,r9  
1120 POKE_AY chip,10,r10  
1130 POKE_AY chip,11,r11  
1140 POKE_AY chip,12,r12  
1150 POKE_AY chip,13,r13  
1160 END DEFine SET_AY
```

PEEK_AY programmable sound generator

PEEK_AY is a function to return the value that is set in one of the registers of the two AY-3 chips. If no chip parameter is supplied, then chip 0 will be the default one used.

syntax: *ay_chip* := *numeric_expression* {0 or 1}
 reg_no := *numeric_expression* {0 to 13}

PEEK_AY([*ay_chip*,] *reg_no*)

example: i. **PRINT PEEK_AY(6)** {display noise period register of chip 0}
 ii. **PRINT PEEK_AY(1, 6)** {display noise period register of chip 1}

note: For more information on the AY-3 sound system, see the QPC Concepts document.

PLAY programmable sound generator

PLAY sends a string of musical instructions into the interrupt driven list of the supplied sound channel. The string may contain various characters (case is not distinctive) to denote the required action, or note to be played.

Construction of the sound string

| Function | Values |
|------------------------------|--|
| Notes | C D E F G A H (H corresponds to B, HB to B flat) |
| Sharps | # |
| Flats | b |
| Rests | p (one length unit) |
| Change in octave | o0 o1 .. o7 (default o2) |
| Change in volume | v0 v1 .. v15 V16 switches to envelope control |
| Duration of note in 1/50 sec | 10 .. 1255 (default: 15) |
| Change of noise frequency | n0 n1 .. 31 (default n0) |
| Determine warp curve | w0 w1 .. w15 (default w0) |
| Change length of warp | x0 x1 .. x32767 (default is x0) |
| Synchronisation stop | s causes a sound channel to wait |
| Activate a waiting channel | r1 r2 .. r6 |

After setting a sound string with **PLAY**, you need to **RELEASE** the sound channel to start it playing. You may also need a set a short **PAUSE** before the **RELEASE**.

syntax: *ay_channel* := *numeric_expression* {1 to 6}
 sound := *string_expression*

PLAY *ay_channel*, *sound*

example: **PLAY 1, 'pv15o4sCDEFGAHo5CDEFGAHp'**

comment: The above example breaks down as

| | |
|----------------|-----------------------------------|
| p | one rest |
| v15 | maximum volume |
| o4 | set octave |
| s | end of synchronisation |
| CDEFGAH | play a scale |
| o5 | set a new octave |
| CDEFGAH | play the scale at the next octave |
| p | one rest |

note: For more information on the AY-3 sound system, see the QPC Concepts document.

PLAYING programmable sound generator

PLAYING is a function which will return 1 (true) if the supplied channel is currently playing and 0 (false) if it is not currently playing.

syntax: *ay_channel := numeric_expression* {1 to 6}

PLAYING (*ay_channel*)

example: **PRINT PLAYING(2)** {display 1 if channel 2 is currently playing}

note: Currently **PLAYING** stops with an error 'invalid channel ID' if you try to test a channel that is currently not in use.

For more information on the AY-3 sound system, see the QPC Concepts document.

POKE_AY programmable sound generator

POKE_AY allows setting of any of the AY-3 registers. If no chip parameter is supplied, then chip 0 will be the default one used.

syntax: *ay_chip := numeric_expression* {0 or 1}
 reg_no := numeric_expression {0 to 13}
 value := numeric_expression {0 to 255}

POKE_AY [*ay_chip*,] *reg_no*, *value*

example: i. **POKE_AY 2,100** {set register 2, in chip 0, to 100}
 ii. **POKE_AY 1,8,15** {set register 8, in chip 1, to 15}

note: For more information on the AY-3 sound system, see the QPC Concepts document.

RELEASE programmable sound generator

RELEASE causes all, or the specified interrupt sound lists to be played, or resumed if held. If no parameter is supplied, then all sound lists will be played, or resumed.

After setting a sound string with **PLAY**, you may need a set a short **PAUSE** before releasing the sound channel.

syntax: *ay_channel := numeric_expression* {1 to 6}

RELEASE [*ay_channel*]

example: i. **RELEASE** {release all channels}
 ii. **RELEASE 1** {release channel 1 only}

note: Currently **RELEASE** stops with an error 'invalid channel ID' if you try to test a channel that is currently not in use.

For more information on the AY-3 sound system, see the QPC Concepts document.

warning: This command is currently broken. It does not fail on channel numbers above 6. Using channel numbers above 6, may cause undesired effects or crash the driver. See the QPC Concepts document for a patch program.

SOUND_AY programmable sound generator

SOUND_AY will either clear the sound channel and the registers for the supplied channel and the corresponding interrupt list. Or sets the sound output to the supplied sound channel, the supplied frequency in hertz, and volume 0 - 15.

If the volume is set to 16 then the **ENVELOPE** setting are used.

If no parameters are supplied, then all sound channels will be cleared.

syntax: *ay_channel* := *numeric_expression* {1 to 3}
 frequency := *numeric_expression* {23 to 93750}
 volume := *numeric_expression* {0 to 16}

SOUND_AY [*ay_channel*]
SOUND_AY *ay_channel*, *frequency*, *volume*

example: i. **SOUND_AY** {clear all sound channels}
 ii. **SOUND_AY 3** {clear sound channel 3}
 iii. **SOUND_AY 1,1000,15** {set channel 1 to 1KHz at maximum volume}
 iv. **ENVELOPE 12,4000**
 SOUND_AY 1,1000,16

note: For more information on the AY-3 sound system, see the QPC Concepts document.

warning: **SOUND_AY** only works on AY-3 chip 0. Not on AY-3 chip 1.

SOUND_AY with parameters is currently broken as it does not write to the registers correctly.
See the QPC Concepts document for a patch program.