Sound in Forth

This time we cover sounds. I have found the Basic Programm below to which creates a hoover sound.

In VolksFORTH there is no build in SOUND command, so we have to build one that is compatible with the BASIC Sound command.

With that new Word *SOUND* we can recreate the hoover sound in Forth.

The BASIC Porgramm needs 5 variables (LOW, HIGH, P, AGAIN, WAIT), the Forth versions works without any variable declaration, all values are passed on the stack. This is one reason why Forth is memory efficient.

The content of this word in detail

CH# = Pokey Sound Channel 0-3

FREQ - Frequency

DIST - distortion

VOL - Sound volume

The comment show what is on the Stack

```
: SOUND       ( CH# FREQ DIST VOL -- )
   SWAP       ( CH# FREQ VOL DIST -- we swap volume and distortion )
   $10        ( CH# FREQ VOL DIST $10 -- hexadecimal $10 or decimal 16 on the stack )
   \*          ( CH# FREQ VOL DIST\*$10 -- distortion is now multiplied by 16 )
   +          ( CH# FREQ VOL+DIST\*$10 -- volume and distortion are added together )
   ROT        ( FREQ VOL+DIST\*$10 CH# -- we rotate the topmost 3 stack items, brining
   DUP        ( FREQ VOL+DIST\*$10 CH# CH# -- we duplicate the topmost stack item, the
   +          ( FREQ VOL+DIST\*$10 CH#+CH# -- the Pokey Frequency Registers are two by
   AUDBASE    ( FREQ VOL+DIST\*$10 CH#\*2 $D200 -- we put the constant AUDBASE on the
   +          ( FREQ VOL+DIST\*$10 CH#\*2+$D200 -- we add the channel to AUDBASE )
   ROT        ( VOL+DIST\*$10 CH#\*2+$D200 FREQ -- we rotate the topmost 3 Stackitems,
   OVER       ( VOL+DIST\*$20 CH#\*2+$D200 FREQ CH#\*2+$D200 -- OVER copies the 2nd St
   C!         ( VOL+DIST\*$20 CH#\*2+$D200 -- we store [think POKE] the Frequency valu
   1+         ( VOL+DIST\*$20 CH#\*2+$D200+1 -- we increment the topmosat stackvalue,
   C!         ( -- we store volume and distortion in the Pokeys volume and distortion
  ;           ( end of definition of SOUND word )
```

And now the same stuff with real values for the command 0 54 10 14 SOUND

```
: SOUND    ( 0 54 10 14 -- )
   SWAP    ( 0 54 14 10 -- we swap volume and distortion )
   $10     ( 0 54 14 10 16 -- hexadecimal $10 or decimal 16 on the stack )
   \*       ( 0 54 14 160 -- distortion is now multiplied by 16 )
   +       ( 0 54 174 -- volume and distortion are added together )
   ROT     ( 54 174 0 -- we rotate the topmost 3 stack items, brining CH# on the top )
   DUP     ( 54 174 0 0 -- we duplicate the topmost stack item, the Channel )
   +       ( 54 174 0 -- the Pokey Frequency Registers are two bytes apart, so we mult
   AUDBASE ( 54 174 0 $D200 -- we put the constant AUDBASE on the Stack, value is $D20
   +       ( 54 174 $D200 -- we add the channel to AUDBASE )
   ROT     ( 174 $D200 54 -- we rotate the topmost 3 Stackitems, bringing the Frequenc
   OVER    ( 174 $D200 54 $D200 -- OVER copies the 2nd Stackitem on the Top of the Sta
```

```
   C!      ( 174 $D200 -- we store [think POKE] the Frequency value 64 in the Pokey Fr
   1+      ( 174 $D201 -- we increment the topmosat stackvalue, which points now on th
   C!      ( -- we store volume and distortion 174 in the Pokeys volume and distortion
  ;        ( end of definition of SOUND word )
```

BASIC Version

```
10 REM european hooter
15 LOW=57:HIGH=45:P=45
20 FOR AGAIN=1 TO 20
30 SOUND 0,P,10,14
40 FOR WAIT = 1 TO 180: NEXT WAIT
50 P=LOW:LOW=HIGH:HIGH=P
60 NEXT AGAIN
70 SOUND 0,0,0,0
80 END
```

VolksForth Sound Command

```
\\ Atari 8bit Sound Command

$D200 CONSTANT AUDBASE

: SOUND ( CH# FREQ DIST VOL -- )
  SWAP $10 * + ROT DUP + AUDBASE +
  ROT OVER C! 1+ C! ;


: hoover
  57 54  ( Sound values for hoover sound )
  20 0 DO
     OVER
     0 SWAP 10 14 SOUND
     500 0 DO LOOP ( delay loop )
     SWAP  ( change sounds )
  LOOP
  0 0 0 0 SOUND ;
```

Heute gehts es um Geräusche. Folgendes Basic Programm zum Erzeugen eines Sirenentons habe ich im Buch "Atari Basic spielend lernen" gefunden:

```
10 REM EUROPAEISCHE SIRENE
15 LOW=57:HIGH=45:P=45
20 FOR AGAIN=1 TO 20
30 SOUND 0,P,10,14
40 FOR WAIT = 1 TO 180: NEXT WAIT
50 P=LOW:LOW=HIGH:HIGH=P
60 NEXT AGAIN
70 SOUND 0,0,0,0
80 END
```

Im volksFORTH ist kein SOUND Befehl eingebaut, also bauen wir uns einen Atari-Basic kompatiblem Sound-Befehl in Forth:

```
\ Atari 8bit Sound Befehl

$D200 CONSTANT AUDBASE
```

```
: SOUND ( CH# FREQ DIST VOL -- )
  SWAP $10 * + ROT DUP + AUDBASE +
  ROT OVER C! 1+ C! ;
```

Und nun kommt die Sirene in Forth:

```
: SIRENE
  57 54  ( Sound Werte fuer Sirenentoene )
  20 0 DO
     OVER
     0 SWAP 10 14 SOUND
     500 0 DO LOOP ( Warteschleife )
     SWAP  ( Toene wechseln )
  LOOP
  0 0 0 0 SOUND ;
```

Das BASIC Programm benutzt 5 Variablen (LOW, HIGH, P, AGAIN, WAIT), das Forth Programm kommt wieder ohne jede Variablendeklaration aus, alle Werte liegen auf dem Stack. Daher ist Forth so Speichereffizient.