256 bytes tune+player by FTC/HT

Sorry, don't remember what assembler I used to assemble this. Might have been Dreamass by Doc Bacardi/The Dreams.

This tune+player was coded as a compo entry for the Tiny SID #1 competition. The aim was to make a quite long tune rather than something short with fancy instruments.

NOTE: Freshness79 wrote a post on CSDb where he said: “You can cut 2 more bytes by using $032c to “automagically” run your code. This routine gets called at the end of the load process to close all channels ($F32F). It doesn't need to be followed by particular sequences of bytes.”

```
; Title:  "NewKidOnTheBlock"
; Chip:   6581 C64 used when composing.
; Author: Frantic/Hack'n'Trade

;---------------------------------
; CONSTANTS
;
DEBUG           = 0
NUMBEROFVOICES      = 3    ;Using all three Voices
S           = $10    ;Song "speed"

;---------------------------------
; ZERO PAGE ADDRESSES
;

TICKCOUNTERS        = $02    ;Use zp for tickcounters.. TICKCOUNTERS+0,
TICKCOUNTERS+7,
                ;TICKCOUNTERS+14 will be used.
DATAPOS         = $03    ;Use zp for SONGPOSITIONS... 0, 7, 14..
FILTERHI        = $23

;---------------------------------
; CODE
;
    * = $326

        .word start ;Four byte init code. Shamelessly ripped from Alihs
entry, who ripped it from Steven
        .word $f6ed ;Judd... :) Since Alih "ripped" this I guess I could
just as well use it too.???
                ;I hope that won't break any competition rules. Ninjas
method, used in xxxlarge was
                ;nice too, but still some bytes larger even if one includes
the extra code here
                ;needed for raster sync..

start:      sei
```

```
        ldy #$02    ;Set it to $02 so the first incorrectly timed
                ;iteration won't affect anything
        sty FILTERHI    ;Init filtersweep to make it a bit more
deterministic.

        ;Init counters and sonpositions
        lda #(NUMBEROFVOICES-1)*7    ;Voice indeX
@initlp:    tax
        sta DATAPOS,x        ;Init datapos+0 to 0, datapos+7 to 7 and
datapos+14 to 14.
        sty TICKCOUNTERS,x    ;y can be anything.. doesn't matter, but now I
happen to
                    ;init the filter so set it to $01 too
        sec
        sbc #7
        bpl @initlp

;--
; Main player loop
@outerloop:
@wrast:     cpx $d012    ;After the loop, x is a negative number, thus well
above $3e
                ;(or whatever the critical raster value is again..)
        bne @wrast    ;First iteration won't be correct though, but since
the
                ;counters are set to 2 initially it doesn't matter.

        #if DEBUG
            lda #1
            sta $d020
            sta $d021
        #endif

        jsr $b5ff    ;hijack some BASIC ROM shit to do the filter sweep
                    ;b5ff looks like this:
                    ;    inc $23
                    ;    ldx $23
                    ;    ldy #$00
                    ;    rts
        stx $d416    ;filter hi

        ldx #(NUMBEROFVOICES-1)*7    ;Voice indeX
        ;Right here is a good place for Voice specific code, if there is any
reason for that.
@innerloop:
        dec TICKCOUNTERS,x
        bne @loopend
        ;Time for new sound settings, turn gate and oscillator off..
        lda #8
        sta $d404,x
```

```
        ;Turn on global volume to make sure we'll hear anything at all.
        ;Reason for having this code *inside* the loop:
        ;  To make sure the player won't run too fast and get executed
        ;  twice on the same rasterline.
        lda #$5f    ;Hi-pass + Lo-pass filter on.
        sta $d418
        lda #$a2    ;Filter used on middle (melody) voice.
        sta $d417
        ;Parse data sequence data
        ldy DATAPOS,x
    @newseq:
        iny
        lda @musicdata-1,y    ;Get databyte
        bne @nonewseq
        lda @musicdata-0,y    ;Get jumpval if it's jumptime
        tay             ;..and use new one instead
        bpl @newseq        ;This means data may not be larger than $80 bytes
    @nonewseq:
        sty DATAPOS,x
        pha
        and #$0f        ;Note value
        tay
        lda @freqhi,y
        sta $d401,x        ;Freq hi
        lda @freqlo,y
        sta $d400,x        ;Freq lo
        pla
        lsr
        lsr
        lsr
        lsr
        tay
        lda adtab,y
        sta $d405,x
        lda durtab,y
        sta TICKCOUNTERS,x
        lda ctrltab,y
        sta $d404,x
@loopend:
        txa
        .byte $cb,7        ;axs #7 / sbx #7 / whatever..
        bpl @innerloop
        #if DEBUG
            dec $d020
            dec $d021
            jmp @outerloop
        #endif

        bmi @outerloop
```

```
;---------------------------------
; "Instruments"
;
; Using AD only to save space. (SR is set to 00 as default)
;
; A pattern is S*8 ticks long, so using "instrument" 5 we
; can represent a whole empty pattern by just one byte in
; the sequence data. At the same time, this format allows
; for changes to the waveform every tick, which means we
; can also make drums and such things. But, not in this
; tune. Perhaps in the next one..
;
;              00   01   02   03   04   05   06
@adtab:     .byte $1c, $1b, $cd, $2b, $1a, $00, $ad
@ctrltab:   .byte $11, $11, $21, $21, $21, $00, $21
@durtab:    .byte S*1, S*3, S*6, S*2, S*1, S*8, S*8

;---------------------------------
; Sequence data
;
; Voc0 starts at 0
; Voc1 starts at 7
; Voc2 starts at 14
;
; Note and duration stored in one byte and the byte following a $00 (JP)
; byte is interpreted as the destination of a jump to another place in
; the data.

@musicdata:
@Voc0start:
    .byte $00 | G4
    .byte $00 | Az4
    .byte $00 | C5
    .byte $00 | D5
    .byte $00 | Dz5
    .byte JP
    .byte <(@Voc0komp-@musicdata)
@Voc1start:
    .byte $20 | D5
    .byte $30 | Dz5
    .byte $20 | D5
    .byte $40 | A4
    .byte $40 | Az4
    .byte JP
    .byte <(@Voc1melody-@musicdata)
@Voc2start:
    ;-
    .byte $50 | 0
    .byte $50 | 0
    .byte $50 | 0
    .byte $50 | 0
```

```
    ;-
    .byte $50 | 0
@Voc2loop:
    .byte $50 | 0
    .byte $50 | 0
    .byte $20 | C5
    .byte $40 | Dz4
    .byte $40 | F4

    ;-
    .byte $50 | 0
    .byte $60 | D5
    .byte $20 | Dz5
    .byte $30 | Az4
    .byte $20 | A4
    .byte $30 | F4
    ;-
    .byte $60 | G4
    .byte $50 | 0
    .byte $50 | 0
    .byte $60 | C5

    ;-
    .byte $50 | 0
    .byte $60 | D5
    .byte $60 | Dz5
    .byte $60 | F5
    ;-
    .byte $60 | G5
    .byte JP
    .byte <(@Voc2loop-@musicdata)
@Voc0komp:
    .byte $10 | G5

    .byte $00 | D4
    .byte $00 | Fz4
    .byte $00 | A4
    .byte $00 | Az4
    .byte $00 | C5
    .byte $10 | D5

    .byte $00 | Dz4
    .byte $00 | F4
    .byte $00 | G4
    .byte $00 | Az4
    .byte $00 | C5
    .byte $10 | Dz5
    .byte $00 | F4
    .byte $00 | A4
    .byte $00 | C5
    .byte $00 | D5
```

```
    .byte $00 | Dz5
    .byte $10 | F5

    .byte JP
    .byte <(@Voc0start-@musicdata)

@Voc1melody:
    .byte $20 | C5
    .byte $30 | Az4

    .byte $20 | A4
    .byte $30 | F4
    ;-
    .byte $60 | G4
    .byte $50 | 0

    .byte $20 | F4
    .byte $30 | Dz4

    .byte $20 | F4
    .byte $30 | Dz4

    ;-
    .byte $20 | D4
    .byte $30 | Dz4

    .byte $20 | D4
    .byte $30 | Fz4

    .byte $20 | G4
    .byte $30 | Dz4

    .byte $00 | C4
    .byte $00 | Dz4
    .byte $00 | F4
    .byte $00 | G4
    .byte $00 | A4
    .byte $00 | C5
    .byte $00 | G5
    .byte $00 | F5
    .byte JP
    .byte <(@Voc1start-@musicdata)

;---------------------------------
;Note freq data
;
; Only using needed notes.

        JP    =  0
        C4    =  1
        D4    =  2
```

```
        Dz4    =  3
        ;E4    =  2;Not used
        F4    =  4
        Fz4    =  5
        G4    =  6
        ;Gz4    =  6;Not used
        A4    =  7
        Az4    =  8
        ;B4    =  8;Not used
        C5    =  9
        ;Cz5    =  9;Not used
        D5    = 10
        Dz5    = 11
        ;E5    = 11;Not used
        F5    = 12
        ;Fz5    = 13;Not used
        G5    = 13
        ;Gz5    ;Not used
        ;A5    ;Not used
        ;Az5    = 14;Not used

@freqlo = *-1    ;-1 to exclude the corresponding first byte (in @freqhi)
used as a "wrap flag"
        .byte $77
        .byte $61,$e1
        .byte $f7,$8f,$30
        .byte $8f,$4e
        .byte $ef
        .byte $c3,$c3
        .byte $ef
        .byte $60

@freqhi:
        .byte $00 ;WRAP
        .byte $07
        .byte $08,$08
        .byte $09,$0a,$0b
        .byte $0c,$0d
        .byte $0e
        .byte $10,$11
        .byte $13
        .byte $16
```

From:
https://codebase64.org/ - **Codebase 64 wiki**

Permanent link:
https://codebase64.org/doku.php?id=base:256_bytes_tune_player

Last update: **2015-04-17 04:30**