```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PSG ADSR Envelope Manager            ;
; Author:       Rob Eaglestone         ;
;                                      ;
; System:       Commander X16 R.41     ;
; Compiler:     CC65                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

VERA_LOW                          = $9F20
VERA_MID                          = $9F21
VERA_HIGH                         = $9F22
VERA_DATA0                        = $9F23
VERA_CTRL                         = $9f25
PSG_CHANNEL_LO                    = $C0

.macro SAVE_VERA_REGISTERS
     lda VERA_LOW
     sta data_store
     lda VERA_MID
     sta data_store+1
     lda VERA_HIGH
     sta data_store+2
     lda $9F25
     sta data_store+3
.endmacro

.macro RESTORE_VERA_REGISTERS
     lda data_store
     sta VERA_LOW
     lda data_store+1
     sta VERA_MID
     lda data_store+2
     sta VERA_HIGH
     lda data_store+3
     sta $9F25
.endmacro
```

```
.macro WRITE_VOLUMES
     ;
     ; set data to channel 0
     ;
     stz VERA_CTRL

     ; volumes -> channels 3,2,1,0
     ;
     ; stride = 4
     ; direction = negative
     ; VRAM bank = 1
     ;
     lda #($30 | $08 | $01)
     sta VERA_HIGH
     lda #$f9
     sta VERA_MID
     ;
     ; volume reg. of channel 3
     ;
     lda #($C0 + (4 * 3) + 2)
     sta VERA_LOW
     ldx #3
@loop:
     lda volume,x
     ora #%11000000    ; L/R channel
     sta VERA_DATA0
     dex
     bpl @loop
.endmacro


.org $0400
.segment "CODE"
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                 ;
;  Player API jump table          ;
;                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
jmp trigger_voice     ; $400
jmp init_voice        ; $403
jmp reset_voice       ; $406
jmp disable           ; $409
jmp toggle            ; $40c
                      ; $40f ->
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                 ;
;  Installer                      ;
;                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
IRQ_VECTOR            = $0314
OLD_IRQ_HANDLER       = $06

installer:
     sei                   ; 1 byte
     lda IRQ_VECTOR        ; 3 bytes
     sta OLD_IRQ_HANDLER   ; 3 bytes
    lda #<player           ; 3 bytes
    sta IRQ_VECTOR         ; 3 bytes
    lda IRQ_VECTOR+1       ; 3 bytes
    sta OLD_IRQ_HANDLER+1  ; 3 bytes
    lda #>player           ; 2 bytes
    sta IRQ_VECTOR+1       ; 3 bytes
    cli                    ; 1 byte
; write "RTS" to installer
     lda #$60
     sta installer
; to re-enable install,
; just write "SEI" (#$78) back

     rts
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                ;
;  Variables                     ;
;                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
data_store:      .res 4,0   ; R = $042b
system_has_work: .byte 1    ; R+4 = $042f

;
;  envelopes (for four voices)
;
envelope_state:        .byte  4, 0, 0, 0     ; $0430 R+5 .. R+8

volume:                .byte  5, 0, 0, 0     ; $0434 R+9 .. R+12
volume_lo:             .byte  6, 0, 0, 0     ; $0438 R+13 .. R+16

release:               .byte  0, 0, 0, 0     ; $043c R+17 .. R+20
release_lo:            .byte 10, 0, 0, 0     ; $0440 R+21 .. R+24

sustain_level:         .byte 30, 0, 0, 0     ; $0444 R+25 .. R+28
sustain:               .byte  0, 0, 0, 0     ; $0448 R+29 .. R+32
sustain_lo:            .byte 50, 0, 0, 0     ; $044c R+33 .. R+36

decay:                 .byte  1, 0, 0, 0     ; $0450 R+37 .. R+40
decay_lo:              .byte  0, 0, 0, 0     ; $0454 R+41 .. R+44

attack:                .byte 60, 0, 0, 0     ; $0458 R+45 .. R+48
attack_lo:             .byte  0, 0, 0, 0     ; $045c R+49 .. R+52

sustain_timer:         .byte  0, 0, 0, 0     ; $0460 R+53
sustain_timer_lo:      .byte  0, 0, 0, 0     ; $0464 R+57
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                ;
;  Player API                    ;
;                                ;
;  Jump Table Targets            ;
;                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
reset_voice: ; (voice=$02)
     ldy $02
     jsr reset_envelope ; y is preserved
     rts

trigger_voice: ; (voice=$02)
     jsr reset_voice ; loads from $02
     lda #01
     sta envelope_state,y
     rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; voice    $02
; attack   $03  decay      $04
; sustain  $05  release    $06
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
init_voice:
     ; jsr reset_voice?
     ldy $02         ; V
     lda $03         ; A
     sta attack, y
     lda $04         ; D
     sta decay, y
     lda $05         ; S
     sta sustain, y
     lda $06         ; R
     sta release, y
     ; jsr trigger_voice?
     rts
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                ;
;  Turn off player               ;
;                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
disable:
     stz system_has_work
     rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                ;
;  Toggle player on/off          ;
;                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
toggle:
    lda system_has_work
     bne disable
     lda #01
     sta system_has_work
     ; inc system_has_work?
     rts
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                ;
;  Player BEGIN.                 ;
;                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
player:
      lda system_has_work
      beq player_done  ; 0 = OFF
    SAVE_VERA_REGISTERS
    ldy #00
      jsr try_envelope
      ; debug with voice 0 first
      WRITE_VOLUMES
    RESTORE_VERA_REGISTERS
player_done:
    jmp (OLD_IRQ_HANDLER)

adsr_handlers:
      .word goofing_off ; +0
      .word try_attack  ; +2
      .word try_decay   ; +4
      .word try_sustain ; +6
      .word try_release ; +8

;
;  Try to run an envelope
;
try_envelope:
      lda envelope_state,y
      ; A=0,1,2,3,4
      cmp #5
      bcc envelope_state_is_valid
      jmp reset_envelope
envelope_state_is_valid:

      asl
      ; A=0,2,4,6,8
      tax
      ; X=0,2,4,6,8
      jmp (adsr_handlers,x)

goofing_off: ; state 0
      rts

try_attack:  ; state 1
      lda volume_lo,y
      adc attack_lo,y
      sta volume_lo,y
      lda volume,y
      adc attack,y
      ;
      ;  see if A>=64.
      ;
      cmp #64  ; max vol?
      bcc vol_in_range ; A<64
      lda #63
      sta volume,y ; vol=max
      bra advance_state ; A>=64
vol_in_range:
    sta volume,y  ; else OK
      rts

try_decay: ; state 2
      lda volume_lo,y
      sbc decay_lo,y
      sta volume_lo,y
      lda volume,y
      sbc decay,y
      cmp sustain_level,y
      bcc prepare_sustain
      ; at or a bit below sustain level
      sta volume,y
      rts
prepare_sustain:
      lda sustain_lo,y
      sta sustain_timer_lo,y
      lda sustain,y
      sta sustain_timer,y
    bra advance_state

try_sustain:    ; state 3
      tya
      tax
      ; transfer y to x
      dec sustain_timer_lo,x
      lda sustain_timer_lo,x
      beq sustain_rollover
      rts
sustain_rollover:
      lda sustain_timer,x
      beq advance_state
      dec sustain_timer,x
      rts

try_release:  ; state 4
      lda volume,y ; sets Z
      beq reset_envelope
      ; checks Z
      sec
      lda volume_lo,y
      sbc release_lo,y
      sta volume_lo,y
      lda volume,y
      sbc release,y
    bcc advance_state
      sta volume,y
      rts

advance_state:
      ; y is preserved
      tya
      tax
      ; transfer y to x
      inc envelope_state,x
      rts

reset_envelope:
      ; y is preserved
      tya
      tax
      ; transfer y to x
    stz volume_lo,x
      stz volume,x    ; Silent
      stz envelope_state,x  ; OFF
      rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                 ;
;  Player END.                    ;
;                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```