
Subject: NinjaTracker
Posted by [Dagen](#) on Wed, 03 Jun 2015 14:24:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hey guys,

If you haven't already heard about NinjaTracker, it's a new player/converter from NinjaForce. Check it out here: http://www.ninjaforce.com/html/products_ninjatracker.html

Anyway, it uses their NinjaForce assembler, which is fine. But I decided to convert it for use with Merlin 16/32.

This version below compiles with Merlin32 to an exact match of the binary they provide:

```
*****
*                               *
*  NinjaTracker Engine      *
*                               *
*    based on              *
*                               *
*  SoundSmith Player v0.95 *
*    (c) 1990              *
*                               *
*  Huibert AALBERS,      *
*  Olivier GOGUEL.      *
*    &                    *
*    Jesse Blue          *
*****
                LST OFF
                ORG $0f0000
*-----

SonREG      =  $e1C03E
SonDATA     =  $e1C03D
SonCTRL     =  $e1C03C
* Params.
                bra  Init_Sound

Reference_Freq =  $00F0                ; Interrupt Frequency
Music_File     adrl $100000            ; Music Adr
Wave          adrl $120000            ; Wave Adr
Nb_Track      =  14                    ; Nb defined Tracks
Nb_PlayedTrack =  8                    ; Nb played Tracks

Init_Sound    =  *
```

```
sei
phb
phk
plb
```

```
clc
xce
rep #$30
```

```
;patch all wave ptrs
```

```
lda Wave           ;+2
clc
adc #2
sta wpatch001+1
sep #$30
lda Wave+2
adc #0
sta wpatch001+3
rep #$30
```

```
lda Wave           ;+0
sta wpatch002+1
sep #$30
lda Wave+2
sta wpatch002+3
rep #$30
```

```
lda Wave           ;+$10022
clc
adc #$0022
sta wpatch003+1
sep #$30
lda Wave+2
adc #$01
sta wpatch003+3
rep #$30
```

```
lda Wave           ;+$1005e
clc
adc #$005e
sta wpatch004+1
sep #$30
lda Wave+2
adc #$01
sta wpatch004+3
rep #$30
```

```
clc
```

```

xce
sep #30

lda #01100000
stal SonCTRL
lda #0
stal $E1C03E
stal $E1C03F

rep #10

ldx #0
wpatch001    = *
]ICI        ldal Wave+2,x
            stal SonDATA
            inx
            cpx #0
            bne ]ICI

            stz Performing

            rep #30

;patch all pointers
lda Music_File    ;+6
clc
adc #6
sta mpatch001+1
sta mpatch002+1
sta mpatch003+1
sta mpatch004+1
sep #30
lda Music_File+2
adc #0
sta mpatch001+3
sta mpatch002+3
sta mpatch003+3
sta mpatch004+3
rep #30

lda Music_File    ;+470
clc
adc #470
sta mpatch005+1
sta mpatch013+1
sep #30
lda Music_File+2
adc #0

```

```

sta mpatch005+3
sta mpatch013+3
rep #$30

lda Music_File      ;+472
clc
adc #472
sta mpatch006+1
sta mpatch012+1
sta mpatch014+1
sep #$30
lda Music_File+2
adc #0
sta mpatch006+3
sta mpatch012+3
sta mpatch014+3
rep #$30

lda Music_File      ;+8
clc
adc #8
sta mpatch007+1
sep #$30
lda Music_File+2
adc #0
sta mpatch007+3
rep #$30

lda Music_File      ;+0
sta mpatch008+1
sep #$30
lda Music_File+2
sta mpatch008+3
rep #$30

lda Music_File      ;+2
clc
adc #2
sta mpatch009+1
sep #$30
lda Music_File+2
adc #0
sta mpatch009+3
rep #$30

lda Music_File      ;+3
clc
adc #3

```

```

sta mpatch010+1
sep #$30
lda Music_File+2
adc #0
sta mpatch010+3
rep #$30

lda Music_File      ;+600
clc
adc #600
sta mpatch011+1
sep #$30
lda Music_File+2
adc #0
sta mpatch011+3
rep #$30

]lp      ldx #0
        STZ Clear_Deb,X
        INX
        INX
        CPX #Clear_End-Clear_Deb
        bcc ]lp

        lda #$5c
        stal $e1002c
        phk
        phk
        pla
        and #$ff00
        stal $e1002c+2
        lda #SoundIRQrtn
        stal $e1002c+1

        sep #$30

        lda #$00
        stal SonCTRL

        LDA #Reference_Freq
        STA Freq_L
        LDA #>Reference_Freq
        STA Freq_H

]loop    ldy #0
        lda Table_Son,Y
        stal SonREG

```

```

    lda Table_Son+1,Y
    stal SonDATA
    iny
    iny
    cpy #7*2
    bne ]loop

    rep #$30

    lda Music_File
    clc
    adc #600
    clc
mpatch001    adcl Music_File+6
    sta Effects1+1
    sep #$20
    lda Music_File+2
    adc #0
    sta Effects1+3
    rep #$20
    lda Effects1+1
    clc
mpatch002    adcl Music_File+6
    sta Effects2+1
    sep #$20
    lda Effects1+3
    adc #0
    sta Effects2+3
    rep #$20

```

* Sauve la stereo pour chaque track.

```

    LDA Music_File+2
    STA $02

mpatch003    ldal Music_File+6
    asl
    bcc *+5
    inc $02
    clc
mpatch004    adcl Music_File+6
    bcc *+5
    inc $02
    clc
    adc #600
    bcc *+5
    inc $02
    clc

```

```

        adc Music_File
        bcc *+4
        inc $02
        sta $00

        ldy #$1E
]loop   lda [$00],y
        sta StereoTable,y
        dey
        dey
        bpl ]loop

```

* Move les parametres des instruments...

```

        ldx #0
        ldy #0

wpatch002   ldal Wave
            and #$00FF
            sta InstIndex
]loopaga    pha

            lda #6
]loop      pha
wpatch003   ldal Wave+$010022,X
            sta instdef,Y
            iny
            iny
            inx
            inx
            pla
            dec
            bne ]loop

            txa
            clc
            adc #$5C-12
            tax

            pla
            dec
            bne ]loopaga

            ldy #0
wpatch004   = *
]loop      ldal Wave+$01005E,x
            sta CompactTable,y
            iny

```

```

    iny
    inx
    inx
    cpy #32
    bcc ]loop

    jsr Play

    cli
    clc
    xce
    rep #$30
    plb
    rti

    mx %00

Play      stz Timer
mpatch005  ldal Music_File+470
           and #%0000000011111111
           sta NumberOfBlocks
           beq SkipPlay
           stz NotePlayed
           stz BlockIndex
mpatch006  ldal Music_File+472
           and #%0000000011111111
           asl
           tax
           lda BlockTable,x
           sta NoteIndex

mpatch007  ldal Music_File+8
           sta Tempo

           ldy #0
           ldx #$2C
mpatch008  = *
]loop     ldal Music_File,X
           sta VolumeTable,Y
mpatch009  ldal Music_File+2,x
           and #$ff
           sta Ninjaloop,y
mpatch010  ldal Music_File+3,x
           and #$f
           phy
           asl
           tay
           lda FineTuneCnvrty

```



```

ply
sta NinjaFineT,y
txa
clc
adc #$1E
tax
iny
iny
cpy #30
bcc ]loop

lda #1
sta Performing
SkipPlay rts

FineTuneCnvrt dw 0,2,4,6,8,10,12,14,0-16,0-14,0-12,0-10,0-8,0-6,0-4,0-2

Get_Effects1 = *
Effects1 ldal 0,X
RTS

Get_Effects2 = *
Effects2 ldal 0,X
RTS

mx %11

SoundIRQrtn = *

phb
phd

phk
plb

ldal SonCTRL
and #%10011111 ; Disable auto-inc. and access DOC reg.
stal SonCTRL

lda #$E0
stal SonREG ; On lit le registre d'interruptions
ldal SonDATA ; pour savoir quel osc. a genere
ldal SonDATA ; l'interruption.

and #%00111110

```

```

lsr
cmp #\$1E
beq TimerInterrupt ; c'est l'interruption 50Hz.

sta OscNumber
lsr ;Osc*2 (of Track which generated the int.)
; dec

asl
tax

lda #0
sep #\$20

]lp ldal SonCTRL
bmi ]lp
ora #%00100000 ; Auto-incrementation
and #%10111111
stal SonCTRL

lda OscNumber
clc
adc #\$80
stal SonREG
lda NinjaWaveAdr,x
stal SonDATA ; Wave Adress pair
lda OscNumber
clc
adc #\$C0
stal SonREG
lda NinjaWaveSiz,x
stal SonDATA ; Wave Size pair
lda OscNumber
clc
adc #\$A0
stal SonREG
lda NinjaDocReg,x
and #%1111_0111 ; int. off
stal SonDATA ; Control register pair

rep #\$20
brl EndInterrupt

TimerInterrupt lda Performing ; Les interruptions 50Hz etant generees
bne WeCanPlay ; en permanence, on utilise Performing
jmp EndInterrupt ; pour savoir si on doit jouer les notes

WeCanPlay stz Temporary ; compteur contenant le numero du track

```

```

;                                courant

    inc  Timer
    lda  Timer
    cmp  Tempo                    ; on joue les notes lorsque Timer=Tempo
    beq  PlayTracks
    jmp  HandleEffects            ; les effets sont mis a jour tous les
;                                1/50 de secondes

PlayTracks    stz  Timer                ; remise a zero du Timer

NewTrack      rep  #$30
              sep  #$20

              ldx  NoteIndex
mpatch011    ldal Music_File+600,x      ; Lit la note a jouer

              rep  #$20

              and  #$7f
; cmp #128 ; Si la note estl 128, c'est une
              bra  NoteFound            ; commande

NotValid      inc  NoteIndex

NotSTP        =  *
NotACommand   sep  #$20

              inc  Temporary            ; On passe au track suivant.
              lda  Temporary
              cmp  #Nb_PlayedTrack
              beq  NextTrack
              jmp  NewTrack

              mx   %00
NextTrack     rep  #$20

              jmp  EndPlay

NoteFound     sta  Semitone            ; On sauvegarde la note lue
              cmp  #0
              beq  DontSave0
              lda  Temporary
              asl
              tay
              lda  Semitone
              sta  SemitoneTbl,y

```

```

DontSave0    sep  #$20
             jsr  Get_Effects1          ; si 0
             ldy  Temporary              ; que l'on doit reutiliser le dernier
             and  #$f0                   ; sample joue.
             sta  curr_instnum
             bne  ThereIsASample
             lda  SampleTable,y
ThereIsASample sta SampleTable,y          ; Sinon, on sauve le numero du
             lsr                               ; sample.
             lsr
             lsr
             lsr
             dec
             asl
             tay
             lda  VolumeTable,y
             lsr
; and #$7f
             sta  VolumeInt
             rep  #$20
             lda  NinjaFineT,y
             sta  FineTune

             lda  Temporary
             asl
             tay
             lda  FineTune                ; Save Finetune for this track
             sta  TrackTune,y

             jsr  Get_Effects1
             and  #$f

             bne  NotArpeggiatto          ; Si l'effet est 0, c'est peut-etre
             jsr  Get_Effects2
             and  #$ff
             beq  NttArpeggiatto

             lda  Temporary
             asl
             tay
             lda  SemitoneTbl,y
             sta  Semitone_

             sep  #$20
             ldy  Temporary              ; set arp.
             lda  #2
             sta  ArpeggiattoTbl,y

```

```

lda Temporary
asl
clc
adc Temporary
tay
lda Semitone_
sta ArpegeToneTbl,y
jsr Get_Effects2
pha
and #$f0
lsr
lsr
lsr
lsr
clc
adc Semitone_
cmp #60
blt arp_toobig1
lda #60-1
arp_toobig1  sta ArpegeToneTbl+1,y
pla
and #$f
clc
adc Semitone_
cmp #60
blt arp_toobig2
lda #60-1
arp_toobig2  sta ArpegeToneTbl+2,y
rep #$20
jmp clear_eff          ; Fin de la preparation de l'Arpeggiatto

NttArpeggiatto  sep #$20
ldy Temporary
lda #0
sta ArpeggiattoTbl,y
rep #$20
jmp clear_eff

NotArpeggiatto  = *
pha

sep #$20

ldy Temporary          ; Il faut arreter l'effet d'Arpeggiatto
lda #0
sta ArpeggiattoTbl,y

rep #$20

```

```

lda Temporary          ;set to no fineslide as default
asl
tay
lda #0
sta Fineslider,y

lda 1,s
asl
phx
tax
lda jumptbl-2,x
sta the_jump+1
plx
pla
the_jump      jmp  jumptbl

nin_1        dw  0
nin_next_pos dw  0

jumptbl      dw  the_tonePup
dw  the_tonePdown
dw  the_toneP
dw  the_vibrato
dw  the_tonePvols
dw  the_vibravols
dw  clear_eff      ;7
dw  clear_eff      ;8
dw  clear_eff      ;9
dw  the_volslide
dw  the_posjump
dw  the_setvol
dw  the_pattbreak
dw  e_effects
dw  the_setspeed

jumptbl2     dw  0
dw  e_finslideup
dw  e_finslidedwn
dw  0
dw  0
dw  0
dw  0
dw  0
dw  0
dw  0
dw  0
dw  e_finvslidup
dw  e_finvsliiddwn

```

```

dw 0
dw 0
dw 0
dw 0

e_effects      =      *                ;e-commands jumper
                jsr  Get_Effects2
                and  #$ff
                pha
                and  #$f0
                lsr
                lsr
                lsr
                tax
                lda  jumptbl2,x
                beq  e_undefined
                sta  the_jump2+1
                pla
                and  #$f
the_jump2      jmp  the_jump2
e_undefined    pla
                jmp  clear_eff

e_finvslidup   =      *                ;slide up only once (Volume)
                pha
                lda  Temporary
                asl
                tax
                stz  IncFreqTbl,x
                stz  NinVslidedwn,x
                pla
                sta  NinVslideup,x
                lda  #1
                sta  Fineslider,y
                jmp  NoTempoChange2
e_finvsliiddwn =      *                ;slide up only once (Volume)
                pha
                lda  Temporary
                asl
                tax
                stz  IncFreqTbl,x
                stz  NinVslideup,x
                pla
                sta  NinVslidedwn,x
                lda  #1
                sta  Fineslider,y
                jmp  NoTempoChange2

```

```

e_finslidPR    dw    0
e_finslidedwn =    *           ;slide down only once (Portamento)
                eor    #$ffff
                inc
e_finslideup   =    *           ;slide up only once
                sta    e_finslidPR
                lda    #0
                sta    NinVslideup,y
                sta    NinVslidedwn,y
                lda    e_finslidPR
                sta    IncFreqTbl,y
                lda    #1
                sta    Fineslider,y
                jmp    NoTempoChange2

```

```

the_vibrato    jsr    Get_Effects2       ;Vibrato (sine only)
; stz vibra_tbl
; stz vibra_speed
                and    #$ff
                beq    the_vib_keep
                pha
                and    #$f
                beq    the_vibra1
                asl
                tay
                lda    vib_tab,y
                sta    vibra_tbl
the_vibra1     pla
                and    #$f0
                lsr
                lsr
                lsr
                sta    vibra_speed
                lda    Temporary
                asl
                tay
                lda    vibra_tbl
                beq    *+5
                sta    Vibrato_Tbl,y
                sta    Vibrato_Tbl_,y
                lda    vibra_speed
                beq    *+5
                sta    VibratoAdd_Tbl,y
                lda    #0
                sta    VibratoPtr_Tbl,y
                sta    NinVslideup,y
                sta    NinVslidedwn,y
                sta    IncFreqTbl,y

```



```

        brl NoTempoChange2
the_vib_keep   lda  Vibrato_Tbl,y
               sta  Vibrato_Tbl,y
               lda  Temporary
               asl
               tay
               lda  #0
               sta  NinVslideup,y
               sta  NinVslidedwn,y
               sta  IncFreqTbl,y
               brl  NoTempoChange2

```

```

vibra_tbl     dw  0
vibra_speed   dw  0

```

```

the_toneP     jsr  Get_Effects2           ;Tone Portamento, slide to another note

```

```

        pha
        lda  Temporary
        asl
        tay
        pla

```

```

        and  #$ff
        bne  the_toneP5
        lda  ninTonePsp,y
the_toneP5    asl
               sta  ninTonePsp,y

```

```

        lda  Semitone
        bne  the_toneP4
        lda  ninTonePsp,y
        beq  no_toneP

```

```

; lda StartFreqTbl,y
; cmp EndFreqTbl,y
; beq no_toneP
; lda ninTonePsp,y

```

```

        idx  IncFreqTbl,y
        bmi  the_toneP3
        bra  the_toneP1

```

```

the_toneP4    stz  Semitone

```

```

the_toneP2    asl
               tax
               lda  Temporary
               asl
               tay
               lda  ZeroTunOffset,x

```

```

        clc
        adc FineTune           ; Add the Instrument's FineTune
        sta EndFreqTbl,y
        cmp StartFreqTbl,y
; beq no_toneP
        lda ninTonePsp,y
        bge the_toneP1
the_toneP3  eor #$ffff
        inc
the_toneP1  sta IncFreqTbl,y
no_toneP    lda #0
        sta NinVslidedwn,y
        sta NinVslideup,y
        sta Vibrato_Tbl,y
        brl NoTempoChange2

```

```

ninTonePspX  dw 0
the_tonePup  = *                ;max. $269
        jsr Get_Effects2        ;Tone Portamento, slide up
        and #$ff
        asl
        sta ninTonePspX
        lda #5*12

```

```

the_tonePup1  asl
        tax
        lda Temporary
        asl
        tay
        lda ZeroTunOffset,x
        clc
        adc FineTune           ; Add the Instrument's FineTune
        sta EndFreqTbl,y
        cmp StartFreqTbl,y
        lda ninTonePspX
        bge the_toneP1X
        eor #$ffff
        inc
the_toneP1X  sta IncFreqTbl,y
        lda #0
        sta NinVslidedwn,y
        sta NinVslideup,y
        sta Vibrato_Tbl,y
        brl NoTempoChange2

```

```

the_tonePdown  = *                ;min. $52
        jsr Get_Effects2        ;Tone Portamento, slide down

```

```

    and #$ff
    asl
    sta ninTonePspX
    lda #2*12
    bra the_tonePup1

the_setvol    JSR  Get_Effects2        ;Change play volume
              and  #$ff
              asl
              cmp  #$81
              blt  the_setvol_1
              lda  #$80
the_setvol_1  sta  VolumeInt

ChangeVol     lda  Semitone
              beq  Changelt
              jmp  clear_eff

Changelt     lda  Temporary
              asl
              sta  OscNumber

              sep  #$20

]lp          lda  SonCTRL
              bmi ]lp
              ora  #%00100000        ; Auto-incrementation
              and  #%10111111
              stal SonCTRL

              lda  OscNumber
              clc
              adc  #$40
              stal SonREG
              idx  VolumeInt
              lda  VolumeConversion,x
              stal SonDATA            ; Volume pair
              stal SonDATA            ; Volume impair

              rep  #$20

              brl  clear_eff

the_volslide  jsr  Get_Effects2        ; slide volume
              pha
              lda  Temporary

```

```

    asl
    tay
    lda #0
    sta IncFreqTbl,y
    sta Vibrato_Tbl,y
    pla
    and #$ff
    beq volslide_2
    sta nin_1
    and #$f
    beq volslide_1
    asl
    sta nin_1
    lda Temporary
    asl
    tay
    lda nin_1
    sta NinVslidedwn,y
    lda #0
    sta NinVslideup,y
volslide_2    brl NoTempoChange2
volslide_1    lda nin_1
              and #$f0
              lsr
              lsr
              lsr
              sta nin_1
              lda Temporary
              asl
              tay
              lda nin_1
              sta NinVslideup,y
              lda #0
              sta NinVslidedwn,y
              brl NoTempoChange2

```

```

the_tonePvols    jsr Get_Effects2           ; slide volume during Portamento
                pha
                lda Temporary
                asl
                tay
                lda #0
                sta Vibrato_Tbl,y
                pla
                and #$ff
                beq Pvolslide_2
                sta nin_1
                and #$f

```

```

    beq Pvolslide_1
    asl
    sta nin_1
    lda Temporary
    asl
    tay
    lda nin_1
    sta NinVslidedwn,y
    lda #0
    sta NinVslideup,y
Pvolslide_2    brl NoTempoChange2
Pvolslide_1    lda nin_1
                and #$f0
                lsr
                lsr
                lsr
                sta nin_1
                lda Temporary
                asl
                tay
                lda nin_1
                sta NinVslideup,y
                lda #0
                sta NinVslidedwn,y
                brl NoTempoChange2

```

```

the_vibravols    jsr Get_Effects2           ; slide volume during Vibrato
                pha
                lda Temporary
                asl
                tay
                lda #0
                sta IncFreqTbl,y
                pla
                and #$ff
                beq Vvolslide_2
                sta nin_1
                and #$f
                beq Vvolslide_1
                asl
                sta nin_1
                lda Temporary
                asl
                tay
                lda nin_1
                sta NinVslidedwn,y
                lda #0

```

```

        sta NinVslideup,y
Vvolslide_2  brl NoTempoChange2
Vvolslide_1  lda nin_1
            and #$f0
            lsr
            lsr
            lsr
            sta nin_1
            lda Temporary
            asl
            tay
            lda nin_1
            sta NinVslideup,y
            lda #0
            sta NinVslidedwn,y
            brl NoTempoChange2

```

```

the_setspeed  jsr Get_Effects2           ; Le tempo est code sur un nibble
              and #$f
              beq clear_eff
              sta Tempo
clear_eff     lda Temporary
              asl
              tay
              lda #0
              sta NinVslideup,y
              sta NinVslidedwn,y
              sta IncFreqTbl,y
              sta Vibrato_Tbl,y
              brl NoTempoChange2

```

```

the_pattbrk  dw 0
the_pattbreak jsr Get_Effects2
             and #$ff
             cmp #64
             bge the_pattb2
             asl
             sta the_pattbrk
             asl
             asl
             asl
             sec
             sbc the_pattbrk
             sta nin_next_pos
the_pattb2   lda #63
             sta NotePlayed
             bra clear_eff

```

```

the_posjump    jsr  Get_Effects2
               and  #$ff
               dec
               sta  BlockIndex
               lda  #63
               sta  NotePlayed
               bra  clear_eff

```

```

NoTempoChange2 = *
               lda  Semitone
               bne  NoTempoChange

               lda  curr_instnum
               beq  NoTempoChange3

```

```

               lda  Temporary
               asl
               tax
               lda  VolumeInt
               sta  TrueVolumeTbl,x
               stx  OscNumber

```

```

               sep  #$20

```

```

]lp           ldal SonCTRL
               bmi ]lp
               ora  #%00100000           ; Auto-incrementation
               and  #%10111111
               stal SonCTRL

```

```

               lda  OscNumber
               clc
               adc  #$40
               stal SonREG
               idx  VolumeInt
               lda  VolumeConversion,x
               stal SonDATA           ; Volume pair
               stal SonDATA           ; Volume impair

```

```

               rep  #$20

```

```

NoTempoChange3 inc NoteIndex
               jmp  NotSTP

```

```

NoTempoChange lda  Temporary
               asl
               tax

```

```
lda VolumeInt
sta TrueVolumeTbl,x
```

```
lda Semitone
bne PlayIt
```

```
inc NoteIndex
jmp NotSTP
```

```
PlayIt      lda Temporary          ; La paire 0-1 d'oscillos etant utilisee
; inc ; pour generer les interruptions, le
```

```
asl          ; track 0 utilise la paire 2-3, etc.
sta OscNumber
```

```
sep #$20
```

```
ldal SonCTRL
and #%10011111
stal SonCTRL
```

```
lda OscNumber
clc
adc #$A0
stal SonREG
ldal SonDATA
ldal SonDATA
and #%11110111
ora #%00000001
stal SonDATA          ; Arrete l'oscillateur pair
```

```
lda OscNumber
clc
adc #$A1
stal SonREG
ldal SonDATA
ldal SonDATA
and #%11110111
ora #%00000001
stal SonDATA          ; Arrete l'oscillateur impair
```

```
ldy Temporary
lda SampleTable,y
```

```
rep #$20
```

```
and #%0000000011110000
lsr
lsr
lsr
```



```

lsr
dec
cmp InstIndex
bcc SampleExists
jmp IgnoreSample
SampleExists sta CurrInstInt
asl
tax
lda InstIndexTable,x ; Offset du debut de la definition de inst.
tax

```

; removed code: we know that we always have only one wavelist, so we take
; the one that's there.

```

stx IndexInterrupt
lda Instrument1+1,x ; On lit la taille et l'adresse de la
sta Temp1Interrupt ; wave pour l'osc. pair
lda Instrument1+3,x ; On lit le mode a utiliser pour l'osc.
and #$ff ; pair
sta Temp2Interrupt
lda StereoMode ; Si StereoMode vaut zero, on utilise
beq StereoOk ; le Mode de l'osc. pour la stereo.
lda Temp2Interrupt ; Sinon on utilise la table StereoTable
and #$f ; pour determiner si le son doit sortir
sta Temp2Interrupt ; a droite ou a gauche.
lda Temporary ; (0=droite $FFFF=gauche)
asl
tax
lda StereoTable,x
beq StereoOk
lda Temp2Interrupt
ora #%0000000000010000
sta Temp2Interrupt

```

```

StereoOk    Idx IndexInterrupt ;same for Osc. B
lda Instrument1+7,x
sta Temp3Interrupt
lda Instrument1+9,x
and #%0000000011111111
sta Temp4Interrupt
lda StereoMode
beq StereoOk2
lda Temp4Interrupt
and #%000000000001111
sta Temp4Interrupt
lda Temporary
asl
tax

```

```

lda StereoTable,x
beq StereoOk2
lda Temp4Interrupt
ora #%0000000000010000
sta Temp4Interrupt

StereoOk2    =    *
lda Temporary
asl
tax
stz SwapperMode,x

lda CurrInstInt
asl
tax
lda Ninjaloop,x
beq conv_note    ;inst is not looped

cmp #3
bne swapmode3
lda Temp2Interrupt
and #%1111_0000
ora #%0000_0110
sta Temp2Interrupt
lda Temp4Interrupt
and #%1111_0000
ora #%0000_0111    ;set swap mode
sta Temp4Interrupt
bra swapmode2

swapmode3    idx IndexInterrupt
lda Instrument1+7+12,x    ;copy wave pos. and len.
sta Temp3Interrupt
lda Instrument1+9+12,x    ;get doc mode
and #%1111_0000
ora #%0000_0111    ;set swap mode
sta Temp4Interrupt
lda Temp2Interrupt
ora #%0000_1000
sta Temp2Interrupt

swapmode2    lda StereoMode
beq conv_note
lda Temp4Interrupt
and #%0000000000011111
sta Temp4Interrupt
lda Temporary
asl
tax

```

```

    lda StereoTable,x
    beq conv_note
    lda Temp4Interrupt
    ora #%0000000000010000
    sta Temp4Interrupt

    lda Temporary
    asl
    tax
    inc SwapperMode,x

conv_note    lda Semitone           ; On convertit un semitone en une
            asl                   ; frequence comprehensible pour le
            tax                   ; DOC.
            lda Temporary         ; Get position in big Finetune Table
            asl                   ; so that we can easily do portamento
            tay
            lda ZeroTunOffset,x
            clc
            adc FineTune          ; Add the Instrument's FineTune
            sta StartFreqTbl,y
            tax
            lda FineTuneTbl,x
            sta TempFreqInt

            lda CurrInstInt
            asl
            tax
            lda CompactTable,x
            tax

bcleFreq    cpx #0
            beq EndBcleFreq
            lsr TempFreqInt
            dex
            bra bcleFreq

EndBcleFreq = *

No_Voice    lda #0
            sep #$20

]lp        ldal SonCTRL
            bmi ]lp
            ora #%00100000        ; Auto-incrementation
            and #%10111111
            stal SonCTRL

```

```

lda  OscNumber
stal SonREG
lda  TempFreqInt
stal SonDATA           ; Frequency low pair
stal SonDATA           ; Frequency low impair
lda  OscNumber
clc
adc  #$20
stal SonREG
lda  TempFreqInt+1
stal SonDATA           ; Frequency high pair
stal SonDATA           ; Frequency high impair
lda  OscNumber
clc
adc  #$40
stal SonREG
ldx  VolumeInt
lda  VolumeConversion,x
stal SonDATA           ; Volume pair
stal SonDATA           ; Volume impair
lda  OscNumber
clc
adc  #$80
stal SonREG
lda  TempInterrupt
stal SonDATA           ; Wave Adress pair
lda  Temp3Interrupt
stal SonDATA           ; Wave Adress impair
lda  OscNumber
clc
adc  #$C0
stal SonREG
lda  TempInterrupt+1
stal SonDATA           ; Wave Size pair
lda  Temp3Interrupt+1
stal SonDATA           ; Wave Size impair
lda  OscNumber
clc
adc  #$A0
stal SonREG
lda  Temp2Interrupt
stal SonDATA           ; Control register pair
lda  Temporary
asl
tax
lda  SwapperMode,x
eor  #1
ora  Temp4Interrupt

```

```

    stal SonDATA          ; Control register impair

    lda CurrInstInt      ; if sample is looped, backup
    asl                  ; place and length of loop, and control
    tax
    lda NinjaLoop,x
    beq IgnoreSample
    lda Temporary
    asl
    tax
    lda Temp3Interrupt
    sta NinjaWaveAdr,x
    lda Temp3Interrupt+1
    sta NinjaWaveSiz,x
    lda Temp4Interrupt
    sta NinjaDocReg,x

IgnoreSample    rep    #$20

                inc    NoteIndex          ; C'est fini, on passe au track
                inc    Temporary          ; suivant...
    lda    Temporary
    cmp    #Nb_PlayedTrack
    beq    EndPlay

                jmp    NewTrack

EndPlay        =    *
                REP    #$20

; DO Nb_PlayedTrack-Nb_Track
    LDA    NoteIndex
    CLC
    ADC    #Nb_Track-Nb_PlayedTrack
    STA    NoteIndex

; FIN

                inc    NotePlayed          ; Si la position de la ligne jouee
    lda    NotePlayed          ; vaut 64, on doit lire un nouveau
    cmp    #64                ; block
    bge    ReadNewBlock
    jmp    EndInterrupt
ReadNewBlock   stz    NotePlayed
                inc    BlockIndex          ; On verifie si on n'a pas fini
    idx    BlockIndex
    cpx    NumberOfBlocks
    bge    Finished
mpatch012     ldal    Music_File+472,x    ; Sinon, on cherche le numero du

```

```

    and #$ff          ; block a jouer
    sta BlockPlayed
    asl
    tax
    lda BlockTable,x  ; et on actualise NoteIndex en fonction
    clc
    adc nin_next_pos  ; (+add pos. in next blk.)
    sta NoteIndex     ; du block a jouer.
    stz nin_next_pos
    bra EndInterrupt

Finished      stz Performing      ; Si on a fini, on met Performing a
;              zero.

    lda LoopMode
    bne NoLoop

mpatch013    lda Music_File+470
    and #%0000000011111111
    sta NumberOfBlocks
    stz NotePlayed
    stz BlockIndex
mpatch014    lda Music_File+472
    and #%0000000011111111
    asl
    tax
    lda BlockTable,x
    sta NoteIndex
    lda #1
    sta Performing
    bra EndInterrupt

NoLoop      sep #$30

bcleStopAll  lda SonCTRL
    bmi bcleStopAll
    and #%10011111      ; acces aux registres du DOC et pas
    stal SonCTRL        ; d'auto-incrementation

    stz OscNumber

bcleStopAll2  lda OscNumber
    clc
    adc #$A0
    stal SonREG
    lda #$01
    stal SonDATA        ; Arrete l'oscillateur pair
    lda OscNumber

```

```

sec
adc #$A0
stal SonREG
lda #$01
stal SonDATA           ; Arrete l'oscillateur impair

lda OscNumber
clc
adc #2
sta OscNumber
cmp #30
bne bcleStopAll2

EndInterrupt sep #$30
pld
plb
clc
rtl

HandleEffects sep #$30           ; Gestion des effets

stz Temporary
bcleHandleArp idx Temporary

lda ArpegiattoTbl,x
bne *+5
jmp NoArpegiatto
cmp #4
bne *+3
lsr
clc
adc Temporary
adc Temporary
adc Temporary
tay
lda Temporary
asl
tax
arp_get_freq lda ArpegeToneTbl-1,y

mx %00           ; Merlin 32 not picking this up?
rep #$30
and #$ff
asl
tay
lda ZeroTunOffset,y
clc

```

```

    adc TrackTune,x          ; Add the Track's FineTune
    bpl *+5
    lda #0
    cmp #$600
    blt *+5
    lda #$600-2
    tay
    lda FineTuneTbl,y
    sta ninfreqInt

    ldx Temporary
    sep #$20
    inc ArpegiattoTbl,x
    lda ArpegiattoTbl,x
    cmp #5
    blt arp_maker_1
    lda #1
    sta ArpegiattoTbl,x
arp_maker_1    =    *

    lda Temporary
    asl
    sta OscNumber

]lp    ldal SonCTRL          ; et on modifie les Frequency registers
    bmi ]lp
    ora #%00100000
    and #%10111111
    stal SonCTRL

    lda OscNumber
    stal SonREG
    lda ninfreqInt
    stal SonDATA          ; Frequency low pair
    stal SonDATA          ; Frequency low impair
    lda OscNumber
    clc
    adc #$20
    stal SonREG
    lda ninfreqInt+1
    stal SonDATA          ; Frequency high pair
    stal SonDATA          ; Frequency high impair

    jmp NoVolSlide

NoArpegiatto    rep #$30          ;Pitch up/down, (=Portamento, ToneP)

```



```

        lda Temporary
        asl
        tax
        lda IncFreqTbl,x
        bne *+5
        jmp NoPitch
; lda Fineslider,x
; beq no_finesl3
; lda Temporary
; bne NoPitch
no_finesl3    lda IncFreqTbl,x
              clc
              adc StartFreqTbl,x
              sta StartFreqTbl,x

              lda IncFreqTbl,x
              bmi tonepslide1

              lda StartFreqTbl,x
              cmp EndFreqTbl,x
              blt tonepslide2
              lda EndFreqTbl,x
              sta StartFreqTbl,x
              stz IncFreqTbl,x
              bra tonepslide2

tonepslide1   lda StartFreqTbl,x
              bmi tonepslide3
              cmp EndFreqTbl,x
              beq *+4
              bge tonepslide2
tonepslide3   lda EndFreqTbl,x
              sta StartFreqTbl,x
              stz IncFreqTbl,x

tonepslide2   =    *

              lda StartFreqTbl,x
              tax
              lda FineTuneTbl,x
              sta ninfreqInt

              sep #$20

              lda Temporary
              asl
              sta OscNumber

```

```

]lp      lda  SonCTRL          ; et on modifie les Frequency registers
        bmi ]lp
        ora  #%00100000
        and  #%10111111
        stal SonCTRL

        lda  OscNumber
        stal SonREG
        lda  ninfreqInt
        stal SonDATA          ; Frequency low pair
        stal SonDATA          ; Frequency low impair
        lda  OscNumber
        clc
        adc  #$20
        stal SonREG
        lda  ninfreqInt+1
        stal SonDATA          ; Frequency high pair
        stal SonDATA          ; Frequency high impair

        jmp  NoVibrato

NoPitch  mx   %00
         =   *                ;Handle Vibrato

        lda  Temporary
        asl
        tax
        lda  Vibrato_Tbl,x
        beq  NoVibrato
        sta vibrato_lokup
        ldy VibratoPtr_Tbl,x
        lda  StartFreqTbl,x
        clc
vibrato_lokup =  *+1
        adc  vib0,y
        bpl  *+5
        lda  #0
        cmp  #$600
        blt  *+5
        lda  #$600-2
        tay
        lda  FineTuneTbl,y
        sta  ninfreqInt

        lda  VibratoAdd_Tbl,x
        clc
        adc  VibratoPtr_Tbl,x
        cmp  #32*2

```

```

        blt vibrato_ovli
; sec
; sbc #32*2
        lda #0
vibrato_ovli    sta VibratoPtr_Tbl,x

        sep #20

        lda Temporary
        asl
        tax
        sta OscNumber

]lp        lda SonCTRL                ; et on modifie les Frequency registers
        bmi ]lp
        ora #%00100000
        and #%10111111
        stal SonCTRL

        lda OscNumber
        stal SonREG
        lda ninfreqInt
        stal SonDATA                ; Frequency low pair
        stal SonDATA                ; Frequency low impair
        lda OscNumber
        clc
        adc #20
        stal SonREG
        lda ninfreqInt+1
        stal SonDATA                ; Frequency high pair
        stal SonDATA                ; Frequency high impair

NoVibrato    = *                ;VolumeSlide up/down
        rep #20

        lda Temporary
        asl
        tax
        lda NinVslidedwn,x
        beq checkVslideup
; lda Fineslider,x
; beq no_finesl1
; lda Temporary
; bne NoVolSlide
no_finesl1    lda TrueVolumeTbl,x
        sec
        sbc NinVslidedwn,x
        bpl changevolsl1

```

```

        lda #0
        stz NinVslidedwn,x
        bra changevolsl1

checkVslideup lda NinVslideup,x
              beq NoVolSlide
; lda Fineslider,x
; beq no_finesl2
; lda Temporary
; bne NoVolSlide
no_finesl2   lda NinVslideup,x
              clc
              adc TrueVolumeTbl,x
              cmp #$81
              blt changevolsl1
              lda #$80
              stz NinVslideup,x
changevolsl1 sta TrueVolumeTbl,x

        lda Temporary
        asl
        sta OscNumber

        lda TrueVolumeTbl,x
        tax
        lda VolumeConversion,x
        and #$ff
        sta Volumina

        sep #$20

]lp      ldal SonCTRL
        bmi ]lp
        ora #%00100000          ; Auto-incrementation
        and #%10111111
        stal SonCTRL

        lda OscNumber
        clc
        adc #$40
        stal SonREG
        lda Volumina
        stal SonDATA          ; Volume pair
        stal SonDATA          ; Volume impair

NoVolSlide = *

        sep #$30

```

```
inc Temporary
lda Temporary
cmp #Nb_PlayedTrack
beq Fini
jmp bcleHandleArp
```

```
Fini      BRL  EndInterrupt
```

```
BlockTable = *
dw Nb_Track*64*0
dw Nb_Track*64*1
dw Nb_Track*64*2
dw Nb_Track*64*3
dw Nb_Track*64*4
dw Nb_Track*64*5
dw Nb_Track*64*6
dw Nb_Track*64*7
dw Nb_Track*64*8
dw Nb_Track*64*9
dw Nb_Track*64*10
dw Nb_Track*64*11
dw Nb_Track*64*12
dw Nb_Track*64*13
dw Nb_Track*64*14
dw Nb_Track*64*15
dw Nb_Track*64*16
dw Nb_Track*64*17
dw Nb_Track*64*18
dw Nb_Track*64*19
dw Nb_Track*64*20
dw Nb_Track*64*21
dw Nb_Track*64*22
dw Nb_Track*64*23
dw Nb_Track*64*24
dw Nb_Track*64*25
dw Nb_Track*64*26
dw Nb_Track*64*27
dw Nb_Track*64*28
dw Nb_Track*64*29
dw Nb_Track*64*30
dw Nb_Track*64*31
dw Nb_Track*64*32
dw Nb_Track*64*33
dw Nb_Track*64*34
dw Nb_Track*64*35
dw Nb_Track*64*36
dw Nb_Track*64*37
```

```
dw Nb_Track*64*38
dw Nb_Track*64*39
dw Nb_Track*64*40
dw Nb_Track*64*41
dw Nb_Track*64*42
dw Nb_Track*64*43
dw Nb_Track*64*44
dw Nb_Track*64*45
dw Nb_Track*64*46
dw Nb_Track*64*47
dw Nb_Track*64*48
dw Nb_Track*64*49
dw Nb_Track*64*50
```

```
StereoMode dw $FFFF
```

```
InstIndexTable = *
dw 12*0
dw 12*1
dw 12*2
dw 12*3
dw 12*4
dw 12*5
dw 12*6
dw 12*7
dw 12*8
dw 12*9
dw 12*10
dw 12*11
dw 12*12
dw 12*13
dw 12*14
dw 12*15
```

```
VolumeConversion dfb 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
dfb 32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,68,70
dfb 72,74,76,78,80,82,84,86,88,90,92,94,96,98,100,102,104,106
dfb 108,110,112,114,116,118,120,122,124,126,128,130,132,134,136
dfb 138,140,142,144,146,148,150,152,154,156,158,160,162,164,166
dfb 168,170,172,174,176,178,180,182,184,186,188,190,192,194,196
dfb 198,200,202,204,206,208,210,212,214,216,218,220,222,224,226
dfb 228,230,232,234,236,238,240,242,244,246,248,250,252,254,255
dfb 255
```

```
;FreqTable dw $00,$16,$17,$18,$1A,$1B,$1D,$1E,$20,$22,$24,$26 ; Octave 0
; dw $29,$2B,$2E,$31,$33,$36,$3A,$3D,$41,$45,$49,$4D ; Octave 1
; dw $52,$56,$5C,$61,$67,$6D,$73,$7A,$81,$89,$91,$9A ; Octave 2
;
```

; dw 81,86,91,97,103,109,115,122,129,137,145,154
; dw 163,172,183,193,205,218,231,244,259,274,290,308
; dw 325,345,366,387,410,435,461,487,516,548,580,616
;
; dw \$0A3,\$0AD,\$0B7,\$0C2,\$0CE,\$0D9,\$0E6,\$0F4,\$102,\$112,\$122,\$133 ; Octave 3
; dw \$146,\$15A,\$16F,\$184,\$19B,\$1B4,\$1CE,\$1E9,\$206,\$225,\$246,\$269 ; Octave 4
; dw \$28D,\$2B4,\$2DD,\$309,\$337,\$368,\$39C,\$3D3,\$40D,\$44A,\$48C,\$4D1 ; Octave 5
; dw \$51A,\$568,\$5BA,\$611,\$66E,\$6D0,\$737,\$7A5,\$81A,\$895,\$918,\$9A2 ; Octave 6
; dw \$A35,\$AD0,\$B75,\$C23,\$CDC,\$D9F,\$E6F,\$F4B,\$1033,\$112A,\$122F,\$1344 ; Octave 7
; dw \$1469,\$15A0,\$16E9,\$1846,\$19B7,\$1B3F
; dw \$1CDE,\$1E95,\$2066,\$2254,\$245E,2688 ; Octave 8

dw 80,80,80,80,80,80,80,80,80,80,80,80,80,80,80
dw 80,80,80,80,80,80,80,80,80,80,80,80,80,80,80
dw 80,80,80,80,80,80,80,80,80,80,80,80,80,80,80
dw 81,81,81,81,81,81,81,81,81,81,81,81,81,81,81
FineTuneTbl dw 81,81,81,81,81,81,81,81,82,82,82,82,82,82
dw 82,82,82,82,82,82,83,83,83,83,83,83,83,83,83
dw 83,83,83,83,84,84,84,84,84,84,84,84,84,84,84
dw 84,85,85,85,85,85,85,85,85,85,85,85,85,85,86
dw 86,86,86,86,86,86,86,86,86,87,87,87,87,87,87
dw 87,87,87,87,87,88,88,88,88,88,88,88,88,88,88
dw 88,88,89,89,89,89,89,89,89,89,89,89,89,89,90
dw 90,90,90,90,90,90,90,90,91,91,91,91,91,91,91
dw 91,91,91,91,92,92,92,92,92,92,92,92,92,92,92
dw 93,93,93,93,93,93,93,93,94,94,94,94,94,94,94
dw 94,94,94,94,95,95,95,95,95,95,95,95,95,95,96
dw 96,96,96,96,96,96,96,96,97,97,97,97,97,97,97
dw 97,97,98,98,98,98,98,98,98,98,98,98,98,98,99
dw 99,99,99,99,100,100,100,100,100,100,100,100,100,100,101
dw 101,101,101,101,101,101,101,102,102,102,102,102,102,102,102,102
dw 103,103,103,103,103,103,103,104,104,104,104,104,104,104,104,104
dw 104,105,105,105,105,105,105,105,105,106,106,106,106,106,106
dw 106,106,107,107,107,107,107,107,107,108,108,108,108,108,108
dw 108,108,109,109,109,109,109,109,110,110,110,110,110,110,110
dw 110,111,111,111,111,111,111,111,112,112,112,112,112,112,112
dw 113,113,113,113,113,113,114,114,114,114,114,114,114,114,115
dw 115,115,115,115,116,116,116,116,116,116,117,117,117,117,117
dw 117,117,118,118,118,118,118,118,119,119,119,119,119,119,119
dw 120,120,120,120,120,121,121,121,121,121,121,122,122,122,122
dw 122,122,123,123,123,123,123,124,124,124,124,124,124,124,125,125
dw 125,125,125,125,126,126,126,126,127,127,127,127,127,127,127
dw 128,128,128,128,128,129,129,129,129,129,130,130,130,130,130
dw 130,131,131,131,131,132,132,132,132,132,133,133,133,133,133
dw 133,134,134,134,134,135,135,135,136,136,136,136,136,136,136
dw 137,137,137,137,138,138,138,138,139,139,139,139,139,139,140
dw 140,140,140,141,141,141,142,142,142,142,143,143,143,143,143
dw 143,144,144,144,144,145,145,145,145,146,146,146,146,147

dw 147,147,148,148,148,149,149,149,150,150,150,151
 dw 151,151,152,152,152,153,153,153,154,154,154,155
 dw 155,155,156,156,156,157,157,157,158,158,158,159
 dw 159,160,160,160,161,161,161,162,162,163,163,163
 dw 164,164,165,165,165,166,166,166,167,167,168,168
 dw 169,169,169,170,170,171,171,171,172,172,173,173
 dw 174,174,174,175,175,176,176,177,177,178,178,179
 dw 179,179,180,180,181,181,182,182,183,183,184,184
 dw 185,185,186,186,187,187,188,188,189,189,190,190
 dw 191,191,192,192,193,193,194,195,195,196,196,197
 dw 197,198,198,199,200,200,201,201,202,203,203,204
 dw 204,205,206,206,207,207,208,209,209,210,211,211
 dw 212,212,213,214,214,215,216,216,217,218,218,219
 dw 220,221,221,222,223,223,224,225,226,226,227,228
 dw 229,229,230,231,232,232,233,234,235,236,236,237
 dw 238,239,240,240,241,242,243,244,245,246,246,247
 dw 248,249,250,251,252,253,254,255,256,256,257,258
 dw 259,260,261,262,263,264,265,266,267,268,269,270
 dw 272,273,274,275,276,277,278,279,280,281,283,284
 dw 285,286,287,288,290,291,292,293,295,296,297,298
 dw 300,301,302,304,305,306,308,309,310,312,313,315
 dw 316,317,319,320,322,323,325,326,328,330,331,333
 dw 334,336,338,339,341,343,344,346,348,349,351,353
 dw 355,357,358,360,362,364,366,368,370,372,374,376
 dw 378,380,382,384,386,389,391,393,395,397,400,402
 dw 404,407,409,412,414,416,419,422,424,427,429,432
 dw 435,437,440,443,446,449,452,455,458,461,464,467
 dw 470,473,476,480,483,486,490,493,497,500,504,508
 dw 512,515,519,523,527,531,535,539,544,548,552,557
 dw 561,566,570,575,580,585,590,595,600,605,610,616
 dw 621,627,633,638,644,650,656,663,669,676,682,689
 dw 696,703,710,717,725,732,740,748,756,765,773,782

ZeroTunOffset ds 24*2

dw 0,100,192,276,360,436,508,576,640,700,756,810
 dw 860,908,954,996,1038,1076,1112,1146,1178,1208,1236,1264
 dw 1288,1312,1336,1356,1376,1396,1414,1430,1446,1462,1476,1490
 dw 1490

TempoDiv Dw \$28

Dw Reference_Freq/1+1
 Dw Reference_Freq/2+1
 Dw Reference_Freq/3+1
 Dw Reference_Freq/4+1
 Dw Reference_Freq/5+1
 Dw Reference_Freq/6+1
 Dw Reference_Freq/7+1
 Dw Reference_Freq/8+1


```

Table_Son      = *
                dfb $1E,$FA
Freq_L         = *-1
                dfb $3E,>$FA
Freq_H         = *-1
                dfb $5E,0           ; Volume
                dfb $9E,0           ; RamSon
                dfb $DE,0           ; Taille
                dfb $E1,$3C
                dfb $BE,$08         ; Mode FreeRun

```

```

Clear_Deb      = *

```

```

Volumina      dw 0
curr_instnum   dw 0

```

```

Timer         DS 2
NumberOfBlocks DS 2
BlockIndex    DS 2
BlockPlayed   DS 2
NoteIndex     DS 2
PositionBlock DS 2
Tempo         DS 2
InstIndex     DS 2

```

```

SemitoneTbl   ds Nb_Track*2
SampleTable   ds Nb_Track
ArpeggiattoTbl ds Nb_Track
ArpegeToneTbl ds Nb_Track*3
StartFreqTbl  ds Nb_Track*2
IncFreqTbl    ds Nb_Track*2
EndFreqTbl    ds Nb_Track*2
TrueVolumeTbl ds Nb_Track*2
TrackTune     ds Nb_Track*2

```

```

OscNumber     DS 2
Temporary     DS 2
Semitone      DS 2
Semitone_     DS 2
FineTune      ds 2
TempInterrupt DS 2
Temp2Interrupt DS 2
Temp3Interrupt DS 2
Temp4Interrupt DS 2
TempFreqInt   DS 2
VolumeInt     ds 2
CurrInstInt   ds 2
Performing    DS 2

```

```

LoopMode      DS  2
IndexInterrupt DS  2

ninfreqInt    dw  0

SwapperMode   ds  $20           ;1=swap mode active

Fineslider    ds  $20

NinVslideup   ds  $20
NinVslidedwn  ds  $20

ninTonePsp    ds  $20

Vibrato_Tbl   ds  $20           ;ptr to the table 0=none
Vibrato_Tbl_  ds  $20           ;ds
VibratoAdd_Tbl ds  $20           ;add to ptr
VibratoPtr_Tbl ds  $20          ;ptr in table

Ninjaloop     ds  15*2
NinjaFineT    ds  15*2
NinjaWaveAdr  ds  15*2
NinjaWaveSiz  ds  15*2
NinjaDocReg   ds  15*2

VolumeTable   DS  $20
StereoTable   DS  $20
CompactTable  DS  $20
Instrument1    =  *
instdef       DS  16*12

Clear_End     =  *

```

Subject: Re: NinjaTracker
Posted by [Dagen](#) on Wed, 03 Jun 2015 14:41:34 GMT
[View Forum Message](#) <> [Reply to Message](#)

Here's an example of how I use the ninjatracker source in my code.

In my case, I'm including it directly in my code with the "put" directive. The only difference is that I remove the "org" statement and just let it compile wherever. I don't believe the player or data files need to be page aligned, but I wouldn't go cross any bank boundaries (\$FFFF+).

*--- In my S16 program after starting the memory manager tools

```
lda #Mod1SSM
ldx #^Mod1SSM
jsr ReadFile          ; READ SSM FILE
_err "Module Not Found! : $" ; CHECK FOR ERRORS
```

```
lda $06              ; PATCH POINTERS FOR NINJATRACKER INIT
sta Music_File+2
lda $04
sta Music_File
```

```
lda #Mod1W
ldx #^Mod1W
jsr ReadFile          ; READ WAVE FILE
_err "Module Not Found! : $" ; CHECK FOR ERRORS
```

```
lda $06              ; PATCH POINTERS FOR NINJATRACKER INIT
sta Wave+2
lda $04
sta Wave
```

```
jsl Init_Sound
```

*--- sound should be playing after this

```
                ; do something ...
```

```
Mod1SSM  str 'STARDUST.SSM'      ; Module to be played
Mod1W    str 'STARDUST.W'        ; Module to be played
        put ntengine            ; the ninjatracker source above (minus 'org' statement)
```

I can post a full example application later with the complete ReadFile routine. This post was mostly to make sure that no one duplicates my effort to make the source usable in Merlin 16 & Merlin 32.

I hope this helps in some way. :d

Subject: Re: NinjaTracker
Posted by [jesseblue](#) on Sun, 06 Mar 2016 17:31:48 GMT
[View Forum Message](#) <> [Reply to Message](#)

Great job, Dagen.

I thought of translating the French comments to English, but it's more authentic and fun like this. :)
