The dramaturgy of music : its impact on my composition

Javier Alejandro Garavaglia

Awarded by

London metropolitan University

September 2010

Arte poética (I)

for 4-Channel Tape

Mirar el río hecho de tiempo y agua
Y recordar que el tiempo es otro río,
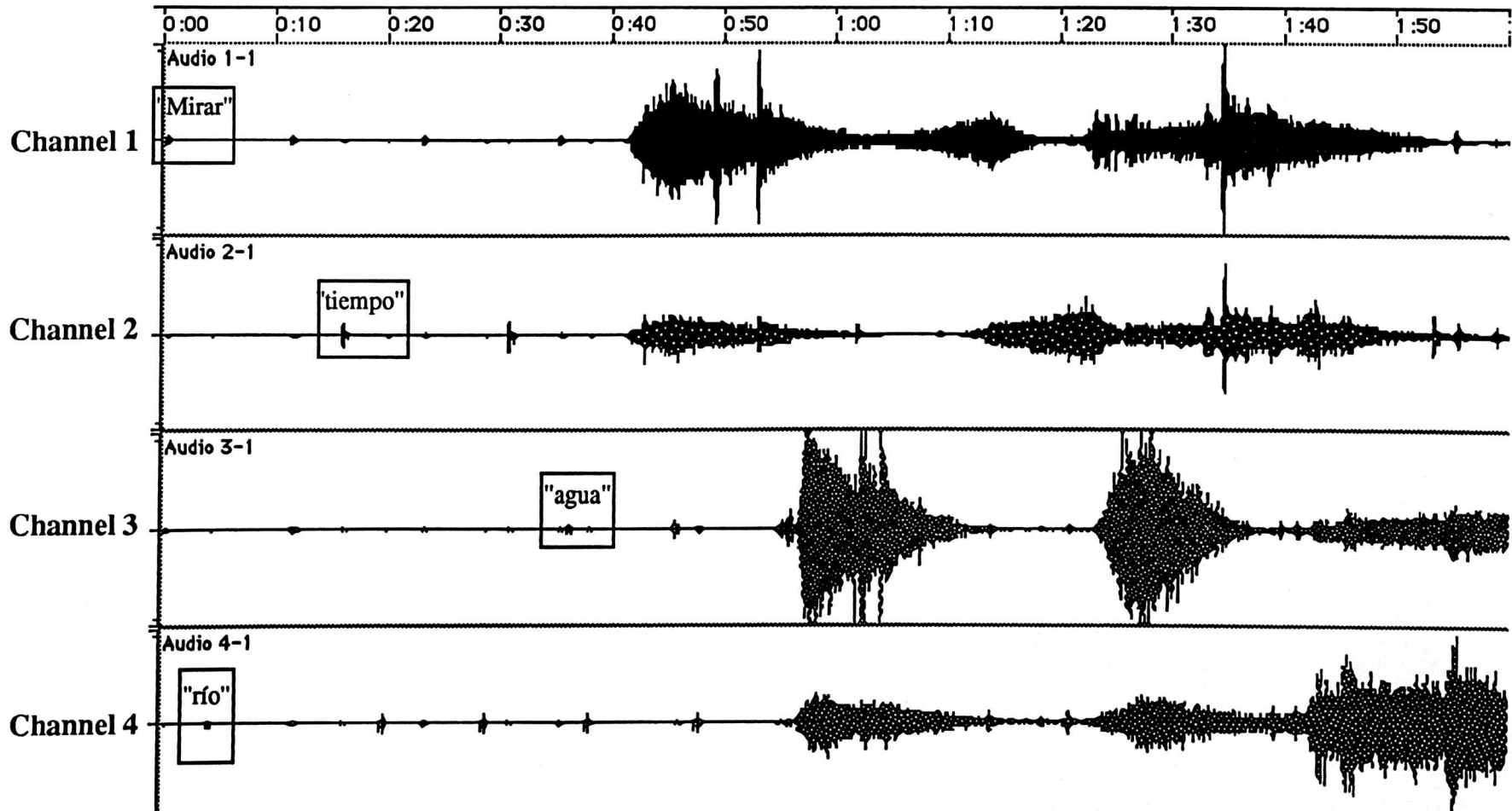Saber que nos perdemos como el río
Y que los rostros pasan como el agua.

Jorge Luis Borges
(from "Arte Poética")

Waveforms

Arte Poética (I)
Waveforms Score

Javier A. Garavaglia (1995)

Merge " CompleteWor ds"
CommonMusic / Channels 1-4
00:00-08:09

"AM + FM (a)"
C-Sound file
Ch. 1 --> 2 (*)
00:40-01:12

"AM + FM (c)"
C-Sound file
Ch. 3 <-- 4 / 00:55-01:25
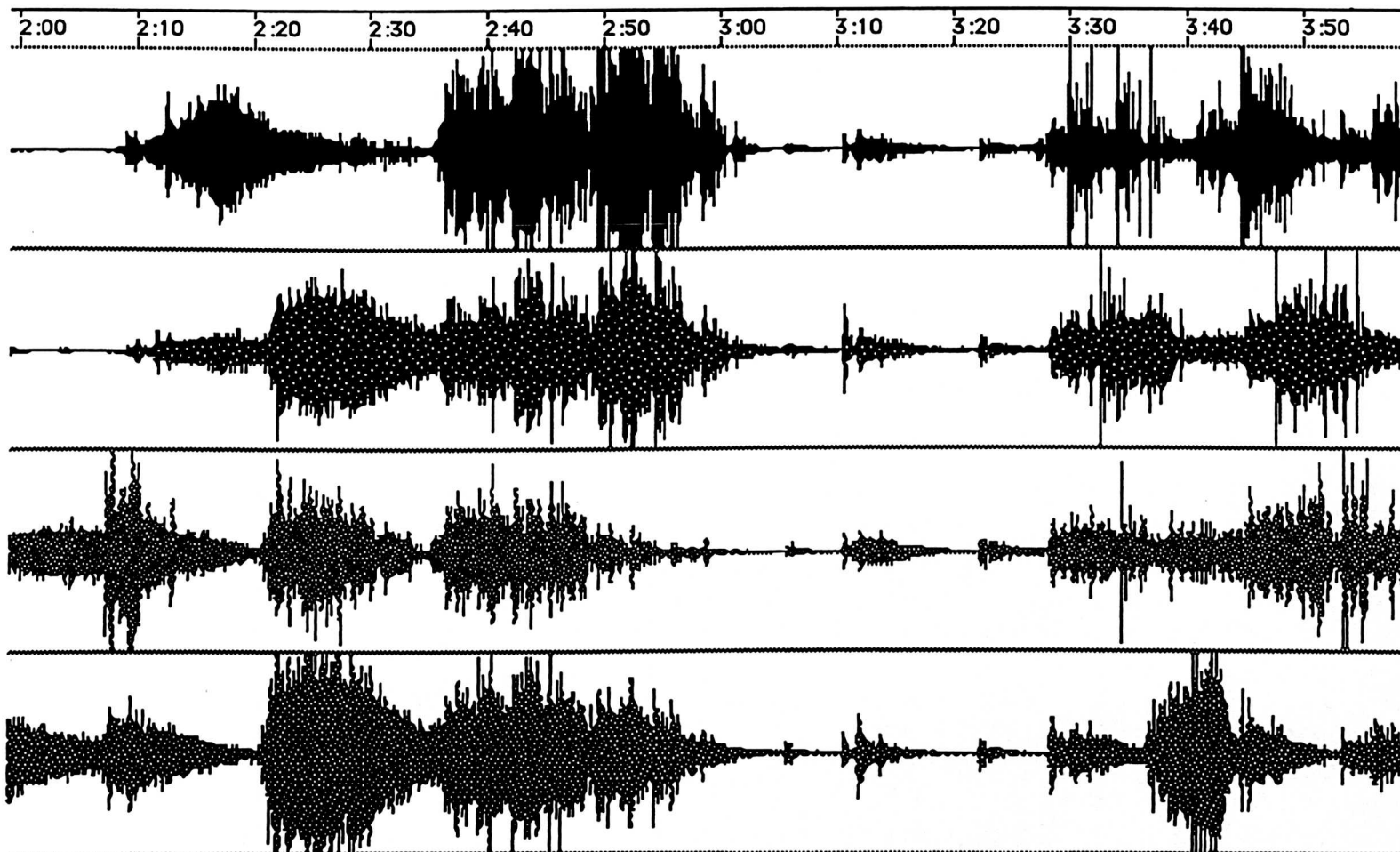
"y recordar + S&H"
C-Sound file
Ch.2 --> 4 / 01:35-02:04

"Mirar + S&H" - C-Sound file
Ch. 1-->2 / 01:00-01:45

"Saber + S&H"
C-Sound file
Ch. 3 <-- 4 / 01:50-02:13

Merge "TruncatedWords"
CommonMusic / Channels 1-4 / 01:00-02:00

(*) "Ch. 1 --> 2" *shows the Panorama direction.*

-1-

Timeline markings: 2:00 2:10 2:20 2:30 2:40 2:50 3:00 3:10 3:20 3:30 3:40 3:50
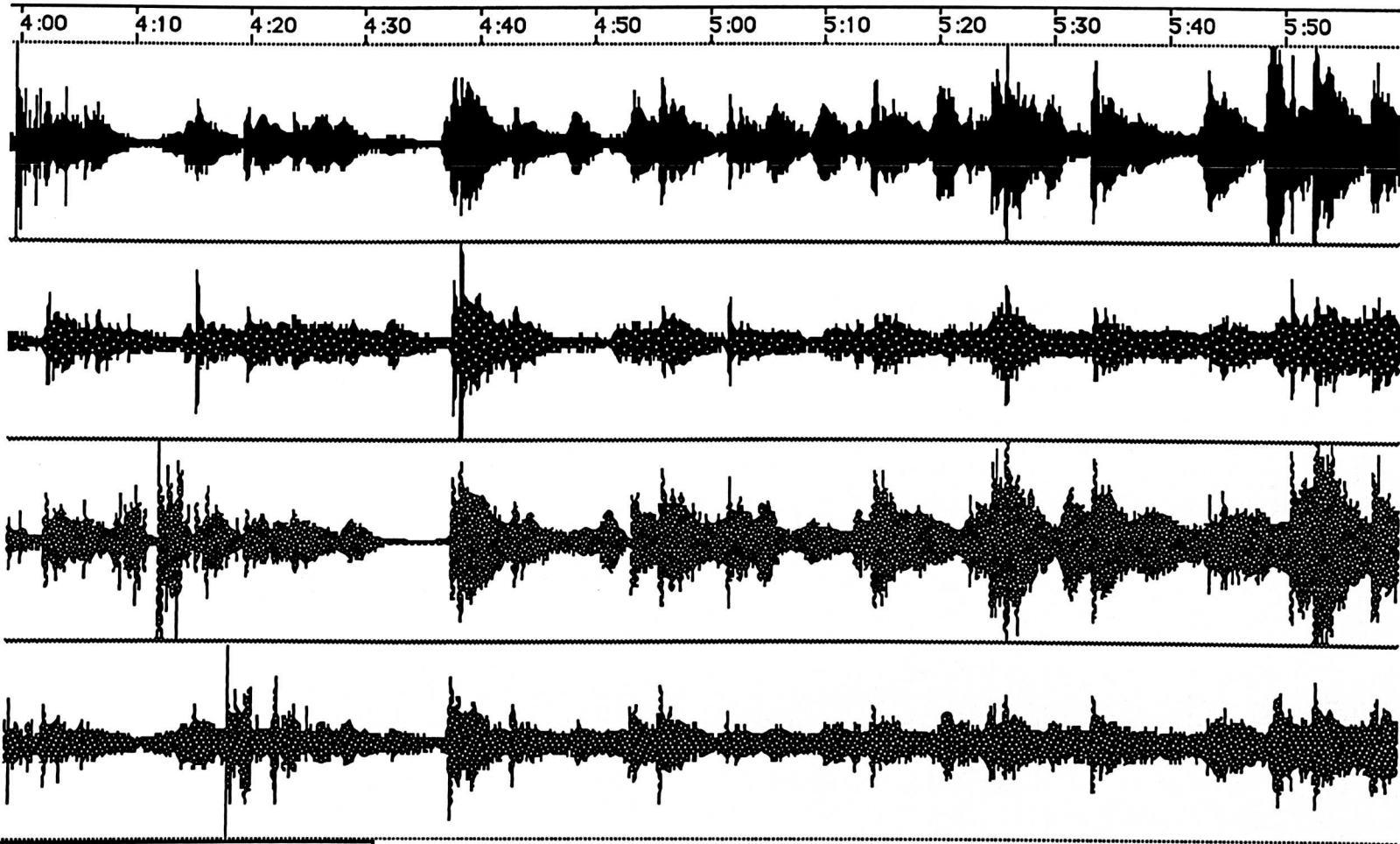
"y que los...+ S&H"
C-Sound file
Ch. 1<--3 / 02:05-02:27

"AM + FM (b)"
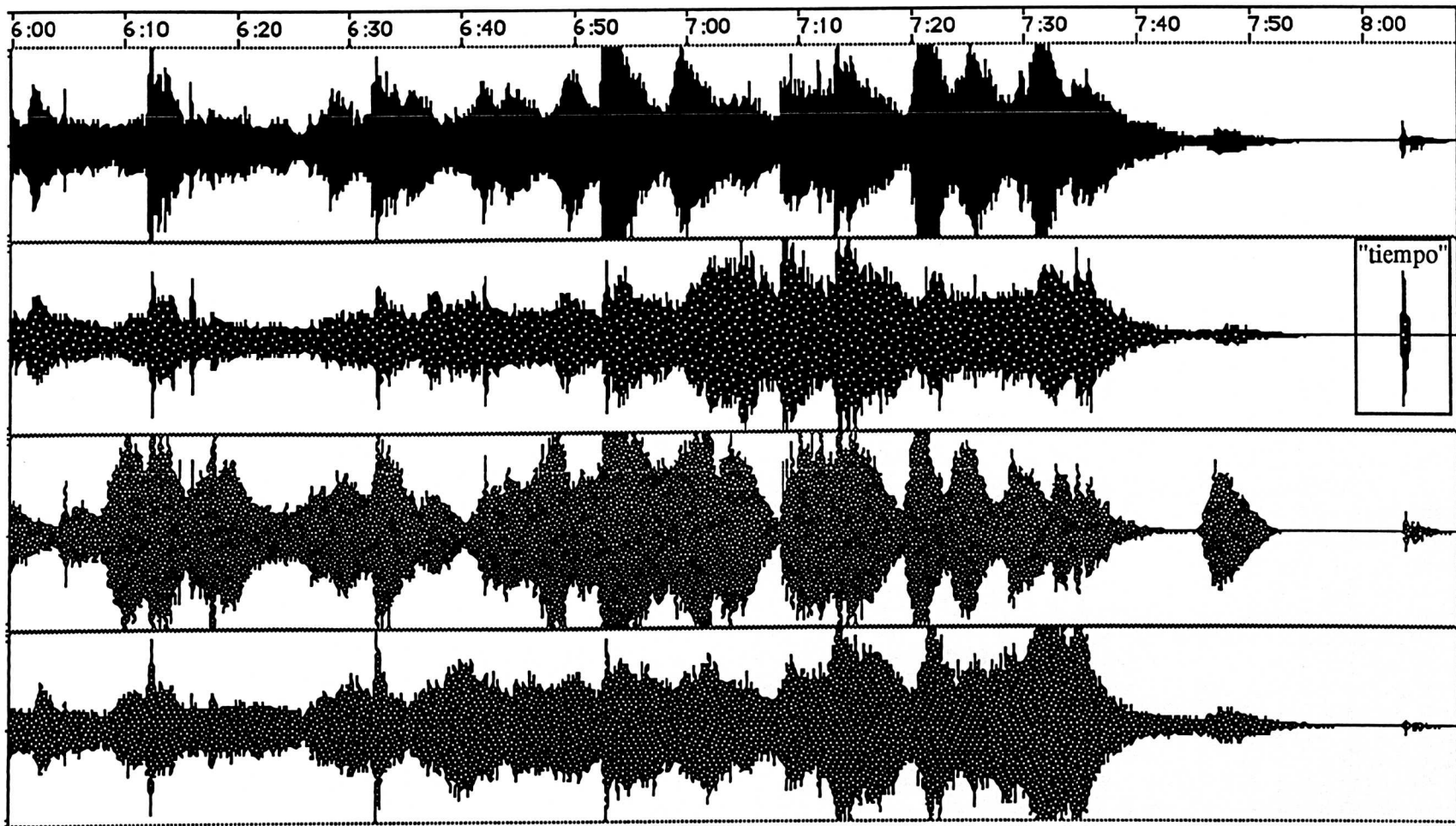C-Sound file
Ch. 1<-- 2 / 02:20-03:06

"FilArte 1" + "FilArte 2"
C-Sound files
    Ch. 1-4 / 03:00-04:03

Merge "TruncatedWords"
CommonMusic / Channels 1-4
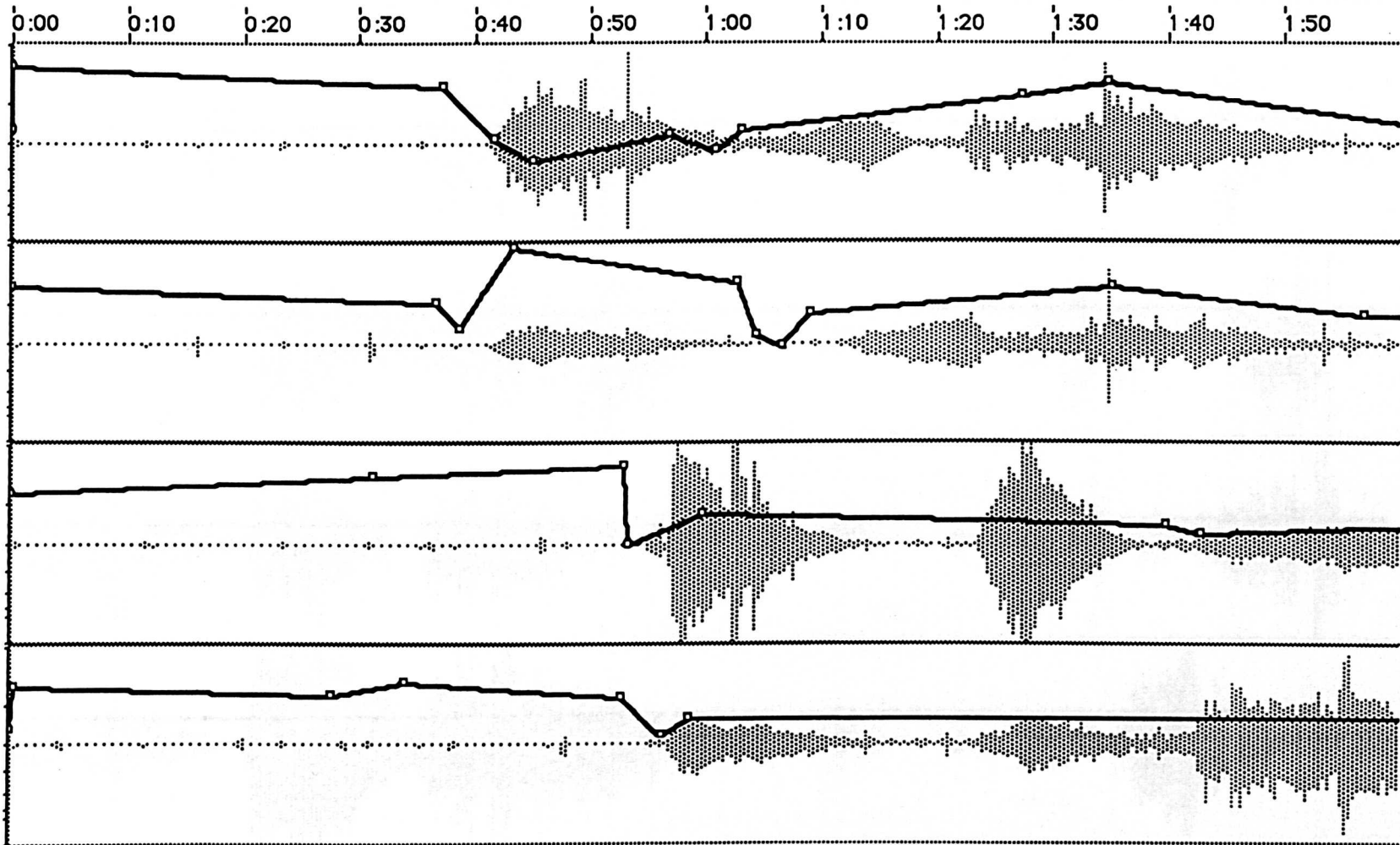   03:20-04:28

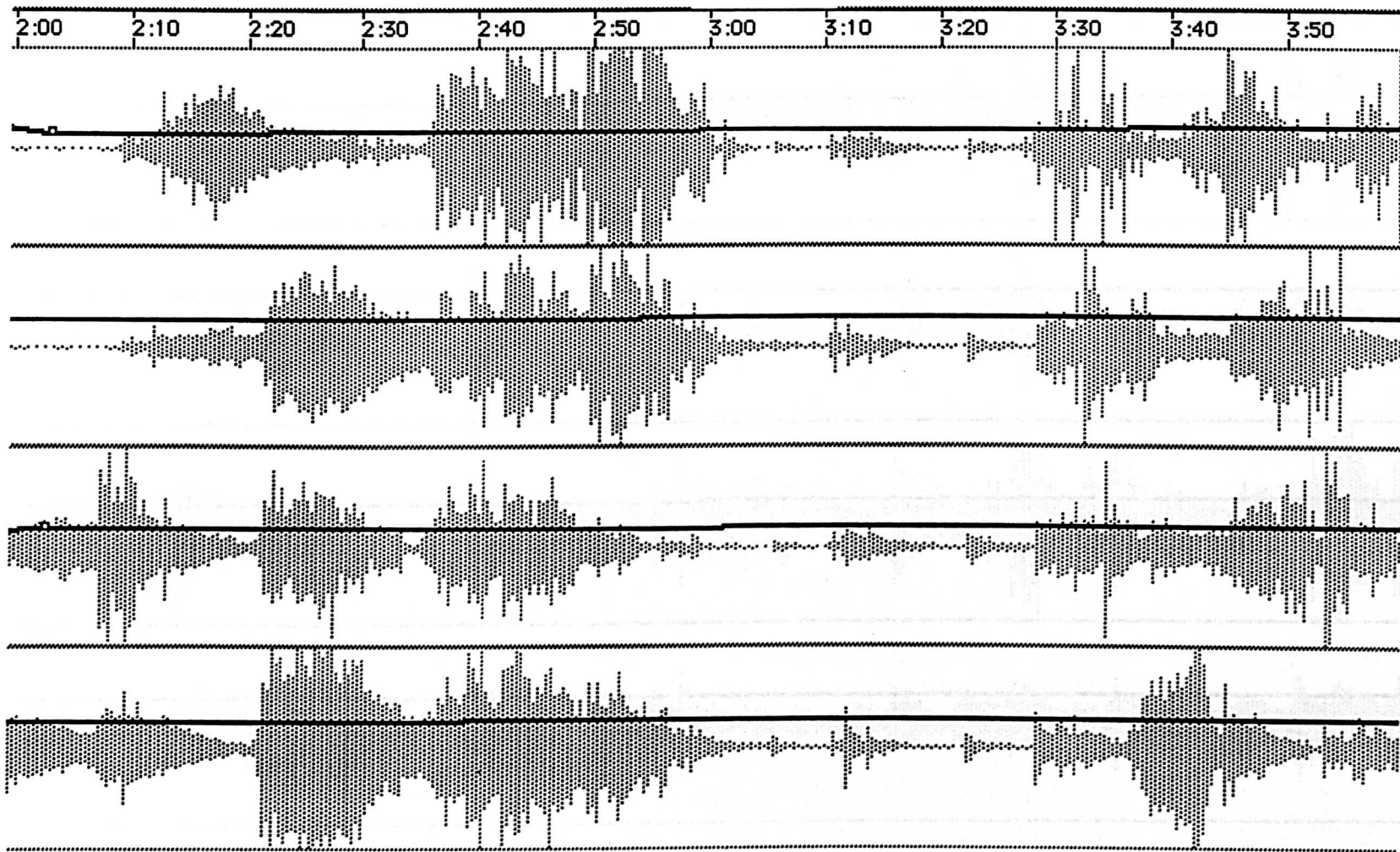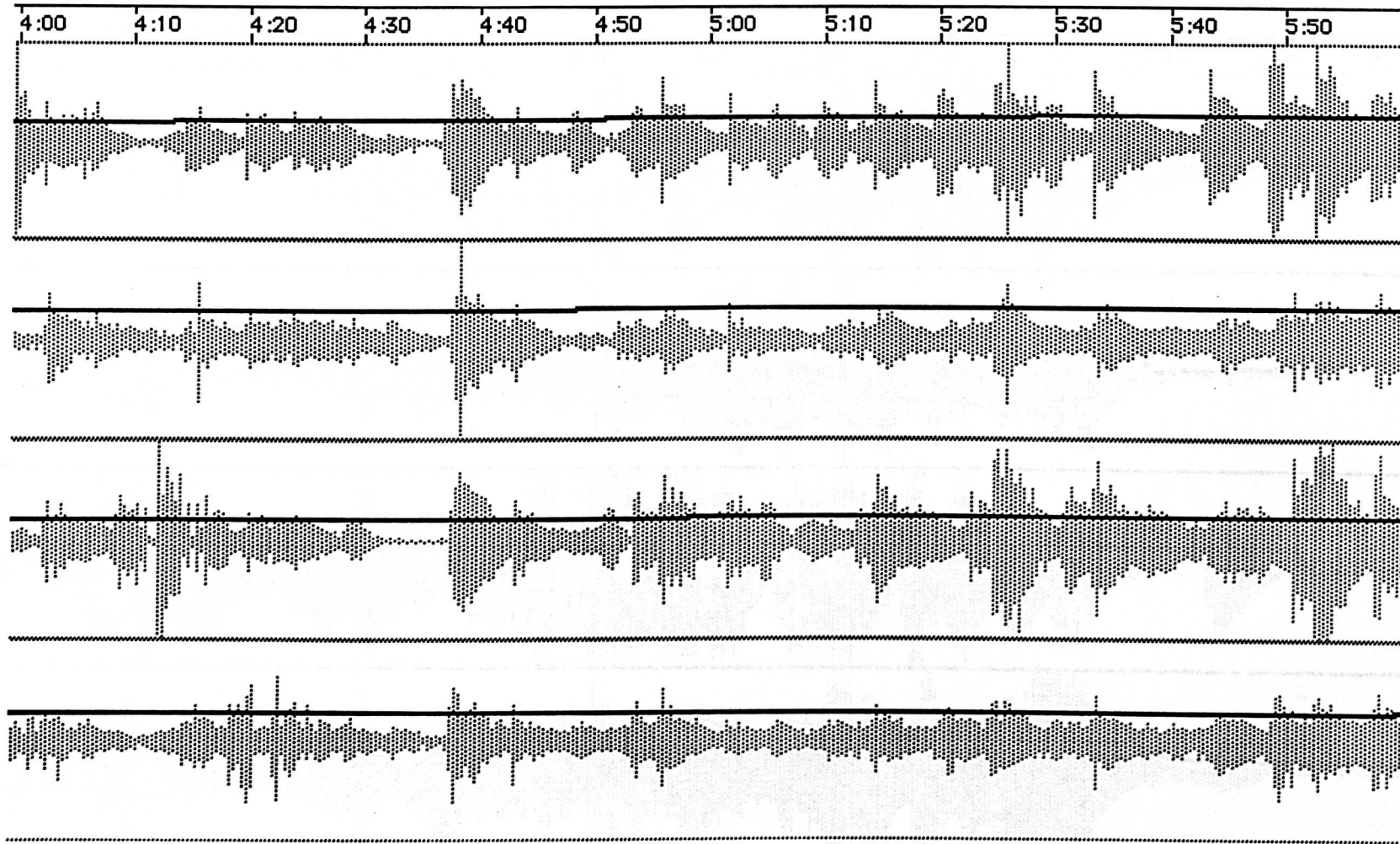Merge "ChownFM2" (2 Takes)
CommonMusic / Channels 1-4
04:00 / 08.09

"tiempo"

Merge "ChownFM1"
CommonMusic / Channels 1-4
06:00-08:09

Arte Poética (I)

Waveforms Score with Volume graphic

C-Sound

and

CommonMusic Data

Arte Poética (I)

C-Sound orc. and sco. Files ("*Water Files* ")

1)

```
;;AM+FM+STIMME(09a).orc

        sr=44100
        kr=441
        ksmps=100
        nchnls=2


            instr 1

iamp          =    ampdb(p4)
iskiptime    =    p6
ifmamp       =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise      =    p8*p3           ;%of total dur, 1=entire dur of note
ifmdec       =    p9*p3           ;% of total duration
ifmoff       =    p3 - (ifmrise + ifmdec)


asig1        soundin  "1)Mirar el río (Trans).aiff", iskiptime;; Voice, transposed 3 Octaves down.
kamp         linseg  0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a1           =        asig1*kamp

kfm          linseg  0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kindex       line    .36,p3,3
kndx         =       kfm * kindex
kbalance     line    0,p3,1
afm0                 foscil  iamp, p5,.36,.57,kndx,1
afm1                 foscil  iamp, p5*1.003,1.003,2.003,kndx,1
amfm         =       kfm*(afm0 * afm1);; the 2 FM will be amplitude modulated with themselves

ainfmam      =       (a1*amfm) /18 ;;;AM from the 2 FM will amplitude modulated with the Voice
```

```
          outs   ainfmam * kbalance,ainfmam *(1 - kbalance)

endin
```

instr 2

```
iamp          =    ampdb(p4)/7
iskiptime     =    p6
ifmamp        =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =    p8*p3           ;%of total dur, 1=entire dur of note
ifmdec        =    p9*p3           ;% of total duration
ifmoff        =    p3 - (ifmrise + ifmdec)


asig2         soundin       "2)Y recordar (TRANS).aiff", iskiptime ;;Voice, transposed 3 Octaves down.
kamp          linseg        0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a2            =             asig2*kamp

kfm       linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kindex    line     .57,p3,4
kndx      =        kfm * kindex
kbalance  line     1,p3,0
afm0               foscil  iamp, p5,.63,1.57,kndx,1
afm1               foscil  iamp, p5*1.003,1.003,2.003,kndx,1
afm       =        kfm*(afm0 + afm1);; sume of the 2 FM

ainfmam       =        a2*afm;; Voice with sume of the 2 FM will amplitude modulated

          outs   ainfmam * kbalance,ainfmam *(1 - kbalance)
endin
```

```
;;AM+FM+STIMME(09a).sco

t0 60
f1 0 2048 10 7   ; sinus

;AM=(Stimme Transponiert)(+)(*) 2 FM.
```

;ins	Start	Dur	AMP	Pitch	Skiptime	fmamp	fmrise	ifmdec
i1	0	28.87	27.5	136	0	.6	.1	.9
i2	7.0	25.65	27.5	157	0	.6	.1	.9

e

2)

;;AM+FM+STIMME(09c).orc

sr=44100
kr=441
ksmps=100
nchnls=2

instr 3

```
iamp          =    ampdb(p4)/7
iskiptime    =    p6
ifmamp       =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise      =    p8*p3           ;%of total dur, 1=entire dur of note
ifmdec       =    p9*p3           ;% of total duration
ifmoff       =    p3 - (ifmrise + ifmdec)


asig3        soundin  "3) Saber .. (TRANS) (aiff)", iskiptime
kamp         linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a3           =       asig3*kamp

kfm          linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kindex       line    .63,p3,7
kndx         =       kfm * kindex
kbalance     line    0,p3,1
afm0                 foscil  iamp, p5,1.75,2.75,kndx,1
afm1                 foscil  iamp, p5*1.003,1.003,2.003,kndx,1
afm          =       kfm*(afm0 * afm1)

ainfmam      =       a3*afm

             outs   ainfmam * kbalance,ainfmam *(1 - kbalance)
endin
```

instr 4

```
iamp          =    ampdb(p4)/7
iskiptime     =    p6
ifmamp        =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =    p8*p3           ;%of total dur, 1=entire dur of note
ifmdec        =    p9*p3           ;% of total duration
ifmoff        =    p3 - (ifmrise + ifmdec)


asig4         soundin  "4) Y que los (Trans).aiff", iskiptime
kamp          linseg  0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a4            =       asig4*kamp

kfm          linseg  0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kindex       line    1.57,p3,8
kndx         =       kfm *  kindex
kbalance     line    1,p3,0
afm0                 foscil  iamp, p5,3,4,kndx,1
afm1                 foscil  iamp, p5*1.003,1.003,2.003,kndx,1
afm          =       kfm*(afm0 + afm1)


ainfmam       =       a4*afm

outs   ainfmam * kbalance,ainfmam *(1 - kbalance)
endin
```

;;AM+FM+STIMME(09c).sco

```
t0 60
f1 0 2048 10 7                 ; sinus

;AM=(Stimme Transponiert)*fm
;ins Start    Dur   AMP Pitch Skiptime  fmamp  ifmrise  ifmdec
i3    0       23.52  27.5 175    0         .7      .1      .9
i4    7.0     22.57  27.5 275    0         .7      .1      .9
e
```

****

3)

;;1-MIRAR+SH.orc

```
sr=44100
kr=441
ksmps=100
nchnls=2

;;Sample&Holds
```

instr 1

```
iamp          =   ampdb(p4)*4
iskiptime     =   p6
ifmamp        =   p7            ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =   p8*p3         ;%of total dur, 1=entire dur of note
ifmdec        =   p9*p3         ;% of total duration
ifmoff        =   p3 - (ifmrise + ifmdec)
irvbgain      =   p11


asig2     soundin  "1)Mirar el río (Trans).aiff", iskiptime;;Voice, transposed 3 Octaves down (1st Verse)
kamp      linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a2        =        asig2*kamp



ascr      buzz      30000,400,10,int(4)
adiff     diff      ascr
anew      balance   adiff,ascr
agate     reson     a2,400,175;; Voice used as GATE for the S&H
asamp     samphold  anew,agate
ash       tone      asamp,1000

kbalance  line      0,p3,1
kfm0      linseg    0,p3/2,p4*1.75,p3,p4
a3        oscil     kfm0, (p5*1.36)+ash,int(4)
ash       =         a3+a2 ;; Sume of the S&H and the Voice

      outs  ash * kbalance,ash*(1-kbalance)
          ;garvbsig =garvbsig + ash*irvbgain
endin

;;1-MIRAR+SH.sco
```

```
t0 60
f1 0 2048 10 3 .5 .3 .25 .2 .167 .14 .125 .111; Sawtooth
f2 0 2048 10 7;Sinus
f3 0 2048 10 4 0 .3 0 .2 0 .14 0 .111; Square
f4 0 2048 10 8 1 1 1 .7 .5 .3 .1;Pulse
```

```
;ins   Start   Dur    AMP   Pitch  Skiptime  fmamp  ifmrise  ifmdec  rvbgain
i1      0      29.00   70     100      0        .4      .2       .9      .2
e
```

****

4)

;;2-Y RECORDAR+SH.orc

```
sr=44100
kr=441
ksmps=100
nchnls=2
```

;;S&H

instr 1

```
iamp          =    ampdb(p4)*4
iskiptime     =    p6
ifmamp        =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =    p8*p3            ;%of total dur, 1=entire dur of note
ifmdec        =    p9*p3           ;% of total duration
ifmoff        =    p3 - (ifmrise + ifmdec)
irvbgain      =    p11
```

```
asig2       soundin  "2)Y recordar (TRANS).aiff", iskiptime;;Voice, transposed 3 Octaves down (2nd Verse)
kamp        linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a2          =        asig2*kamp
```

```
ascr     buzz           30000,400,10,int(4)
adiff    diff           ascr
anew     balance        adiff,ascr
agate    reson          a2,400,275;;Voice used as GATE for the S&H
asamp    samphold       anew,agate
ash      tone           asamp,1000

kbalance line           0,p3,1
kfm0     linseg         0,p3/2,p4*2.75,p3,p4
a3       oscil          kfm0, (p5*1.57)+ash,int(4)
ash      =              a3+a2 ;;Sume of the S&H and the Voice

         outs   ash * kbalance,ash*(1-kbalance)
         ;garvbsig =garvbsig + ash*irvbgain
endin
```

;;2-Y RECORDAR+SH.sco

```
t0 60
f1 0 2048 10 3 .5 .3 .25 .2 .167 .14 .125 .111; Sawtooth
f2 0 2048 10 7;Sinus
f3 0 2048 10 4 0 .3 0 .2 0 .14 0 .111; Square
f4 0 2048 10 8 1 1 1 .7 .5 .3 .1;Pulse
```

;ins	Start	Dur	AMP	Pitch	Skiptime	fmamp	ifmrise	ifmdec	rvbgain
i1	0	27.00	70	100	0	.4	.2	.9	.2
e									

****

5)

;;3- SABER+SH.orc

```
sr=44100
kr=441
ksmps=100
nchnls=2
```

;;S&H

instr 1

```
iamp          =   ampdb(p4)*4
iskiptime     =   p6
ifmamp        =   p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =   p8*p3           ;%of total dur, 1=entire dur of note
ifmdec        =   p9*p3           ;% of total duration
ifmoff        =   p3 - (ifmrise + ifmdec)
irvbgain      =   p11


asig2         soundin  "3) Saber .. (TRANS) (aiff)", iskiptime ;;Voice, transposed 3 Octaves down (3rd Verse)
kamp          linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a2            =        asig2*kamp



ascr          buzz       30000,400,10,int(4)
adiff         diff       ascr
anew          balance    adiff,ascr
agate         reson      a2,400,300 ;;Voice used as GATE for the S&H
asamp         samphold   anew,agate
ash           tone       asamp,1000

kbalance      line       0,p3,1
kfm0          linseg     0,p3/2,p4*3,p3,p4
a3            oscil      kfm0, (p5*1.63)+ash,int(4)
```

```
ash            =          a3+a2;;Sume of the S&H and the Voice

       outs   ash * kbalance,ash*(1-kbalance)
endin
```

;;3- SABER+SH.sco

```
t0 60
f1 0 2048 10 3 .5 .3 .25 .2 .167 .14 .125 .111; Sawtooth
f2 0 2048 10 7;Sinus
f3 0 2048 10 4 0 .3 0 .2 0 .14 0 .111; Square
f4 0 2048 10 8 1 1 1 .7 .5 .3 .1;Pulse
```

;ins	Start	Dur	AMP	Pitch	Skiptime	fmamp	ifmrise	ifmdec	rvbgain
i1	0	24.00	70	100	0	.4	.2	.9	.2
e									

```
********************
```

6)

;;4-Y QUE LOS+SH.orc

```
       sr=44100
       kr=441
       ksmps=100
       nchnls=2
```

;;S&H
instr 1

```
iamp         =   ampdb(p4)*4
iskiptime    =   p6
ifmamp       =   p7           ;% of total amp, 1 ist gleich db amp in p4
ifmrise      =   p8*p3        ;%of total dur, 1=entire dur of note
ifmdec       =   p9*p3        ;% of total duration
ifmoff       =   p3 - (ifmrise + ifmdec)
irvbgain     =   p11
```

```
asig2          soundin      "4) Y que los (Trans).aiff", iskiptime ;;Voice, transposed 3 Octaves down (4th Verse)
kamp           linseg       0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a2             =            asig2*kamp


ascr           buzz         30000,400,10,int(4)
adiff          diff         ascr
anew           balance      adiff,ascr
agate          reson        a2,400,400 ;;Voice used as GATE for the S&H
asamp          samphold     anew,agate
ash            tone         asamp,1000

kbalance    line      0,p3,1
kfm0        linseg    0,p3/2,p4*4,p3,p4
a3          oscil     kfm0, (p5*2.57)+ash,int(4)
ash         =         a3+a2;;Sume of the S&H and the Voice

        outs  ash * kbalance,ash*(1-kbalance)
            ;garvbsig =garvbsig + ash*irvbgain
endin
```

;;4-Y QUE LOS+SH.sco

```
t0 60
f1 0 2048 10 3 .5 .3 .25 .2 .167 .14 .125 .111; Sawtooth
f2 0 2048 10 7;Sinus
f3 0 2048 10 4 0 .3 0 .2 0 .14 0 .111; Square
f4 0 2048 10 8 1 1 1 .7 .5 .3 .1;Pulse
```

;ins	Start	Dur	AMP	Pitch	Skiptime	fmamp	ifmrise	ifmdec	rvbgain
i1	0	23.00	70	100	0	.4	.2	.9	.2
e									

****

7)

;;AM+FM+STIMME(09b).orc

```
        sr=44100
```

```
kr=441
ksmps=100
nchnls=2
```

instr 5

```
iamp            =    ampdb(p4)/11
iskiptime       =    p6
ifmamp          =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise         =    p8*p3           ;%of total dur, 1=entire dur of note
ifmdec          =    p9*p3           ;% of total duration
ibalance        =    p10          ; 1=left-0=right
index           =    p11
ifmoff          =    p3 - (ifmrise + ifmdec)

asig5       soundin  "ARTE POET 1-EST.AIFF", iskiptime;; Whole first Stanza in original Pitch

kamp        linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
a5          =        asig5*kamp

kfm         linseg      0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kndx        =        kfm * index
kfreqfm     line     p5,p3,p5*1.36
afm0        foscil   iamp, kfreqfm,1,1,kndx,1
afm1        foscil   iamp, kfreqfm*1.003,1.003,2.003,kndx,1
afm         =        kfm*(afm0 + afm1)
ainfm       =        a5*afm;; Whole first Stanza will amplitude modulated with 2 FM.

        outs   ainfm * ibalance,ainfm *(1 - ibalance)
endin
```

;;AM+FM+STIMME(09b).sco

```
t0 60
f1 0 2048 10 7                ; sinus
```

;2 FM + 1. Strophe komplett (Gesamtdauer = 42.56 Sek.)

;ins Start	Dur	AMP	Pitch	Skiptime	fmamp	ifmrise	ifmdec	ibalance	index(FM)
i5 0	14.18	30	40	0	.7	.1	.99	.01	4
i5 .36	14.18	>	70	0	.7	.1	.99	>	>

i5	.57	14.18	>	80	0	.7	.1	.99	>	>
i5	.63	14.18	>	110	0	.7	.1	.99	>	>
i5	1.57	14.18	>	120	0	.7	.1	.99	>	>
i5	1.75	14.18	>	140	0	.7	.1	.99	>	>
i5	2.75	14.18	>	150	0	.7	.1	.99	>	>
i5	3.00	14.18	>	160	0	.7	.1	.99	>	>
i5	4.00	14.18	>	180	0	.7	.1	.99	>	>
i5	7.00	14.18	>	210	0	.7	.1	.99	>	>
i5	8.00	14.18	>	220	0	.7	.1	.99	>	>
i5	11.00	14.18	>	280	0	.7	.1	.99	>	>
i5	12.00	14.18	>	300	0	.7	.1	.99	>	>
i5	14.00	14.18	>	330	0	.7	.1	.99	>	>
i5	15.00	14.18	>	360	0	.7	.1	.99	>	>
i5	16.00	14.18	>	440	0	.7	.1	.99	>	>
i5	18.00	14.18	>	450	0	.7	.1	.99	>	>
i5	21.00	14.18	>	490	0	.7	.1	.99	>	>
i5	22.00	14.18	>	540	0	.7	.1	.99	>	>
i5	28.00	14.18	>	660	0	.7	.1	.99	>	>
i5	30.00	14.18	33	770	0	.7	.1	.99	.99	15
e										

****

8)

;;FiltARTE1.orc
sr = 44100
kr = 441
ksmps = 100
nchnls = 2

instr 1

iamp	=	(ampdb(p4)) - 3
ifreqstart	=	p5 ; Ausgang Freq. für Filter
ifreqend	=	p6 ; Ende Freq. für Filter
ibandwidth	=	p7 ; Bandbreite des Filters
iskiptime	=	p8
ifmrise	=	p9*p3 ;%of total dur, 1=entire dur of note

```
ifmdec          =   p10*p3          ;% of total duration
iampl           =   p11             ;% of total amp, 1 ist gleich db amp in p4  für FM
ifmoff          =   p3 - (ifmrise + ifmdec)


kfiltfreq       line    p5, p3, p6
kbandwidth      linseg  p7,p3,p7*1.5
kbalanceleft    linseg  0,7,0.25,7,0.50,8,0.8
kbalanceright   linseg  0.8,7,0.50,7,0.25,8,0
```

;AM between 2 Signals with the voice, then filtered

```
asig1           soundin   "ARTE POET 1-EST.AIFF", iskiptime
asig1filt       =                   asig1/40; wenig Pegel, um die Filter und AM  nicht zu übersteuern
aamasig1        =       (asig1filt * asig1filt)/1.6 ; AM1
afilt           reson             aamasig1,kfiltfreq, kbandwidth
asigamfilt      =       afilt
```

;Chowning FM (TWICE)

```
ktabelle        randh   int(100),p3,0
kfreqfm1        line    300,p3,400
kfreqfm2                                line     300,p3,275
kampfm          linseg  0,ifmrise,iampl,ifmdec,0,ifmoff,0
kindex          line    0,p3,7
kndx            =       kampfm * kindex

afm0            foscil  p4*7, kfreqfm1 ,ktabelle,ktabelle,kndx,1
afm1            foscil  p4*7, kfreqfm2,1,3,kndx,1
afm             =       kampfm* (afm0 + afm1);
afmamfilt       =       (afm * asigamfilt)/77      ;; filtered AM1 *(FM1+FM2) =AM2


                outs            afmamfilt * kbalanceleft, afmamfilt * kbalanceright

endin
```

;;FiltARTE1.sco

```
t0 60
f1 0 2048 10 7  ;Sine wave
```

```
; ins    strt    dur    amp    startfr    endfr    bdw  skptm  fmrise  fmdec  fmamp
  i10     0      15.0    7      7000       8000     210   0      .15     1      .9
e
```

; The voice will be amplitude modulated with itself then it will be filtered by RESON (dynamically), then 2 FM
; will be produced and added. Output: the FMs will be multiplied with the filtered amplitude modulated voice (AM between
; AMfilt and the 2 FMs)

****
9)

;;FiltARTE2.orc

```
sr = 44100
kr = 441
ksmps = 100
nchnls = 2

garvbsig init 0
```

instr 1

```
iamp            =    (ampdb(p4)) - 3
ifreqstart      =    p5  ; Ausgang Freq. für Filter
ifreqend        =    p6  ; Ende Freq. für Filter
ibandwidth      =    p7  ; Bandbreite des Filters
irvbgain        =    p8
iskiptime       =    p9
ifmrise         =    p10*p3        ;%of total dur, 1=entire dur of note
ifmdec          =    p11*p3        ;% of total duration
iamp1           =    p12           ;% of total amp, 1 ist gleich db amp in p4  für FM
ifmoff          =    p3 - (ifmrise + ifmdec)


kfiltfreq       line     p5, p3, p6
kbandwidth      linseg   p7,p3,p7*1.5
kbalanceleft    linseg   0,7,0.25,7,0.50,8,0.8
kbalanceright   linseg   0.8,7,0.50,7,0.25,8,0
```

;AM between 2 Signals with human voice, then filtered


```
asig1           soundin    "ARTE POET 1-EST.AIFF", iskiptime
asig1filt       =          asig1/40;wenig Pegel, um die Filter und AM  nicht zu übersteuern
aamasig1        =          (asig1filt * asig1filt)/1.6 ; AM1
afilt           reson              aamasig1,kfiltfreq, kbandwidth
asigamfilt      =          afilt
```

;Chowning FM (TWICE)


```
ktabelle        randh      int(100),p3,0
kfreqfm1        line       300,p3,400
kfreqfm2                              line      300,p3,275
kampfm          linseg     0,ifmrise,iamp1,ifmdec,0,ifmoff,0
kindex          line       0,p3,7
kndx            =          kampfm * kindex

afm0            foscil     p4*7, kfreqfm1 ,ktabelle,ktabelle,kndx,1
afm1            foscil     p4*7, kfreqfm2,1,3,kndx,1
afm             =          kampfm* (afm0 + afm1);
afmamfilt       =          (afm * asigamfilt)/77      ;; filtered AM1 *(FM1+FM2) =AM2


                outs          afmamfilt * kbalanceleft, afmamfilt * kbalanceright

   garvbsig     =         garvbsig + afmamfilt * p8
endin
```

instr 100
```
krvbtime  line  .36,p3-8,7

asig0           reverb     garvbsig, krvbtime
kbalanceleft    linseg     0,7,0.25,7,0.50,8,0.8
kbalanceright   linseg     1,7,0.50,7,0.25,8,0
        outs    asig0 * kbalanceleft , asig0 * kbalanceright
   garvbsig     =        0
endin
```

;;FilARTE2.sco

```
t0 60
f1 0 2048 10 7  ;Sine wave

; ins    strt    dur
i100     0       22
```

```
; ins   strt    dur    amp    startfr   endfr    bdw   rvsnd   skptm   fmrise   fmdec   fmamp
 i10    0       15.0   7      7000      8000     210   .1      0       .15      1       .9
e
```
; The voice will be amplitude modulated with itself then it wil be filtered by RESON (dynamically) then 2 FM will be produced and added.
; Output: the FMs will be multiplied with the filtered amplitude modulated voice (AM between AMfilt and the 2 FMs)

****

C-Sound orc. and sco. Files (*"Time Files "*)
Chowning FM Samples for the "TIEMPO"-SAMPLES

;;TiempoSample 1 - Orchestralfile

```
sr=44100
kr=441
ksmps=100
nchnls=1


instr  1

iamp         =    ampdb(p4)
iskiptime    =    p6
ifmamp       =    p7             ;% of total amp, 1 ist gleich db amp in p4
ifmrise      =    p8*p3          ;%of total dur, 1=entire dur of note
ifmdec       =    p9*p3          ;% of total duration
ifmoff       =    p3 - (ifmrise + ifmdec)


kfreqfm      line     30,p3,33
kampfm       linseg   0,ifmrise, ifmamp, ifmdec, 0, ifmoff, 0
kindex       line     0,p3,15
kndx         =        kampfm * kindex
ktabelle1    randh    ( int(200)),p3,0
```

```
afm0            foscil   iamp, kfreqfm , ktabelle1, ktabelle1+1, kndx,1
afm1            foscil   iamp, kfreqfm*1.001, 1.001, 2.001,kndx,1
afm             =        kampfm*(afm0 + afm1)


                out      afm
endin


;;TiempoSample 1 -  Scorefile

t0 60
f1 0 2048 10 7                               ; Sine
f2 0 2048 10 1 .5 .3 .25 .2 .167 .14 .125 .111   ; Sawtooth
f3 0 2048 10 1 0 .3 0 .2 0 .14 0 .111       ; Square
f4 0 2048 10 1 1 1 1 .7 .5 .3 .1            ;Pulse

;AM
;ins Start   Dur  AMP  Pitch  Skiptime  fmamp  ifmrise  ifmdec
i1    0      7.0   80    0       0        .6      .2       .9


e

;;TiempoSample 2 - Orchestralfile

sr=44100
kr=441
ksmps=100
nchnls=1

instr 1

iamp            =    ampdb(p4)
iskiptime       =    p6
ifmamp          =    p7          ;% of total amp, 1 ist gleich db amp in p4
ifmrise         =    p8*p3       ;%of total dur, 1=entire dur of note
ifmdec          =    p9*p3       ;% of total duration
ifmoff          =    p3 - (ifmrise + ifmdec)

ktabelle        randh  (100 + int(100)), p3, 0
kfreqfmcr       line   157,p3,175
```

```
;kfreqfmmd      line      275,p3,300
kfreqfmmd       line       57,p3,36
kampfm          linseg    0, ifmrise, ifmamp, ifmdec, 0, ifmoff, 0
kindex          line      4,p3,11
kndx            =         kampfm * kindex
ktabelle        randh     int (100),p3,0
kbalance        line      0,p3,1


afm0            foscil    iamp, kfreqfmcr, ktabelle-100, ktabelle-100, kndx,1
afm1            foscil    iamp, kfreqfmmd*1.004, 1.004, 2.004, kndx, 1
afm             =         kampfm* (afm0 + afm1)
kbalance        line      0,p3,1
ainam           =         afm
                out       ainam
endin
```

;;TiempoSample 2 - Scorefile

```
t0 60
f1 0 2048 10 7                           ; Sine
f2 0 2048 10 1 .5 .3 .25 .2 .167 .14 .125 .111  ; Sawtooth
f3 0 2048 10 1 0 .3 0 .2 0 .14 0 .111    ; Square
f4 0 2048 10 1 1 1 1 .7 .5 .3 .1          ;Pulse
```

;AM

```
;ins Start    Dur  AMP Pitch Skiptime  fmamp  ifmrise  ifmdec
i1    0       7.0  80   0       0        .6      .2       .9


e
```

;;TiempoSample 3 - Orchestralfile

```
sr=44100
kr=441
ksmps=100
nchnls=1

instr 1
```

```
iamp          =    ampdb(p4)
iskiptime     =    p6
ifmamp        =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =    p8*p3           ;%of total dur, 1=entire dur of note
ifmdec        =    p9*p3           ;% of total duration
ifmoff        =    p3 - (ifmrise + ifmdec)


kfreqfm       line      66,p3,77
kampfm        linseg    0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kindex        line      0,p3,16
kndx          =         kampfm * kindex
ktabelle1     randh   ( int (360)),p3,0


afm0          foscil    iamp, kfreqfm , ktabelle1, ktabelle1+1, kndx,1
afm1          foscil    iamp, kfreqfm*1.001,1 .001, 2.001,kndx,1
afm           =         kampfm*( afm0 + afm1)

              out       afm
endin
```

;;TiempoSample 3 - Scorefile

```
t0 60
f1 0 2048 10 7                                ; Sine
f2 0 2048 10 1 .5 .3 .25 .2 .167 .14 .125 .111   ; Sawtooth
f3 0 2048 10 1 0 .3 0 .2 0 .14 0 .111         ; Square
f4 0 2048 10 1 1 1 1 .7 .5 .3 .1              ;Pulse

;AM
;ins Start   Dur  AMP Pitch Skiptime  fmamp  ifmrise  ifmdec
i1   0       7.0  80   0      0        .6      .2       .9


e
```

;;TiempoSample 4 - Orchestralfile

```
sr=44100
kr=441
```

```
ksmps=100
nchnls=1


instr  1

iamp          =    ampdb(p4)
iskiptime     =    p6
ifmamp        =    p7              ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =    p8*p3           ;%of total dur, 1=entire dur of note
ifmdec        =    p9*p3        ;% of total duration
ifmoff        =    p3 - (ifmrise + ifmdec)


kfreqfm            line      490,p3,450
kampfm             linseg    0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kindex             line      0,p3,18
kndx               =         kampfm * kindex
ktabelle1          randh   ( int(36)),p3,100


afm0          foscil    iamp, kfreqfm , ktabelle1+1, int(3), kndx, 1
afm1          foscil    iamp, kfreqfm*1.001, 1.001, 2.001, kndx,1
afm           =         kampfm*(afm0 + afm1)
       out    afm
endin


;;TiempoSample  4 - Scorefile

t0 60
f1 0 2048 10 7                              ; Sine
f2 0 2048 10 1 .5 .3 .25 .2 .167 .14 .125 .111    ; Sawtooth
f3 0 2048 10 1 0 .3 0 .2 0 .14 0 .111         ; Square
f4 0 2048 10 1 1 1 1 .7 .5 .3 .1              ;Pulse

;AM
;ins Start   Dur  AMP Pitch Skiptime fmamp ifmrise ifmdec
i1    0     7.0   80    0      0       .6     .2      .9


e
```

```
;;TiempoSample 5 - Orchestralfile

sr=44100
kr=441
ksmps=100
nchnls=2
garvbsig   init 0


instr  1

iamp          =   ampdb(p4)
iskiptime     =   p6
ifmamp        =   p7            ;% of total amp, 1 ist gleich db amp in p4
ifmrise       =   p8*p3         ;%of total dur, 1=entire dur of note
ifmdec        =   p9*p3         ;% of total duration
irvbgain      =   p10
ifmoff        =   p3 - (ifmrise + ifmdec)


ktabelle      randh    (100 + int(100)),p3,0
kfreqfmcr     line     57,p3,63
kfreqfmmd     line     63,p3,36
kampfm        linseg   0,ifmrise,ifmamp,ifmdec,0,ifmoff,0
kindex        line     4,p3,11
kndx          =        kampfm * kindex
kbalance      line     0,p3,1


afm0          foscil   iamp, kfreqfmcr , ktabelle-100, ktabelle-100, kndx,int(4)
afm1          foscil   iamp, kfreqfmmd*1.002, 1.002, 2.002, kndx, int(4)
afm           =        kampfm*(afm0 + afm1)

outs      afm * kbalance, afm *(1 - kbalance)

garvbsig= garvbsig+afm*p10
endin

instr 2

krvbtime   line    .36,p3,15
asig       reverb          garvbsig,krvbtime
          outs    asig,asig
```

```
garvbsig=0

endin
```

```
t0 60
f1 0 2048 10 9                              ; Sine
f2 0 2048 10 1 .5 .3 .25 .2 .167 .14 .125 .111    ; Sawtooth
f3 0 2048 10 1 0 .3 0 .2 0 .14 0 .111       ; Square
f4 0 2048 10 1 1 1 1 .7 .5 .3 .1            ;Pulse

;AM
;ins Start   Dur   AMP  Pitch  Skiptime  fmamp  ifmrise  ifmdec  irvbgain
i1    0      4.0   80    0        0        .6      .2       .9      .5
i2    0      15.0
e
```

****

Arte Poética (I)
CommonMusic Algorithms

1)

(**Merge** CompleteWords ()
;;Grundalgorithmus für "ARTE POETICA (I)". Dauer= 8 Min. 02 Sek.(Samples auf den Akai S1000=> "agua",
"mirar", ;;"tiempo", "río", "recordar".)
;; Die Länge (ohne das Inkrement) jedes Algorithmus wird von *CalculateLength* bestimmt,wobei die Länge vom
;;Inkrement immer verschieden sein soll.
;; Samplesdauer auf dem Sampler (AKAI) ---> (C4->Ursprüngliche Tonhöhe)
;; Alle Algorithmen enden (fast) an der selben Zeitpunkt (Minute 8)

```
(loop for beg from 1 to 5
for n in '(Mirar2 río2 tiempo2 agua2 recordar2 )
for begin in '(0 4 16 36 77)
for Inkrement in '(.36 .57 .63 1.57 1.75)
for Ton in '( c4 c4 c4 c4 c4 )
for MidiKanal in '(0 3 1 2 4 );; MidiChannel 1 2 3 4 5 (Die Reihenfolge der Samples beim Akai ist (Mirar-Tiempo-agua-río-recordar)
for Durchlauf in '(0 1 2 3 4)
for Samplelength in '( 0.580 0.543 0.911 0.685 0.722 )
for totalDauer in '(160 280 190 30 22);;Die Grenze von der maximalen Dauer (ohne Inkrement) wird in Sek. gesetzt.
for DauerInterval in '(11 7 14 8 12)
for AnfangAmplitude in '(.236 .257 .363 .257 .236)
for MaxAmplitude in '( .8 .8 .9 .8 .8 )
do


(let* ((Noten Ton)
(Kanal Midikanal)
(Ink Inkrement)
(Durchl Durchlauf)
(Spllght Samplelength)
(TotDauer TotalDauer)
(DauerIntervall DauerInterval)
(AnfAmp AnfangAmplitude)
(MaxAmp MaxAmplitude))


(algorithm (name n) midi-note (start begin)
(vars (i Ink) (FreqRatio 1) (MomentDauer DauerIntervall) (amp1 AnfAmp) (amp2 MaxAmp)
(Länge 0) (CumulateDauer 0) (Increment 0)(DL Durchl))


(setf DL dl)
(setf Länge (CalculateLength i (+ Spllght Momentdauer) TotDauer)) ;; Die Zahl der Elementen
(setf Channel kanal)
(setf Increment i )
(setf FreqRatio (between 1 1.007))
(setf Note (item (steps 1 from Noten):kill länge))
(setf Note (item(pitches (steps 1 from Noten))))
(setf note (* note FreqRatio)) ;;Multiply the Pitch by Random number with a Ratio < 1 Octave
(setf Rhythm (incf MomentDauer Increment))
(setf Duration Rhythm)
(setf Amplitude (interpl (mod count Länge)  0 amp1
(- Länge 1) amp2))
(setf MomentDauer Duration)
(setf CumulateDauer (+ MomentDauer CumulateDauer))
```

```
(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE CumulateDauer-------->) (/ CumulateDauer 60.0) (quote min.--->)dl)))
(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE Länge-------->)  Länge  (quote Length--->)Channel)))

);;End of algorithm

(defun CalculateLength (Increment SampleDauer GesamtDauer)
;;Durch diese Funktion wird die Länge jedes Alg. kalkuliert
(let* ((index 0)(Cumulate 0)(Länge 0)(MomentDauer 0));;(Länge ist wie Length dh. wieviele Elemente wird index haben)
(setf Increment
(cond
((And(> Increment 1) (< increment 10)(/ Increment 10.0)))
((And(> Increment 10) (< increment 100)(/ Increment 100.0)))
((And(> Increment 100) (< increment 1000)(/ Increment 1000.0)))
( Increment )))
(setf Länge  (-(Truncate SampleDauer Increment)1));;DURCH DIESE DIVISION WIRD LÄNGE KALKULIERT
(loop
(setf index(+ index 1))
(setf MomentDauer (incf MomentDauer Increment))
(setf Cumulate    (incf Cumulate MomentDauer))
(cond
((Or(> index Länge) (> Cumulate GesamtDauer))(return  (- index 1))))
;(Print (list (/ Cumulate 60)  (quote min.bei.Länge)  länge ))
);;of Loop


);;of let*

);; end of CalculateLength



);; end of Let* (of Loop)
);;end of LOOP
) ;; End of Merge
```

Dauer des Merges

```
;(CUMULATEDAUER--------> 6.5 MIN.---> 4)
;(lLäNGE-------->l 15 LENGTH---> 4)
;(CUMULATEDAUER--------> 7.505 MIN.---> 3)
;(lLäNGE-------->l 19 LENGTH---> 2)
;(CUMULATEDAUER--------> 7.9266666666666664 MIN.---> 0)
;(lLäNGE-------->l 29 LENGTH---> 0)
;(CUMULATEDAUER--------> 7.917500000000002 MIN.---> 1)
;(lLäNGE-------->l 30 LENGTH---> 3)
;(CUMULATEDAUER--------> 8.749999999999996 MIN.---> 2)
;(lLäNGE-------->l 24 LENGTH---> 1)
```

****
2)


```
(Merge TruncatedWords ()
;; Die Länge jedes Algorithmus wird von CalculateLength bestimmt,wobei die Länge vom
;;Inkrement immer gleich sein soll
;; Samplesdauer auf dem Sampler (AKAI) ---> 7.0 Sek (C4->Ursprüngliche  Tonhöhe)
;;  Alle  Algorithmen enden an der selben Zeitpunkt

(loop for beg from 1 to 5
for n in '(Mirar río tiempo agua recordar )
for begin in '(0 0 0 0 0)
for Inkrement in '(.003 .003 .003 .003 .003)
for Inkrement2 in '(.000036 .000057 .000063 .0000157 .0000175 )
for Ton in '( c4 c4 c4 c4 c4 )
for MidiKanal in '(0 3 1 2 4 );; MidiChannel 1 2 3 4 5 !!!!
for Durchlauf in '(0 1 2 3 4);;
for Samplelength in '( 0.580 0.543 0.911 0.685 0.722 )
for totalDauer in '(66 66 66 66 66);;Die Grenze von  der maximalen Dauer wird in Sek. gesetzt.
do

(let* ((Noten Ton)
(Kanal Midikanal)
(Ink Inkrement)
(Ink2 Inkrement2)
(Durchl Durchlauf)
(Spllght Samplelength)
(TotDauer TotalDauer))
```

```
(algorithm (name n) midi-note (start begin)
(vars (i Ink)  (FreqRatio 1) (MomentDauer 0) (amp .0175) (Länge 0) (CumulateDauer 0) (Increment 0)(DL Durchl))

(setf DL dl)
(setf Länge (CalculateLength i Spllght TotDauer)) ;; Die Zahl der Elementen des Algorithmus wird berechnet.
;(setf Channel (between 0 4)) ;;MIDIChannel Wahl in Random
(setf Channel kanal)
(setf Increment i )
(setf FreqRatio (between 0.9 1.1))
(setf Note (item (steps 1 from Noten):kill länge))
(setf Note (item(pitches (steps 1 from Noten))))
(setf note (* note FreqRatio)) ;;Multiply the Pitch by Random number with a Ratio < 1 Octave
(setf Rhythm (incf MomentDauer (+ ink2 Increment)))
(setf Duration Rhythm)
(setf Amplitude (interpl (mod count Länge)   0 amp
(/ länge 1.57).7
(- Länge 1) amp))
(setf MomentDauer Duration)
(setf CumulateDauer (+ MomentDauer CumulateDauer))




(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE CumulateDauer-------->) (/ CumulateDauer 60.0) (quote min.--->)dl)))
(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE Länge-------->) Länge  (quote Length--->)dl)))

);;End of algorithm.

(defun CalculateLength (Increment SampleDauer GesamtDauer)
;;Durch diese Funktion wird die Länge jedes Alg. kalkuliert
(let* ((index 0)(Cumulate 0)(Länge 0)(MomentDauer 0));;(Länge ist wie Length dh. wieviele Elemente wird index haben)
(setf Länge (-(Truncate SampleDauer Increment)1));;DURCH DIESE DIVISION WIRD LÄNGE KALKULIERT
(loop

(setf index(+ index 1))
(setf MomentDauer (incf MomentDauer Increment))
(setf Cumulate    (incf Cumulate MomentDauer))
(cond
((Or(> index Länge) (> Cumulate GesamtDauer))(return  (- index 1)))
```

```
(t 'LängeundGesamtdauersindkleiner))
;(Print (list (/ Cumulate 60)  (quote min.bei.Länge)  länge ))
);;of Loop


);;of let*


);; end of CalculateLength



);; end of Let* (of Loop)
);;end of LOOP
) ;; End of Merge
```

Dauer des Merges

```
;(CUMULATEDAUER--------> 0.8299754999999985 MIN.---> 1)
;(|LäNGE-------->| 180 LENGTH---> 1)
;(CUMULATEDAUER--------> 0.9375167999999993 MIN.---> 0)
;(|LäNGE-------->| 192 LENGTH---> 0)
;(CUMULATEDAUER--------> 1.1029922750000012 MIN.---> 3)
;(|LäNGE-------->| 209 LENGTH---> 3)
;(CUMULATEDAUER--------> 1.1036506250000016 MIN.---> 4)
;(|LäNGE-------->| 209 LENGTH---> 4)
;(CUMULATEDAUER--------> 1.1202922499999992 MIN.---> 2)
;(|LäNGE-------->| 209 LENGTH---> 2)
```


****

3)


```
(Merge ChownFM1 ()
;;Die Samples werden am Anfang teilweise gelesen.
;; Die Länge jedes Algorithmus wird von CalculateLength bestimmt,wobei die Länge vom
;;Inkrement immer gleich sein soll(Length = SampleDauer / Increment
;; Samplesdauer auf dem Sampler (AKAI) ---> 7.0 Sek (C4->Ursprüngliche  Tonhöhe)
;; NICHT Alle  Algorithmen enden an der selben Zeitpunkt (TotalDauer:1Min:10Sek)

(loop for beg from 1 to 4
```

```
for n in '(FMIncfi0 FMIncfi1 FMIncfi2 FMIncfi3)
for begin in '(0 7 15 59) ;;Start wird von der GesamtDauer von c4 (Akai S1000)
for Inkrement in '( .175 .175 .175 .175)
for Inkrement2 in '( .0063 .0157 .175 .175)
for Ton in '(c4 c4 c4 c4)
for Durchlauf in '(0 1 2 3 );;
for Samplelength in '(7 7 7 7)
for totalDauer in '(125 118 110 74);;Die Grenze von  der maximalen Dauer wird in Sek. gesetzt.(ohne Inkrement)
do


(let* ((Noten Ton)
(Ink Inkrement)
(Ink2 Inkrement2)
(Durchl Durchlauf)
(Spllght Samplelength)
(TotDauer TotalDauer))



(algorithm (name n) midi-note (start begin)
(vars (i Ink)  (FreqRatio 1) (MomentDauer 0) (amp .275) (Länge 0) (CumulateDauer 0) (Increment 0) (DL Durchl))


(setf DL dl)
(setf Länge (CalculateLength i Spllght TotDauer)) ;; Die Zahl der Elementen des Algorithmus wird berechnet
(setf Channel (item(items 0 1 2 6 in heap)));;MidiChannels 1 2 3 7 (Samples "Tiemposample1" "Tiemposample2" "Tiemposample3" und
"ARTE Poetica                                          ;;1.verso")
(setf Increment  i )
(setf FreqRatio (between .5 2.5))
(setf Note (item (steps 1 from Noten):kill länge))
(setf Note (item(pitches (steps 1 from Noten))))
(setf NOTE (COND
((= Channel 6)(* note 1))
((* note FreqRatio))))
(setf Rhythm (incf MomentDauer (+ ink2 Increment)))
(setf Duration Rhythm)
(setf Amplitude (interpl (mod count Länge) 0 amp
(- Länge 1) .8))
(setf MomentDauer Duration)
(setf CumulateDauer (+ MomentDauer CumulateDauer))

(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE CumulateDauer-------->) (/ CumulateDauer 60.0) (quote min.--->)DL)))
(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE Länge-------->) Länge  (quote Length--->)DL)))
```

```
);;End of algorithm

(defun CalculateLength (Increment SampleDauer GesamtDauer)
;;Durch diese Funktion wird die Länge jedes Alg. kalkuliert
(let* ((index 0)(Cumulate 0)(Länge 0)(MomentDauer 0));;(Länge ist wie Length dh. wieviele Elemente wird index haben)
(setf Increment
(cond
((And(> Increment 1) (< increment 10)(/ Increment 10.0)))
((And(> Increment 10) (< increment 100)(/ Increment 100.0)))
((And(> Increment 100) (< increment 1000)(/ Increment 1000.0)))
( Increment )))
(setf Länge  (-(Truncate SampleDauer Increment)1));;DURCH DIESE DIVISION WIRD LÄNGE KALKULIERT
(loop
(setf index(+ index 1))
(setf MomentDauer (incf MomentDauer Increment))
(setf Cumulate    (incf Cumulate MomentDauer))
(cond
((Or(> index Länge) (> Cumulate GesamtDauer))(return  (- index 1))))
;(Print (list (/ Cumulate 60)  (quote min.bei.Länge)  länge ))
);;of Loop
);;of let*

);; end of CalculateLength

);; end of Let* (of Loop)
);;end of LOOP
) ;; End of Merge


*******************
```

4)

```
(Merge ChownFM2 ();; Die Länge jedes Algorithmus wird von CalculateLength bestimmt,wobei die Länge vom
;;Inkrement immer gleich sein soll(Length = SampleDauer / Increment
;; Samplesdauer auf dem Sampler (AKAI) ---> 7.0 Sek (C4->Ursprüngliche  Tonhöhe)
;; NICHT Alle  Algorithmen enden an der selben Zeitpunkt (TotalDauer:3Min:28Sec)


(loop for beg from 1 to 4
```

```
for n in '(FMIncfi5 FMIncfi6 FMIncfi7 FMIncfi8)
for begin = (random 20)
for Inkrement in '( .175 .175 .175 .175)
for Inkrement2 in '( .0063 .0157 .175 .175)
for Ton in '(c3 c3 c3 c3);;; Oktave tiefer als beim Merge ChownFM1!!!!!!
for Durchlauf in '(5 6 7 8);;
for Samplelength in '(7 7 7 7)
for totalDauer in '(125 118 110 74);;Die Grenze von  der maximalen Dauer wird in Sek. gesetzt.(ohne Inkrement)
do

(let* ((Noten Ton)
(Ink Inkrement)
(Ink2 Inkrement2)
(Durchl Durchlauf)
(Spllght Samplelength)
(TotDauer TotalDauer))


(algorithm (name n) midi-note (start begin)
(vars (i Ink)  (FreqRatio 1) (MomentDauer 0) (amp .275) (Länge 0) (CumulateDauer 0) (Increment 0) (DL Durchl))


(setf DL dl)
(setf Länge (CalculateLength i Spllght TotDauer)) ;; Die Zahl der Elementen des Algorithmus wird berechnet
(setf Channel (item(items 0 1 2 3 in heap)));;MidiChannels 1 2 3 4 (Samples "TiempoSample1" "TiempoSample2" "TiempoSample3"
;; und "TiempoSample4")
(setf Increment  i )
(setf FreqRatio (between .5 1.5))
(setf Note (item (steps 1 from Noten):kill länge))
(setf Note (item(pitches (steps 1 from Noten))))
(setf NOTE (COND
((= Channel 6)(* note 1))
((* note FreqRatio))))
(setf Rhythm (incf MomentDauer (+ ink2 Increment)))
(setf Duration Rhythm)
(setf Amplitude (interpl (mod count Länge) 0 amp
(- Länge 1) .7))
(setf MomentDauer Duration)
(setf CumulateDauer (+ MomentDauer CumulateDauer))

(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE CumulateDauer-------->) (/ CumulateDauer 60.0) (quote min.--->)DL)))
(if ( equal ( + 1 count) Länge)(PRINT (LIST (QUOTE Länge-------->)  Länge  (quote Length--->)DL)))
```

);;End of **algorithm**.

```lisp
(defun CalculateLength (Increment SampleDauer GesamtDauer)
;;Durch diese Funktion wird die Länge jedes Alg. kalkuliert
(let* ((index 0)(Cumulate 0)(Länge 0)(MomentDauer 0));;(Länge ist wie Length dh. wieviele Elemente wird index haben)
(setf Increment
(cond
((And(> Increment 1) (< increment 10)(/ Increment 10.0)))
((And(> Increment 10) (< increment 100)(/ Increment 100.0)))
((And(> Increment 100) (< increment 1000)(/ Increment 1000.0)))
( Increment )))
(setf Länge  (-(Truncate SampleDauer Increment)1));;DURCH DIESE DIVISION WIRD LÄNGE KALKULIERT
(loop
(setf index(+ index 1))
(setf MomentDauer (incf MomentDauer Increment))
(setf Cumulate    (incf Cumulate MomentDauer))
(cond
((Or(> index Länge) (> Cumulate GesamtDauer))(return  (- index 1))))
;(Print (list (/ Cumulate 60)  (quote min.bei.Länge)  länge ))
);;of Loop
);;of let*

);; end of CalculateLength
);; end of Let* (of Loop)
);;end of LOOP
);; End of Merge
```

****