Godfried-Willem Raes

"Counting Down From Minus 747"

een algoritmische elektroakoestische kompositie geschreven en gerealiseerd in opdracht van de Studio voor Experimentele Muziek te Antwerpen

Gent, 12/1997-01/1998

" Counting Down From -747 "

Godfried-Willem RAES
1997/1998

Inleiding tot en volledige source kode voor de partituur en realisatie van deze algoritmische elektroakoestische kompositie

---

1. Koncertversie (versie 1)

- een real-time electroakoestische kompositie waarvoor een Pentium type computer, een EMU procussion sampleplayer, en een meerkanaals audiosysteem met twee tot zes verschillende luidsprekerkanalen gebruikt wordt.

Solist : Godfried-Willem Raes. Andere uitvoerders zijn mogelijk: bvb. Joachim Brackx, Marc Maes, Karin De Fleyt...

De software voor deze koncertversie is vrij beschikbaar voor uitvoerders die over de vereiste hardware beschikken: een EMU ProCussion klankmodule. Alternatieve klankmodules kunnen hier absoluut niet worden gebruikt. De PC moet een MPU401 kompatibele midi-poort hebben op het standaard adres &H330. Het programma kan worden verkregen via aanvraag met e-mail. (Het zal U worden toegestuurd als een binaire attach-file genaamd **gmt747.exe**, het bestand beslaat ca.150kB)

2. Versie voor tape/CD (versie 2 : 747+30)

- Een Mix-down versie voor gewone stereo weergave beschikbaar op DAT of CD.
- Een Tascam DA88 8-kanaals digitale tape versie is eveneens op verzoek beschikbaar.

Programmanotas en kommentaartekst

Dit stuk behoort tot de uitgebreide reeks komposities van de auteur waarvoor <GMT>, (acroniem for 'Godfried's Multi Tasker' or, 'General Multitasker') werd ingezet. <GMT> fungeert als het PC-alternatief voor het Max programma op de verouderde Macintosh computer. In dit stuk treden tien autonome maar met elkaar interagerende musici op in een kamermuzikaal verband. Deze musici bestaan alleen virtueel: d.w.z. in software. Als ensemble voeren zij de partituur -eveneens in software- uit. Deze partituur -anders dan we dat in trraditionele muziek kennen- bestaat helemaal niet uit een kronologisch geordende reeks instrukties voor de spelers. Wel integendeel, wordt zij in real-time berekend en inhoudelijk afhankelijk gemaakt van de speeltechnische mogelijkheden van de spelers, en ook van de 'dirigent': de enige speler van vlees en bloed die in dit stuk een rol te vervullen krijgt. De partituur in deze konceptie moet veeleer gezien worden als een samenhangend systeem van regels inzake semantiek en syntax waarmee de relaties tussen de muzikale realisatieopdrachten aan de spelers worden georganiseerd. Een stukje van de onderliggende filosofie hebben we toegelicht in 'Multitaskers as a tool for composition', beschikbaar op onze webstek.
De software zoals we die schreven voor deze kompositie organiseert zowel de kompositie alsook de interpretatie van het muzikaal verloop, en biedt bovendien heel mogelijkheden voor real-time interventie vanwege de dirigent. Deze kan daarbij beslist niet gaan 'improviseren', maar is wel vrij alle spelers van interpretatieve instrukties te voorzien. Hij krijgt de macht over de interpretatie van de uitvoering en moet daarbij uiteraard beroep doen op zijn rechtstreekse auditieve feedback, die mede afhankelijk is van de koncertomstandigheden.
Omdat het stuk ontstond als gevolg van een opdracht ter gelegenheid van de 50ste verjaardag van Joris Delaet, kon het niet anders dan dat het een zuiver elektroakoestische kompositie zou worden. Een hommage aan Joris' vele en onverdroten inspanningen voor het doen ingang vinden van de elektroakoestische muziek in onze kontreien. Daarom wordt voor

het klankmateriaal gebruik gemaakt van zowel syntetische (additieve en FM-synteze) als 'reele' geluiden (in casu 'samples' van veelal percussieve instrumenten). De bedoeling stond voor om te vermijden in de valkuil van heel wat elektroakoestische muziek te trappen: nml. het gevaar op stereotype 'traag verlopende' muziek met in eerste plaats graduele timbrale overgangen, veel galm en suggestie van ruimtelijkheid en vooral, zonder een algemeen richtingsgevoel.Vaak immers lijkt het of heel wat van deze muziek net zo goed achterstevoren zou kunnen gespeeld worden, en alsof ze geheel en alleen gedacht werd vanuit het sonoor materiaal dat haar samenstelt. Vandaar dus onze keuze om hier nu eens gebruik te maken van een grote verzameling individueel verschillende perkussieve klanken en geluiden. Deze geluiden werden in klassen ondergebracht volgens hun meest opvallende specifieke eigenschappen. Vervolgens werden ze gevektoriseerd, wat noodzakelijk was voor hun zuiver algoritmische behandeling in een kompositie. Snelheid -meer bepaald reaktiesnelheid tussen de mogelijkheden van de 10 spelers afzonderlijk- was een van onze belangrijkste streefdoelen. Daarom ook werd geen moment overwogen hier gebruik te maken van de nochtans interessante mogelijkheden geboden door real-time syntheze op het PC platform, maar werkten we binnen de strenge beperkingen van het midi-protocol. Dit protocol werd hier echter wel tot op zijn uiterste grenzen gebruikt, vooral omdat de software alle mogelijkheden op het gebied van real-time controllers uitbuit. De midi-bytes stromen als het ware uit de komputer in een nek aan nek race.
De titel van het stuk is afgeleid van het aantal dagen dat de 50e verjaardag van Joris verwijderd is van het jaar 2000. Dit bleek 747 dagen te zijn. Nu is dit getal ontbindbaar in priemfaktoren alsvolgt: 3 * 3 * 83. Daarom werd het cijfer 3 en het kwadraat ervan (9) het vormgevend beginsel achter alle ritmische strukturen in het stuk. Een volmaakte drieledigheid en drievuldigheid dus. De duur van het stuk werd op voorhand bepaald als 747 sekonden (12'27") terwijl de grote vorm in drie delen uiteenvalt. Dat je met numerologie zowat alles kan bewijzen blijkt uit de wijze waarop we de tweede versie van dit stuk -ter gelegenheid van het 25jarig bestaan van de Antwerpse Studio voor Experimentele Muziek- realizeerden: 25 jaar komt overeen met precies 9000 dagen. Dit is 216.000 uren of 12.960.000 minuten, wat ons in sekonden 777.600.000 oplevert. Wanneer we van dit getal 777Megasekonden, de exponent laten vallen, dan bekomen we 777. Dit getal nu is precies 30 sekonden verschillend van 747. Maar, dit getal was berekend van de eerste klank tot de inzet van de laatste klank in het stuk. Nu bleek dit eigenlijk foutief, aangezien die laatste klanken nog heel wat real-time modulatie ondergaan eer zij het stuk met de finale stilte afsluiten. Dit nu duurt precies 30 sekonden! Hiermee is dan aangetoond dat er een noodzakelijk verband moest bestaan tussen de 50ste verjaardag van Joris enerzijds en het 25jarig bestaan van SEM anderzijds en dat dit feit zich alleen kon voordoen op deze precieze grens nabij het nieuwe millenium...
Vijf van de musici die we hier aan het werk horen maken exkluzief gebruik van perkussieve geluiden waaraan geen toonhoogte kan worden toegeschreven. De overige vijf spelers hebben een instrumentarium toebedeeld gekregen dat voor zowat de helft bestaat uit geluiden met een min of meer tonig karakter. Ook hier is de andere helft van de geluiden brij van toonhoogte. Omdat zodoende ongeveer 747 promil van de geluiden geen toonhoogte in de traditionele betekenisn hebben, dienden we uit te kijken naar andere syntaktische principes dan de gebruikelijke toonhoogte-georienteerde. Voor de organisatie van de minderheid aan tonige geluiden maakten we uiteraard gebruik van onze eigen biblioteek met software harmonie funkties <Harmony Library>, , maar voor het gros van het klankmateriaal was dit volkomen ongeschikt. Daarom schreven we een biblioteek met funkties en procedures waarmee intrinsieke evoluties en ontwikkelingspatronen voor ritme en metrum op syntaktische wijze konden worden behandeld. Alle spelers worden in hun partituur gekonfronteerd met doorlopend veranderende patronen en dit ondanks het feit dat elke speler in een totaal verschillend en veranderend metrum en tempo speelt. Zoiets zou volkomen ondenkbaar zijn, indien we het zouden willen gespeeld krijgen door spelers van vlees en bloed. Naar kompleksiteit gingen we in dit stuk nog heel wat verder dan de aanzetten gegeven in onze vroegere experimenten met polymetriek waarbij we de komputer gebruikten om de metronooms van elke muzikus afzonderlijk aan te sturen. (cfr. <Fall'96: voor player piano , 6 musici en Polymetronome). We hopen binnenkort onze harmonie biblioteek te kunnen uitbreiden met een nieuw hoofstuk dat ook andere experimentele komponisten wellicht nuttig kan zijn.

Bronkode:

```
' ****************************
' * MULTI-TASKER: GMT_CDF747 *
' ****************************
' 07.11.1997  start coding for "Counting Down from -747"  (codename: CDF747)
'             747 sec = 12'23"
'             747 = 3 x 3 x 83 = 3 x 249
'  derived metastructure:
'  1---------------249 250----------------498 499-----------------747
'  1-83 84-166 167-249 250-333 334-416 417-498 499-582 583-665 666-747
' factorial structure:
'  1-2-3-7-11-13-17-19-23-29-31-37-43-47-53-59-67-71-73-79-83
'  0-0-2-0-0 -0  0  0   0  0  0  0  0  0 0  0  0  0  0  1
' 08.11.1997: Ritme patroon generatie en realisatie geimplementeerd.
'             Start work on EMU procussion - mapping problem.
'             This piece will only work in combination with this
'             particular sound module!
' 09.11.1997  Pitch code added
' 10.11.1997  Ritme kode toegevoegd aan alle taken.
' 11.11.1997  To be done: add sustain modulated 'narrative'. (FB01?)
'             we still have midi channel 0, 11,12,13,14,15 free...
'             Real time controller tasks added... (cc1-cc4, press, pwm)
'             Compiled version works...
' 06.12.1997  Submix mapping added in Procussion initialisation
' 07.12.1997  Some further refinements added.
'             Runs O.K. on <Hera> - inserted in desktop. Tested with SB64 card
'             Support for 12 function keys added. The now inverse the sounds
'             CC1 in voice 9 has no effect on that patch. Code is o.k.
' 12.12.1997  do we need /Ah compilation here???
' 13.12.1997  Run on <Hera>, [Contec-docking], although too slow...
'             This version used for concert in Antwerp (premiere)
' 15.12.1997  now uses new harmony library build.
' 26.12.1997  CC2 added to task 7, CcFree function defined
'             note off's added
'             Off line DAT recording made.
' 01-12.01.1997 Opnieuw opgenomen & verbeterd - Soundscape studio bij Stichting Logos

'$DYNAMIC
'$INCLUDE: 'C:\bc7\Harmlib\main_lib.bi'     :' declaraties voor de Harmony library
'$INCLUDE: 'GMT_TYPE.BI'     :' Typed variables for task description
'$INCLUDE: 'GMT_KONS.BI'     :' shared constants
'$INCLUDE: 'GMT_VARS.BI'     :' common shared variables and arrays
'$INCLUDE: 'GMT_DEBU.BI'     :' procedures for Debug module
'$INCLUDE: 'GMT_MIDI.BI'     :' procedures for Midi-module
'$INCLUDE: 'GMT_PROC.BI'     :' procedures for GMT-main module
'$INCLUDE: 'GMT_USER.BI'     :' user code...
'$INCLUDE: 'GMT_747.BI'      :' code for this ea piece...
'$INCLUDE: 'INITSYNT.BI'     :' initialisation of synthe modules

' variable declarations:**********
REDIM SHARED FKmap(0 TO 11) AS STRING * 5
DIM SHARED MuBuf%(0 TO MidiBuffer)
DIM SHARED Task(0 TO 60) AS MTtask
DIM SHARED Har(15 TO 30) AS HarmType:  'limit to nr. of music generators
DIM SHARED regs AS RegType
REDIM SHARED MTritm!(LBOUND(Har) TO UBOUND(Har), 0 TO 0)
SCREEN 12: WIDTH 80, 60
FOR i% = LBOUND(Har) TO UBOUND(Har)
    Har(i%).vel = STRING$(128, 0)
NEXT i%
Begin$ = "... push any key to start <Counting down from -747>"

InitMuis
InitFK
InitMT
MpuUart
InitProcussion
Show1 -1
Show2 -1
Show3

LOCATE 44, 15: PRINT Begin$;
SLEEP
Begin$ = SPACE$(LEN(Begin$))
```

```
LOCATE 44, 15: PRINT Begin$;
Begin$ = ""
ot$ = TIME$
TIME$ = "00:00:00"
STACK 4096
Lps% = 6000

' ************************** start multitasker ***************************
DO
Cnt& = (Cnt& + 1) AND &H1FFFFFF
            ' GROUP 1: tasks set by individual bits d0-d14 in TaskWord0%
IF TaskWord0% AND d3 THEN
    IF Cnt& MOD Task(3).rsi = False THEN Keyhandler
END IF
IF TaskWord0% AND d4 THEN
    IF Cnt& MOD Task(4).rsi = False THEN Midi
END IF
IF TaskWord0% AND d12 THEN
    IF Cnt& MOD Task(12).rsi = False THEN Autoregulate
END IF
IF TaskWord0% AND d13 THEN
    IF Cnt& MOD Task(13).rsi = False THEN
        Muis mhor%, mver%, mbut%
        IF mbut% <> oldmbut% THEN
            IF mbut% = 1 THEN
                MuisHandler mhor%, mver%
                oldmbut% = mbut%
            ELSE
                oldmbut% = mbut%
            END IF
        END IF
    END IF
END IF
IF TaskWord0% AND d14 THEN EXIT DO

' TaskWord1% control word
IF TaskWord1% THEN
     IF TaskWord1% AND d0 THEN
        IF Cnt& MOD Task(15).rsi = False THEN Voice1
     END IF
     IF TaskWord1% AND d1 THEN
        IF Cnt& MOD Task(16).rsi = False THEN Voice2
     END IF
     IF TaskWord1% AND d2 THEN
        IF Cnt& MOD Task(17).rsi = False THEN Voice3
     END IF
     IF TaskWord1% AND d3 THEN
        IF Cnt& MOD Task(18).rsi = False THEN Voice4
     END IF
     IF TaskWord1% AND d4 THEN
        IF Cnt& MOD Task(19).rsi = False THEN Voice5
     END IF
     IF TaskWord1% AND d5 THEN
        IF Cnt& MOD Task(20).rsi = False THEN Voice6
     END IF
     IF TaskWord1% AND d6 THEN
        IF Cnt& MOD Task(21).rsi = False THEN Voice7
     END IF
     IF TaskWord1% AND d7 THEN
        IF Cnt& MOD Task(22).rsi = False THEN Voice8
     END IF
     IF TaskWord1% AND d8 THEN
        IF Cnt& MOD Task(23).rsi = False THEN Voice9
     END IF
     IF TaskWord1% AND d9 THEN
        IF Cnt& MOD Task(24).rsi = False THEN Toolkit
     END IF
     IF TaskWord1% AND d10 THEN
        IF Cnt& MOD Task(25).rsi = False THEN GWRap
     END IF
     IF TaskWord1% AND d13 THEN
        IF Cnt& MOD Task(28).rsi = False THEN Global
     END IF
     IF TaskWord1% AND d14 THEN
        IF Cnt& MOD Task(29).rsi = False THEN Ritmiek
     END IF
END IF
```

```
' code block for Taskword2:
IF TaskWord2% THEN
     IF TaskWord2% AND d0 THEN
        IF Cnt& MOD Task(30).rsi = False THEN Pan1
     END IF
     IF TaskWord2% AND d1 THEN
        IF Cnt& MOD Task(31).rsi = False THEN Pan2
     END IF
     IF TaskWord2% AND d2 THEN
        IF Cnt& MOD Task(32).rsi = False THEN Pan3
     END IF
     IF TaskWord2% AND d3 THEN
        IF Cnt& MOD Task(33).rsi = False THEN Pan4
     END IF
     IF TaskWord2% AND d4 THEN
        IF Cnt& MOD Task(34).rsi = False THEN Pan5
     END IF
     IF TaskWord2% AND d5 THEN
        IF Cnt& MOD Task(35).rsi = False THEN Pan6
     END IF
     IF TaskWord2% AND d6 THEN
        IF Cnt& MOD Task(36).rsi = False THEN Pan7
     END IF
     IF TaskWord2% AND d7 THEN
        IF Cnt& MOD Task(37).rsi = False THEN Pan8
     END IF
     IF TaskWord2% AND d8 THEN
        IF Cnt& MOD Task(38).rsi = False THEN Pan9
     END IF
     IF TaskWord2% AND d9 THEN
        IF Cnt& MOD Task(39).rsi = False THEN cc1
     END IF
     IF TaskWord2% AND d10 THEN
        IF Cnt& MOD Task(40).rsi = False THEN cc2
     END IF
     IF TaskWord2% AND d11 THEN
        IF Cnt& MOD Task(41).rsi = False THEN cc3
     END IF
     IF TaskWord2% AND d12 THEN
        IF Cnt& MOD Task(42).rsi = False THEN cc4
     END IF
     IF TaskWord2% AND d13 THEN
        IF Cnt& MOD Task(43).rsi = False THEN Press
     END IF
     IF TaskWord2% AND d14 THEN
        IF Cnt& MOD Task(44).rsi = False THEN Pwm
     END IF
END IF

' ********************** Multitasking debug code block *****************
IF TaskWord3% THEN
     IF TaskWord3% AND d0 THEN
         IF Cnt& MOD Task(45).rsi = False THEN NRTasks
     END IF
     IF TaskWord3% AND d1 THEN
         IF Cnt& MOD Task(46).rsi = False THEN ShowMIN
     END IF
     IF TaskWord3% AND d2 THEN
         IF Cnt& MOD Task(47).rsi = False THEN ShowRsi
     END IF
     IF TaskWord3% AND d3 THEN
         IF Cnt& MOD Task(48).rsi = False THEN MTSpeed
     END IF
     IF TaskWord3% AND d4 THEN
         IF Cnt& MOD Task(49).rsi = False THEN ShowTime
     END IF
     IF TaskWord3% AND d5 THEN
         IF Cnt& MOD Task(50).rsi = False THEN ShowProMil
     END IF
     IF TaskWord3% AND d11 THEN
         IF Cnt& MOD Task(56).rsi = False THEN
            COLOR 7: LOCATE 21, 40: PRINT "Stack=";
            LOCATE 21, 46: PRINT USING "######"; STACK;
            Task(56).rsi = Lps% / 3.7
         END IF
     END IF
     IF TaskWord3% AND d12 THEN
         IF Cnt& MOD Task(57).rsi = False THEN
            ' show free stackspace
            COLOR 7: LOCATE 21, 20: PRINT "Free-Stack";
            LOCATE 21, 31: PRINT USING "######"; FRE(-2);
            Task(57).rsi = Lps% / 3.9
         END IF
     END IF
     IF TaskWord3% AND d13 THEN
         IF Cnt& MOD Task(58).rsi = False THEN
            ' show free far memory FRE(-1)
            COLOR 7: LOCATE 21, 60: PRINT "Memory=";
            LOCATE 21, 68: PRINT USING "######"; FRE(-1);
            Task(58).rsi = Lps% / 4.1
         END IF
     END IF
     IF TaskWord3% AND d14 THEN
         IF Cnt& MOD Task(59).rsi = False THEN ShowMuis mhor%, mver%
     END IF
END IF
LOOP

'clean-up on exit:
Midi
'CLOSE
ResetTime ot$
CLEAR
END

SUB Autoregulate
STATIC ot#
    dt# = TIMER - ot#: ot# = TIMER
    IF dt# < .001 THEN Task(12).rsi = Task(12).rsi + (Task(12).rsi / 10): EXIT SUB
    IF dt# > .05 THEN Task(12).rsi = Task(12).rsi - (Task(12).rsi / 10): EXIT SUB
        l& = Task(12).rsi / dt#
        IF l& < d15 THEN Lps% = l& ELSE Task(12).rsi = Task(12).rsi - (Task(12).rsi /
5)
        IF l& <= 1 THEN Lps% = 1
END SUB

REM $STATIC
SUB InitFK
    ' initialize the labels for the function key mapping F1-F12
    FKmap(0) = "Thru="
    FKmap(1) = "MiIn="
    FKmap(2) = "Inv1="
    FKmap(3) = "Inv2="
    FKmap(4) = "Trig!"
    FKmap(5) = "Inv4="
    FKmap(6) = "Inv5="
    FKmap(7) = "Inv6="
    FKmap(8) = "Inv7="
    FKmap(9) = "Inv8="
    FKmap(10) = "Inv9="
    FKmap(11) = "InvA="
END SUB

REM $DYNAMIC
SUB InitMuis
    regs.ax = &H1
    CALL Interrupt(&H33, regs, regs)
END SUB

SUB Keyhandler
Task(3).rsi = Lps% / 19: ' 19Hz
DO
 k$ = INKEY$
 IF k$ = "" THEN EXIT SUB
SELECT CASE LEN(k$)
    CASE 1
    i% = ASC(k$)
    SELECT CASE i%
        CASE 27
                       COLOR 15: LOCATE 49, 27: PRINT "Sure to Quit? (Y/N)";
                       DO: k$ = INKEY$: LOOP UNTIL k$ <> ""
                       k$ = UCASE$(k$)
                       IF k$ = "Y" THEN
```

```
                TaskWord0% = TaskWord0% OR d14: ' exit!!!
                    ELSE
                LOCATE 49, 27: PRINT SPACE$(20);
                    END IF
    CASE 32: SLEEP 1:  ' spatiebalk
    CASE 33: Show2 -1: ' uitroepteken !
    CASE 63: Show1 -1: ' ?
    CASE 64: Show3:  ' @
    CASE 65 TO 94
                    ' uppercase characters A-Z switch tasks ON
        nm% = (i% - 65)
        SELECT CASE nm%
            CASE IS < 15
                TaskWord1% = TaskWord1% OR (2 ^ nm%)
                Show1 1
                dummy! = Tprop!(15 + nm%)
            CASE ELSE: ' loopt van P tot ^
                TaskWord2% = TaskWord2% OR (2 ^ (nm% - 15))
                Show1 2
                dummy! = Tprop!(30 + nm%)
        END SELECT
    CASE 97 TO 126
                    ' lowercase characters a-z switch tasks OFF
        nm% = i% - 97
        SELECT CASE nm%
            CASE IS < 15
                TaskWord1% = TaskWord1% AND (NOT (2 ^ nm%))
                Show1 1
                Task(15 + nm%).stoptime = ProMil%
            CASE ELSE
                TaskWord2% = TaskWord2% AND (NOT (2 ^ (nm% - 15)))
                Show1 2
                Task(30 + nm%).stoptime = ProMil%
        END SELECT
END SELECT
CASE 2
i% = ASC(LEFT$(k$, 1))
j% = ASC(RIGHT$(k$, 1))
SELECT CASE i%
    CASE 0
    SELECT CASE j%
        CASE 16 TO 28
                            ' Alt + QWERTYUIOP{}  (onafh.van shift)
                            ' switching tasks TaskWord0%
            TaskWord0% = TaskWord0% XOR (2 ^ (j% - 16))
            Show1 0
                        ' Function keys F1-F10:
        CASE 59 TO 68
            FKswitch% = FKswitch% XOR (2 ^ (j% - 59)): Show2 j%
            Pedals
            ' switch controlls via FKswitch%
        CASE 71
            ' Home-key
        CASE 73
                            ' PageUP: Double Tempo (Faster)
            Tempo% = Tempo% * 2: IF Tempo% > 240 THEN Tempo% = 240
            Show2 11
        CASE 75
                            ' left arrow: Slower
            Tempo% = Tempo% - 1: IF Tempo% < 30 THEN Tempo% = 30
            Show2 11
        CASE 77
                            ' right arrow: Faster
            Tempo% = Tempo% + 1: IF Tempo% > 240 THEN Tempo% = 240
            Show2 11
        CASE 81
                            ' PageDOWN: half tempo
            Tempo% = Tempo% / 2:  IF Tempo% < 30 THEN Tempo% = 30
            Show2 11
        CASE 83
            ' Delete: switches all notes off! (Midi only!)
            FOR i% = 0 TO 15: AllNotesOff i%: NEXT i%
        CASE 120 TO 128
            ' Alt + 1,2,3,...,= (bovenrij)  met of zonder shift-toets.
            ' toggles for Debug-tasks
            TaskWord3% = TaskWord3% XOR (2 ^ (j% - 120))
            Show1 3
            CASE 129
                                 ' de NUL toets schakelt alle debug tasks uit
                TaskWord3% = False:
                Show1 3
            CASE 133
                ' function key 11
                FKswitch% = FKswitch% XOR (2 ^ (10)): Show2 69
                Pedals
            CASE 134
                ' function key 12
                FKswitch% = FKswitch% XOR (2 ^ (11)): Show2 70
                Pedals
        END SELECT
    END SELECT
END SELECT
LOOP

END SUB

SUB Muis (h%, v%, b%)
' task_13
     regs.ax = &H3
     CALL Interrupt(&H33, regs, regs)
     h% = regs.cx:      ' 0->639
     v% = regs.dx:      ' 0->479
     b% = regs.bx:     ' 0= no button
                     ' 1= left button  d0
                     ' 2= right button d1
                     ' 3= both buttons
                     ' 4= middle button d2
Task(13).rsi = Lps% / 20: ' 20 x per sec.
END SUB

SUB MuisHandler (hor%, ver%)
    Khor% = (hor% \ 8) + 1
    Kver% = (ver% \ 8) + 1
SELECT CASE Kver%
    CASE 2 TO 19
        ' menuscreen
        IF Kver% - 2 <= 14 THEN
        ' kijk eerst horizontaal in welke task-groep de pointer zich bevindt:
        SELECT CASE Khor%
            CASE 3 TO 18
                ' Tasks 15-29  - application tasks
                IF Task(Kver% - 2 + 15).rsi > 0 THEN
                TaskWord1% = TaskWord1% XOR (2 ^ (Kver% - 2))
                Show1 1
                ' ON-geval:
                nm% = 15 + Kver% - 2
                IF TaskWord1% AND 2 ^ (Kver% - 2) THEN
                    dummy! = Tprop!(nm%)
                ELSE
                    ' OFF-geval
                    Task(nm%).starttime = -1: ' reset!
                    Task(nm%).stoptime = -1
                    IF nm% <> 29 THEN
                        AllNotesOff Task(nm%).kanaal
                        Har(nm%).vel = STRING$(128, 0)
                    END IF
                END IF
                 END IF

            CASE 21 TO 36
                ' Task(30-44) - groep 2
                 IF Task(Kver% - 2 + 30).rsi > 0 THEN
                TaskWord2% = TaskWord2% XOR (2 ^ (Kver% - 2))
                Show1 2
                nm% = 30 + (Kver% - 2)
                ' ON-geval:
                IF TaskWord2% AND 2 ^ (Kver% - 2) THEN
                    dummy! = Tprop!(nm%)
                ELSE
                    Task(nm%).starttime = -1: ' reset
                    Task(nm%).stoptime = -1: 'ProMil%
                         ' OFF-geval:
                    AllNotesOff Task(nm%).kanaal
                    'Har(nm%).vel = STRING$(128, 0)
                END IF
                 END IF
            CASE 41 TO 58
                ' Task(45-59) - debugging tasks
                IF Task(Kver% - 2 + 45).rsi > 0 THEN
                    TaskWord3% = TaskWord3% XOR (2 ^ (Kver% - 2))
                    Show1 3
                END IF
            CASE 61 TO 77
                ' Task(0-14) - basic MT tasks
                IF Task(Kver% - 2).rsi > 0 THEN
                    TaskWord0% = TaskWord0% XOR (2 ^ (Kver% - 2))
                    Show1 0
                END IF
        END SELECT
        ELSE
                IF Kver% = 18 THEN
            SELECT CASE Khor%
                CASE 3 TO 19
                    ' = DEL key command: Silence
                CASE 21 TO 39
                    ' =? show menus
                    Show1 -1
                CASE 41 TO 59
                    ' =! Workscreen
                    Show2 -1
                CASE 61 TO 79
                    ' =@ Debugscreen
                    Show3
            END SELECT
                END IF
        END IF
    CASE 22 TO 49
        ' first we check the sliders...
        ' get the position of the slider block:
        Slider 3, sh1%, sv1%, dummy%, sh2%, sv2%
        IF hor% >= sh1% AND hor% <= sh2% THEN
            IF ver% >= sv2% AND ver% <= sv1% THEN
                Slider 0, hor%, ver%, 1, ctrl%, retval%
                IF ctrl% > -1 THEN
                    ' use sliders as channel level controllers:
                    Uit &HB0 + ctrl%
                    Uit 7
                    Uit retval%
                    ' de 14 hier is geen bug!!!, ctrl% start van 1
                    Task(14 + ctrl%).level = retval%
                END IF
            END IF
            EXIT SUB
        END IF
        ' switches block
        SELECT CASE Khor%
            CASE 2 TO 12
                ' left block: controlls
                SELECT CASE Kver%
                    CASE 22: Tempo% = Tempo% * 2
                    CASE 23: Tempo% = Tempo% / 2
                    CASE 24: Tempo% = Tempo% + 1
                    CASE 25: Tempo% = Tempo% - 1
                END SELECT
                IF Tempo% > 240 THEN Tempo% = 240
                IF Tempo% < 30 THEN Tempo% = 30
                Show2 11
            CASE 30 TO 45
                ' center block: function keys
                SELECT CASE Kver%
                    CASE 23 TO 34: '32
                        FKswitch% = FKswitch% XOR (2 ^ (Kver% - 23))
                        Show2 Kver% + 36
                        Pedals
                END SELECT
            CASE 47 TO 79
                ' rechterblok
        END SELECT

    'CASE 34 TO 49
```

```
        ' user display also used by slider-block now
    CASE 50 TO 60
        ' debug screen
END SELECT
END SUB

REM $STATIC
SUB ResetTime (ot$)
odth& = VAL(MID$(ot$, 1, 2)) * 3600
odtm% = VAL(MID$(ot$, 4, 2)) * 60
odts% = VAL(MID$(ot$, 7, 2))
' tel de start-time stand op bij de timer op het eind van het programma:
newtime& = odth& + odtm% + odts% + TIMER:  ' in sekonden
' zet opnieuw om naar een string:
hh& = newtime& \ 3600:  newtime& = newtime& - (hh& * 3600)
mm& = newtime& \ 60:  newtime& = newtime& - (mm& * 60)
ss& = newtime& MOD 60:
hh$ = LTRIM$(STR$(hh&)): IF LEN(hh$) < 2 THEN hh$ = "0" + hh$
mm$ = LTRIM$(STR$(mm&)): IF LEN(mm$) < 2 THEN mm$ = "0" + mm$
ss$ = LTRIM$(STR$(ss&)): IF LEN(ss$) < 2 THEN ss$ = "0" + ss$
nt$ = hh$ + ":" + mm$ + ":" + ss$
TIME$ = nt$
END SUB

SUB Slider (FuncNumber%, h%, v%, but%, ctrl%, retval%)
STATIC Sliderareas() AS INTEGER, Slidervalues() AS INTEGER, Toggle%
CONST sliderlengte% = 127: ' in pixels (7-bit resolution) - this is a constant!!!
                      ' this also limits retval% to this value
CONST SliderBackColor = 10: ' green
CONST SliderBorderColor = 13: ' purple
CONST SliderHotColor = 12: ' red

IF Toggle% = 0 THEN
    Toggle% = True: ' reset by deleting sliders.
    DIM Sliderareas(0 TO 3, 0 TO 7) AS INTEGER
    DIM Slidervalues(0 TO 7) AS INTEGER: ' pro forma
END IF

SELECT CASE FuncNumber%
    CASE -1
        ' this removes all sliders from the screen
        x1% = Sliderareas(0, 1)
        y1% = Sliderareas(1, 1)
        x2% = Sliderareas(2, UBOUND(Sliderareas, 2))
        y2% = Sliderareas(3, UBOUND(Sliderareas, 2))
        LINE (x1% - 1, y1% + 1)-(x2% + 1, y2% - 1), False, BF
                ' the +1/-1's are because without them, the border is not
                ' removed properly.
        ERASE Sliderareas
        ' as now it keeps Slidervalues()
        ' we could also erase this one...
        Toggle% = False
    CASE 0
        ' here we make the sliders react on mouse movement
        ' first we check mouse horizontal position
        Sliderhit% = False
        FOR i% = 1 TO UBOUND(Sliderareas, 2)
            IF h% >= Sliderareas(0, i%) AND h% <= Sliderareas(2, i%) THEN
                ' check vertical within range condition...
                FOR j% = 1 TO UBOUND(Sliderareas, 2)
                    IF v% >= Sliderareas(3, i%) AND v% <= Sliderareas(1, i%) THEN
                        ' now the mouse must be at slider i%
                        Sliderhit% = True
                        ' for debug : LOCATE 50, 10: PRINT "Slider"; i%
                        IF but% = 1 THEN
                            ' if left button is down...
                            ' hide the mouse cursor...
                            regs.ax = 2
                            CALL Interrupt(&H33, regs, regs)
                            LINE (Sliderareas(0, i%) + 1, Sliderareas(1, i%) - 1)-
(Sliderareas(2, i%) - 1, v%), SliderHotColor, BF
                            LINE -(Sliderareas(0, i%) + 1, Sliderareas(3, i%) - 1),
SliderBackColor, BF
                            ' show cursus again:
                            regs.ax = 1
                            CALL Interrupt(&H33, regs, regs)
                            ctrl% = i%
                            retval% = Sliderareas(1, i%) - v%
                            Slidervalues(i%) = retval%
                        END IF
                        EXIT FOR
                    END IF
                NEXT j%
                IF Sliderhit% = True THEN EXIT FOR
            END IF
        NEXT i%

    CASE 1
        ' this case creates sliders
        sliderhor% = h%: ' horizontaal vertrekpunt van de reeks sliders
        sliderbottom% = v%: 'laagste punt op scherm in absolute koordinaten
        sliderafstand% = retval%
        sliderbreedte% = ctrl%:
        Slideraantal% = but%

        REDIM Sliderareas(0 TO 3, 1 TO Slideraantal%) AS INTEGER:
        REDIM Slidervalues(1 TO Slideraantal%) AS INTEGER
                ' needed to remember the positions of the
                ' sliders for later calls...
        FOR i% = 1 TO Slideraantal%
            stap% = (i% - 1) * sliderafstand%
            Sliderareas(0, i%) = sliderhor% + stap%
            Sliderareas(1, i%) = sliderbottom%
            Sliderareas(2, i%) = sliderhor% + sliderbreedte% + stap%
            Sliderareas(3, i%) = sliderbottom% - sliderlengte%
                ' here we 'misuse' Basic VIEW to draw boxes...
            VIEW (Sliderareas(0, i%), Sliderareas(1, i%))-(Sliderareas(2, i%),
Sliderareas(3, i%)), SliderBackColor, SliderBorderColor
        NEXT i%
        ' now we remember where the sliders are on  the screen
        ' the values/setting are kept in Slidervalues()

        VIEW:  ' reset to full screen view

    CASE 2
        ' redraw/rescale sliders but save slider positions!
        ' the number of sliders should of course be the same than
        ' at the time of their original creation
        ' First we delete the existing sliderbox:
        x2% = Sliderareas(2, UBOUND(Sliderareas, 2))
        y2% = Sliderareas(3, UBOUND(Sliderareas, 2))
        LINE (Sliderareas(0, 1) - 1, Sliderareas(1, 1) + 1)-(x2% + 1, y2% - 1), False,
BF
                ' the +1/-1's are because without them, the border is not
                ' removed properly.
        ' then we create the new resized one:
        sliderhor% = h%: ' new horizontal starting point for sliderblock
        sliderbottom% = v%: 'lowest point in screen in  absolute coordinates
        sliderafstand% = retval%
        sliderbreedte% = ctrl%:
        Slideraantal% = UBOUND(Sliderareas, 2)
        FOR i% = 1 TO Slideraantal%
            stap% = (i% - 1) * sliderafstand%
            Sliderareas(0, i%) = sliderhor% + stap%
            Sliderareas(1, i%) = sliderbottom%
            Sliderareas(2, i%) = sliderhor% + sliderbreedte% + stap%
            Sliderareas(3, i%) = sliderbottom% - sliderlengte%

                ' here we again 'misuse' Basic VIEW to draw boxes...
            VIEW (Sliderareas(0, i%), Sliderareas(1, i%))-(Sliderareas(2, i%),
Sliderareas(3, i%)), SliderBackColor, SliderBorderColor
            VIEW
                ' replace the previous slider positions:
            ' recalculate the vertical position:
            v% = Sliderareas(1, i%) - Slidervalues(i%)
            LINE (Sliderareas(0, i%) + 1, Sliderareas(1, i%) - 1)-(Sliderareas(2, i%) -
1, v%), 12, BF
            LINE -(Sliderareas(0, i%) + 1, Sliderareas(3, i%) - 1), 10, BF
        NEXT i%
    CASE 3
        ' returns the active area for the sliderbox
        h% = Sliderareas(0, LBOUND(Sliderareas, 2))
        v% = Sliderareas(1, LBOUND(Sliderareas, 2))
        ctrl% = Sliderareas(2, UBOUND(Sliderareas, 2))
        retval% = Sliderareas(3, UBOUND(Sliderareas, 2))
    CASE 4
        ' returns the most recent position of a slider in the box.
        ' the slider must be passed in ctrl%
        retval% = Slidervalues%(ctrl%)

END SELECT

END SUB

REM $DYNAMIC
SUB WriteSeqScore (f%)
    ' filewriter for multitaskers
STATIC Toggle%, initim!, oldtick&, old$()
IF Toggle% = False THEN
    Toggle% = True
    DIM old$(0 TO 15): FOR i% = 0 TO 15: old$(i%) = STRING$(128, 0): NEXT i%
    initim! = TIMER
END IF
tick& = (TIMER - initim!) * 100
IF tick& <= oldtick& THEN tick& = oldtick& + 1
oldtick& = tick&

FOR i% = 0 TO 15
    IF TaskWord2% AND 2 ^ i% THEN
        IF Har(i% + 30).vel <> old$(i%) THEN
            PRINT #f%, Task(i% + 30).kanaal; tick&; "H"; Har(i% + 30).vel
            old$(i%) = Har(i% + 30).vel
        END IF
    END IF
NEXT i%
' for GMT:
Task(5).rsi = Lps% / 100
END SUB
```

CDF-747 module kode:

```
' ***************************************************************************
' *                      <Counting Down from 747>                           *
' *                      dedicated to Joris De Laet                         *
' *              an algorithmic electroacoustic composition by              *
' *                         Godfried-Willem Raes                            *
' *                              11/1997-01/1998                            *
' ***************************************************************************
' this modules contains procedures only and can only be runned from
' and together with the module GMT_MAIN
'$DYNAMIC
'$INCLUDE: 'C:\bc7\harmlib\main_lib.bi'
'$INCLUDE: 'GMT_TYPE.BI'     :' Typed variables for task description
'$INCLUDE: 'GMT_KONS.BI'     :' shared constants
'$INCLUDE: 'GMT_VARS.BI'     :' common shared variables and arrays
'$INCLUDE: 'GMT_DEBU.BI'     :' procedures for Debug module
'$INCLUDE: 'GMT_MIDI.BI'     :' procedures for Midi-module
'$INCLUDE: 'GMT_PROC.BI'     :' procedures for GMT-main module
'$INCLUDE: 'GMT_USER.BI'     :' user code...
'$INCLUDE: 'GMT_747.BI'      :' procedures in this module

' declare and dimension rythm arrays:
' (declared in GMT_MAIN)
REDIM SHARED MTritm!(15 TO 23, 0 TO 0)   ' dynamic arrays for rythm controll
                                ' the indexes correspond to the
                                ' musical task-numbers in MT
                                ' The second dimension will change
                                ' throughout the programm.
                                ' For this reason we did not add
                                ' rhythm support to the type structure.

REM $STATIC
SUB cc1
    ' internal task nr.39   - real time continuous controller 1
    ' De taak die van deze controller gebruik maakt moet zelf de
    ' parameters instellen via:
    ' Task(39).kanaal
    ' Task(39).rsi = Lps% / frq%
    ' Task(39).level = value to set the controller to at start
    ' Task(39).duur  = used as endvalue.
    '                  as soon as this value is reached, the task releases itself.

IF Task(39).level <> Task(39).duur THEN
   cnt% = SGN(Task(39).duur - Task(39).level): ' negative = reduce cc
   Task(39).level = Task(39).level + cnt%
   Uit &HB0 + Task(39).kanaal
   Uit 1
   Uit Task(39).level
   TaskWord2% = TaskWord2% OR d9
ELSE
   Task(39).rsi = Task(39).InitRSI
   TaskWord2% = TaskWord2% AND (NOT d9):   ' schakelt zichzelf uit
   ' display off-status
   Show1 2
END IF
END SUB

SUB cc2
    ' internal task nr.40   - ctrl 2
IF Task(40).level <> Task(40).duur THEN
   cnt% = SGN(Task(40).duur - Task(40).level)
   Task(40).level = Task(40).level + cnt%
   Uit &HB0 + Task(40).kanaal
   Uit 2
   Uit Task(40).level
   TaskWord2% = TaskWord2% OR d10
ELSE
   Task(40).rsi = Task(40).InitRSI
   TaskWord2% = TaskWord2% AND (NOT d10)
   Show1 2
END IF
END SUB

SUB cc3
    ' internal task nr.41   - ctrl 3
IF Task(41).level <> Task(41).duur THEN
   cnt% = SGN(Task(41).duur - Task(41).level)
   Task(41).level = Task(41).level + cnt%
   Uit &HB0 + Task(41).kanaal
   Uit 3
   Uit Task(41).level
   TaskWord2% = TaskWord2% OR d11
ELSE
   Task(41).rsi = Task(41).InitRSI
   TaskWord2% = TaskWord2% AND (NOT d11)
   Show1 2
END IF
END SUB

SUB cc4
    ' internal task nr.42   - ctrl 4
IF Task(41).level <> Task(41).duur THEN
   cnt% = SGN(Task(42).duur - Task(42).level)
   Task(42).level = Task(42).level + cnt%
   Uit &HB0 + Task(42).kanaal
   Uit 4
   Uit Task(42).level
   TaskWord2% = TaskWord2% OR d12
ELSE
   Task(42).rsi = Task(42).InitRSI
   TaskWord2% = TaskWord2% AND (NOT d12)
   Show1 2
END IF
END SUB

FUNCTION CcFree% (nr%)
    ' this function returns True or False
    ' True = continuous controller nr% is not used by another task
    ' False = cc is in use at the moment
SELECT CASE nr%
    CASE 1: mask% = d9:    ' cc1
    CASE 2: mask% = d10:   ' cc2
    CASE 3: mask% = d11:   ' cc3
    CASE 4: mask% = d12:   ' cc4
    CASE 5: mask% = d13:   ' use for channel pressure controller, task 43
    CASE 6: mask% = d14:   ' use for pitchwheel controller
```

```
END SELECT
IF (TaskWord2% AND mask%) = 0 THEN CcFree% = True ELSE CcFree% = False
END FUNCTION

FUNCTION GetRitmSize% (taaknummer%)
i% = -1
DO
    i% = i% + 1
    IF MTritm!(taaknummer%, i%) = 0 THEN i% = i% - 1: EXIT DO
LOOP UNTIL (i% = UBOUND(MTritm!, 2))
                              ' wanneer in het array een nul staat
                              ' is het ritmisch patroon gedaan.
                              ' Positieve waarden = klank
                              ' Negatieve waarde = rust
                              ' Nul = einde patroon
                              ' de nul zelf mag niet worden geteld!
                              ' (-> divide 0 error in MT!)
IF i% < 0 THEN BEEP: STOP:        ' is illegal!
GetRitmSize% = i%

END FUNCTION

FUNCTION GetRitmTiks% (taaknummer%)
' wanneer in het array een nul staat is het ritmisch patroon gedaan.
' Positieve waarden = klank, Negatieve waarde = rust
' Nul = einde patroon
i% = 0
tiks% = 0
DO
    tiks% = tiks% + ABS(MTritm!(taaknummer%, i%))
    i% = i% + 1
LOOP UNTIL i% > UBOUND(MTritm!, 2)
GetRitmTiks% = tiks%
END FUNCTION

SUB Global
    ' task 28: maintains globalised data on all other music generating tasks.
Har(28).vel = STRING$(128, 0)
FOR i% = 15 TO 27
    ' we check only tasks that are active
    IF (TaskWord1% AND (2 ^ (i% - 15))) > 0 THEN
     ' check only pitched tasks
     SELECT CASE i%
       CASE 16, 18, 19, 20, 21, 23
        Har(28).vel = SumHar$(Har(28), Har(i%))
       CASE ELSE
        ' these tasks have no pitches...
     END SELECT
    END IF
NEXT i%
FillHarType Har(28)
END SUB

SUB GWRap
STATIC oldnote%
    ' new task 25.12.1997
a% = Promil%
SELECT CASE a%
    CASE IS < 150
        EXIT SUB
    CASE IS < 990
        b% = (((a% - 150) / 840!) * 9!) * RND(1)
        SELECT CASE b%
            CASE 1: noot% = 37: ' droog tikje
            CASE 2: noot% = 39: ' trr-pschhh
            CASE 3: noot% = 51: ' perc. dzjing dun
            CASE 4: noot% = 52: '
            CASE 5: noot% = 55
            CASE 6: noot% = 46
            CASE 7: noot% = 47
            CASE 8: noot% = 54:  ' elektronisch
            CASE ELSE: noot% = oldnote%
        END SELECT
    CASE IS < 1003
        oldnote% = 0
        noot% = 54
END SELECT
```

```
    IF noot% <> oldnote% THEN
        IF oldnote% THEN
            Uit 144 + Task(25).kanaal
            Uit oldnote%
            Uit 0
        END IF
        Uit 144 + Task(25).kanaal
        Uit noot%
        Uit Task(25).level
        oldnote% = noot%
    END IF
    Uit &HB0 + Task(25).kanaal:    ' autopan
    Uit 10
    Uit RND(1) * 127
Task(25).rsi = Lps% - (RND(1) * 100)
END SUB

SUB Pan1
    ' autoregulating task.
STATIC oldval%
    ' procussion panning has only 16 steps (-8 to +7)
    ' Thus we can use a sine function to modulate panning
mult% = (Tang!(30) / 4!) * Task(30).duur:  ' kwartsinus verloop
                           ' Tang!(30)/4 : loopt van 0 -> Pi/2
panval% = 18 * INT(SIN(Tang!(30) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(30).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 10Hz
    IF Task(30).rsi > Lps% / 10 THEN
        IF Lps% > d7 THEN Task(30).rsi = Task(30).rsi - (Task(30).rsi / 10)
    END IF
ELSE
    IF Task(30).rsi < 29467 THEN
        Task(30).rsi = Task(30).rsi + (Task(30).rsi / 10)
    ELSE
        Task(30).rsi = &H7FFF
    END IF
END IF

END SUB

SUB Pan2
    ' autoregulating task nr. 31
STATIC oldval%
mult% = (Tang!(31) / 4!) * Task(31).duur:  ' kwartsinus verloop
                           ' Tang!(30)/4 : loopt van 0 -> Pi/2
panval% = 18 * INT(SIN(Tang!(31) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(31).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 20Hz
    IF Task(31).rsi > Lps% / 20 THEN
        IF Lps% > d7 THEN Task(31).rsi = Task(31).rsi - (Task(31).rsi / 10)
    END IF
ELSE
    IF Task(31).rsi < 29467 THEN
        Task(31).rsi = Task(31).rsi + (Task(31).rsi / 10)
    ELSE
        Task(31).rsi = &H7FFF
    END IF
END IF
END SUB

SUB Pan3
    ' autoregulating task nr. 32
STATIC oldval%
mult% = (Tang!(32) / 4!) * Task(32).duur:  ' kwartsinus verloop
                           ' Tang!(30)/4 : loopt van 0 -> Pi/2
panval% = 18 * INT(SIN(Tang!(32) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(32).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 15Hz
    IF Task(32).rsi > Lps% / 15 THEN
        IF Lps% > d7 THEN Task(32).rsi = Task(32).rsi - (Task(32).rsi / 10)
    END IF
ELSE
    IF Task(32).rsi < 29467 THEN
        Task(32).rsi = Task(32).rsi + (Task(32).rsi / 10)
    ELSE
        Task(32).rsi = &H7FFF
    END IF
END IF

END SUB

SUB Pan4
    ' autoregulating task nr. 33
STATIC oldval%
mult% = (Tang!(33) / 4!) * Task(33).duur:  ' kwartsinus verloop
                           ' Tang!(30)/4 : loopt van 0 -> Pi/2
panval% = 18 * INT(SIN(Tang!(33) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(33).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 7Hz
    IF Task(33).rsi > Lps% / 7 THEN
        IF Lps% > d7 THEN Task(33).rsi = Task(33).rsi - (Task(33).rsi / 10)
    END IF
ELSE
    IF Task(33).rsi < 29467 THEN
        Task(33).rsi = Task(33).rsi + (Task(33).rsi / 10)
    ELSE
        Task(33).rsi = &H7FFF
    END IF
END IF

END SUB

SUB Pan5
    ' autoregulating task nr. 34
STATIC oldval%
mult% = (Tang!(34) / 4!) * Task(34).duur:  ' kwartsinus verloop
panval% = 18 * INT(SIN(Tang!(34) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(34).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 7Hz
    IF Task(34).rsi > Lps% / 7 THEN
        IF Lps% > d7 THEN Task(34).rsi = Task(34).rsi - (Task(34).rsi / 10)
    END IF
ELSE
    IF Task(34).rsi < 29467 THEN
        Task(34).rsi = Task(34).rsi + (Task(34).rsi / 10)
    ELSE
        Task(34).rsi = &H7FFF
    END IF
END IF
END SUB

SUB Pan6
    ' autoregulating task nr. 35
STATIC oldval%
mult% = (Tang!(35) / 4!) * Task(35).duur:  ' kwartsinus verloop
panval% = 18 * INT(SIN(Tang!(35) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(35).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 4Hz
    IF Task(35).rsi > Lps% / 4 THEN
        IF Lps% > d7 THEN Task(35).rsi = Task(35).rsi - (Task(35).rsi / 10)
    END IF
ELSE
    IF Task(35).rsi < 29467 THEN
        Task(35).rsi = Task(35).rsi + (Task(35).rsi / 10)
    ELSE
        Task(35).rsi = &H7FFF
    END IF
END IF

END SUB

SUB Pan7
    ' autoregulating task nr. 36
STATIC oldval%
mult% = (Tang!(36) / 4!) * Task(36).duur:  ' kwartsinus verloop
panval% = 18 * INT(SIN(Tang!(36) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(36).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 3Hz
    IF Task(36).rsi > Lps% / 3 THEN
        IF Lps% > d7 THEN Task(36).rsi = Task(36).rsi - (Task(36).rsi / 10)
    END IF
ELSE
    IF Task(36).rsi < 29467 THEN
        Task(36).rsi = Task(36).rsi + (Task(36).rsi / 10)
    ELSE
        Task(36).rsi = &H7FFF
    END IF
END IF
END SUB

SUB Pan8
    ' autoregulating task nr. 37
STATIC oldval%
mult% = (Tang!(37) / 4!) * Task(37).duur:  ' kwartsinus verloop
panval% = 18 * INT(SIN(Tang!(37) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(37).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 3Hz
    IF Task(37).rsi > Lps% / 2 THEN
        IF Lps% > d7 THEN Task(37).rsi = Task(37).rsi - (Task(37).rsi / 10)
    END IF
ELSE
    IF Task(37).rsi < 29467 THEN
        Task(37).rsi = Task(37).rsi + (Task(37).rsi / 10)
    ELSE
        Task(37).rsi = &H7FFF
    END IF
END IF
END SUB

SUB Pan9
    ' autoregulating task nr. 38 - last panning task
STATIC oldval%
mult% = (Tang!(38) / 4!) * Task(38).duur:  ' kwartsinus verloop
panval% = 18 * INT(SIN(Tang!(38) * mult%) * 7)
IF panval% <> oldval% THEN
    Uit 176 + Task(38).kanaal
    Uit 10
    Uit panval%
    oldval% = panval%
        ' panning should not go faster then 1Hz
    IF Task(38).rsi > Lps% THEN
        IF Lps% > d7 THEN Task(38).rsi = Task(38).rsi - (Task(38).rsi / 10)
    END IF
ELSE
    IF Task(38).rsi < 29467 THEN
        Task(38).rsi = Task(38).rsi + (Task(38).rsi / 10)
    ELSE
        Task(38).rsi = &H7FFF
    END IF
END IF
END SUB

SUB Pedals
STATIC OldFK%
    ' send real time switch controllers 64,65,66,67
    ' excecute Function key 3,4,5,6,7,8,9,10,11,12
CONST PedCtrl = 64
FOR i% = 2 TO 11
    ix% = 2 ^ i%
    ol% = OldFK% AND ix%
    nl% = FKswitch% AND ix%
    IF ol% <> nl% THEN
        Uit &HB0 + Task(13 + i%).kanaal
        Uit PedCtrl
        IF nl% THEN Uit 127 ELSE Uit 0
    END IF
NEXT i%
OldFK% = FKswitch%
END SUB
```

Debug Module voor <Counting Down from -747>

```
' *********************************
' * Debugging code for multitasker *
' * Module GMT_DEBU.BAS            *
' *********************************
'$INCLUDE: 'C:\bc7\Harmlib\main_lib.bi'
'$INCLUDE: 'GMT_TYPE.BI'    :' Typed variables for task description
'$INCLUDE: 'GMT_KONS.BI'
'$INCLUDE: 'GMT_VARS.BI'
'$INCLUDE: 'GMT_DEBU.BI'
'$INCLUDE: 'GMT_USER.BI'

SUB MTSpeed
       ' display loops per second of the multitasker.
       ' the lps% calculation is part of the main MT-code.(Autoregulate)
    COLOR 13
    LOCATE 54, 12: PRINT USING "#####"; Lps%;
    COLOR 7
Task(48).rsi = Lps% / 4
END SUB

SUB NRTasks
    ' print number of active tasks per second in the debug screen
nrt% = 0: Frq& = 0
FOR i% = 0 TO 14
    b% = 2 ^ i%
    IF b% AND TaskWord0% THEN nrt% = nrt% + 1: Frq& = Frq& + (Lps% \ Task(i%).rsi)
    IF b% AND TaskWord1% THEN nrt% = nrt% + 1: Frq& = Frq& + (Lps% \ Task(15 +
i%).rsi)
    IF b% AND TaskWord2% THEN nrt% = nrt% + 1: Frq& = Frq& + (Lps% \ Task(30 +
i%).rsi)
    IF b% AND TaskWord3% THEN nrt% = nrt% + 1: Frq& = Frq& + (Lps% \ Task(45 +
i%).rsi)
NEXT i%
' average frequency of tasks can now be calculated as:
FrqAvg% = Frq& / nrt%
IF Lps% > 0 THEN Task(45).rsi = Lps%
COLOR 13
LOCATE 55, 12: PRINT USING "#####"; nrt%;
COLOR 7
END SUB

SUB ShowMIN
STATIC Toggle%
IF Toggle% = False THEN
    Toggle% = True
    COLOR 5: LOCATE 58, 20: PRINT "Midi-in:"; SPACE$(48);
END IF
    ' displays information from Midi-IN in buffer
    COLOR 13: LOCATE 58, 29:
    FOR i% = 1 TO 10
         PRINT ASC(MID$(MiBuf, i%, 1));
    NEXT i%
    PRINT "/";
    COLOR 7
Task(46).rsi = Lps% / 5: ' 5Hz
END SUB

SUB ShowMuis (h%, v%)
    Khor% = (h% \ 8) + 1
    Kver% = (v% \ 8) + 1
    COLOR 13
    LOCATE 59, 3
    PRINT "H="; Khor%; " V="; Kver%; "  ";
    COLOR 7
    Task(59).rsi = Lps% / 11
END SUB

SUB ShowProMil
    COLOR 13: LOCATE 57, 12: PRINT USING "#####"; ProMil%;
    COLOR 7
Task(50).rsi = Lps% / 10
END SUB

SUB ShowRsi
' displays the actual state of the reschedule values for the different active tasks.
STATIC Toggle%, l0%
IF Toggle% = False THEN
    l0% = 34
    FOR il% = l0% TO 49
        LOCATE il%, 1: PRINT SPACE$(80);
    NEXT il%
    Toggle% = True
END IF

COLOR 5
FOR il% = 0 TO 14
    l% = l0% + il%
    IF TaskWord0% AND 2 ^ il% THEN
        m$ = "Task" + LTRIM$(STR$(il%)) + ")="
        LOCATE l%, 60: PRINT m$; : COLOR 13: PRINT Task(il%).rsi; : COLOR 5
    END IF
    IF TaskWord1% AND 2 ^ il% THEN
        m$ = "Task" + LTRIM$(STR$(15 + il%)) + ")="
        LOCATE l%, 2: PRINT m$; : COLOR 13: PRINT Task(15 + il%).rsi; : COLOR 5
    END IF
    IF TaskWord2% AND 2 ^ il% THEN
        m$ = "Task" + LTRIM$(STR$(30 + il%)) + ")="
        LOCATE l%, 20: PRINT m$; : COLOR 13: PRINT Task(30 + il%).rsi; : COLOR 5
    END IF
    IF TaskWord3% AND 2 ^ il% THEN
        m$ = "Task" + LTRIM$(STR$(45 + il%)) + ")="
        LOCATE l%, 40: PRINT m$; : COLOR 13: PRINT Task(45 + il%).rsi; : COLOR 5
    END IF
NEXT il%
COLOR 7
Task(47).rsi = Lps% / 16: ' 16Hz
END SUB

SUB ShowTime
    ot! = TIMER
    COLOR 13
    LOCATE 56, 12: PRINT USING "#####"; INT(ot!); : ' runtijd in sekonden
    COLOR 7
Task(49).rsi = Lps% + 1
END SUB
```

Midi besturings module

```
' ***************
' * Midi-module *
' ***************
'$DYNAMIC
'$INCLUDE: 'C:\bc7\Harmlib\main_lib.bi'
'$INCLUDE: 'GMT_TYPE.BI'    :' Typed variables for task description
'$INCLUDE: 'GMT_KONS.BI'
'$INCLUDE: 'GMT_VARS.BI'
```

```
'$INCLUDE: 'GMT_MIDI.BI'

REM $STATIC
SUB AllNotesOff (k%)
    Uit &HB0 + k%: Uit &H7B: Uit 0:   ' all notes off
    Uit &HB0 + k%: Uit &H79: Uit 0:   ' controllers reset
END SUB

FUNCTION B2S$ (b%)
' returns a 2-bytes hex-string for a byte integer, with leading 0 !
b% = b% AND &HFF
IF b% < 16 THEN B2S$ = "0" + HEX$(byte%) ELSE B2S$ = HEX$(byte%)
END FUNCTION

FUNCTION GetBendWheel% (k%)
SearchString$ = CHR$(224 + k%)
pointer% = INSTR(MiBuf, SearchString$):    ' startpoint for the search...
IF pointer% >= 3 THEN
    ' look for the most recent pitch-bend information.
    ' Needed since running status is likely to be in use here.
     lp% = pointer% - 1
     DO
    Msb% = ASC(MID$(MiBuf, lp%, 1))
    Lsb% = ASC(MID$(MiBuf, lp% - 1, 1))
    IF Msb% > 127 OR Lsb% > 127 THEN lp% = lp% + 2: EXIT DO
    IF lp% <= 2 THEN EXIT DO
    lp% = lp% - 2
     LOOP
    Lsb% = ASC(MID$(MiBuf, lp%, 1))
    Msb% = ASC(MID$(MiBuf, lp% - 1, 1))
    GetBendWheel% = ((Msb% * d8) + Lsb%) - d14:  ' bipolar 14 bits number
ELSE
    GetBendWheel% = False
END IF
END FUNCTION

FUNCTION GetController% (k%, c%)
SearchString$ = CHR$(c%) + CHR$(176 + k%)
pointer% = INSTR(MiBuf, SearchString$):    ' startpoint for the search...
IF pointer% >= 3 THEN
    ' look for the most recent controller information.
    ' Needed since running status is likely to be in use here.
     lp% = pointer% - 1
     DO
    Lsb% = ASC(MID$(MiBuf, lp%, 1))
    ' check controller number:
    Msb% = ASC(MID$(MiBuf, lp% + 1, 1))
    IF Msb% <> c% THEN lp% = lp% + 2: EXIT DO
    IF Lsb% > 127 THEN lp% = lp% + 2: EXIT DO
    IF lp% <= 2 THEN EXIT DO
    lp% = lp% - 2
     LOOP
    Lsb% = ASC(MID$(MiBuf, lp%, 1))
    GetController% = Lsb%
ELSE
    GetController% = False
END IF

END FUNCTION

FUNCTION GetLastNote% (k%)
SearchString$ = CHR$(144 + k%)
pointer% = INSTR(MiBuf, SearchString$):    ' startpoint for the search...
IF pointer% >= 3 THEN
    ' look for the most recent note on/off information.
    ' Needed since running status is likely to be in use here.
     lp% = pointer% - 1
     DO
    Msb% = ASC(MID$(MiBuf, lp%, 1))
    Lsb% = ASC(MID$(MiBuf, lp% - 1, 1))
    IF Msb% > 127 OR Lsb% > 127 THEN lp% = lp% + 2: EXIT DO
    IF lp% <= 2 THEN EXIT DO
    lp% = lp% - 2
     LOOP
    Lsb% = ASC(MID$(MiBuf, lp%, 1))
    Msb% = ASC(MID$(MiBuf, lp% - 1, 1))
    GetLastNote% = ((Lsb% * d8) + Msb%)
```

```
ELSE
    GetLastNote% = False
END IF

END FUNCTION

FUNCTION GetNewMiNoot% (k%)
SearchString$ = CHR$(144 + k%)
pointer% = INSTR(MiBuf, SearchString$)
IF pointer% >= 3 THEN
    Msb% = ASC(MID$(MiBuf, pointer% - 1, 1))
    Lsb% = ASC(MID$(MiBuf, pointer% - 2, 1))
    GetNewMiNoot% = (Msb% * d8) + Lsb%: ' long int, to avoid negative numbers
ELSE
    GetNewMiNoot% = False
END IF
            ' deleting this note could be done as follows:
        '     MID$(MiBuf, pointer%) = ""
        '     MID$(MiBuf, pointer% + 1) = ""
        '     MID$(MiBuf, pointer% + 2) = ""
END FUNCTION

FUNCTION GetPitchBend% (k%)
SearchString$ = CHR$(224 + k%)
pointer% = INSTR(MiBuf, SearchString$):    ' startpoint for the search...
IF pointer% >= 3 THEN
    ' look for the most recent pitch-bend information.
    ' Needed since running status is likely to be in use here.
     lp% = pointer% - 1
     DO
    Msb% = ASC(MID$(MiBuf, lp%, 1))
    Lsb% = ASC(MID$(MiBuf, lp% - 1, 1))
    IF Msb% > 127 OR Lsb% > 127 THEN lp% = lp% + 2: EXIT DO
    IF lp% <= 2 THEN EXIT DO
    lp% = lp% - 2
     LOOP
    Lsb% = ASC(MID$(MiBuf, lp%, 1))
    Msb% = ASC(MID$(MiBuf, lp% - 1, 1))
    bendvalue% = ((Msb% * d8) + Lsb%) - d14:  ' bipolar 14 bits number
        ' normalisatie - omrekening naar Cents-units:
    bv! = Bendrange * ((100! * bendvalue%) / d14)
    GetPitchBend% = INT(bv!)
ELSE
    GetPitchBend% = False
END IF
END FUNCTION

REM $DYNAMIC
SUB Midi
' midi I/O handler (bidirectional)  - Task in multitasker
STATIC wpp%
DO
    WHILE INP(Madr OR 1) < d7
         d% = INP(Madr)
         IF FKswitch% AND d1 THEN
        IF d% < 248 THEN MiBuf = CHR$(d%) + MiBuf
                ' truncation on the right side is automatic with fixed lenght strings
                ' strings are automatically left justified. They reside in Dgroup.
        IF FKswitch% AND d0 THEN Uit d%:          ' midi THRU switch
         END IF
    WEND
    IF wpp% <> wp% THEN
         WAIT Madr OR 1, d6, d6
         OUT Madr, MuBuf%(wpp%)
         wpp% = (wpp% + 1) AND MidiBuffer
    END IF
LOOP UNTIL wpp% = wp%
END SUB

REM $STATIC
SUB MpuUart
InitUART:
         IF INP(Madr OR 1) AND d7 THEN
                WAIT Madr OR 1, d6, d6
                OUT Madr OR 1, &H3F
         ELSE
                DO
                dummy% = INP(Madr)
                LOOP UNTIL INP(Madr OR 1) AND 128
                GOTO InitUART
         END IF
END SUB

REM $DYNAMIC
SUB Uit (byte%)
' wp%= write pointer in ringbuffer - must be SHARED
         MuBuf%(wp%) = byte% AND &HFF: wp% = (wp% + 1) AND MidiBuffer
END SUB
```

Gebruikers kode module

```
' *********************************
' *          <GMT_USER>          *
' *********************************
' Should be linked with the GMT multitasker as main code module.
'$DYNAMIC
'$INCLUDE: 'C:\bc7\harmlib\main_lib.bi'
'$INCLUDE: 'GMT_TYPE.BI'    :' Typed variables for task description
'$INCLUDE: 'GMT_KONS.BI'
'$INCLUDE: 'GMT_VARS.BI'
'$INCLUDE: 'GMT_MIDI.BI'     :' procedures for midi I/O
'$INCLUDE: 'GMT_DEBU.BI'     :' procedures voor Debug module
'$INCLUDE: 'GMT_PROC.BI'     :' procedures in the main MT-module
'$INCLUDE: 'GMT_USER.BI'

SUB InitMT
    ' Hier moeten eigen tasks benoemd en geinitialiseerd worden.
    ' sets initial values for the taskactivation
    TaskWord0% = d3 + d4 + d11 + d12 + d13: ' tasks 0-14
    TaskWord1% = d0 + d13 + d14:            ' tasks 15-29 - Metriek ON
    TaskWord2% = d0                         ' tasks 30-44
    TaskWord3% = d3 + d5:                   ' tasks 45-59
    ' sets initial default periodicities:
    FOR i% = 0 TO 59
        Task(i%).InitRSI = &H7FFF
        Task(i%).level = 127
    NEXT i%

    CDF747duur% = 747
    Task(0).name = "": Task(0).InitRSI = 1
    Task(1).name = ""
    Task(2).name = ""
    Task(3).name = "Keyboard": Task(3).InitRSI = 50
    Task(4).name = "Midi I/O": Task(4).InitRSI = 20
    Task(5).name = "WriteSeq": Task(5).InitRSI = 1000
    Task(6).name = "        "
    Task(7).name = "        "
    Task(8).name = "        "
    Task(9).name = "        "
    Task(10).name = "        "
    Task(11).name = "        "
    Task(12).name = "Autoreg ": Task(12).InitRSI = 100
    Task(13).name = "Mouse   ": Task(13).InitRSI = 50
    Task(14).name = "EXIT GMT"

    Task(15).name = "Voice1  ": Task(15).level = 100: Task(15).kanaal = 1:
Task(15).InitRSI = 18001: Task(15).patch = 18:  'found sound
    Task(16).name = "Voice2 T": Task(16).level = 90: Task(16).kanaal = 2:
Task(16).InitRSI = 9000: Task(16).patch = 24:   'industry
    Task(17).name = "Voice3  ": Task(17).level = 90: Task(17).kanaal = 3:
Task(17).InitRSI = 6000: Task(17).patch = 29:   'Sound FX
    Task(18).name = "Klok 4 T": Task(18).level = 90: Task(18).kanaal = 4:
Task(18).InitRSI = 30000: Task(18).patch = 34:  'Churchyard
    Task(19).name = "Thunda5T": Task(19).level = 90: Task(19).kanaal = 5:
Task(19).InitRSI = 27000: Task(19).patch = 12:  'Thundadome
    Task(20).name = "Bass 6 T": Task(20).level = 80: Task(20).kanaal = 6:
Task(20).InitRSI = 15000: Task(20).patch = 43:  'Ritual Night- tuned. Ruisig. 2 zones
    Task(21).name = "Voice7 T": Task(21).level = 90: Task(21).kanaal = 7:
Task(21).InitRSI = 18000: Task(21).patch = 52:  'Beach Party- OK pitched woody sound
    Task(22).name = "Voice8  ": Task(22).level = 120: Task(22).kanaal = 8:
Task(22).InitRSI = 21000: Task(22).patch = 41: 'Multi FX - no pitches
```

```
    Task(23).name = "Vibras T": Task(23).level = 100: Task(23).kanaal = 9:
Task(23).InitRSI = 30000: Task(23).patch = 14: 'pitched bells...
    Task(24).name = "Toolkit ": Task(24).level = 120: Task(24).kanaal = 10:
Task(24).InitRSI = 32000: Task(24).patch = 9: ' toolkit - OK
    Task(25).name = "GW-Rap  ": Task(25).level = 110: Task(25).kanaal = 11:
Task(25).InitRSI = 31000: Task(25).patch = 6: ' Rap session
    Task(26).name = "------  ": Task(26).level = 90: Task(26).kanaal = 12:
Task(26).InitRSI = 1200: Task(26).patch = 0
    Task(27).name = "------  ": Task(27).level = 90: Task(27).kanaal = 13:
Task(27).InitRSI = 1000: Task(27).patch = 0
    Task(28).name = "Global  ": Task(28).level = 0: Task(28).kanaal = 0:
Task(28).InitRSI = 13279: Task(28).patch = 0

    Task(29).name = "Ritmiek ": Task(29).level = 0: Task(29).kanaal = 0:
Task(29).InitRSI = 10000: Task(29).patch = 0

' for the panning tasks we mis-use Task().duur as a parameter for the pannning
frequencymodulator...
    Task(30).name = "Pan1    ": Task(30).kanaal = 1: Task(30).InitRSI = &H7FFF:
Task(30).duur = 747
    Task(31).name = "Pan2    ": Task(31).kanaal = 2: Task(31).InitRSI = &H7FFE:
Task(31).duur = 2241
    Task(32).name = "Pan3    ": Task(32).kanaal = 3: Task(32).InitRSI = &H7FFD:
Task(32).duur = 1494
    Task(33).name = "Pan4    ": Task(33).kanaal = 4: Task(33).InitRSI = &H7FFC:
Task(33).duur = 498
    Task(34).name = "Pan5    ": Task(34).kanaal = 5: Task(34).InitRSI = &H7FFB:
Task(34).duur = 249
    Task(35).name = "Pan6    ": Task(35).kanaal = 6: Task(35).InitRSI = &H7FFA:
Task(35).duur = 560
    Task(36).name = "Pan7    ": Task(36).kanaal = 7: Task(36).InitRSI = &H7FF9:
Task(36).duur = 373
    Task(37).name = "Pan8    ": Task(37).kanaal = 8: Task(37).InitRSI = &H7FF8:
Task(37).duur = 186
    Task(38).name = "Pan9    ": Task(38).kanaal = 9: Task(38).InitRSI = &H7FF7:
Task(38).duur = 83
    Task(39).name = "cc1   A ": Task(39).kanaal = 0: Task(39).InitRSI = &H7FF5
    Task(40).name = "cc2   B ": Task(40).kanaal = 0: Task(40).InitRSI = &H7FF4
    Task(41).name = "cc3   C ": Task(41).kanaal = 0: Task(41).InitRSI = &H7FF3
    Task(42).name = "cc4   D ": Task(42).kanaal = 0: Task(42).InitRSI = &H7FF2
    Task(43).name = "Press   ": Task(43).kanaal = 0: Task(43).InitRSI = &H7FF6
    Task(44).name = "Pwm     ": Task(44).kanaal = 0: Task(44).InitRSI = &H7FF0

    Task(45).name = "Nr-Tasks": Task(45).InitRSI = 500
    Task(46).name = "Midi-In ": Task(46).InitRSI = 50
    Task(47).name = "Show_RSI": Task(47).InitRSI = 1000
    Task(48).name = "MT-Speed": Task(48).InitRSI = 79
    Task(49).name = "ShowTime": Task(49).InitRSI = 3001
    Task(50).name = "Promil%%": Task(50).InitRSI = 783

    Task(56).name = "Stack   ": Task(56).InitRSI = 102
    Task(57).name = "D-Group ": Task(57).InitRSI = 101
    Task(58).name = "Memory  ": Task(58).InitRSI = 100
    Task(59).name = "ShowMuis":
    ' set all rsi's to their initial value:
    FOR i% = 0 TO UBOUND(Task)
        Task(i%).rsi = Task(i%).InitRSI
    NEXT i%
END SUB

FUNCTION ProMil%
STATIC Toggle%, Fk!
IF Toggle% = False THEN Toggle% = True: Fk! = 1000! / CDF747duur%
ProMil% = TIMER * Fk!: ' promille
    ' 1%% duurt (CDF747%/1000) sekonden = 0.747"
    ' dit komt overeen met (CDF747%/1000)* Lps% counts in the multitasker
END FUNCTION

SUB Show1 (par%)
    ' par% = -1: complete redraw
    ' par% = 0 to 15: redraw active task colors only.
    ' uses lines 1-20 inclusive
SELECT CASE par%
    CASE -1
        FOR il% = 1 TO 20: LOCATE il%, 1: PRINT SPACE$(80); : NEXT il%
        ' draw frame: - hoeken:
        COLOR 7
        LOCATE 1, 1: PRINT CHR$(201);
        LOCATE 1, 80: PRINT CHR$(187);
        LOCATE 20, 1: PRINT CHR$(200);
        LOCATE 20, 80: PRINT CHR$(188);
        '             - kadertje:
        LOCATE 1, 2: PRINT STRING$(78, CHR$(205));
        LOCATE 20, 2: PRINT STRING$(78, CHR$(205));
        FOR il% = 2 TO 19
            LOCATE il%, 1: PRINT CHR$(186);
            LOCATE il%, 80: PRINT CHR$(186);
        NEXT il%

        COLOR 6
        LOCATE 18, 3: PRINT "DEL = Silence°°°°";
        LOCATE 18, 20: PRINT " ? = show menus°°°°°";
        LOCATE 18, 40: PRINT " ! = show workscr.°°";
        LOCATE 18, 60: PRINT " @ = show debugscr.°";

        COLOR 14
        FOR il% = 0 TO 14
            Label$ = "<" + CHR$(65 + il%) + "/" + CHR$(97 + il%) + ">= " + (Task(il% +
15).name)
            LOCATE 2 + il%, 3: PRINT Label$;
            IF TaskWord1% AND (2 ^ il%) THEN cl% = 11 ELSE cl% = 1
            LOCATE 2 + il%, 18: COLOR cl%: PRINT CHR$(219);
            COLOR 14

            Label$ = "<" + CHR$(80 + il%) + "/" + CHR$(112 + il%) + ">= " + (Task(il% +
30).name)
            LOCATE 2 + il%, 20: PRINT Label$;
            IF TaskWord2% AND (2 ^ il%) THEN cl% = 11 ELSE cl% = 1
            LOCATE 2 + il%, 36: COLOR cl%: PRINT CHR$(219);
            COLOR 14
        NEXT il%

        COLOR 12
        LOCATE 2, 40: PRINT "<Alt1>="; Task(45).name;
        LOCATE 3, 40: PRINT "<Alt2>="; Task(46).name;
        LOCATE 4, 40: PRINT "<Alt3>="; Task(47).name;
        LOCATE 5, 40: PRINT "<Alt4>="; Task(48).name;
        LOCATE 6, 40: PRINT "<Alt5>="; Task(49).name;
        LOCATE 7, 40: PRINT "<Alt6>="; Task(50).name;
        LOCATE 8, 40: PRINT "<Alt7>="; Task(51).name;
        LOCATE 9, 40: PRINT "<Alt8>="; Task(52).name;
        LOCATE 10, 40: PRINT "<Alt9>="; Task(53).name;
        LOCATE 11, 40: PRINT "<Alt0>="; Task(54).name;
        LOCATE 12, 40: PRINT Task(55).name;
        LOCATE 13, 40: PRINT Task(56).name;
        LOCATE 14, 40: PRINT Task(57).name;
        LOCATE 15, 40: PRINT Task(58).name;
        LOCATE 16, 40: PRINT Task(59).name;
        FOR ij% = 2 TO 16
            IF TaskWord3% AND 2 ^ (ij% - 2) THEN cl% = 11 ELSE cl% = 1
            LOCATE ij%, 58: COLOR cl%:   PRINT CHR$(219);
        NEXT ij%

        COLOR 2
        LOCATE 2, 60: PRINT "<Alt+q>="; Task(0).name;
        LOCATE 3, 60: PRINT "<Alt+w>="; Task(1).name;
        LOCATE 4, 60: PRINT "<Alt+e>="; Task(2).name;
        LOCATE 5, 60: PRINT "<Alt+r>="; Task(3).name;
        LOCATE 6, 60: PRINT "<Alt+t>="; Task(4).name;
        LOCATE 7, 60: PRINT "<Alt+y>="; Task(5).name;
        LOCATE 8, 60: PRINT "<Alt+u>="; Task(6).name;
        LOCATE 9, 60: PRINT "<Alt+i>="; Task(7).name;
        LOCATE 10, 60: PRINT "<Alt+o>="; Task(8).name;
        LOCATE 11, 60: PRINT "<Alt+p>="; Task(9).name;
        LOCATE 12, 60: PRINT "<Alt+[>="; Task(10).name;
        LOCATE 13, 60: PRINT "<Alt+]>="; Task(11).name;
        LOCATE 14, 60: PRINT "<Alt+`>="; Task(12).name;
        LOCATE 15, 60: PRINT Task(13).name;
        LOCATE 16, 60: PRINT "<ESC>  ="; Task(14).name;
        FOR ij% = 2 TO 16
            IF TaskWord0% AND 2 ^ (ij% - 2) THEN cl% = 11 ELSE cl% = 1
            LOCATE ij%, 77: COLOR cl%: PRINT CHR$(219);
        NEXT ij%
```

```
    CASE 0
        FOR ij% = 2 TO 16
            IF TaskWord0% AND 2 ^ (ij% - 2) THEN cl% = 11 ELSE cl% = 1
            LOCATE ij%, 77: COLOR cl%: PRINT CHR$(219);
        NEXT ij%
    CASE 1
        FOR ij% = 2 TO 16
            IF TaskWord1% AND 2 ^ (ij% - 2) THEN cl% = 11 ELSE cl% = 1
            LOCATE ij%, 18: COLOR cl%: PRINT CHR$(219);
        NEXT ij%
    CASE 2
        FOR ij% = 2 TO 16
            IF TaskWord2% AND 2 ^ (ij% - 2) THEN cl% = 11 ELSE cl% = 1
            LOCATE ij%, 36: COLOR cl%:  PRINT CHR$(219);
        NEXT ij%
    CASE 3
        FOR ij% = 2 TO 16
            IF TaskWord3% AND 2 ^ (ij% - 2) THEN cl% = 11 ELSE cl% = 1
            LOCATE ij%, 58: COLOR cl%:   PRINT CHR$(219);
        NEXT ij%
END SELECT
COLOR 7
END SUB

SUB Show2 (par%)
    ' uses lines 21-49  (= vertical = 168 -392)
SELECT CASE par%
    CASE -1
        FOR ij% = 21 TO 49
            LOCATE ij%, 1: PRINT STRING$(80, " ");
        NEXT ij%
        ' new 10-slider block for volume controllers
        Slider 1, 10, 300, 10, 8, 16
        ' sub scripts:
        n$ = " 1 2 3 4 5 6 7 8 9 A"
        COLOR 14: LOCATE 40, 1: PRINT n$;
        LOCATE 41, 1: PRINT " Level controllers ";

        ' scoring blok:
        COLOR 3
        FOR il% = 22 TO 35
            LOCATE il%, 58: PRINT STRING$(22, "°");
        NEXT il%
        COLOR 12: LOCATE 28, 60

        ' switches blok:
        COLOR 6
        FOR il% = 22 TO 35
            LOCATE il%, 27: PRINT STRING$(25, "°");
        NEXT il%
        FOR il% = 0 TO 11
            COLOR 7: LOCATE 23 + il%, 30: PRINT "F" + HEX$(il% + 1) + "=";
            COLOR 14: PRINT SGN(FKswitch% AND (2 ^ il%));
            COLOR 6: PRINT "°"; FKmap(il%);
        NEXT il%
        COLOR 7
        ' function key display:   - switches block
    CASE 59 TO 70
        COLOR 7: LOCATE 23 + (par% - 59), 30: PRINT "F" + HEX$(par% - 58) + "=";
        COLOR 14: PRINT SGN(FKswitch% AND (2 ^ (par% - 59)));
END SELECT
COLOR 7
END SUB

SUB Show3
    ' draws screen on lines 50-60
' draw frame: - hoeken:
COLOR 7
LOCATE 50, 1: PRINT CHR$(201);
LOCATE 50, 80: PRINT CHR$(187);
LOCATE 60, 1: PRINT CHR$(200);
LOCATE 60, 80: PRINT CHR$(188);
'             - kadertje:
    LOCATE 50, 2: PRINT STRING$(78, CHR$(205));
    LOCATE 60, 2: PRINT STRING$(78, CHR$(205));
FOR il% = 51 TO 59
    LOCATE il%, 1: PRINT CHR$(186);
    LOCATE il%, 80: PRINT CHR$(186);
NEXT il%
' author
COLOR 9
LOCATE 51, 5: PRINT "<GMT> Generic RealTime Multitasker by Prof.Dr.Godfried-Willem
RAES";
COLOR 1
LOCATE 52, 22: PRINT "<Counting Down From Minus 747>";
COLOR 5: ' violet
LOCATE 54, 2: PRINT "°MT-Speed:";
LOCATE 55, 2: PRINT "°Nr-Tasks:";
LOCATE 56, 2: PRINT "°TIME    :";
LOCATE 57, 2: PRINT "°CDF747%%:";
LOCATE 54, 17: PRINT "°±²";
LOCATE 55, 17: PRINT "°±²";
LOCATE 56, 17: PRINT "°±²";
LOCATE 57, 17: PRINT "°±²";

' MM tempo monitor used for display of initial rsi's in this application)
COLOR 2
LOCATE 54, 21: PRINT "°A="; INT(Lps% * 60! / Task(15).InitRSI);
LOCATE 55, 21: PRINT "°B="; INT(Lps% * 60! / Task(16).InitRSI);
LOCATE 56, 21: PRINT "°C="; INT(Lps% * 60! / Task(17).InitRSI);
LOCATE 57, 21: PRINT "°D="; INT(Lps% * 60! / Task(18).InitRSI);
LOCATE 54, 29: PRINT "°E="; INT(Lps% * 60! / Task(19).InitRSI);
LOCATE 55, 29: PRINT "°F="; INT(Lps% * 60! / Task(20).InitRSI);
LOCATE 56, 29: PRINT "°G="; INT(Lps% * 60! / Task(21).InitRSI);
LOCATE 57, 29: PRINT "°H="; INT(Lps% * 60! / Task(22).InitRSI);
LOCATE 54, 37: PRINT "°I="; INT(Lps% * 60! / Task(23).InitRSI);
LOCATE 55, 37: PRINT "°J="; INT(Lps% * 60! / Task(24).InitRSI);
LOCATE 56, 37: PRINT "°K=";
LOCATE 57, 37: PRINT "°L=";
LOCATE 54, 45: PRINT "°M=";
LOCATE 55, 45: PRINT "°N=";
LOCATE 56, 45: PRINT "°O=";
LOCATE 57, 45: PRINT "°P=";
LOCATE 54, 53: PRINT "°°°±±²Û";
LOCATE 55, 53: PRINT "°°°±±²Û";
LOCATE 56, 53: PRINT "°°°±±²Û";
LOCATE 57, 53: PRINT "°°°±±²Û";
COLOR 7
END SUB

FUNCTION Tang! (k%)
    ' returns the angle in radians of time passed in task k%
    Tang! = Tprop!(k%) * Pi2:     ' full circle
END FUNCTION

FUNCTION Tprop! (k%)
    ' k%= tasknumber 0-59
    ' returns a normalized value (0->1) for the time of the
    ' task with the tasknumber passed.
a% = ProMil%
IF a% < False THEN Tprop = False: EXIT FUNCTION
IF a% > 1000 THEN Tprop = 1: EXIT FUNCTION
Strt% = Task(k%).starttime
IF Strt% < False THEN Strt% = a%: Task(k%).starttime = a%
IF a% < Strt% THEN Tprop = False: EXIT FUNCTION

Stp% = Task(k%).stoptime
IF Stp% <= False THEN Stp% = 1000
IF a% >= Stp% THEN Tprop = 1: EXIT FUNCTION

b% = a% - Strt%:    ' set to 0
c% = Stp% - Strt%:  ' duur
IF c% > 0 THEN Tprop = b% / c% ELSE Tprop = False
END FUNCTION
```

Module voor Initialisatie van de klankbronnen

```
' Synthesizer & Midi setup initialisation for CDF747
```

```
' machines to use: Procussion & FB01
```

```
'$DYNAMIC
```

```
'$INCLUDE: 'C:\bc7\harmlib\main_lib.bi'
```

```
'$INCLUDE: 'GMT_TYPE.BI'     :' Typed variables for task description
```

```
'$INCLUDE: 'GMT_KONS.BI'     :' shared constants
'$INCLUDE: 'GMT_VARS.BI'     :' common shared variables and arrays
'$INCLUDE: 'GMT_MIDI.BI'     :' procedures for Midi-module
'$INCLUDE: 'GMT_PROC.BI'     :' procedures for GMT-main module
'$INCLUDE: 'GMT_USER.BI'     :' user code...
'$INCLUDE: 'InitSynt.Bi'

REM $STATIC
SUB InitProcussion
' task15 = voice 1  etc...
' channel 0 is not used!  - we use channels 1-11
CONST DeviceID = 2, NoOut = 0, Mainmix = 1, Sub1 = 2, Sub2 = 3, Sub1L = 4, Sub1R = 5,
Sub2L = 6, Sub2R = 7, Layer = 8

FOR i% = 15 TO 25
    Uit &HC0 + Task(i%).kanaal
    Uit Task(i%).patch
    Uit &HB0 + Task(i%).kanaal
    Uit 7
    Uit Task(i%).level
NEXT i%
Midi
' sysex mastersetting for assignment of submixes: (6 channels)
' the 15 zones are mapped over the 3-stereo outputs:
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 32: Uit 96: Uit Mainmix: Uit 0:
Uit &HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 33: Uit 96: Uit Sub1: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 34: Uit 96: Uit Sub2: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 35: Uit 96: Uit Mainmix: Uit 0:
Uit &HF7

Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 36: Uit 96: Uit Sub1: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 37: Uit 96: Uit Sub2: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 38: Uit 96: Uit Mainmix: Uit 0:
Uit &HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 39: Uit 96: Uit Sub1: Uit 0: Uit
&HF7

Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 40: Uit 96: Uit Sub2: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 41: Uit 96: Uit Mainmix: Uit 0:
Uit &HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 42: Uit 96: Uit Sub1: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 43: Uit 96: Uit Sub2: Uit 0: Uit
&HF7

Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 44: Uit 96: Uit Mainmix: Uit 0:
Uit &HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 45: Uit 96: Uit Sub1: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 46: Uit 96: Uit Sub2: Uit 0: Uit
&HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 47: Uit 96: Uit Mainmix: Uit 0:
Uit &HF7
Uit &HF0: Uit &H18: Uit &H6: Uit DeviceID: Uit 3: Uit 48: Uit 96: Uit Sub2: Uit 0: Uit
&HF7
Midi

END SUB
```

Include files voor hiervoor gegeven kode modules

```
' Include file for GMT_747.BAS
```

```
DECLARE SUB Pan1 ()
```

```
DECLARE SUB Pan2 ()
```

```
DECLARE SUB Pan3 ()
```

```
DECLARE SUB Pan4 ()
DECLARE SUB Pan5 ()
DECLARE SUB Pan6 ()
DECLARE SUB Pan7 ()
DECLARE SUB Pan8 ()
DECLARE SUB Pan9 ()
DECLARE SUB cc1 ()
DECLARE SUB cc2 ()
DECLARE SUB cc3 ()
DECLARE SUB cc4 ()
DECLARE SUB Press ()
DECLARE SUB Pwm ()
DECLARE SUB Pedals ()

DECLARE SUB Voice1 ()
DECLARE SUB Voice2 ()
DECLARE SUB Voice3 ()
DECLARE SUB Voice4 ()
DECLARE SUB Voice5 ()
DECLARE SUB Voice6 ()
DECLARE SUB Voice7 ()
DECLARE SUB Voice8 ()
DECLARE SUB Voice9 ()
DECLARE SUB Toolkit ()
DECLARE SUB GWRap ()
DECLARE SUB Global ()
DECLARE SUB Ritmiek ()

DECLARE FUNCTION GetRitmTiks% (taaknummer%)
DECLARE FUNCTION GetRitmSize% (taaknummer%)
DECLARE FUNCTION Sinc! (hoek!)
DECLARE FUNCTION CcFree% (nr%)
```

' ************************************

```
' * GMT_DEBU: procedure declarations  *
```

' ************************************

```
DECLARE SUB MTSpeed ()
```

```
DECLARE SUB ShowMuis (h%, v%)
```

```
DECLARE SUB ShowMIN ()
```

```
DECLARE SUB ShowRsi ()
DECLARE SUB NRTasks ()
DECLARE SUB ShowProMil ()
DECLARE SUB ShowTime ()
```

' ********************************

```
' * GMT Constant Declaration File *
```

' ********************************

```
CONST MidiBuffer = &H1FFF
```

```
CONST Pi2 = 2! * 3.141593

' procedures in module MIDIPROC.BAS
' **********************************
DECLARE SUB Uit (byte%) :         ' midi output instruction for
multitasker!
DECLARE FUNCTION B2S$ (b%) :      ' byte to 2-digit string.
DECLARE SUB Midi () :             ' midi I/O task           Task 4 /
11
DECLARE FUNCTION GetBendWheel% (k%)
DECLARE FUNCTION GetController% (k%, c%)
DECLARE FUNCTION GetLastNote% (k%)
DECLARE FUNCTION GetNewMiNoot% (k%)
DECLARE FUNCTION GetPitchBend% (k%)
DECLARE SUB MpuUart ()
DECLARE SUB AllNotesOff (k%)
```

' *

```
' * declares GMT_MAIN.BAS *
```

' ************************

```
DECLARE SUB Keyhandler ()                ' task 3
```

```
DECLARE SUB Muis (h%, v%, b%)     ' get mouse info
```

```
DECLARE SUB InitMuis ()
```

```
DECLARE SUB MuisHandler (hor%, ver%) : ' task 13
DECLARE SUB ResetTime (a$)
DECLARE SUB Autoregulate ()            ' Task 12
DECLARE SUB InitFK ()
DECLARE SUB WriteSeqScore (f%)         ' task 5
DECLARE SUB Slider (funktienummer%, h%, v%, button%, ctrl%, bt%)
                                       ' called by MuisHandler
```

```
TYPE MTtask
```

```
InitRSI   AS INTEGER          ' ex InitRSI (0-3,0-14)
```

```
    rsi         AS INTEGER          ' ex Task0(0-14)
```

```
starttime AS INTEGER
```

```
stoptime  AS INTEGER
```

```
     name      AS STRING * 8       ' ex Taskname0,3...
     level     AS INTEGER
     kanaal    AS INTEGER
     chord     AS INTEGER          ' could become H() AS Harmtype...
     patch     AS INTEGER
     duur      AS INTEGER
END TYPE

' ****************************************
' * GMT_USER: procedure declaration file *
' ****************************************
' compositions:

' utils:
DECLARE SUB InitMT ()
DECLARE SUB Show1 (k%)
DECLARE SUB Show2 (k%)
DECLARE SUB Show3 ()
DECLARE FUNCTION ProMil% ()
DECLARE FUNCTION Tprop! (k%)
DECLARE FUNCTION Tang! (k%)
```

' **************************

```
' * COMMON SHARED VARIABLES *
```

```
'  ***************************
```

```
' for MT.BAS, GMT_747.BAS, MTDEBUG.BAS
```

```
COMMON SHARED wp%, rp%          ' read/write pointer for midi
COMMON SHARED TaskWord0%, TaskWord1%, TaskWord2%, TaskWord3%
                                ' TaskWord3%=controll-word for
debugger=tasks
                                ' TaskWord0%=controll-word for data-
I/O tasks
                                ' TaskWord1%=controll-word for
application tasks
COMMON SHARED MuBuf%()          ' buffer for outgoing midi-info
COMMON SHARED MiBuf AS STRING * MidiBuffer: ' string format!
COMMON SHARED Task() AS MTtask  ' user type task descriptor variable
COMMON SHARED Har() AS HarmType
COMMON SHARED Lps%              ' Multitasking speed in Hz for
autoregulation

' timing:
COMMON SHARED CDF747duur%, Tfakt!

' shared variables for <Counting down to 747> application:
COMMON SHARED Toets!()          ' needed for harmony tasks,
ornamentation
COMMON SHARED GlobTon%          ' global Tc% pointer for <Counting
down from 747>
COMMON SHARED FKswitch%         ' 16 bits for switches
COMMON SHARED FKmap() AS STRING * 5
COMMON SHARED Tempo%            ' Global tempo in MM units.
```

```
' include file procedures initialisation synthesizers
```

```
DECLARE SUB InitProcussion ()
```

Includes voor de Harmony-Library van Godfried-Willem Raes

```
'$INCLUDE: 'C:\bc7\harmlib\KONSTANT.BI'
```

```
'$INCLUDE: 'C:\bc7\harmlib\COM_TYPE.BI'
'$INCLUDE: 'C:\bc7\harmlib\HARM_GEN.BI'
'$INCLUDE: 'C:\bc7\harmlib\HARM_ANA.BI'
'$INCLUDE: 'C:\bc7\harmlib\HARM_FUZ.BI'
'$INCLUDE: 'C:\bc7\harmlib\HARM_PSY.BI'
'$INCLUDE: 'C:\bc7\harmlib\HARM_VIZ.BI'
'$INCLUDE: 'C:\bc7\harmlib\HARM_FIL.BI'
'$INCLUDE: 'C:\bc7\harmlib\HARM_AKU.BI'
DECLARE SUB Interrupt (intnum AS INTEGER, inreg AS RegType, outreg AS
RegType)
DECLARE SUB InterruptX (intnum AS INTEGER, inreg AS RegTypeX, outreg
AS RegTypeX)
DECLARE SUB Absolute (address AS INTEGER)
DECLARE SUB Int86Old (intnum AS INTEGER, inarray() AS INTEGER,
outarray() AS INTEGER)
DECLARE SUB Int86XOld (intnum AS INTEGER, inarray() AS INTEGER,
outarray() AS INTEGER)
```

```
' Type declarations for libraries HARM_PSY en HARM_AKU
```

```
' ****************************************************
TYPE HarmType
   Vel AS STRING * 128        ' complete harmony descriptor
   C(0 TO 11) AS SINGLE       ' fuzzy shepard chord descriptor
   Dis AS SINGLE              ' fuzzy dissonance of the harmony
   Con AS SINGLE              ' fuzzy consonance of the harmony
   Iprop(0 TO 6) AS SINGLE    ' interval property strenghts
END TYPE
TYPE RegType
   ax    AS INTEGER
   bx    AS INTEGER
   cx    AS INTEGER
   dx    AS INTEGER
   bp    AS INTEGER
   si    AS INTEGER
   di    AS INTEGER
   flags AS INTEGER
END TYPE
TYPE RegTypeX
   ax    AS INTEGER
   bx    AS INTEGER
   cx    AS INTEGER
   dx    AS INTEGER
   bp    AS INTEGER
   si    AS INTEGER
   di    AS INTEGER
   flags AS INTEGER
   ds    AS INTEGER
   es    AS INTEGER
END TYPE
```

```
' declares for Harm_Aku

DECLARE FUNCTION DifTone! (n1%, n2%)
DECLARE FUNCTION F2N% (f!)
DECLARE FUNCTION GetAkuCons! (Har AS HarmType)
DECLARE FUNCTION GetAkuDis! (Har AS HarmType)
DECLARE FUNCTION GetDipAkuDis! (n1%, BYVAL v1%, n2%, BYVAL v2%)
DECLARE FUNCTION GetDipoleDis! (BYVAL f1%, BYVAL v1%, BYVAL f2%,
BYVAL v2%)
DECLARE SUB GetSpecDefault (Spec!(), NrHarmonics%)
DECLARE SUB GetSpecData (Spec!())
DECLARE FUNCTION N2F% (noot%)
DECLARE FUNCTION NormVol2Midi% (v!)
DECLARE FUNCTION Midi2NormVol! (v%)
DECLARE SUB InvDFT (Spectrum!(), Samp!())
DECLARE SUB DFT (Samp!(), Spectrum!())
DECLARE SUB Har2Samp (Har AS HarmType, Samp!())
DECLARE SUB Samp2Har (Samp!(), Har AS HarmType)
DECLARE SUB LinSpec2Har (Sp!(), Har AS HarmType)
DECLARE SUB Har2LinSpec (Har AS HarmType, Sp!())
DECLARE SUB Har2Shape (Har AS HarmType, Shape!())
```

```
' declaration of functions in Harm_ana.bas
```

```
DECLARE FUNCTION IsChordClassic% (ChordType&)
```

```
DECLARE FUNCTION Name3Chord$ (ChordNumber%)
```

```
DECLARE FUNCTION NameChord$ (ChordNumber%)
```

```
DECLARE FUNCTION Tonality$ (n%, Tc%)
DECLARE FUNCTION ReadHarFile& (Har AS HarmType, track%, filenr%)
```

```
DECLARE SUB WriteHar2File (Har AS HarmType, track%, filenr%)
```

```
' include file for subs and functions in the fuzzy-harmony module
```

```
DECLARE FUNCTION Cons% (crd%)
```

```
DECLARE FUNCTION Dishar! (b1%, b2%)
DECLARE FUNCTION Dishar3! (b1%, b2%, b3%)
DECLARE FUNCTION Dismel! (b1%, b2%)
DECLARE FUNCTION Flue! (b1%, b2%, b3%)
DECLARE FUNCTION FuzFrameVar! (Hm%(), size%)
DECLARE SUB GetFuzzyData (FuzInt!())
DECLARE SUB GetFuzzyMelo (FuzMel!())
DECLARE FUNCTION HarmFrameQual! (Hm%(), size%)
DECLARE FUNCTION HarmQual! (Hm%())
DECLARE FUNCTION HarmQualWeight! (Hm%())
DECLARE FUNCTION MelFrameQual! (Ml%(), size%)
DECLARE FUNCTION MeloQual! (Ml%())
DECLARE FUNCTION MelQualWeight! (Ml%())
DECLARE FUNCTION SecSolveQual! (b1%, b2%)
DECLARE FUNCTION TritSolveQual! (b1%, b2%)
```

```
' Include for HARM_GEN
```

' *******************

```
DECLARE FUNCTION AddChords% (ChordNumber1%, ChordNumber2%)
```

```
DECLARE FUNCTION AddNoteInChord% (ChordNumber%, a%)
DECLARE FUNCTION Cnr2Ctp& (BYVAL c%, BYVAL Tc%)
DECLARE FUNCTION ComChords% (ChordNumber1%, ChordNumber2%)
DECLARE FUNCTION Ctp2Cnr% (ChordType&, n%)
DECLARE FUNCTION DelNoteInChord% (ChordNumber%, a%)
DECLARE FUNCTION DifChords% (ChordNumber1%, ChordNumber2%)
DECLARE FUNCTION Get3Kadens! (cn1%, cn2%, cn3%, cn4%)
DECLARE FUNCTION GetChordType& (ChordNumber%, BYVAL Tc%)
DECLARE FUNCTION GetConsonance! (ChordNumber%)
DECLARE FUNCTION GetDissonance! (ChordNumber%)
DECLARE FUNCTION GetNrNotes% (ChordNumber%)
DECLARE FUNCTION GetRndNote% (modus%, Tc%)
DECLARE FUNCTION GetScaleCnr% (modus%, Tc%)
DECLARE FUNCTION GetTc% (ChordNumber%)
DECLARE FUNCTION IsNoteInChord% (ChordNumber%, note%)
DECLARE FUNCTION MakeHarmChord% (ChordNumber%, modus%, harmsystem%)
DECLARE FUNCTION MakeChordNum% (a%, b%, c%, d%, e%, f%, g%, h%, i%,
j%, k%, l%)
DECLARE FUNCTION Make3ChordNum% (a%, b%, c%)
DECLARE FUNCTION Make4ChordNum% (a%, b%, c%, d%)
DECLARE FUNCTION MirCnr% (ChordNumber%, note%)
DECLARE FUNCTION Neg% (ChordNumber%)
DECLARE FUNCTION NxNt% (modus%, Tc%, note%, sg%)
DECLARE FUNCTION ParConChord% (t%, c!, tol!)
DECLARE FUNCTION ParaChord% (t%, d!, c!, tol!)
DECLARE FUNCTION ParDisChord% (t%, d!, tol!)
DECLARE FUNCTION Richt% (b1%, b2%)
DECLARE FUNCTION Rol% (ChordNumber%, n%)
DECLARE FUNCTION Ror% (ChordNumber%, n%)
DECLARE FUNCTION SetTc% (ChordNumber%, Tc%)
DECLARE FUNCTION TransChordNum% (ChordNumber%, n%)
DECLARE FUNCTION TransChordType& (ChordType&, n%)
DECLARE FUNCTION VarChord% (ChordNumber%, modus%)
```

```
' declare 'HARM_PSY'
```

```
DECLARE FUNCTION AbsDifHar$ (H1 AS HarmType, H2 AS HarmType)
```

```
DECLARE SUB AddCnr2Har (H AS HarmType, Cnr%, lo%, hi%, v%)
```

```
DECLARE SUB AddNote2Har (H AS HarmType, n%, v%)
```

```
DECLARE SUB AddShNo2Har (H AS HarmType, n%, v%)
DECLARE FUNCTION CommonHar$ (H1 AS HarmType, H2 AS HarmType)
DECLARE FUNCTION ConvergeHar$ (H AS HarmType, refnote%, faktor!)
DECLARE SUB DelNote2Har (H AS HarmType, n%)
DECLARE SUB DelShNo2Har (H AS HarmType, n%)
DECLARE FUNCTION DifHar$ (H1 AS HarmType, H2 AS HarmType)
DECLARE FUNCTION DimIntInHar$ (H AS HarmType, tc%, mode%, degree%,
sg%)
DECLARE FUNCTION DiminuteHar$ (H AS HarmType, tc%, mode%, steps%)
DECLARE SUB FillHarType (H AS HarmType)
DECLARE FUNCTION Fit2Mode$ (H AS HarmType, mode%, tc%)
DECLARE FUNCTION GetConPsy (H AS HarmType)
DECLARE FUNCTION GetDisPsy (H AS HarmType)
DECLARE SUB GetIntProp (H AS HarmType)
DECLARE FUNCTION GetNrInt% (H AS HarmType, interval%, norm!)
DECLARE SUB GetPsiChord (H AS HarmType)
DECLARE FUNCTION GetScaleHar$ (mode%, tc%, vel%)
DECLARE FUNCTION GetShepVal! (note%)
DECLARE FUNCTION GetStrongest% (H AS HarmType, n%)
DECLARE FUNCTION Har2Cnr% (H AS HarmType, norm!)
DECLARE FUNCTION MirHar$ (H AS HarmType, n%)
DECLARE FUNCTION MorfHar$ (H1 AS HarmType, H2 AS HarmType, modus%,
tc%, db%)
DECLARE SUB P2Har (H AS HarmType, i&, p%())
DECLARE FUNCTION SolveHar$ (H AS HarmType, n%, norm!)
DECLARE FUNCTION SolveMaj2$ (H AS HarmType, n%, norm!)
DECLARE FUNCTION SolveMin2$ (H AS HarmType, n%, norm!)
DECLARE FUNCTION SolveTrit$ (H AS HarmType, n%, norm!)
DECLARE SUB SubstNtInHar (H AS HarmType, n1%, n2%)
DECLARE FUNCTION SumHar$ (H1 AS HarmType, H2 AS HarmType)
DECLARE FUNCTION SumVelo% (BYVAL v1%, BYVAL v2%)
DECLARE FUNCTION SymDimHar$ (H AS HarmType, tc%, sg%)
```

```
' Include for visualisation module.
```

```
DECLARE SUB ShowHarmFuz (b%)
```

```
DECLARE SUB ShowMelFuz (b%)
```

```
DECLARE SUB ShowPsiChord (h AS HarmType, hor%, ver%, hs%, vs%)
```

```
DECLARE SUB ShowHar (h AS HarmType, hor%, ver%, s!)
```

```
DECLARE SUB ShowCrd (crd%, v%, h%)
DECLARE SUB ShowStaff (crd%, v%, h%)
DECLARE SUB ShowLargeStaff (crd%, v%, h%)
DECLARE SUB DrawGclef (v%, h%)
DECLARE SUB DrawFclef (v%, h%)
```

```
' Constant Declaration
```

```
CONST False = 0, True = NOT False
```

```
' bit constants for boolelogic
CONST d0 = 1, d1 = 2, d2 = 4, d3 = 8, d4 = 16, d5 = 32, d6 = 64, d7 =
128, d8 = 256
CONST d9 = 512, d10 = 1024, d11 = 2048, d12 = 4096, d13 = 8192, d14 =
16384, d15 = 32768
CONST d16 = 65536, d17 = 131072, d18 = 262144, d19 = 524288, d20 =
1048576, d21 = 2097152
CONST d22 = 4194304, d23 = 8388608, d24 = 16777216, d25 = 33554432,
d26 = 67108864
CONST d27 = &H8000000, d28 = &H10000000, d29 = &H20000000, d30 =
&H40000000
' modi
CONST md0 = 2741, md1 = 2477, md2 = 1453, md3 = 2733
CONST md4 = 2731, md5 = 2795, md6 = 2507, md7 = 2475
CONST md8 = 1365, md9 = 585, md10 = 677, md11 = &HFFF
' midi-support module
CONST Madr% = &H330
' aku-module - tuning base
CONST La = 440!
CONST GrondDo! = 8.175799: ' = La * (2^(3/12))/64
CONST Domq! = 7.94305: ' quartertone lower for semitone bands
                       ' = La * (2^(5/24))/64
CONST Pi = 3.141592654#: ' needed for the Fourier transforms
CONST Pi2 = 6.283185307#
```

De bronkode voor de harmonie biblioteek evenals de gekompileerde versies, zowel voor Microsoft Basic PDS BC7.1 als voor Power Basic Compiler versie 3.5, is beschikbaar via internet. Midi-file versies van deze kompositie worden in geen geval beschikbaar gesteld, gezien de eigenheid van de hier beoogde klanken en klankeffekten. (Http://www.ping.be/logos/harmlib/HarmonyLibrary.html)

Godfried-Willem Raes
Januari, 1998