Turbo Pascal deel 4

MSX CLUB MAGAZINE 37

Erik van 'Bilsen

Scanned, ocr'ed and converted to PDF by HansO, 2001

Muziek en geluid met Turbo Pascal. Dat zijn de ingrediënten voor deze aflevering. De Play-procedure vormt de leidraad voor een verdere verkenning in deze programmeertaal.

SOUNDS.LIB

De procedures voor het produceren van geluid zijn te vinden in het bestand SOUNDS.LIB. Deze procedures zijn tevens opgenomen aan het eind van deze aflevering. Tik ze in met de editor van Turbo Pascal en sla ze op onder de naam SOUNDS.LIB. Het bestand MSXBIOS.LIB heeft u voor het gebruik van deze procedures niet nodig. Neem dus de volgende regel op in uw programma:

```
{$I Sounds.lib}
```

De mogelijkheden

Alvorens ik de procedures ga behandelen geef ik eerst een kort overzicht van de mogelijkheden van de procedures. Deze mogelijkheden zijn uitgebreider dan het PLAY-commando in BASIC! Er wordt gebruik gemaakt van drie procedures: de procedures Sound en Play (bekend uit BASIC) en de procedure InitPlay. Voordat je begint met het maken van muziek, moet altijd eerst de procedure InitPlay worden aangeroepen. Deze procedure initialiseert enkele variabelen. Vervolgens kunnen met behulp van de procedure Play drie strings worden doorgegeven, bijvoorbeeld:

Play('ceg','egb-', '');

Er moeten altijd drie strings doorgegeven worden. Elke string is net zoals in BASIC een macro, die ver- schillende commando's kan bevatten. Deze commando's zijn:
A t/m G: De noten. Een noot kan gevolgd worden door een #, +, -en/of een cijfer (voor de lengte) en/of een punt. Er is maar één punt toegestaan.
O, L, R, V, S, M en T: Deze commando's hebben dezelfde betekenis als in BASIC. Zie hiervoor de BASIC-handleiding. Het tempo hoeft maar één keer worden ingesteld (in BASIC 3 keer, een keer voor elk kanaal).
De commando's N en X worden niet ondersteund. Ter compensatie kunnen een aantal extra commando's worden gebruikt

> **en** **<**: Deze zijn ook bij de FM-Pac bekend en worden gebruikt om een octaaf omhoog of omlaag te gaan.

Yn,m: Dit commando is eveneens afkomstig van de FM-Pac. Het wordt gebruikt om rechtstreeks een PSG-regjster te schrijven. Daarbij geeft n het registernummer en m de data aan. Dit commando kan bijvoorbeeld worden gebruikt om aan te geven of een kanaal geluid en/of ruis moet bevatten (met behulp van register 7):

Play('Y7,240c' ,","");

Na dit commando wordt door kanaal l zowel ruis als geluid geproduceerd.

Zn,m: Dit is een geheel nieuw commando waarmee een eenvoudige vibrato kan worden geproduceerd. De n geeft het bereik van de vibrato aan (1-4095). Dit is de waarde waarmee de frequentie rond de standaard toonhoogte schommelt. De m geeft hier de stapgrootte aan. Bij een positieve stapgrootte gaat de toon in eerste instantie omlaag, bij een negatieve stapgrootte omhoog. Met een stapgrootte van nul wordt de vibrato uitgezet. Zorg er wel voor dat het bereik een veelvoud van de stapgrootte is.
Alle commando's kunnen zowel met hoofd- als met kleine letters worden ingetypt. Desgewenst kunnen spaties worden toegevoegd. Een voorbeeld van een eenvoudige compositie is T-Tune bovenaan deze pagina.. Wellicht kunt u later zelf de Play-procedure nog uitbreiden met commando's voor het creëren van bijvoorbeeld glissando's en fade-outs.

Bespreking procedures

In het navolgende zullen de procedures die aan het eind van de aflevering staan worden besproken.
Eerst zullen de gebruikte variabelen aan bod komen.

De variabele Time is een systeem-variabele die zich bevindt de op adressen $FCA2 en $FCA3 in het geheugen. Deze twee geheugen-plaatsen (een integer) worden automatisch door de computer 50 keer per seconde opgehoogd (op een Europese machine). De variabele Htime is een hulpvariabele. FreqTab is een array van 12 inte-gers groot. Deze tabel wordt geladen met de nootwaarden van de 12 grondnoten (c,c#,d,d# enz.) van octaaf 1. Deze waarden zijn een afgeleide van de frequentie. De waarde van een noot van een hoger liggend octaaf is gelijk aan de waarde van dezelfde noot van een lager octaaf gedeeld door 2. In het vervolg zal ik voor het gemak steeds de term fre-quentie gebruiken als ik het over de noorwaarde heb. De variabelen NoteLength, Octave en Volume zijn elk arrays van 3 bytes groot, l byte voor elk kanaal. De variabele vibrato is voor elk kanaal als volgt opgebouwd: Het eerste element (Vibrato[x,0]) bevat het bereik, het tweede element de beginfrequen-tie, het derde element de eindfre-quentie en de laatste twee elementen de stapgrootte. De overige variabelen komen later aan bod.
De basis voor het produceren van geluid is natuurlijke de Sound-pro-cedure (gelijk aan de SOUND-in-structie in BASIC). Deze zou er met behulp van de MsxBios routine $93 als hieronder uit kunnen zien:

```
PROCEDURE SOUND (register, data : integer) ;

BEGIN

  RegA := register ;
  RegDE := dta ;
  MSXBIos($93)

END
```

Ik heb echter voor een oplossing gekozen die geen gebruikt maakt van het BIOS maar van de PSG-poorten $AO en $A1. De procedure is in machinetaal geschreven met behulp van de standaard InLine procedure. Deze wordt in een van de volgende afleveringen behandeld.

Met behulp van de procedure Init-Play worden de bovengenoemde variabelen geïnitialiseerd. FreqTab wordt geladen met de standaard nootwaarden. De standaardlengte van een noot wordt op 4 gezet (een kwartnoot). Het octaaf wordt standaard op 4 gezet en het volume op 8. De vibrato-functie wordt uitgezet. Tot slot wordt het tempo op 120 gezet en de hulpvariabele Htime gelijk gemaakt aan de huidige waarde van Time. Met behulp van de Sound procedure wordt register 7 zo gezet, dat alle kanalen alleen geluid produceren (geen ruis).

Sub-procedures

Het hoofdgedeelte vormt de Play-procedure. Zoals je ziet bestaat deze procedure nog uit 3 sub-pro-cedures en -functies, te weten Inte-gerValue, RealLength en Music. Deze drie sub-procedures kunnen alleen in de hoofdprocedure Flay worden aangeroepen. Binnen sub-procedures kunnen ook weer sub-sub-procedures worden aangemaakt. Op die manier kunnen bij elkaar horende procedures worden gegroepeerd. De sub-procedures worden gedeclareerd voor de hoofdtekst van de procedure.

Laten we beginnen met het hoofdgedeelte van de procedure Play. Er worden drie parameters doorgegeven, te weten pO, pi en p2, elk een string van maximaal 254 karakters groot Deze strings worden overgezet in de array Macro. Vervolgens wordt voor elk kanaal (voice) de teller (Counter) en de Index op l gezet. De teller geeft het aantal tellen aan dat de huidige noot nog voortduurt (in stappen van 1/50 seconde). De variabele Index geeft de huidige positie in de string aan.
Voor het omrekenen van het tempo en de nootlengte naar het aantal tellen dat een noot duurt wordt de variabele Factor gebruikt. Play-Music tenslotte is een boolean, die aangeeft of het kanaal in gebruik is. Als alle drie array-elementen de waarde FALSE bevatten geeft dat aan dat de Play-procedure is afgelopen. De eerste REPEAT...UNTIL-lus herhaalt een aantal stappen totdat deze drie elementen inderdaad de waarde FALSE bevatten.
Met de tweede REPEAT...UNUL-lus wordt gewacht totdat de hulpvariabele Home niet meer gelijk is aan de systeemvariabele Time, met andere woorden de lus wacht totdat

1/50 seconde verstreken is. Vervolgens wordt voor de drie kanalen de teller verlaagd. Als deze de waarde O bereikt heeft, dan wordt de procedure Music aangeroepen, die de tekststrings aftast. In het andere geval wordt gekeken of de vi-brato-functie actief is. Zo ja, dan wordt een vibrato uitgevoerd. Daartoe wordt de huidige frequentie van het betreffende kanaal (FreqfVoiceNr]) vermeerderd met de stapgrootte en naar de juiste registers geschreven (registers O en l voor kanaal l, registers 2 en 3 voor kanaal 2, en registers 3 en 4 voor kanaal 3). Hierbij valt het gebruik van de procedures Hi en Lo op.

Hi en Lo

Hi en Lo zijn twee standaardprocedures van Turbo Pascal en zijn een afkorting van High en Low. Ze worden gebruikt om een integer in een hoog (8 hoogste bits) een laag (8 laagste bits) deel te splitsen. Een integer is immers een combinatie van twee bytes (van 0-255) met de waarde laag+256*hoog. Register O van de PSG (kanaal 1) verwacht het lage gedeelte en register l het hoge gedeelte van de frequentie.
De procedure Play bevat de sub-functies IntegerValue en Real-Length. De functie IntegerValue wordt gebruikt om de waarde van een getal in de string Text, vanaf positie Index+1 te achterhalen. De variabele Index is hier een VAR-parameter, hetgeen betekent dat het adres van de variabele aan de functie IntegerValue wordt doorgegeven. Veranderingen aan de parameter Index hebben dus tot gevolg dat de variabele waarmee de functie wordt aangeroepen zelf ook verandert (zie deel l van de cursus).
De functie kan bijvoorbeeld als volgt worden aangeroepen:

```
Positie:=1;
Getal:=IntegerValue(‘al23b456c' .Positie) ;
```

Vervolgens zal de variabele Getal de waarde 123 bevatten en de variabele Positie de nieuwe waarde 5 (de positie van het eerstvolgende niet-cijfer). Als u het gebruik van VAR-parameters goed heeft begrepen, dan moet u weten waarom de volgende aanroep tot een foutmelding leidt

```
Getal:=IntegerValue ('al23b456c',1);
```

De verdere behandeling van deze functie laat ik over aan zelfstudie.

De functie RealLength wordt gebruikt om het aantal tellen (van 1/50 seconde) te berekenen dat een noot duurt aan de hand van het tempo en de lengte van de noot. De formule daarvoor is:

Tellen = 12000 / (Lengte * Tempo)

Hieruit valt af te leiden dat een tempo van 93,125,187 of 250 het beste resultaat oplevert (het aantal tellen is dan een geheel getal). De factor wordt zo aangepast dat bij een aantal tellen van bijvoorbeeld 7.5 afwisselend 7 en 8 tellen worden gebruikt.

De sub-procedure Music

De sub-procedure Music tast de string Macro af naar het volgende commando. Het nummer (0-2) van het kanaal wordt doorgegeven aan de variabele Voice. Als de huidige positie in de string groter is dan de lengte van die string, dan wordt de procedure afgebroken. Daartoe wordt de variabele Play-Music op FALSE gezet en het volume van het betreffende kanaal op nul. Met Exit wordt de procedure (niet netjes) verlaten. In sommige gevallen is het aan te bevelen om een procedure voortijdig met Exit te verlaten. De regel is echter dat elk willekeurig programma kan worden geschreven zonder gebruik te maken van Exit. In dit geval had de Exit kunnen worden vermeden door aan het eind van het IF-statement met ELSE de rest van de procedure te vervolgen.
De variabele Ch van het type CHAR wordt geladen met het huidige karakter in de string. Het karakter wordt daarbij omgezet naar een hoofdletter (UpCase). De variabele C bevat vervolgens de ASOI-waarde van het karakter. Als het karakter een van de letters van A t/m G is (ASCn-waarde tussen 64 en 72), dan wordt de betreffende noot gespeeld.

Achtereenvolgens wordt:
- De noot omgezet in een nummer (NoteNr);
- Gekeken of de noot gevolgd wordt door een #, + of -
- Gekeken of de noot gevolgd wordt door een getal (leng-te);
- De teller (Counter) geladen met het aantal tellen dat de noot duurt;
- Gekeken of de noot wordt gevolgd door een punt, zo ja, dan wordt het aantal tellen met 1.5 vermenigvuldigd (=*3/2);
- Afhankelijk van het octaaf de frequentie gedeeld;
- De juiste registers van de PSG geschreven;
- De variabele Vibrato bijgewerkt;
- De index aangepast.

Als het karakter géén noot is, dan wordt de variabele Number geladen met het getal in de string dat achter het commando staat. Vervolgens worden aan de hand van een CASE-statement de mogelijke commando's afgelopen.

Enkele opmerkingen:
- Bij het R-commando (rust) wordt de teller geladen met het aantal tellen dat de rust duurt, en wordt het volume van het betreffende kanaal
op o gezet. Het gebruik van de punt is bij het R-commando niet toegestaan. De procedure Music wordt voortijdig verlaten;
- Om het S-commando te kunnen gebruik moet het volume op 16 worden gezet. Een volume van 16 geeft aan dat de S-code in gebruik is;
- Bij de commando's Z en Y worden twee getallen gevraagd, gescheiden door een komma. Dit scheidingsteken wordt echter niet gecontroleerd.

Het gebruik van deze commando's leidt niet tot het produceren van geluid, er wordt alleen een variabele of register veranderd. Vandaar dat binnen de procedure Music nog een keer de procedure Music wordt aangeroepen, net zolang totdat een van de letters van A t/m G wordt aangetroffen. Dit verschijnsel van een procedure die zichzelf aanroept wordt recursief programmeren genoemd. Op de (onmogelijkheden van recursief programmeren kom ik wellicht in een van de volgende afleveringen terug.

Hiermee wordt deze muzikale aflevering besloten. Zoals je wellicht gemerkt zult hebben ligt het niveau wat hoger dan gebruikelijk voor cursussen voor beginners. Het leereffect wordt echter grotendeels bepaald door hetgeen je zelf probeert en experimenteert (op die manier heb ik zelf ook Turbo Pascal onder de knie gekregen). Deze cursus vormt daarvoor een handvat en steunpilaar. Veel oefenen is dus het credo, en daarbij wens ik u veel succes!

```
PROGRAM T_Tune;

{*********************************************************************
 *                                                                   *
 *                    Aanzet tot Triple Soft Tune                    *
 *                       Door  Erik van Bilsen                       *
 *                                                                   *
 *********************************************************************}

CONST
   a0 = 'aar16aaar16a ggr16gggr16g ffr16fffr16f ffr16fffr16g';
   a1 = 'aa>a<aaa>a<a gg>g<ggg>g<g ff>f<fff>f<f ff>f<fff>g<g';
   b0 = 'c<bagagagfefgfgab>';
   c0 = 's0z2000,20,8o5l16ffffe-e-e-e-cccc<aaaa>';
   c1 = 'v13z800,4o7l1c';
   c2 = 'z800,-4o4c';

{$I sounds.lib}

BEGIN
   InitPlay;
   Play('t125s0m5000o2l16','v14z3,1o5l8','');
   Play(a0,'','');  Play(a0,'','');
   Play(a1,'','');  Play(a1,'','r1'+c0);
   Play(a1,b0,'');  Play(a1,b0,'');
   Play(a1,b0,'');  Play(a1,b0,'r1'+c0);
   Play(a1,b0,c1+c2);  Play(a1,b0,c1+c2);
   Play(a1,b0,c1+c2);  Play(a1,b0,c1+c0);
   Play('aa>a<r16v15a>a<r16v14a>a<r16v13a>a<r16v12a>a<r16','','');
   Play('v11a>a<r16v10a>a<r16v9a>a<r16v8a>a<r16v8a>a<r16','','');
   Play('v7a>a<r16v6a>a<r16v5a>a<r16v4a>a<r16v3a>a<r16','','');
   Play('v2a>a<r16v1a>a<','','')
END.
```

SOUNDS.LIB

```
TYPE
   Str254 = STRING[254];

VAR
   Time: INTEGER ABSOLUTE $FCA2;
   Htime: INTEGER;
   FreqTab: ARRAY [0..11] OF INTEGER;
   NoteLength,Octave,Volume: ARRAY [0..2] OF BYTE;
   Counter,Freq,Factor: ARRAY [0..2] OF INTEGER;
   Vibrato: ARRAY [0..2,0..4] OF INTEGER;
   Tempo,S_code: BYTE;

PROCEDURE Sound (Register,Data: INTEGER);
BEGIN
   InLine($0E/$A0/$3A/Register/$ED/$79/
          $0C/$3A/Data/$ED/$79/$FB)
END;

PROCEDURE InitPlay;
VAR Teller: BYTE;
BEGIN
   FreqTab[ 0]:=$D5D; FreqTab[ 1]:=$C9C;
   FreqTab[ 2]:=$BE7; FreqTab[ 3]:=$B3C;
   FreqTab[ 4]:=$A9B; FreqTab[ 5]:=$A02;
   FreqTab[ 6]:=$973; FreqTab[ 7]:=$8EB;
   FreqTab[ 8]:=$86B; FreqTab[ 9]:=$7F2;
   FreqTab[10]:=$780; FreqTab[11]:=$714;
   FOR Teller:=0 TO 2 DO
      BEGIN
         NoteLength[Teller]:=4; Octave[Teller]:=4;
         Volume[Teller]:=8; Vibrato[Teller,3]:=0;
         Vibrato[Teller,4]:=0
      END;
   Tempo:=120; Htime:=Time; Sound(7,248)
END;

PROCEDURE Play (p0,p1,p2: Str254);
VAR Macro: ARRAY [0..2] OF Str254;
    Index: ARRAY [0..2] OF BYTE;
    PlayMusic: ARRAY [0..2] OF BOOLEAN;
    VoiceNr: BYTE;
    Frequency: INTEGER;

FUNCTION IntegerValue (Text: Str254;
                       VAR Index: BYTE):INTEGER;
VAR SubString: STRING[5];
    Ascii,Dummy,Negative: INTEGER;
BEGIN
   SubString:='0'; Negative:=1; Index:=Index+1;
   IF Text[Index]='-' THEN
      BEGIN
         Negative:=-1; Index:=Index+1
      END;
   Ascii:=Ord(Text[Index]);
```

```
   WHILE (Ascii<58) and (Ascii>47) DO
      BEGIN
         SubString:=SubString+Chr(Ascii);
         Index:=Index+1; Ascii:=Ord(Text[Index])
      END;
   Val (SubString,Ascii,Dummy);
   IntegerValue:=Ascii*Negative
END;

FUNCTION RealLength (Nlength,Nr: BYTE):INTEGER;
VAR Temp: INTEGER;
BEGIN
   Temp:=Nlength*Tempo;
   RealLength:=Factor[Nr] div Temp;
   Factor[Nr]:=12000+(Factor[Nr] mod Temp)
END;

PROCEDURE Music (Macro: Str254; Voice: BYTE);
VAR Posit,C,NoteNr: BYTE;
    Number,F: INTEGER;
   Ch: CHAR;
BEGIN
   Posit:=Index[Voice];
   IF Posit>Length(Macro) THEN
      BEGIN
         PlayMusic[Voice]:=FALSE; Sound(8+Voice,0);
         Exit
      END;
   Ch:=UpCase(Macro[Posit]); C:=Ord(Ch);
   IF (C>64) and (C<72) THEN
      BEGIN
         IF C<67 THEN NoteNr:=C-60 ELSE NoteNr:=C-67;
         NoteNr:=NoteNr shl 1;
         IF NoteNr>5 THEN NoteNr:=NoteNr-1;
         CASE Macro[Posit+1] OF
            '+','#': BEGIN NoteNr:=NoteNr+1;
                           Posit:=Posit+1 END;
            '-'    : BEGIN NoteNr:=NoteNr-1;
                           Posit:=Posit+1 END;
            END;
         Number:=IntegerValue(Macro,Posit);
         IF Number=0 THEN Number:=NoteLength[Voice];
         Counter[Voice]:=RealLength(Number,Voice);
         IF Macro[Posit]='.' THEN
            BEGIN
               Counter[Voice]:=(Counter[Voice]*3)shr 1;
               Posit:=Posit+1
            END;
         F:=FreqTab[NoteNr];
         IF Octave[Voice]<>1 THEN
            F:=F shr (Octave[Voice]-1);
         Freq[Voice]:=F;
         Sound(Voice shl 1,Lo(F));
         Sound(Voice shl 1+1,Hi(F));
         IF Volume[Voice]=16 THEN Sound(13,S_code);
         Sound(Voice+8,Volume[Voice]);
         IF Vibrato[Voice,4]<>0 THEN
```

```
            BEGIN
               Vibrato[Voice,1]:=F-Vibrato[Voice,0]
                                 shr 1;
               Vibrato[Voice,2]:=Vibrato[Voice,1]
                                +Vibrato[Voice,0];
               Vibrato[Voice,3]:=Vibrato[Voice,4]
            END;
         Index[Voice]:=Posit
      END ELSE
      BEGIN
         Number:=IntegerValue(Macro,Posit);
         CASE Ch OF
            'O': Octave[Voice]:=Number;
            'V': Volume[Voice]:=Number;
            'L': NoteLength[Voice]:=Number;
            '>': Octave[Voice]:=Octave[Voice]+1;
            '<': Octave[Voice]:=Octave[Voice]-1;
            'R': BEGIN
                    Counter[Voice]:=
                       RealLength(Number,Voice);
                    Index[Voice]:=Posit;
                    Sound(8+Voice,0); Exit
                 END;
            'S': BEGIN
                    S_code:=Number; Volume[Voice]:=16
                 END;
            'M': BEGIN
                    Sound(11,Lo(Number));
                    Sound(12,Hi(Number))
                 END;
            'T': Tempo:=Number;
            'Z': BEGIN
                    Vibrato[Voice,0]:=Number;
                    Vibrato[Voice,4]:=
                       IntegerValue(Macro,Posit);
                    Vibrato[Voice,3]:=Vibrato[Voice,4]
                 END;
            'Y': Sound(Number,IntegerValue(Macro,Posit))
            END;
         Index[Voice]:=Posit;
         Music(Macro,Voice)
      END;
END;

BEGIN
   Macro[0]:=p0+' '; Macro[1]:=p1+' '; Macro[2]:=p2+' ';
   FOR VoiceNr:=0 TO 2 DO
      BEGIN
         Counter[VoiceNr]:=1; Index[VoiceNr]:=1;
         Factor[VoiceNr]:=12000;
         PlayMusic[VoiceNr]:=TRUE
      END;
   REPEAT
      REPEAT UNTIL Time<>Htime;
      Htime:=Time;
      FOR VoiceNr:=0 TO 2 DO IF PlayMusic[VoiceNr] THEN
         BEGIN
```

```
            Counter[VoiceNr]:=Counter[VoiceNr]-1;
            IF Counter[VoiceNr]=0 THEN
               Music(Macro[VoiceNr],VoiceNr) ELSE
            IF Vibrato[VoiceNr,3]<>0 THEN
               BEGIN
                  Frequency:=Freq[VoiceNr]
                            +Vibrato[VoiceNr,3];
                  Freq[VoiceNr]:=Frequency;
                  IF (Frequency=Vibrato[VoiceNr,1]) or
                     (Frequency=Vibrato[VoiceNr,2]) THEN
                     Vibrato[VoiceNr,3]:=
                     -Vibrato[VoiceNr,3];
                  Sound(VoiceNr shl 1,Lo(Frequency));
                  Sound(VoiceNr shl 1+1,Hi(Frequency))
               END;
         END;
   UNTIL not(PlayMusic[0] or PlayMusic[1]
          or PlayMusic[2])
END;
```