Toch de bios

MCM

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

In onze bespreking van Turbo Pascal in MCM nummer 19 hebben we kritiek geleverd op de afwezigheid van grafische functies in deze compiler. Dat hebben we geweten; sindsdien zijn er verschillende brieven en diskettes binnengekomen, waarin ons werd gedemonstreerd dat de BIOS van MSX1 en MSX2 wel degelijk te gebruiken is vanuit Turbo Pascal.

Er komt wat ML aan te pas, maar als de 'laag-niveau' functies en procedures maar eenmaal gemaakt zijn, is het gebruik ervan een fluitje van een cent.

De BIOS bereiken

Het probleem met alle programmeertalen die onder MSXDOS werken, is dat de BIOS voor die talen niet zomaar toegankelijk is. Voor MSXDOS wordt de volledige 64K geheugen vrijgegeven, waardoor de BIOS tijdelijk het veld moet ruimen; zijn plaats wordt ingenomen door MSXDOS en gewoon RAM-geheugen.
Nu is het wel zo, dat MSX daarop voorbereid is: in principe is het mogelijk vanuit elke geheugenconfiguratie de BIOS te 'bereiken'. Helaas is daar gewoonlijk ML voor no-dig. Gelukkig is Turbo Pascal uitgerust met de functie IN-LINE, waardoor gewoon Pascal en Z80-machinetaal eenvoudig door elkaar gebruikt kunnen worden.
Alle inzenders hebben dat dan ook gedaan; elk op hun eigen manier. We zijn zo vrij geweest een aantal van die manieren te combineren en aan te passen. Daarvoor hebben we een vorm gekozen, die we hierbij tot standaard proberen te verheffen.

Bibliotheken

Een andere krachtige mogelijkheid van Turbo Pascal is het 'includen' van andere Pas-cal-programma's. Daarmee kan een stuk programma 'ingelast' worden in een hoofdprogramma, op het moment dat dat gecompileerd wordt. De opdracht tot het includen wordt gegeven ia een commentaar-regel van de volgende vorm:

{$ITEST.LIB}

Hiermee wordt dus het bestand TEST.LIB meegecom-pileerd. Het aantal include-bestanden is in principe ongelimiteerd. Het ligt nu voor de hand, om veelgebruikte functies en procedures op te slaan in 'bibliotheek'-bestanden, die dan met een include-op-dracht meegecompileerd worden. Het voordeel is, dat het hoofdprogramma overzichtelijk blijft. De programmeur hoeft eigenlijk niet eens te weten hoe die bibliotheek nu precies in elkaar zit: als hij maar weet, hoe de functies en procedures uit de bibliotheek gebruikt moeten worden.

Een standaard

Zo'n bibliotheek heeft, net als een normaal programma, maximaal vier delen. Dat zijn achtereenvolgens een GONST-, TYPE- en VAR-gedeelte en een stuk met procedures en functies. Een normaal programma heeft daarnaast nog het 'hoofdprogramma'. Door de strikte opbouw van een Pascal-programma moet elk deel van een bibliotheek ui een apart include-bestand zitten. Vaak zijn de delen wel te combineren in één enkel bestand, maar dat levert problemen op als er meer bibliotheken tegelijk gebruikt gaan worden.
Daarom stellen we voor, voor de deelbestanden de extensies CON, TYP, VAR en LIB te nemen. Voor een grafische bibliotheek met de naam GRAPH zouden we dus — hoogstens — de bestanden GRAPH.CON, GRAPH.TYP, GRAPH-.VAR en GRAPH.LIB krijgen. Natuurlijk is het mogelijk dat een bibliotheek geen VARs of CONSTs bevat; in dat geval ontbreekt het desbetreffende bestand gewoon.
Het schema van het eigenlijke programma zou er dan uitzien als in listing 1. Dit lijkt een beetje omslachtig, maar op deze manier kunnen er meer bibliotheken tegelijk gebruikt worden, zonder dat er iets aan de bibliotheek-bestanden veranderd hoeft te worden. En dat veranderen is nu precies niet de bedoeling!

```
{     MSXBIOS.VAR - een include-file met de reg-variabelen voor
      de MSXBIOS-bibliotheek }

   regA, regBC, regDE, regHL, regF, regIX, regIY: integer;
```

Listing 2a: MSXBIOS. VAR

```
{     MSXBIOS.LIB - een include-file met de procedures en
      functies voor de MSXBIOS-bibliotheek }

{ aanroepen van de MSX-BIOS vanuit MSXDOS }
procedure msxbios(entry: integer);
begin
   inline(
      $F5/$C5/$D5/$E5/$DD/$E5/$FD/$E5/   { PUSH alle registers }
      $3A/regA/                          { LD A,(regA) }
      $ED/$4B/regBC/                     { LD BC,(regBC) }
      $ED/$5B/regDE/                     { LD DE,(regDE) }
      $2A/regHL/                         { DL HL,(regHL) }
      $DD/$2A/entry/                     { LD IX,(entry) }
{*}   $FD/$2A/$C0/$FC/                   { LD IY,EXPTBL }
      $CD/$1C/$00/                       { CALL CALSLT }
      $32/regA/                          { LD (regA),A }
      $ED/$43/regBC/                     { LD (regBC),BC }
      $ED/$53/regDE/                     { LD (regDE),DE }
      $22/regHL/                         { LD (regHL),HL }
      $DD/$22/regIX/                     { LD (regIX),IX }
      $FD/$22/regIY/                     { LD (regIY),IY }
      $F5/                               { PUSH AF }
      $E1/                               { POP HL }
      $22/regF/                          { LD (regF),HL }
      $AF/                               { XOR A }
      $32/regA+1/                        { LD (regA+1),A }
      $32/regF+1/                        { LD (regF+1),A }
      $FD/$E1/$DD/$E1/$E1/$D1/$C1/$F1/   { POP alle registers }
      $FB)                               { EI: enable interrupts }
end;

{ aanroepen van de MSX2 SUB-ROM: alleen MSX2! }
procedure msx2bios(entry: integer);
begin
   inline(
      $F5/$C5/$D5/$E5/$DD/$E5/$FD/$E5/   { PUSH alle registers }
      $3A/regA/                          { LD A,(regA) }
      $ED/$4B/regBC/                     { LD BC,(regBC) }
      $ED/$5B/regDE/                     { LD DE,(regDE) }
      $2A/regHL/                         { DL HL,(regHL) }
      $DD/$2A/entry/                     { LD IX,(entry) }
{*}   $FD/$2A/$F7/$FA/                   { LD IY,EXBRSA }
      $CD/$1C/$00/                       { CALL CALSLT }
      $32/regA/                          { LD (regA),A }
      $ED/$43/regBC/                     { LD (regBC),BC }
      $ED/$53/regDE/                     { LD (regDE),DE }
      $22/regHL/                         { LD (regHL),HL }
      $DD/$22/regIX/                     { LD (regIX),IX }
      $FD/$22/regIY/                     { LD (regIY),IY }
      $F5/                               { PUSH AF }
      $E1/                               { POP HL }
      $22/regF/                          { LD (regF),HL }
```

```
      $AF/                                  { XOR A }
      $32/regA+1/                           { LD (regA+1),A }
      $32/regF+1/                           { LD (regF+1),A }
      $FD/$E1/$DD/$E1/$E1/$D1/$C1/$F1/      { POP alle registers }
      $FB)                                  { EI: enable interrupts }
end;
```

Listing 2b: GMSXBIOS.LIB

Gebruik van de BIOS

De eerste bibliotheek is afkomstig van een Belgische lezer: Jean Delestienne. Hij maakte de functies msxbios en msx2bios. We hebben die — op een paar aanpassingen na — overgenomen. Zie listings 2A en 2B.
Listing 2A geeft de inhoud van het bestand MSXBI-OS.VAR. Hierin worden zeven variabelen gedeclareerd,
die de registers van de Z80 symboliseren. Deze set noemen we de reg-variabelen, omdat de namen ervan allemaal met 'reg' beginnen.

Listing 2B bevat het bestand GMSXBIOS.LIB en bevat twee functies.
Msxbios maakt gebruik van een stuk INLINE-machine-taal, dat alle registers eerst bewaart op de stack, ze dan vult met de informatie uit de reg-variabelen, daarna een BlOS-routine aanroept en de reg-variabelen hun nieuwe waarde geeft.
Welke BlOS-routine er wordt aangeroepen wordt bepaald door de variabele 'en-try". De procedure msx2bios is hieraan gelijk, behalve in de regel die gemerkt is met {*}.

```
PROGRAM test;

CONST

{$I GRAPH.CON} { de constanten van de bibliotheek }
{$I TWEE.CON}    { en van nog een bibliotheek } ...
{ eigen constanten }

TYPE
 {$1 GRAPH.TYP}   { bibliotheek-types }
{ eigen types }

VAR
  {SI GRAPH.VAR}  { bibliotheek-variabelen }
  {$1 GRAPH.LIB}  { procedures en functies }

procedure eigen;  { eigen procedures en functies }
begin
end;

begin { hoofdprogramma }
end.
```

Listing 1: een include - voorbeeld

```
{     MSXBIOS.VAR - een include-file met de reg-variabelen voor de
MSXBIOS-bibliotheek }
regA, regBC, regDE, regHL, regF, reglX, reglY: integer;
```

Msxbios roept de 'normale' BIOS aan, msx2bios daarentegen de 'sub-rom': het stuk van de BIOS dat alleen bij MSX2 bestaat. Het gebruik van de beide procedures is eenvoudig: geef eerst de reg-variabelen hun waarde, roep dan msx(2)bios aan met als argument het adres van de gewenste BIOS-routine en lees — indien nodig — daarna de informatie uit de reg-variabelen. Een paar voorbeelden.

```
msxbios($3E);
```

initialiseert de functietoetsen. Hiervoor zijn geen reg-variabelen nodig.

```
msxbios($9F); ch := chr(regA);
```

haalt een karakter van het toetsenbord en slaat het op in de variabele ch van type char.

```
regA: = ord(ch);
msxbios($A5);
```

stuurt het karakter ch naar de printer.

De nodige informatie over de inhouden van de registers voor en na aanroep van de BIOS staat onder andere in de BlOS-tabellen in MCM 16 en 18.

```
{     GRAPH.VAR - include-file met variabelen-declaraties voor de
      GRAPH-bibliotheek }

   bakclr:   byte absolute $F3EA;      { BAcKground CoLoR }
   forclr:   byte absolute $F3E9;      { FOReground CoLoR }
   bdrclr:   byte absolute $F3EB;      { BorDeR CoLoR }
   atrbyt:   byte absolute $F3F2;      { ATtRibute BYTe }

   cloc:     integer absolute $F92A;   { Cursor LOCation }
   cmask:    byte absolute $F92C;      { Cursor MASK }
   logopr:   byte absolute $FB02;      { LOGical OPeRation }

   dppage:   byte absolute $FAF5;      { DisPlay PAGE }
   acpage:   byte absolute $FAF6;      { ACtual PAGE }

   gxpos:    integer absolute $FCB3;   { Graphic X-POSition }
   gypos:    integer absolute $FCB5;   { Graphic Y-POSition }
   grpacx:   integer absolute $FCB7;   { GRaPhic ACcumulator X }
   grpacy:   integer absolute $FCB9;   { GRaPhic ACcumulator Y }
```

Listing3a: GRAPH.VAR

```
{     GRAPH.TYP - typedefinities voor de GRAPH-bibliotheek }

   kleur      = 0..15;   { kleuren van 0 tot en met 15 }
   paletkleur = 0..7;    { rood/groen/blauw componenten }
```

Listing3b: GRAPH.TYP

Grafiek

Leuker zijn natuurlijk de grafische routines van MSX1 en MSX2. Daarvoor danken we AJ.A. van Rossum uit Boxmeer. Hij is de maker van de GRAPH-bibliotheek.
Ook die hebben we opgesplitst, dit keer in drie bestanden: GRAPH.VAR, GRAPH.TYP en GRAPH.-LIB. Verder hebben we de definities aangevuld en aangepast aan GMSXBIOS.LIB. De beide bibliotheken werken nu samen: GRAPH gebruikt de msxbios-procedure.

GRAPH.VAR - listing 3A

bevat een reeks variabeledeclaraties voor plaatsen in het systeem-RAM. In deze definities wordt het Turbo Pascal-woord 'absolute' gebruikt. De variabele *bdrclr', bijvoorbeeld, bevindt zich nu echt op adres $F3EB, zodat we in plaats van te PEEKen en POKEn gewoon
a: = bdrclr; respectievelijk bdrclr: = 4; kunnen gebruiken.

GRAPH.TYP - listing 3B

beslaat de gebruikte TYPEs in de bibliotheek. Deze type-definities zijn niet strikt nodig maar passen wel heel mooi in de 'geest' van Pascal; bovendien voorkomen ze verkeerde aanroepen van somm-mige procedures en functies. GRAPH.LIB, tenslotte, bevat procedures en functies om een aantal grafische mogelijkheden te benutten. Zie listing 3A. Uit plaatsgebrek kunnen we de MSX1 en MSX2-functies niet tegelijk plaatsen. Daarom in dit nummer alleen de BlOS-functies voor MSX1, die natuurlijk ook op een MSX2 te gebruiken zijn. De lijst is zeker niet volledig: we hebben de belangrijkste BlOS-functies er uitgelicht. Bovendien hebben we de namen gekozen zoals die in de MSX-specificatie — bijvoorbeeld het MSX Technical Data B ook - staan. Dat levert hier en daar cryptische namen op, zoals CHGMOD. Deze procedure verandert de scherm-mode, zoals SCREEN dat doet vanuit Basic. Een meer bekende en makkelijke naam was dus 'screen' geweest, maar daar hebben dus niet voor geko/.en om consequent te kunnen blijven
Overigens zijn bepaalde MSX2-mogelijkheden wel te gebruiken met de 'oude' BlOS-routines: CHGMOD, bijvoorbeeld, laat ook de scherm-types 3 tot en met 8 toe. Uiteraard gaat dat niet goed op een MSXl-computer.
We bespreken niet elke routine apart; ook dat zou teveel plaats kosten.
Bovendien hopen we dat de commentaar-regels in listing 3C - GRAPH.LIB - voor zich spreken.

```
{     GRAPH.LIB - procedures en functies voor de GRAPH-bibliotheek.
      maakt gebruik van de functies msxbios en msx2bios uit de
      GMSXBIOS.LIB-bibliotheek }

{ initialiseer de functietoetsen }
procedure inifnk;
begin
   msxbios($3E);
end;

{ schakel beeldscherm uit }
procedure disscr;
begin
   msxbios($41);
end;

{ schakel scherm in }
procedure enascr;
begin
   msxbios($44);
end;

{ lees een adres in VRAM }
function rdvrm(addr: integer): integer;
begin
   regHL := addr;
   msxbios($4A);
   rdvrm := regA;
end;

{ schrijf naar een adres in VRAM }
procedure wrtvrm(addr, data: integer);
begin
   regHL := addr;
   regA := data;
   msxbios($4D);
end;

{ vul VRAM-gebied met data byte }
procedure filvrm(addr, len : integer; data : byte);
begin
   regA := data;
   regBC := len;
   regDE := addr;
   msxbios($56);
end;

{ kopieer VRAM-gebied naar RAM: vramaddr tot vramaddr+len naar addr }
procedure ldirmv(len, addr, vramaddr:integer);
begin
   regBC := len;
   regDE := addr;
   regHL := vramaddr;
   msxbios($59);
end;
```

```
{ kopieer RAM naar VRAM: vanaf adres addr tot addr+len naar vramaddr }
procedure ldirvm(len, addr, vramaddr:integer);
begin
   regBC := len;
   regDE := vramaddr;
   regHL := addr;
   msxbios($5C);
end;

{ zet de scherm-mode: 0-3 op MSX1, 0-8 op MSX2 }
procedure chgmod(mode: integer);
begin
   regA := mode;
   msxbios($5F);
end;

{ zet voorgrond, achtergrond, randkleur }
procedure chgclr(voorgrond, achtergrond, rand : kleur);
begin
  forclr := voorgrond;
  bakclr := achtergrond;
  bdrclr := rand;
  msxbios($62);
end;

{ maak alle sprites leeg }
procedure clrspr;
begin
   msxbios($69);
end;

{ druk een letter af op het grafische scherm. positie wordt bepaald
  door grpacx en grpacy. doe dus eventueel eerst bijvoorbeeld:
  grpacx := 16; grpacy := 120; grpprt('D'); }
procedure grpprt(lett: char);
begin
   regA := ord(lett);
   msxbios($8D);
end;

{ schrijf iets naar de soundchip-register reg }
procedure wrtpsg(reg, iets: integer);
begin
   regA := reg;
   regDE := iets;
   msxbios($93);
end;

{ lees een register van de soundchip }
function rdpsg(reg: integer): integer;
begin
   regA := reg;
   msxbios($96);
   rdpsg := regA;
end;
```

```
{ haal toetsenbord-status: false is leeg, true is letter in buffer }
function chsns: boolean;
begin
   msxbios($9C);
   chsns := (( regF and $40) <> 0);  { test de Z-vlag }
end;

{ stuur een letter naar de printer }
procedure lptout(ch: char);
begin
   regA := ord(ch);
   msxbios($A5);
end;

{ test printer-status: true is klaar, false is niet klaar }
function lptstt: boolean;
begin
   msxbios($A8);
   lptstt := ((regF and $40) <> 0);  { test de Z-vlag }
end;

{ beep }
procedure beep;
begin
   msxbios($C0);
end;

{ wis het scherm in elke scherm-mode }
procedure cls;
begin
   inline($FD/$2A/$C0/$FC/  { LD IY,(EXPT) }
          $DD/$21/$C3/$00/  { LD IX,CLS    }
          $97/              { SUB A - Z-vlag moet gezet! }
          $CD/$1C/$00)      { CALL CALSLT  }
end;

{ haal de functietoetsen weg }
procedure erafnk;
begin
   msxbios($CC);
end;

{ laat de functietoetsen zien }
procedure dspfnk;
begin
   msxbios($CF);
end;

{ lees de joystick uit }
function gtstck(stick: integer): integer;
begin
   regA := stick;
   msxbios($D5);
   gtstck := regA;
end;

{ lees vuurknop-status }
```

```
function gttrig(stick: integer): integer;
begin
   regA := stick;
   msxbios($D8);
   gttrig := regA;
end;

{ lees touchpad-status }
function gtpad(pad: integer): integer;
begin
   regA := pad;
   msxbios($DB);
   gtpad := regA;
end;

{ lees paddle-status }
function gtpdl(pdl: integer): integer;
begin
   regA := pdl;
   msxbios($DE);
   gtpdl := regA;
end;

{ maak de toetsenbord-buffer leeg }
procedure kilbuf;
begin
   msxbios($156);
end;
```

Listing3c: GRAPH.LIB

DEMO.PAS

De laatste listing, nummer 4, is een klein demonstratieprogramma dat het gebruik van de GRAPH-bibliotheek laat zien. Nogmaals: de BlOS-lijst is niet compleet. Dus als u uitbreidingen heeft: stuur ze op, dan plaatsen we ze volgende keer samen met de MSX2-BIOS-lijst. Zo wordt Turbo Pascal toch nog een echte MSX-taal!

```
program graphdemo;

type
   {$I GRAPH.TYP}        { types van de graphg-bibliotheek }
var
   {$I GMSXBIOS.VAR}      { de reg-variabelen }
   {$I GRAPH.VAR}        { GRAPH-variabelen }
   i:  integer;
   ch: char;

{$I MSXBIOS.LIB}         { de bibliotheken zelf }
{$I GRAPH.LIB}

begin
   chgclr(1,8,1);        { voorgr. 1, achtergr. 8, border 1 }
   chgmod(0);            { naar scherm 0 }
   dspfnk;               { zet de functietoetsen neer }
   writeln('zwart-op-rood met F-toetsen');
   beep;
   read(kbd, ch);
   erafnk;               { haal de functietoetsen weer weg }
   chgclr(8,1,8);        { verander kleuren }
   writeln('rood-op-zwart zonder F-toetsen');
   beep;
   read(kbd, ch);
   chgclr(1,15,1);       { zwart op wit }
   chgmod(2);            { SCREEN 2 }
   for i:= 1 to 26 do
   begin
      grpacx := i*8;     { zet de X- ... }
      grpacy := i*6;     { en Y-coordinaten }
      forclr := i mod 16;{ verander voorgrondkleur }
      grpprt(chr(64+i)); { en druk een hoofdletter af }
   end;
   beep;
   read(kbd, ch);
   chgclr(15,4,4);       { normale kleuren }
   chgmod(0);            { terug naar scherm 0 }
   writeln('Klaar!');
end.
```

Listing 4: DEMO.PAS