MDL-lib maakt Pascal eenvoudiger BIBLIOTHEEK VOOR PASCAL PROGRAMMEURS

MSX Computer Magazine nummer 45 - maart 1991

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

Het programmeren in Turbo Pascal is populair in MSX-minnend Nederland. En dat is begrijpelijk; Pascal is een gestructureerde taal, die bovendien beduidend sneller is dan Basic. Dat is voornamelijk te danken aan het feit dat Pascal gecompileerd wordt. Helaas ontbreekt het Turbo Pascal aan grafische mogelijkheden, waarvan MSX-Basic weer veel beter voorzien is.

Verreweg de meeste programma's in MSX-Basic - die bijvoorbeeld gebruik maken van die grafische instructies - zijn daardoor niet zomaar om te zetten naar Turbo Pascal. Het kan wel, maar het kost moeite en vereist een grondige kennis van de MSX-computer.

De bibliotheek MDL-lib van Martijn Dekker brengt daar drastisch verandering in. Vrijwel alle mogelijkheden van Basic komen met deze uitgebreide bibliotheek binnen het bereik van de - gevorderde -Pascal-programmeur.

Succesvol, maar beperkt

Turbo Pascal is een succesvol product. Na de oorspronkelijke CP/M versie kwam er een versie voor MS-DOS, die inmiddels tot 6.0 geëvalueerd is. Gelukkig voor de MSX'ers is de CP/M versie door Philips aangepast voor MSX-DOS. Hoewel de 'goede oude' versie 3.0 zeker niet de modernste is, biedt hij een comfortabele manier om - ook grotere - programma's te ontwikkelen.
Maar zoals gezegd is het nut van Turbo Pascal beperkt, omdat er geen enkele MSX-specifieke opdracht in de hele taal te vinden is. Het tekstscherm wordt slechts ten dele ondersteund - geen kleur! - en het grafische scherm al helemaal niet. Geluid maken via de MSX geluidschip is niet mogelijk en ook de cassetterecorder wordt niet ondersteund, evenmin als de joystick of de muis.
Ergens is het wel begrijpelijk dat de pogingen om alle mogelijkheden van MSX-Basic toegankelijk te maken vanuit Turbo Pascal nooit echt geslaagd zijn. Het is namelijk helemaal niet zo eenvoudig als het lijkt. Een grondige kennis van zowel MSX als Turbo Pascal en machinetaal (!) is een vereiste om Turbo Pascal met de BIOS van zowel MSX l als MSX2 te laten samenwerken. De onderlinge verschillen tussen individuele MSX-computers - die echter wel allemaal binnen de standaard vallen - maken het er ook niet makkelijker op. Valt de snelheid van Pascal dan nooit te combineren met de veelzijdigheid van MSX-Basic?
Het antwoord is: nu wel. Want nu is er de MDL bibliotheek. Op het gevaar af als een reclamespotje te gaan klinken: dit is een droomproduct. Daar komt nog eens bij, dat het ontzettend weinig kost: slechts 25 guldens. U hoort het goed: slechts 25 guldentjes voor dit prima... Maar laten we ons niet te ver laten meeslepen. Wat biedt MDL de

Pascal programmeur?

MDLLIB:	Basis-bibliotheek. Bevat procedures die door de andere bibliotheken gebruikt worden, o.a. aanroep van de BIOS.
VRAM1:	VRAM- en VDP-routines voor MSX1.
VRAM2:	Idem, voor MSX2. Bevat een aantal extra procedures.
GRAPMSX1:	Grafische routines voor MSX1. Circels, vlakken, vullen enzovoort.
GRAPMSX2:	Idem voor MSX2, weer met een aantal extra's.
LOGOROUT:	Routines voor 'turtle-graphics'.
GAMEIO:	Ondersteuning van joystick, paddle, touchpad, MSX2-muis en -lichtpen.
SOUNDS:	Geluidsroutines; gemakkelijker te gebruiken dan de Basic-commando's.
DISKTOOL:	Disk-routines, o.a. sectoren lezen/schrijven, bestanden zoeken.
TAPE:	Cassette-routines.
DATA:	Routines om te sorteren en twee geheugenbereiken om te wisselen ('swap').
MISC:	Overige routines: BEEP, testen of te printer klaar staat, enzovoort.

Tabel 1: de twaalf MDL-bibliotheken

12 bibliotheken

Een eerste inspectie van de MDL-diskette leert dat er 43 bestanden op staan, waarvan er twaalf de extensie LIB dragen. Nog meer telwerk leert, dat MDL-lib 147 functies en procedures bevat. Dat lijkt wat overdreven. Waarom niet één grote bibliotheek? Toch heeft de auteur de hele bibliotheek bijzonder netjes opgezet. Ons recensieexpemplaar droeg het versienummer 2.0, waaruit je zou kunnen opmaken dat er ooit een 1.0 geweest is. Het is in ieder geval te zien dat deze bibliotheek al een tijdje bestaat: hij zit goed in elkaar.
Eigenlijk gaat het hier om een stelsel van bibliotheken die elkaar op allerlei manieren nodig hebben. Er is een basis-bibliotheek - MDLLIB geheten - die de routines bevat die alle.andere bibliotheken nodig hebben. Zo gebruikt bijvoorbeeld de VRAM2-bibliotheek deze functies en biedt hij daarnaast extra procedures zoals 'WriteVram'. De grafische bibliotheek voor MSX2 GRAPMSX2 maakt op zijn beurt weer gebruik van de routines uit VRAM2.
Deze aanpak heeft weliswaar tot gevolg dat er nogal wat verschillende bibliotheken op de werkdiskette rond zullen zwerven, maar het heeft ook voordelen. Zo blijft bijvoorbeeld de compileertijd beperkt; alleen de bibliotheken die echt gebruikt worden, worden meegecompi-leerd. De joystick-routines nemen - zolang ze niet nodig zijn - geen compilatietijd of geheugen in beslag. Een bijkomend voordeel is, dat het uiteindelijke .COM file kleiner is dan anders. Een nadeel is wel, dat er enig planwerk verricht moet worden, voordat het programma werkt. Elke bibliotheek bestaat namelijk nog eens uit maximaal vier delen: een deel met de TYPE, CONST en VAR definities plus een bestand met de eigenlijke procedures en functies. Niet elke bibliotheek heeft echter elk type bestand; dat moet even nagekeken worden. En dan komen we meteen op ons voornaamste punt van kritiek: de documentatie.

Helder maar onvolledig

Het is begrijpelijk dat Martijn de documentatie niet op papier bijlevert; dat kun je voor die prijs ook niet verwachten. De handleiding staat in de vorm van een aantal tekst-bestanden op diskette en met een demonstratieprogramma kunnen ze in de juiste volgorde worden afgedrukt. Maar wat te doen als er geen printer in huis is? Ons voorstel: lever de documentatie indien gewenst ook op papier bij, tegen kostprijs. Pluspunt is het feit dat de handleiding volledig Nederlandstalig is. Dat het schrijven van Nederlandse handleidingen in de toch op Engels gebaseerde computerwereld niet eenvoudig is blijkt bijvoorbeeld uit het door Martij n uitgevonden woord 'includeren'. Waar zou daar de klemtoon liggen? Is het includeren of includeren?
Wat we echter misten is een index. Het is niet mogelijk te achterhalen in welke bibliotheek een bepaalde functie of procedure zich bevindt en er is evenmin een overzicht van globale variabelen, types of constanten die door MDL gebruikt worden. Er is wel een - overigens uitstekende - beschrijving van elke deel-bibliotheek. Daarin staat een duidelijke omschrijving van alle onderdelen met een heleboel nuttige opmerkingen erbij.

Die zijn overigens soms wel wat technisch: bij de prompt-procedure staat bijvoorbeeld dat dit overeenkomt met: "REM of' in direct mode"
onder Basic. Het klopt, daar niet van, maar niet iedereen zal het begrijpen. Over de source-code van de bibliotheek is eigenlijk hetzelfde te zeggen. Hij is helder en van prima Nederlandstalig commentaar voorzien, maar dat commentaar is af en toe te technisch. Wie echter zijn of haar MSX goed kent zal het zeer verhelderend vinden om te zien wat er allemaal gebeurt.

Bibliotheek	**Routine + parameters:**	
MDLLIB	Procedure	_CL(slotptr, adres:integer);
LOGOROUT	Procedure	_Dr;
LOGOROUT	Procedure	_Pt;
GRAPMSX1	Procedure	Arc (x,y, hoek, straal, kleur : integer);
GRAPMSX2	Procedure	Arc (x,y, hoek, straal, kleur : integer);
LOGOROUT	Procedure	Back(afst:integer);
TAPE	Procedure	Baud(rate:integer);
MISC	Procedure	Beep;
MDLLIB	Procedure	Bios(adres:integer);
MDLLIB	Function	BiosMem(adres:integer):byte;
GRAPMSX1	Procedure	Box(x1,y1,x2,y2,color:integer);
GRAPMSX2	Procedure	Box(x1,y1,x2,y2,color:integer);
MDLLIB	Procedure	CalBas (adres:integer);
DATA	Function	CallFunc(var param1,param2):boolean;
GRAPMSX1	Procedure	Circle(x,y,straal,kleur:integer);
GRAPMSX2	Procedure	Circle(x,y,straal,kleur:integer);
LOGOROUT	Procedure	ClearScreen;
VRAM1	Procedure	ClearSprites;
VRAM2	Procedure	ClearSprites;
TAPE	Procedure	CloseCasOutput(var gelukt:boolean);
MISC	Procedure	ClrEos;
VRAM1	Procedure	ClrScr;
VRAM2	Procedure	ClrScr;
MISC	Procedure	ClrY(y1,y2:byte);
VRAM1	Procedure	Color(forclr,bakclr,bdrclr:integer);
VRAM2	Procedure	Color(forclr,bakclr,bdrclr:integer);
VRAM2	Function	ColorTable:integer;
GRAPMSX2	Procedure	CopyPic(x1,y1, x2,y2, sourcepage, xd,yd, destpage:integer);
MISC	Function	CtrlStop:boolean;
VRAM2	Procedure	DefColor(colour,red,green,blue : integer);
MISC	Procedure	DefKey(nummer:integer; tekst:LibStr);
TAPE	Procedure	DrInt;
GRAPMSX1	Procedure	Draw(x1,y1,x2,y2,color:integer);
GRAPMSX2	Procedure	Draw(x1,y1,x2,y2,color:integer);
GRAPMSX1	Procedure	DrawTo(x,y,color:integer);
GRAPMSX2	Procedure	DrawTo(x,y,color:integer);
TAPE	Procedure	DriveOff;
DISKTOOL	Function	Dskf(drive:byte):integer;
SOUNDS	Procedure	Effect(channels:SetOfChannel; nr:byte; time:integer);
GRAPMSX1	Procedure	Ellipse(x,y,straal,kleur,beg, eind:integer; afpl:real);
GRAPMSX2	Procedure	Ellipse(x,y,straal,kleur,beg,eind:integer; afpl:real);
GRAPMSX1	Procedure	FillBox(x1,y1,x2,y2,color:integer);
GRAPMSX2	Procedure	FillBox(x1,y1,x2,y2,color:integer);
GRAPMSX1	Procedure	FillPattern(x1,y1,x2,y2,kleur:integer);
GRAPMSX2	Procedure	FillPattern(x1,y1,x2,y2,kleur:integer);
GRAPMSX1	Procedure	FillScreen(kleur:integer);
GRAPMSX2	Procedure	FillScreen(kleur:integer);
GRAPMSX1	Procedure	FillShape(x,y,kleur,border:integer);
GRAPMSX2	Procedure	FillShape(x,y,clr,rand : integer);
LOGOROUT	Procedure	ForWd(dist:integer);
MISC	Function	FormatNumber(number:real; commas,asterisk, dollar,plus,sign,exp:boolean; bpnt,apnt:byte) : libStr;
GRAPMSX2	Function	GetDotColor(x,y:integer):integer;
GRAPMSX2	Procedure	GetPic(x1,y1, x2,y2:integer; var buffer);
GRAPMSX1	Procedure	gml(commands:libstr);
GRAPMSX2	Procedure	gml(commands:libstr);
VRAM1	Procedure	GotoXY(x,y:byte);
VRAM2	Procedure	GotoXY(x,y:byte);
GRAPMSX1	Procedure	Gwrite(x,y:integer;text:libstr);
GRAPMSX2	Procedure	Gwrite(x,y:integer;text:libstr);
LOGOROUT	Procedure	HideTurtle;
LOGOROUT	Procedure	Home;
VRAM1	Procedure	InitCharMode(width,char1,char2:byte);

Tabel 2: Een deel van de mogelijkheden van MDL-LIB

Interface

De MDL-lib biedt de Pascal programmeur een interface tussen Turbo Pascal en de

MSX BIOS. Dat lijkt misschien simpel,
maar dat is het zeker niet. Voor zover wij hebben kunnen zien is Martijn echter bijzonder netjes te werk gegaan; wij zijn ervan overtuigd dat alle procedures op alle bestaande MSX computers zullen werken - en dat hebben we wel eens anders gezien! De enige uitzondering daarop zijn de procedures die de Basic-interpreter aanroepen, maar daarbij geeft Martijn zelf al aan dat er mogelijk problemen kunnen optreden, hoewel hij ze niet verwacht. Wij verwachten ook weinig problemen, maar je weet het nooit. Het behoort in ieder geval vermeld te worden als er dingen niet volgens de standaard werken.
Maar er is meer. De bibliotheken bieden ook een paar onverwachte extra's, zoals een zeer algemene sorteer-routine. Ook zijn er disk-routines en zelfs 'turtle-gra-phics'. Zie tabel l voor een overzicht van de bibliotheken en hun taken. Ook is de write-opdracht intelligent: als het scherm in de grafische mode staat, kan er toch tekst op afgedrukt worden, gewoon met write() en writeln(). Nogmaals: het is een compleet en doordacht product.

```
Program Circles;
{
   Demonstratieprogramma voor de MDL-bibliotheek
   MSX Computer Magazine
}
Type
   {$I MDLLIB.TYP}    { Verplicht }
   {$I GRAPMSX2.TYP}  { Nodig voor MSX2 grafische bibliotheek }

Const                 { Geen constanten voor VRAM2 of GRAPMSX2 }
   NHor = 8;          { Eigen constanten: schermdelen horizontaal }
   NVert = 4;         { ... en verticaal }
   SCREEN = 5;        { Scherm 5, dus }
   NX = 256;          { Afmetingen van schermtype }
   NY = 212;
   NC = 16;           { Aantal kleuren van schermtype }
   AANTAL = 200;       { Aantal herhalingen }

Var
   {$I MDLLIB.VAR}    { Verplicht }
   {$I VRAM2.VAR}     { Voor VRAM2. GRAPMSX2 heeft geen VAR }
                      { Eigen variabelen: }
   X, Y, C, I, X2, Y2: Integer;
   Dx, Dy: Integer;

{$I MDLLIB.LIB}       { De drie gebruikte bibliotheken }
{$I VRAM2.LIB}        { in de juiste volgorde: GRAPMSX2 }
{$I GRAPMSX2.LIB}     { gebruikt VRAM2 gebruikt MDLLIB }

Procedure Wacht;      { Wacht op een toets }
Var Ch: Char;
Begin
   Read(Kbd, Ch);
End;

Function Min(A, B: Integer): Integer;
Begin
   If A < B Then Min := A Else Min := B;
End;

{ DoeIets: tekent een rechthoek met daarin een kruis
  en twee cirkels }
Procedure DoeIets(X, Y, B, H, K: Integer);
Begin
   Box(X, Y, X + B - 1, Y + H - 1, K);
   Draw(X, Y, X + B - 1, Y + H - 1, NC - K - 1);
   Draw(X, Y + H - 1, X + B - 1, Y, NC - K - 1);
   Circle(X + B div 2, Y + H div 2, Min(B div 2, H div 2), K);
   Ellipse(X + B div 2, Y + H div 2, H div 2, K, 0, 360, H / B);
End;

{ Inverteer: inverteert een rechthoek }
Procedure Inverteer(X, Y, B, H: Integer);
Begin
   Logical(3);    { XOR }
   FillBox(X, Y, X + B - 1, Y + H - 1, NC - 1);
End;

Begin
   ScrMode(SCREEN);      { Werken in SCREEN 5 }
   Color(15, 0, 0);      { Standaard wit op zwart }
   ClrScr;               { Maak scherm schoon }
   Dx := NX div NHor;    { Bereken breedte en hoogte }
   Dy := NY div NVert;   { van een schermdeel }
   C := 0;               { Eerste kleur }
   X := 0;               { Eerste X-coordinaat }
   While X < NX Do Begin
      Y := 0;                          { Eerste Y-coordinaat }
      While Y < NY Do Begin
         DoeIets(X, Y, Dx, Dy, C);
         C := (C + 1) Mod NC;          { Volgende kleur }
         Y := Y + Dy;                  { Volgende Y-coordinaat }
      End;
      X := X + Dx;                     { Volgende X }
   End;
   Wacht;
   For I := 1 to AANTAL Do Begin
      { Kies twee random schermdelen: }
      X := Random(NHor) * Dx;
```

```
      Y := Random(NVert) * Dy;
      Repeat
         X2 := Random(NHor) * Dx;
         Y2 := Random(NVert) * Dy;
      Until (X2 <> X) And (Y2 <> Y);
      { ... totdat ze niet gelijk zijn }
      { Inverteer beide schermdelen: }
      Inverteer(X, Y, Dx, Dy);
      Inverteer(X2, Y2, Dx, Dy);
      Logical(0);       { PSET, voor CopyPic }
      { Kopieer deel 1 naar deel 2: }
      CopyPic(X, Y, X + Dx - 1, Y + Dy - 1, -1, X2, Y2, -1);
      { ... en inverteer weer terug }
      Inverteer(X, Y, Dx, Dy);
      Inverteer(X2, Y2, Dx, Dy);
   End;
   Wacht;
   ScrMode(0);          { Terug naar scherm 0 }
End.
```

PC-compatibel

Wat kunnen we nog meer zeggen? Voor zover wij hebben kunnen controleren is zo ongeveer elke BIOS routine van zowel MSX1 als MSX2 vertegenwoordigd in MDL en nog onder een begrijpelijke en gemakkelijk te onthouden naam ook. De meeste procedures hebben dezelfde naam als in Basic, of de naamgeving komt overeen met Turbo Pascal voor de PC.
Zelfs zonder documentatie op papier en zonder alfabetische inhoudsopgave per bibliotheek konden we het Pascal-programma in listing l in ongeveer tien minuten schrijven. Dat geeft een aardig idee over hoe de bibliotheken gebruikt moeten worden en hoe de grafische capaciteiten van MSX2 in scherm 5 benut kunnen worden. Het programma tekent 32 figuren op het scherm in 16 kleuren en kiest vervolgens steeds willekeurig een figuur, die dan over een ander figuur heen gelegd wordt.
We hebben ook een Basic-versie gemaakt, zie listing 2. In het eerste deel is er weinig verschil te merken tussen MSX-Basic en Turbo Pascal: de eerste doet er bijna 10 seconden over, de tweede ruim 11. Dit is waarschijnlijk te wijten aan het feit dat de CIRCLE-opdracht vanuit Pascal via een aanroep van de Basic-ROM verloopt. Hiervoor is het nodig dat er een Basic-statement aangemaakt wordt, dat vervolgens door de interpreter wordt afgehandeld. Dit is de enige manier om de CIRCLE-opdracht via de ROM te laten verlopen, maar het is natuurlijk niet sneller dan Basic.
In het tweede deel wordt niet de Basic-ROM maar de SUB-ROM gebruikt, namelijk voor het inverteren en kopiëren van de rechthoeken. Hierbij hoeft er alleen maar een inter-slot CALL te worden uitgevoerd en er blijkt wel een duidelijk verschil. Turbo Pascal scoort een kleine 14 seconden, tegen Basic 37. Dat is twee tot drie keer zo snel. Nu is dit natuurlijk een vrij willekeurige test, maar hij heeft wel degelijk waarde. Zoals gezegd is de functie van MDL heel vaak niet meer dan die van een interface; het is de BIOS die steeds het zware werk doet. Sommige functies worden daardoor niet sneller, maar juist een tikje langzamer; andere gaan er wel op vooruit. De winst zit hem echter niet in de aanroep van de BIOS, maar in het verschil tussen een compiler en een interpreter. Bewezen is in ieder geval wel, dat de MDL bibliotheek Turbo Pascal een stuk krachtiger maakt.

Nog even wat feiten: het eigenlijke Pascal-programma is 98 regels lang. Turbo Pascal meldt na compileren echter dat er 1038 regels gecompileerd zijn; voor het gebruik van de bibliotheken MDLLIB, VRAM2 en GRAPMSX2 moet kennelijk de prijs van ruim 900 te compileren regels betaald worden. Het resultaat was een .COM bestand van 13 kB.

```
10 REM CIRCLS - een testprogrammaatje                                    0
20 REM Alleen voor MSX2!                                                 0
30 REM MSX Computer Magazine                                             0
40 REM                                                                   0
50 DEFINT A-Z                                                           30
60 DX=256/8: DY=212/4                                                  215
70 C=0                                                                  84
80 COLOR 15,0,0: SCREEN 5                                              214
90 FOR X=0 TO 256-1 STEP DX                                            107
100    FOR Y=0 TO 212-1 STEP DY                                        226
110       LINE (X,Y)-(X+DX-1,Y+DY-1),C,B                               233
120       LINE (X,Y+DY-1)-(X+DX-1,Y),15-C                              239
130       LINE (X,Y)-(X+DX-1,Y+DY-1),15-C                               19
140       CIRCLE (X+DX/2,Y+DY/2),DX/2,C                                114
150       CIRCLE (X+DX/2,Y+DY/2),DY/2,C,,,DY/DX                         22
160       C=(C+1) MOD 16                                               248
170   NEXT Y                                                            69
180 NEXT X                                                             109
190 A$=INPUT$(1)                                                       171
200 FOR I=1 TO 200                                                     204
210    X=INT(RND(1)*8)*DX: Y=INT(RND(1)*4)*DY                          137
220    X2=INT(RND(1)*8)*DX: Y2=INT(RND(1)*4)*DY                        235
230    IF X2=X AND Y2=Y THEN GOTO 220                                   19
240    LINE (X,Y)-(X+DX-1,Y+DY-1),15,BF,XOR                            111
250    LINE (X2,Y2)-(X2+DX-1,Y2+DY-1),15,BF,XOR                        134
260    COPY (X,Y)-(X+DX-1,Y+DY-1) TO (X2,Y2)                           132
270    LINE (X,Y)-(X+DX-1,Y+DY-1),15,BF,XOR                            117
280    LINE (X2,Y2)-(X2+DX-1,Y2+DY-1),15,BF,XOR                        140
290 NEXT I                                                             218
300 A$=INPUT$(1): SCREEN 0                                              69
```