Turbo Pascal deel 2

MSX CLUB MAGAZINE 35

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

In deze aflevering komen de grafische routines aan bod.
Het is onmogelijk alle routines in het magazine behandelen, zie ook het bestand GRAPH1.LIB als bijlage, een verzameling van grafische routines te gebruiken in Turbo Pascal. Enkele van deze routines zullen in deze aflevering behandeld worden, met name die routines die programmeertechnisch interessant zijn. De overige routines worden aan het eind van deze aflevering in het kort besproken.

ROM-BIOS Routines

Vrijwel alle grafische BASIC-com-mando's maken gebruik van zogenaamde BlOS-routines. Dit zijn routines in het ROM van de computer. Zo zijn er bijvoorbeeld complete routines voor het tekenen van lijnen, zodat de programmeur deze routines niet zelf hoeft te schrijven. Bij MSX2 computers zijn naast de gewone (MSX1) routines ook routines opgenomen voor de extra grafische schermen. Deze bevinden zich in het sub-ROM. Om de BlOS-routines vanuit Turbo Pascal aan te roepen, moet gebruik worden gemaakt van de procedures MsxBios en Msx2Bios. Deze routines zijn te vinden op het diskabonnement in de file MSXBIOS.UB en zijn tevens beschreven in MSX Magazine 21 en 27. Omdat dit een cursus Turbo Pascal is en geen cursus machinetaal, zal ik een bespreking ervan achterwege laten. Het zelfde geldt voor de BIOS routines. Deze zijn te vinden in vrijwel elk MSX (2) zakboekje. De grafische routines zijn overigens allemaal geschreven voor MSX2 Computers.

Include files

Voordat ik verder ga met de bespreking van de grafische routines, moet ik eerst iets zeggen over zogenaamde Include files. Dit zijn stukjes programmatekst, meestal standaardprocedures, die in een programma kunnen worden opgenomen. Een voorbeeld hiervan is het bestand GRAPH1.LIB, dat grafische routines bevat. Om gebruik te kunnen maken van deze grafische routines, moet in het programma worden aangegeven dat deze in-clude file mee gecompileerd moet worden. Dat gaat door middel van een {$1 naam.ext} directive. Door bijvoorbeeld in het programma de regel f$I graphl.lib} toe te voegen, wordt tijdens het compileren GRAPH1 .LIB geladen en met de rest van het programma meegecompileerd Een programma dat gebruik maakt van grafische routines ziet er uit als in voorbeeld l hieronder.

```
PROGRAM  [naam];

CONST;
[Declaratie van constanten]
```

```
TYPE
[Declaratie van  typen]

VAR
[Declaratie van variabelen]

{$I msxbios. lib}
{$i graph1.lib}

{Eigen procedures en functies}

BEGIN
[Hoofdprog rammaJ
END.
```

Zoals je ziet wordt ook het bestand MSXBIOS.LIB meegecompileerd. Dit omdat de meeste grafische routines gebruik maken van het BIOS.

Enkele voorbeelden

Een eenvoudig voorbeeld van een grafische procedure is de Cls procedure, de TP versie van het BASIC CLS-commando. De procedure ziet er als volgt uit:

```
PROCEDURE Cls;

BEGIN
  MsxBios($00C3)
END;
```

De procedure roept de BIOS-routi-ne &H00C3 aan. Deze BlOS-routine maakt het scherm schoon. In tegenstelling tot de standaard procedure ClrScr, die alleen tekstschermen schoon maakt, werkt deze routine ook op grafische schermen. Merk op dat in TP hexadecimale getallen worden aangegeven met een dollarteken.

Een andere procedure is het LINE-commando. De Msx2Bios routine voor het tekenen van lijnen maakt gebruik van systeemvariabelen. Deze variabelen staan op een vaste plaats in het geheugen. Zo staat bijvoorbeeld op adres $FCB3 de sys-teemvariabele GXPos. Deze variabele bevat bijvoorbeeld de eind-X-coördinaat van een lijn. De gemakkelijkste manier om hiervan in TP gebruik te kunnen maken, is het declareren van ABSOLUTE variabelen. Dit zijn normale variabelen, met het verschil dat ze op een vast adres in het geheugen staan. Het gebruik van absolute variabelen gaat als volgt: Het tekenen van een lijn op de schermen 5 t/m 8 gebeurt door middel van de Msx2Bios routine $0085. Als invoer moeten de (machinetaal) registers BC en DE de begjncoördinaat bevatten, de systeemvariabelen GXPos en GYPos de eindcoördinaat en de systeemvariabele AtrByt de kleur. Aan het eind van de procedure worden de systeemvariabelen GrPackX en GrPackY gevuld met de eindcoördinaat. Dit is alleen van

belang als er tekst op een grafisch scherm moet worden gezet. Aan de hand van dit voorbeeld moet u zelf in staat zijn om de procedures LineTo, LineStep en Line ToStep, de TP varianten van de LINE- , LINE STEP en LINE -STEP commando's, te schrijven.

Selectie en iteratie

Veel theorie is nu behandeld, maar van de drie basisprincipes van gestructureerd programmeren is er nog maar één aan bod gekomen, namelijk de modulaire opbouw. Essentieel voor vrijwel elk programma is selectie en iteratie, of-
tewel het gebruik van keuzes en loops. Alle vormen van selectie en iteratie en combinaties hiervan zullen achtereenvolgens worden behandeld.

IF .. THEN .. ELSE

Deze eerste vorm van selectie is ook in BASIC bekend. Als aan een bepaalde voorwaarde wordt voldaan (TRUE is), wordt het statement na THEN uitgevoerd. Het ELSE gedeelte is optioneel. Zie het voorbeeld 4 hierboven.
Er kan ook gebruik worden gemaakt van boolean variabelen:

```
IF KeyPressed THEN....
```

De variabele KeyPressed is een standaardvariabele binnen Turbo Pascal van het type BOOLEAN. Als de waarde TRUE is wil dat zeggen dat een toets is ingedrukt. De bovengenoemde regel komt dus overeen met

```
IF KeyPressed=TRUE THEN
```

Evenzo is

```
 IF not KeyPressed THEN ....
```

gelijk aan

```
IF Key-Pressed=FALSETHEN ....
```

CASE

De tweede vorm van selectie is het CASE statement. Met dit statement kunnen aan de hand van de waarde van een variabele verschillende handelingen worden uitgevoerd. Bijvoorbeeld:

```
CASE SchermMode OF
  5,8:SchermBreedte:=256;
  6,7:SchermBreedte:=512;
END;
```

Als de huidige schermmode 5 of 8 is, dan is de maximale scherm-breedte 256 pixels, anders 512 pixels. Merk op dat een CASE statement altijd wordt afgesloten met een END;.

FOR .. TO/DOWNTO .. DO

Deze vorm van iteratie vertoond veel gelijkenissen met de FOR . . TO loop in BASIC. Er zijn enkele verschillen:

- Turbo Pascal kent geen STEP mogelijkheid.
- Als de besturingsvariabele moet worden verlaagd, gebruik je DOWNTO.
- Alle soorten variabelen mogen in Turbo Pascal worden gebruikt.

WHILE.. DO en REPEAT.. UNTIL

Naast de bovengenoemde vormen van selectie en iteratie kent TP ook nog twee vormen van combinaties hiervan. De eerste vorm is het WHILE . . DO statement, vrij vertaald: "Zolang aan een voorwaarde voldaan wordt doe...". Het voorbeeld 6 (zie rechtboven) totaliseert een rij getallen totdat een O wordt ingegeven.
Merk op dat de handeling achter de WHILE . . DO tussen een BEGIN en END; staat. Dit omdat de handeling een zogenaamd samengesteld statement is. Hierop kom ik later terug.
Een andere vorm van selectie en iteratie is de REPEAT . . UNTIL lus: "Herhaal een bepaalde handeling totdat aan een voorwaarde wordt voldaan". Zie voorbeeld 7 rechtsboven.
Het verschil tussen WHILE . . DO en REPEAT . . UNTIL is dat de handeling na REPEAT minstens één keer wordt uitgevoerd. Bij WHILE . . DO hoeft dat niet het geval te zijn.

Als aan meerdere voorwaarden moet worden voldaan, moet elke voorwaarde tussen haakjes worden geschreven. Dat geldt ook voor het IF . . THEN . . ELSE en WHILE . .

Samengestelde statements

Als na een selectie of iteratie meerdere handelingen moeten worden uitgevoerd, dan moeten deze handelingen tussen een BEGIN en END; worden gezet. Alles wat tussen BEGIN en END; staat wordt dan opgevat als één statement, namelijk een samengesteld statement. De noodzaak van het gebruik van samengestelde statements wordt in voorbeeld 9 duidelijk (de volgende twee procedures staan ook in het bestand GRAPHl.LIB):

Text en TextXY

De procedures Text en TextXY worden gebruikt voor het afdrukken van tekst op een grafisch scherm. De procedure Text drukt de tekst af op de huidige (grafische) cursorpositie, aangegeven met de GrPackX en GrPackY systeemvariabelen. Met de

procedure TextXY kan de tekst op elke willekeurige positie op het scherm worden af gedrukt.
Als in de procedure Text de BEGIN en END; achter de FOR-lus zouden zijn weggelaten dat heeft dat tot resultaat dat alleen de laatste letter van de string op het grafische scherm wordt afgedrukt. De eerste regel van het samengestelde statement wordt even vaak uitgevoerd als de string lang is, terwijl de MsxBios routine $008D maar één keer wordt uitgevoerd omdat die dan niet meer bij het FOR . . TO . . DO statement hoort. De END; geeft als het ware de NEXT in basic aan.
Een string wordt gezien als een array van het type CHAR. In dit voorbeeld is de string Tekst dus een array van maximaal 255 tekens groot. Om een bepaalde positie binnen een array aan te geven maak je gebruik van een index tussen twee vierkante haken. Het element (karakter) nummer n van de string Tekst wordt dus aangegeven met Tekst[n]. Dit komt overeen met BASIC: MID$ (T$, n, l). Op arrays kom ik later in deze aflevering terug.
De procedure TextXY roept de procedure Text aan. Dat kan alleen als de procedure Text vóór de procedure TextXY is gedeclareerd. De reden hiervoor is dat TP een single-pass compiler is, hetgeen inhoudt dat een programma in één keer wordt gecompileerd. Als de procedure Text achter de TextXY procedure zou staan, dan zou de aanroep ervan in de TextXY procedure onbekend zijn.

Puntkomma's

In TP moet achter elk statement een puntkomma staan, behalve achter het laatste statement in een samengesteld statement (het laatste statement voor de END;). Het is echter wel toegestaan om achter dit laatste statement een puntkomma te plaatsen.

Voor de goede orde

In de tot nu toe behandelde voorbeelden is een bepaalde lay-out van de programmatekst te ontdekken. Op MSX gebied staat Turbo Pascal voor, ik zeg het nog maar eens, gestructureerd programmeren. Een overzichtelijke lay-out is hierbij een handig instrument. Het benadrukt namelijk de structuur en maakt het programma leesbaarder. Met dit laatste bedoel ik dat als je je programma overzichtelijk opbouwt, het makkelijker is om het na maanden terug te lezen en nog te begrijpen ook. Iets wat over programmeren in BASIC en machinetaal niet altijd gezegd kan worden. Als je de lay-out opbouwt volgens de volgende afspraken, ziet je programma er veel professioneler en duidelijker uit
Typ alle woorden die de structuur van het programma aangeven in HOOFDLETTERS. Dit zijn woorden die worden gebruikt voor declaraties (VAR, PROCEDURE, BEGIN, END enz.), woorden die betrekking hebben op iteratie en selectie (FOR, WHILE, CASE enz.) en woorden die standaard gegevenstypen aanduiden (INTEGER, BOOLEAN enz.).
Typ de overige woorden (procedurenamen, variabelen enz.) in kleine letters, beginnend met een hoofdletter. Als een naam uit meerdere delen bestaat, gebruik dan meerdere hoofdletters. Denk bijvoorbeeld aan de standaard-variabele KeyPressed, de procedure LineTo en de standaardprocedures en -functies WriteLn, Cos, UpCase, QrScr enz.

Geef procedures, functies en variabelen een overzichtelijke naam, die de inhoud ervan dekt.
Spring na elke BEGIN 3 spaties in en zorg dat elke END in dezelfde kolom begint als de bijbehorende BEGIN. Spring bij samengestelde statements ook 3 spaties in. Voorbeelden:

```
WHILE ... DO

   BEGIN

  END;

IF .... THEN
  BEGIN
  END ELSE
  BEGIN

     REPEAT

    UNTIL ...

END;
```

Arrays en enumeratietypen

Naast de tot nu toe behandelde gegevenstypen (INTEGER, BYTE, BOOLEAN, CHAR en STRING) bestaan er ook nog arrays en enumeratietypen. Arrays worden gedeclareerd als te zien boven aan volgende pagina. In voorbeeld 10 wordt eerst de regel gegeven en direct eronder een voorbeeld. Het gegevenstype kan zelf ook weer een array zijn, zie voorbeeld 11, met na de eerste regel de kortere schrijfwijze. Hetgeen overeenkomt met DIM AR (3 ,4) in BASIC. Als IndexType kunnen alle soorten gegevenstypen worden gebruikt, bijvoorbeeld:

```
Arrayl: ARRAY ['a' . . 'z' ] OF INTEGER;
```

Een waarde in deze array kan als volgt worden toegekend:

```
Array1['f']  := 123;
```

Enumeratietypen zijn typen waarbij elke mogelijke waarde een naam heeft kijk naar voorbeeld 12. De variabele Verkeerslicht kan slechts drie waarden hebben, te weten Rood, Oranje of Groen. De volgende toekenning is bijvoorbeeld mogelijk:

```
Verkeerslicht := Oranje;
```

De namen Rood, Oranje en Groen zijn GEEN strings, ze zijn slechts een benaming voor een mogelijke waarde. Bij het compileren wordt de naam Rood vervangen door waarde O, Oranje door waarde l en Groen door waarde 2. De variabele Verkeerslicht wordt in

feite dus gevuld met waarde 1. Het laatste voorbeeld (13) staat ook in het bestand GRAPH1.LIB.

Logische operaties

De procedure LogOpr wordt gebruikt om de zogenaamde logische operatie code in te stellen. Bij de meeste grafische BASIC-commando's kan deze code optioneel worden meegegeven, voorbeeld: PSET (100,100),1,TPSET. Aangezien procedures in TP geen optionele parameters kennen, heb ik voor het instellen van de logische operatie code (LOC) een aparte procedure gemaakt. De standaard LOC's zijn pset, and, or, xor, preset, tpset, tand, tor, txor en tpreset. Deze komen respectievelijk overeen met de waarden O, l, 2,3,4,8,9,10,11 en 12. In het voorbeeld is voor de eerste 5 woorden de letter "l" van "logisch" gezet omdat de woorden "and" en "or" gereserveerde zijn en dus niet als enumeratietype mogen worden gebruikt. In de procedure wordt gebruik gemaakt van de Ord-func-tie. In de vorige aflevering heb ik gezegd dat deze overeenkomt met het BASIC commando ASC. Dit is niet volledig. De stelling is juist voor gegevens van het type CHAR, maar de Ord-functie kan ook voor enumeratietypen worden gebruikt. Het resultaat van de functie is dan het orde-nummer (volgnummer) van de naam in de lijst. Het orde-nummer van Ipset is dus O en van tpset 5. De variabele Temp wordt geladen met het ordenummer van de Code van het type LogOprCodes. Voor tpset is deze waarde dus 5. Voor de systeem-variabele LogOp moet tpset echter de waarde 8 (zie voorgaand lijstje) vertegenwoordigen, vandaar de tweede regel van de procedure.
De procedure kan bijvoorbeeld als volgt worden aangeroepen:

```
LogOpr   (tpset);
```

Alle grafische procedures die daarna worden aangeroepen zullen volgens de LOC in LogOp worden uitgevoerd.

De volgende keer

Het tempo van deze cursus is hoog, maar er is dan ook nog veel dat ik wil behandelen. In de volgende aflevering worden de grafische procedures afgesloten. Ik zal het dan voornamelijk hebben over een routine voor SMOOTH-SCROLL in maar liefst VIER RICHTINGEN. Deze aflevering besluit ik met een overzicht .. Veel Succes!

Turbo Pascal procedures en functions voor MSX-grafisch

PROCEDURE Screen (Node: BYTE);
Komt overeen met het SCREEN commando.

PROCEDURE SetPage (Display, Active: BYTE);
Komt overeen met het SET PAGE commando.
PROCEDURE Cls;
Komt overeen met CLS.

PROCEDURE Color (Foreground, Background, Border BYTE);
Komt overeen met COLOR.

PROCEDURE SetPalet (Numbex, Red, Oxeen, Blue: BYTE);
Komt overeen met COLOR=(n,r,g,b).

PROCEDURE GetPalet (Number: BYTE; VAR Red, Oxeen, Blue: INTEGER);
Haalt een palet op uit het videoram en zet de waarden in de laatste 3 parameters. Na de volgende regel bevatten de variabelen r, g en b het aantal tinten rood, groen en blauw van kleur 14: GetPalet (14,r,g,b);

PROCEDURE NewPalet;
Komt overeen met COLOR=NEW.

PROCEDURE RestorePalet;
Komt overeen met COLOR=RESTORE.

PROCEDURE Pset (x,y: INTEGER; Color: BYTE);
Komt overeen met PSET.

FUNCTION Point (x,y: INTEGER): BYTE;
Komt overeen met POINT.

PROCEDURE Line (xl, yl, x2, y2 : INTEGER; Colox: BYTE);
Komt overeen met LINE.

PROCEDURE Box (xl,yl,x2,y2: INTEGER; Color: BYTE);
Tekent een vierkant. Komt overeen met LINE (xl,yl)-(x2,y2),c,B.

PROCEDURE BoxF (xl,yl,x2,y2: INTEGER; Color: BYTE);
Tekent een gevuld vierkant. Komt overeen met LINE (xl,yl)-(x2,y2),c,BF.

PROCEDURE Draw (Macro: Str255);
Komt in grote lijnen overeen met het DRAW commando. Het gebruik van variabelen in de tekststring is echter niet toegestaan.

PROCEDURE Circle (mx,my,Radius: INTEGER; Color: BYTE);
Tekent een, afhankelijk van de schermmode, perfect ronde cirkel met middelpunt (mx,my) en een Radius (in verticale richting).

PROCEDURE Ellipse (mx,my,xx,xy: INTEGER; Color: BYTE);
Tekent een ellips met middelpunt (mx,my) en radius (rx,ry). Radius isde halve breedte van de ellips. Radius ry is de halve hoogte van de ellips.

PROCEDURE LogOpx (Code: LogOprCodes);
Stelt de logische operatie code in. De codes die niet met een "t" beginnen (pset, or etc.) moeten worden voorafgegaan door een "l" (Ipset, lor etc.).

PROCEDURE PatternPaint (fx,£y: INTEGER; Pattexn: BYTE);
Deze procedure vult een willekeurig vlak met een bepaald patroon. De patronen worden vastgelegd in de procedure Pattems, in de vorm van een 8*8 bytes patroon. Er zijn al twee motieven in die procedure vastgelegd, namelijk een baksteenmotief (nr. 0) en een vlechtmotief(nr. 1). Het aantal patronen kan naar eigen behoefte worden uitgebreid. Geef behalve de x- en y-coördinaat ook het patroonnummer mee. Enkele belangrijke verschillen met het PAINT commando:
l - Het te vullen gebied moet helemaal omlijnd zijn, dus ook de eventuele randen van het scherm.
2-De omlijning hoeft niet, zoals een BASIC, van één doorlopende kleur te zijn.
3-De patroonvul procedure werkt langzamer dan in BASIC.

PROCEDURE ScreenCopy (sx,sy,nx,ny: INTEGER; SourcePage: BYTE;dx,dy: INTEGER;
DestinationPage: BYTE);
Komt in grote lijnen overeen met het COPY commando voor grafische schermen. Een belangrijk verschil: Met nx en ny wordt NIET de eindcoördinaat aangeduid, maar de breedte en hoogte van het te kopieren blok in pixels.

PROCEDURE CoplnBytes(sx,sy,nx,ny: INTEGER; SouzcePage: BYTE;dx,dy: INTEGER;
DestinationPage: BYTE);
Is in grote lijnen gelijk aan de ScreenCopy procedure. Deze procedure kopieert gedeelten van het beeldscherm in hele bytes. In de schermmodes 5, 6 en 7 bestaat één byte namelijk uit meerdere pixels. Dit houdt in dat deze routine sneller werkt als ScreenCopy. Een voorwaarde is wel dat de coördinaten, breedte en hoogte in scherm 5 en 7 veelvouden moeten zijn van 2 en in scherm 6 een veelvoud van 4. Als u gebruik maakt van een logische operatie code (m. b.v. de procedu re LogOpr), dan werkt alleen de procedure ScreenCopy.

PROCEDURE Text (Tekst: Str255);
Deze procedure schrijft tekst op het grafische scherm, op de huidige grafische cursorpositie.

PROCEDURE TextXY (x,y: INTEGER; Tekst: Str255);
Deze procedure schrijft tekst op het grafische scherm, op de coördinaten aangegeven met x en y.

PROCEDURE Vpoke (Adres: INTEGER; Data: BYTE);
Komt overeen met VPOKE.

FUNCTION Vpeek (Adres: INTEGER): BYTE;
Komt overeen met VPEEK.

PROCEDURE SetAdjust (ax,ay: INTEGER);
Komt overeen met SET ADJUST.

PROCEDURE RamToVram (Source,Destination,Lengte: INTEGER);
Verplaatst een blok vanaf Source in het RAM, met Lengte bytes naar Destination in het VRAM.

PROCEDURE VramToRam (Source,Destination,Lengte: INTEGER);
Verplaatst een blok vanaf Source in het VRAM, met Lengte bytes naar Destination in het RAM.

PROCEDURE FillVram (StartAdress, EndAdress: INTEGER; Data: BYTE);
Vult het videoram van StartAdress t/m EndAdress met Data.

PROCEDURE WrtVdp (Register,Data: BYTE);
Schrijft Data naar een VDP-Register. Dit register komt niet altijd overeen met VDP() in BASIC. Zie voor de verschillen de cursus Ken uw Computer.

PROCEDURE SpriteSize (Size: BYTE; Display; DisplaySizes);
Definieert de sprite-grootte. Met size wordt de grootte in pixels aangegeven (8 of 16 pixels). Met Display wordt aangegeven of de sprites twee maal zo groot moeten worden weergegeven. Display is een enumeratietype en kan de waarden normaal of groot hebben.

PROCEDURE DefSprite (Number: BYTE; Pattern: Str6);
Komt overeen met SPRITE$(N)=P$.

PROCEDURE DefSpriteColor (Number,Color: BYTE);
Komt overeen met COLOR SPRITE(N)=C.

PROCEDURE DefSpriteColors (Number: BYTE; Colors: Strl6);
Komt overeen met COLOR SPRITE$(N)=C$.

PROCEDURE PutSprite (VlakNr: BYTE; x,y: INTEGER; SprNr: BYTE);
Komt in grote lijnen overeen met PUT SPRITE. Er wordt geen parameter met de spritekleur meegegeven. Gebruik hiervoor de procedure DefSpriteColor.