Schermen op MSX

De 2+ schermen
Alex Wulms

MSX Computer & Club Magazine nummer 72 - december '94 / januari '95

We gaan het hebben over de MSX 2+ schermen. Zoals wellicht bekend, zijn dit de schermen met de nummers 10, 11 en 12. Zij zitten op de MSX 2+ en ook de MSX TurboR.

Op deze MSX 2+ schermen kunnen zeer veel kleuren tegelijk worden getoond; op scherm 12 zijn het er 19268 en op de schermen 10 en 11 zijn het 12599. De 2+ schermen hebben dezelfde resolutie als scherm 8 van de MSX 2, dat slechts 256 kleuren tegelijk kan weergeven. Desondanks gebruiken de MSX 2+ schermen niet meer geheugen dan scherm 8. Dit heeft men bereikt door een slimme truc te gebruiken bij het opslaan van de schermen. Heel globaal gesteld hebben vier naast elkaar liggende pixels dezelfde kleur en kan per pixel de helderheid worden ingesteld. De kleurinformatie wordt op deze manier verdeeld over meerdere pixels, waardoor je in totaal meer kleuren kunt weergeven terwijl het toch niet meer geheugen kost. Om deze codering toe te passen heeft de MSX 2+ een nieuw kleur systeem, het YJK systeem. De letters J en K staan voor de kleurcomponenten en de letter Y staat voor de helderheid. Dit YJK systeem zullen we eerst uitleggen aan de hand van scherm 12 en daarna komen de schermen 10 en 11 aan bod.

Scherm 12 en het YJK systeem

Scherm 12 werkt, zoals hierboven al vermeld, met het YJK systeem, waarbij de kleurinformatie is verdeeld over vier pixels en de helderheid per pixel wordt vastgesteld. In het videogeheugen ziet dat er als volgt uit:

pixel	**b7 b6 b5 b4 b3**	**b2 b1 b0**
0	Y0	Kl
1	Y1	Kh
2	Y2	Jl
3	Y3	Jh

Tabel 1

Pixel 0 tot en met 3 zijn de vier naast elkaar liggende pixels en b7 tot en met b0 zijn de acht bits binnen één pixel. Zoals in tabel l te zien is, heeft iedere pixel een 5-bits Y-component en zijn de J- en K-componenten verdeeld over de overige twaalf bits; de K-component zit in de laagste 3 bits van pixel 0 en pixel 1 en de J-component zit in de laagste 3 bits van pixel 2 en 3.

Conversie van YJK naar RGB

Hoewel het YJK systeem handig is om de kleuren compact te kunnen opslaan, is het voor ons handiger om te werken met het zogenaamde RGB systeem. Bij het RGB systeem wordt aangegeven hoe helder de drie primaire kleuren zijn, te weten Rood, Groen en Blauw. Dit zijn de drie basiskleuren waar bijvoorbeeld tv's en monitors mee werken en die ook het menselijk oog waarneemt. Gelukkig is het mogelijk om YJK kleuren om te rekenen naar RGB kleuren. Dit kan met de volgende formules:

R = Y + J
G = Y + K
B = 5/4 Y - J/2 - K/4

Bij deze formules moet echter wel nog rekening worden gehouden met de gebruikte rekenmethode; de VDP rekent in 6-bits 2 complement. De getallen J en K zijn beide 6-bits getallen, die in principe overeenkomen met de getallen O tot en met 63 als geen rekening wordt gehouden met 2 complement. Om nu de J en K waarden om te zetten naar 2 complement kunnen de volgende BASIC instructies worden gebruikt:

```
IF J>31 THEN J-J-64 IF K>31 THEN K=K-64
```

Verder dient bij de conversie van YJK naar RGB ook nog rekening gehouden te worden met het feit dat de drie resultaten slechts 5-bits groot zijn, dus R, G en B zijn alle drie slechts 5-bits groot. Dit houdt in dat ze altijd in het gebied 0-31 moeten liggen. De VDP gebruikt hierbij de volgende methode: als één van de waarden kleiner wordt dan 0, dan wordt 0 gekozen en als één van de waarden groter wordt dan 31, dan wordt 31 gekozen. In BASIC kan dit als volgt worden geprogrammeerd:

```
IF R<0 THEN R=0 ELSE IF R>31 THEN R=31
IF G<0 THEN G=0 ELSE IF G>31 THEN G-31
IF B<0 THEN B=0 ELSE IF B>31 THEN B=31
```

Met behulp van deze informatie kan een programma geschreven worden dat scherm 12 plaatjes omzet in scherm 8 plaatjes. Wij maakten zo'n programma en je vindt het als 12-TO-8.BAS op de volgende pagina.

In dit programma wordt per groepjes van vier pixels gewerkt. Voor zo'n groepje van vier pixels worden eerst de J- en K-componenten berekend. Daarna wordt per pixel de Y-component berekend en vervolgens worden hieruit de RGB waarden berekend en weggeschreven in scherm 8 formaat.

```
10 ' Scherm 12 naar scherm 8 conversie
20 '
30 ' Geschreven door Alex Wulms voor MCCM
40 '
50 DEFINTA-Z
60 PRINT "Geef naam van om te zetten scherm 12 plaatje: ";
70 LINE INPUT N$
80 SCREEN 8:SET PAGE 1,1:CLS:BLOAD N$,S
90 '_TURBO ON
100 FOR Y=0 TO 211
110 FOR X=0 TO 255 STEP 4
120 K=(POINT(X,Y) AND 7)+8*(POINT(X+1,Y) AND 7)
130 J=(POINT(X+2,Y) AND 7)+8*(POINT(X+3,Y) AND 7)
140 IF K>31 THEN K=K-64
150 IF J>31 THEN J=J-64
160 FOR T=0 TO 3
170 YC=POINT(X+T,Y)\8
180 R=YC+J:IF R<0 THEN R=0 ELSE IF R>31 THEN R=31
190 G=YC+K:IF G<0 THEN G=0 ELSE IF G>31 THEN G=31
200 B=(5*YC-2*J-K)\4
210 IF B<0 THEN B=0 ELSE IF B>31 THEN B=31
220 C=(G AND &B11100)*8 OR (R AND &B11100) OR B\8
230 PSET (X+T,Y),C
240 NEXT T,X,Y
250 '_TURBO OFF
260 A$=INPUT$(1)
```

Conversie van RGB naar YJK

Ook de conversie in omgekeerde richting kan. Het is mogelijk om plaatjes in RGB formaat om te zetten in het YJK formaat van scherm 12. Hierbij worden de volgende formules gebruikt:

Y = B/2 + R/4 + G/8
J = R - Y
K = G - Y

Ook hierbij moeten de resultaten weer worden gecontroleerd op het bereik. De Y ligt in het gebied 0 - 31 en de J en K liggen in het gebied -32 - 31.
Bij het berekenen van de YJK waarden treedt er echter nog een probleem op; de J en de K worden namelijk per vier pixels berekend, terwijl de waarden voor rood, groen en blauw die hierbij worden gebruikt, per pixel kunnen variëren. Om dit probleem op te lossen, dient per vier pixels de gemiddelde waarde van rood, groen en blauw te worden berekend. Deze gemiddelde waarden worden vervolgens gebruikt om de J en de K voor de vier pixels te berekenen en daarna wordt per pixel de Y bepaald.
Met deze informatie is een programma te maken dat scherm 8 plaatjes converteert naar scherm 12. Een programma voor deze conversie is te vinden in de listing 8-TO-12.BAS op de volgende bladzijde.

```
10 ' Scherm 8 naar scherm 12 conversie
20 '
30 ' Geschreven door Alex Wulms voor MCCM
40 '
50 DEFINT A-Z
60 PRINT "Geef naam van om te zetten scherm 8 plaatje ";
70 LINE INPUT N$
80 SCREEN 12:SET PAGE 1,1:CLS:BLOAD N$,S
90 '_TURBO ON
100 FOR Y=0 TO 211
110 FOR X=0 TO 255 STEP 4
120 '
130 ' bepaal J en K van de vier pixels
140 '
150 R=0:G=0:B=0:FOR T=0 TO 3:P=POINT(X+T,Y)
160 R=R+(P AND &B11100):G=G+(P\8 AND &B11100)
170 B=B+(P*8 AND &B11100):NEXT T
180 R=R\4:G=G\4:B=B\4
190 YC=(4*B+2*R+G)\8:IF YC>31 THEN YC=31
200 J=R-YC:IF J<-32 THEN J=-32 ELSE IF J>31 THEN J=31
210 K=G-YC:IF K<-32 THEN K=-32 ELSE IF K>31 THEN K=31
220 '
230 ' bereken Y per pixel en vul in
240 '
250 FOR T=0 TO 3:P=POINT(X+T,Y):R=P AND &B11100
260 G=P\8 AND &B11100:B=P*8 AND &B11100
270 YC=(4*B+2*R+G)\8:IF YC>31 THEN YC=31
280 PSET (X+T,Y),8*YC:NEXT T
290 '
300 ' zet de J en de K nog in de vier pixels
310 '
320 PSET (X,Y),K AND 7,OR:PSET (X+1,Y),K\8 AND 7,OR
330 PSET (X+2,Y),J AND 7,OR:PSET (X+3,Y),J\8 AND 7,OR
340 NEXT X,Y
350 '_TURBO OFF
360 A$=INPUT$(1)
```

Scherm 10 en 11

Hoewel het mogelijk is om op scherm 12 erg mooie plaatjes weer te geven, heeft scherm 12 ook een groot nadeel; door de gebruikte codering is het onmogelijk om scherpe kleurovergangen binnen een groep van vier pixels te maken. Je kunt bijvoorbeeld niet een rode lijn door een blauwe achtergrond trekken zoals dat op bijvoorbeeld scherm 5 wel kan. Ook in Japan heeft men dit probleem onderkend. De oplossing die hiervoor verzonnen is, heet scherm 10 en 11. Deze schermen, die beiden precies dezelfde indeling hebben, vormen een kruising van scherm 12 met het YJK systeem en scherm 5 met het systeem met het kleurenpalet. Bij deze schermen wordt een bit minder gebruikt voor de Y-component. In plaats van een 5-bits Y-component hebben scherm 10 en 11 een 4-bits Y-component. Het zogenaamde attribute bit dat hierdoor vrij komt, wordt vervolgens als een schakelbit gebruikt:
Als het bit 0 is, wordt de pixel met het YJK systeem weergegeven. Anders dient het systeem met kleurenpalet gebruikt te worden. In dat geval vormt de 4-bits Y-component het kleurnummer, net zoals ook op scherm 5 een 4-bits kleurnummer wordt gebruikt. In tabelvorm ziet dit er als volgt uit:

pixel	**b7 b6 b5 b4**	**b3**	**b2 b1 bO**
0	Y0/C0	A0	KL
1	Y1/C1	A1	Kh
2	Y2/C2	A2	Jl
3	Y3/C3	A3	Jh

Tabel 2

Dit betekent, dat het heel eenvoudig is om een scherm 12 plaatje te converteren naar een scherm 10 of 11 plaatje; het is voldoende om bit 3 van alle pixels te resetten zodat overal het YJK systeem wordt gebruikt. Dit kan in BASIC als volgt met het commando LINE:

```
LINE (0,0)-(255,211),&B11110111,BE,AND
```

Deze functie gebruikt de logische AND functie om het attribuut bit te resetten van alle pixels. Nadat een plaatje op zo'n manier is omgezet naar scherm 10 of 11 formaat, kan er overheen worden getekend met de normale commando's in BASIC.

Verschil tussen scherm 10 en 11

Hoewel scherm 10 en scherm 11 dezelfde indeling hebben, maakt BASIC toch wel degelijk een onderscheid tussen deze twee schermen. Dit heeft men gedaan om het programmeren eenvoudiger te maken. In scherm 11 kunnen alle bits van de pixel worden veranderd. Voor het kleurnummer worden dan de waarden 0 tot en met 255 gebruikt. De conversie van scherm 12 naar scherm 10 of 11 kan bijvoorbeeld in scherm 11 worden uitgevoerd. Als je echter met behulp van het palet over het YJK plaatje wilt gaan tekenen, is dit best wel lastig; je moet er dan voor zorgen dat alleen de vier hoogste bits van de pixel worden veranderd en dat het attribute bit gelijk wordt gemaakt aan l, zodat het paletsysteem wordt gebruikt. Om bijvoorbeeld een blauw lijntje—met kleur 4—te tekenen van coördinaat (5,5) naar coördinaat (90,70) kunnen

de volgende commando's gebruikt worden in scherm 11:

```
75 LINE (5,5)-(90,70),7,,AND:' reset Color/Y
80 LINE (5,5)-(90,70),4*16+8,,OR:' teken lijn in kleur 4
```

Dit tekenen met de paletkleuren kan echter een stuk eenvoudiger in scherm 10 worden gedaan. In scherm 10 kun je namelijk net zo tekenen als in scherm 5. Hetzelfde lijntje trekken in scherm 10 gaat dan als volgt:

```
LINE (5,5)-(90,70),4
```

Alle commando's in BASIC hebben op scherm 10 ditzelfde effect, ook commando's, zoals CIRCLE, waarbij je de methode van scherm 11 met twee keer tekenen niet kunt toepassen.
Kort samengevat kun je in scherm 11 de pixels volledig veranderen zodat de YJK componenten aangepast worden, terwijl je in scherm 10 over een plaatje heen kunt tekenen alsof je gewoon in scherm 5 zit te werken. Het is trouwens mogelijk om willekeurig tussen scherm 10,11 en 12 te wisselen zonder dat iedere keer BASIC het scherm wist zoals bij normale schermwisselingen gebeurt!

Schermen selecteren in assembly

Hoewel BASIC een aantal commando's biedt om de MSX 2+ schermen te selecteren, zul je vanuit assembly iets meer werk moeten verrichten om deze schermen te zien te krijgen. De BIOS routine om een scherm te selecteren gaat niet
verder dan scherm 8! Om toch scherm 10,11 of 12 te krijgen dien je dan ook via de BIOS scherm 8 te selecteren en vervolgens moet je zelf de VDP instellen op het YJK systeem door de goede waarden naar de MSX 2+ VDP registers te schrijven. De 2+ schermen kun je instellen met behulp van twee bits van VDP register 25:

Bit 3 (YJK) = 0

Werk in de MSX 2 mode, geef dus scherm 8 weer

Bit 3 (YJK) = 1

Werk in de MSX 2+ mode, gebruik dus het YJK systeem.

Bit 4 (YAE) = 0

Gebruik het volledige YJK systeem, geef dus scherm 12 weer.

Bit 4 (YAE) = 1

Gebruik het YJK en het paletsysteem door elkaar, geef dus scherm 10 en 11
weer.
De andere bits van register 25 hebben op de MSX 2+ ook nog een betekenis. Deze mogen dan ook niet zomaar worden veranderd. Gelukkig houdt de BIOS een kopie bij van register 25 in het MSX systeemgebied. Deze kopie staat op adres #FFFA. Het instellen van de 2+ schermen kan zoals weergegeven in SETSCR.GEN.

SETSCR.GEN

```
; ******************************
; * Stel een scherm (MSX 2+) scherm in
; * in: A = gewenste scherm mode
chgmod:           equ   #5f

setscreen:  cp    9
            jp    c,chgmod    ; MSX 2 scherm : gebruik de BIOS
            ret   z           ; scherm 9 bestaat (hier) niet
            cp    13
            ret   nc          ; 13 en hoger bestaat niet
            push  af          ; onthoudt de gewenste mode
            ld    a,8
            call  chgmod            ; pak scherm 8
            ld    hl,#fffa
            ld    a,(hl)
            and   %11100111   ; reset AYE (bit 4) en YJK (bit 3)
            or    %00001000   ; set YJK voor scherm 10,11 en 12
            ld    b,a         ; B = register 25 setting
            pop   af
            cp    12
            jr    z,setscr2   ; scherm 12 => B is goed
            set   4,b         ; scherm 10 of 11 => set AYE
setscr2:    ld    (hl),a            ; sla nieuwe waarde op
            di
            out   (#99),a           ; opm: bij de 2+ ligt het VDP
            ld    a,25+128    ;      I/O addr officieel vast!
            out   (#99),a           ; stel register 25 in
            ei
            ret
```

Sprites op de MSX 2+ schermen

De sprites op scherm 10,11 en 12 werken hetzelfde als op scherm 5 en 7. Het is dus mogelijk om per spritelijn twee kleuren te gebruiken; de spritekleuren worden opgehaald uit het instelbare kleurenpalet. Dit in tegenstelling tot de sprites op scherm 8, waar de kleuren van de sprites worden bepaald met een vast, in de VDP ingebakken, palet. Normaal staan de spritetabellen op de volgende posities in het videogeheugen:

Tabelnaam	Normale adressen	lengte
Sprite$ tabel	#F000-#F7FF	2048 bytes
Sprite-kleurtabel	#F800-#F9FF	512 bytes
Sprite-infotabel	#FA00-#FA7F	1 28 bytes

Tabel 3

Dit kan echter veranderd worden door de goede waarden naar de VDP registers te schrijven waar de sprite adressen in staan. Dit zijn de zelfde VDP registers als op de MSX 2 VDP, namelijk VDP (5), VDP(6) en VDP(l2].

De extra VDP registers

De MSX2+ VDP heeft drie extra registers, genummerd 25, 26 en 27. Ze zijn vanuit BASIC met VDP (2 6) - VDP (2 8) toegankelijk. Om het goede VDP nummer van een register te bepalen dien je dus één bij het register nummer op te tellen. De registers zijn als volgt ingedeeld:

reg	b7	b6	b5	b4	b3	b2	b1	bO
25	-	CMD	VDS	YAE	YJK	WTE	MSK	SP2
26	-	-	H08	H07	H06	H05	H04	H03
27		-	-		-	H02	H01	H00

De afkortingen bij de bits hebben de volgende betekenis:

CMD: command function
Dit bit bepaalt hoe de VDP commando's werken. Als CMD gereset is, werken de commando's alleen op de schermen 5 tot en met 12. In het andere geval werken ze op alle schermen. Op scherm 5 tot en met 12 werken ze dan normaal, terwijl het adresserings systeem van scherm 8 op de overige schermen wordt gebruikt. Dit houdt in dat de coördinaten op deze schermen naar de VDP gestuurd moeten worden alsof het om scherm 8 gaat.

VDS: VRAM data select
Dit bit bepaalt de functie van pin 8 van de VDP. Als VDS gereset is, komt het kloksignaal van 3.58 MHz voor de CPU op pin 8 te staan. Anders wordt het zogenaamde VDS signaal erop gezet.
Dit signaal is laag als de VDP het VRAM aanspreekt voor de weergave van het scherm. Anders is het VDS signaal hoog. Daar bij de meeste MSX2+ computers de

Z80 zijn kloksignaal van pin 8 van de VDP krijgt, is het aan te bevelen om dit bit altijd op 0 te laten staan.

YAE en YJK
Deze staan al genoemd in de paragraaf 'Schermen selecteren in assembly'.

WTE: wait function
Dit bit bepaalt of de wait functie van de VDP werkt. Als WTE gereset is, werkt de wait functie niet. In het andere geval is de wait functie ingeschakeld. Als de CPU het VRAM dan wil aanspreken, genereert de VDP automatisch waitstates totdat de VDP klaar is om de CPU bij het VRAM te laten. Door hier gebruik van te maken, hoeven programma's die de VDP commando's gebruiken niet meer bit 7 van status register 2 te controleren.

MSK: mask bit

Met dit bit kan de mask functie van de VDP worden ingeschakeld. Indien MSK gereset is, staat de mask functie uit. In het andere geval staat ze aan. Dit houdt in dat de acht meest linkse pixels gemaskerd worden met de randkleur. Dit is nodig indien er horizontaal wordt ge-scrolled met register 26 en 27. De uitvoer van de eerste acht pixels wordt namelijk niet gegarandeerd indien register 27 ongelijk aan nul is. Merk op dat op scherm 6 en 7 de eerste zestien pixels worden gemaskerd in plaats van de eerste acht.

SP2: screen size bit
Hiermee wordt de schermgrootte ingesteld voor de horizontale scroll. Indien SP2 gereset is, is het scherm één pagina groot. Alles wat links het scherm uit loopt, komt rechts het scherm weer in. In het andere geval is het scherm twee pagina's groot. Indien nu de eerste pagina links het scherm uitloopt, komt de tweede pagina rechts het scherm in.

H08-H00: horizontal scroll value
De horizontale scrollwaarde wordt met deze bits ingesteld. Er wordt op scherm 6 en 7 met twee pixels per eenheid ge-scrolled en op de overige schermen met één pixel per eenheid. De horizontale scrollwaarde moet verdeeld worden over twee registers. Bit H08-H03 komen hierbij in register 26 terecht en bepalen de verschuiving van het scherm naar links. Bit H02-H00 daarentegen komen in register 27 terecht en bepalen de verschuiving van het scherm naar rechts. Dit maakt het instellen van de scrollwaarde een beetje ingewikkeld. Om bijvoorbeeld het scherm n pixels naar links te schuiven onder BASIC, kunnen de volgende commando's worden gebruikt:

```
VDP(27)=(N+7)\8
VDP(28)=-N AND 7
```

Dit komt overeen met het BASIC commando SET \SCROLL N. In andere programmeertalen, zoals assembler, is het SET SCROLL commando echter niet voorhanden; dan dien je dus zelf de scrollwaarde in te stellen met de hierboven vermelde methode.

Scherm 9 Het ontbrekende scherm

Tussen het hoogste MSX 2 scherm— scherm 8—en het laagste MSX 2+ scherm— scherm 10—ontbreekt een scherm, namelijk scherm 9. In de loop der tijd hebben de meest wilde verhalen over dit scherm de ronde gedaan. Het leukste verhaal was wel dat Konami het zou hebben opgekocht voor hun eigen spellen. Dit is—helaas (?)— niet het geval.
Scherm 9 wordt namelijk in Korea gebruikt als scherm voor de daar gebruikte karakters. Het is gewoon één van de grafische MSX 2 schermen, dat wordt aangestuurd alsof het een tekstscherm is, maar dan met de Koreaanse karakters erop. Waarschijnlijk scherm 7, maar dat weet ik niet zeker. Dit is te vergelijken met de Kanji schermen van de Japanse MSX 2+ modellen en de MSX turbo R.
Met dank aan Bernard Lamers voor de informatie over scherm 9.