De nieuwe schermen nader bekeken TECHNISCHE DETAILS OVER FRAAIE SCHERMEN

MSX Computer Magazine nummer 47 - juni 1991

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

Nu de MSX 2+ ingeburgerd begint de raken, is na veel experimenteren duidelijk geworden wat je nou eigenlijk zelf kunt doen met die drie nieuwe schermen: bar weinig.
Toch is het nuttig om te weten hoe de nieuwe VDP, de 9958, met de nieuwe schermen omgaat. Er worden een aantal leuke en leerzame truuks uit de kast gehaald om uit 54 kB videoRAM toch een afbeelding te toveren met meer dan 19000 kleuren!
Ook op de redactie hadden we lange tijd moeite om de nieuwe schermmodes naar onze pijpen te laten dansen. In het begin werd er druk met de MSX 2+ geëxperimenteerd. De MSX 2+ bleek inderdaad een machine te zijn die verschrikkelijk veel in zijn mars heeft, al was het niet eenvoudig alle mogelijkheden te benutten.
Her en der waren redacteuren natuurlijk wel bezig de nieuwe schermen te onderzoeken, maar voor iedereen bleven er wel een paar details onduidelijk. Tijd dus om de koppen eens bij elkaar te steken. Dit artikel is het resultaat, we leggen de werking van de schermen van de MSX 2+ eens tot in de details uit. In de hoop dat u straks zelf plaatjes kunt gaan maken en/of bewerken. Nederlandse software voor de 2+ hebben we nog veel te weinig gezien!

19268 kleuren, maar hoe?

De makers van de MSX 2+ wilden meer kleuren op het scherm krijgen, maar de 128 kB videoRAM moest nog steeds toereikend zijn voor twee schermpagina's, met andere woorden: de nieuwe scherm-mode moest veel meer kleuren kunnen tonen dan het al bestaande screen 8, maar mocht niet meer ruimte innemen.
Dit lijkt misschien een onmogelijke doelstelling: meer kleuren en evenveel geheugen. Toch is men er in geslaagd de 19268 kleuren in 54 kB te 'proppen'. Hierdoor worden er aan de mogelijkheden wel enige beperkingen opgelegd.
In schermmode 8 is het zo geregeld dat elke byte correspondeert met een pixel op het scherm. Zo'n byte kan 256 verschillende - binaire - getallen bevatten. De acht bits van deze byte zijn verdeeld over de basiskleuren, rood, groen en blauw. Drie voor rood, drie voor groen en twee voor blauw. Schematisch ziet de opbouw van een screen 8 byte er uit zoals getekend in figuur l.
De computer mengt uit de verschillende waarden van de basiskleuren de kleur van het pixel. Op die manier zijn er dus 256 kleuren beschikbaar. Het beschikbare videogeheugen van 256 * 212 = 54272 bytes is op deze manier optimaal benut. Althans, zo lijkt het.

Toch meer...

Maar met wat gegoochel blijkt het toch mogelijk meer uit de enen-en-nullenbrij te halen dan je op het eerste gezicht zou denken. De oplossing die bedacht werd was dat men de pixels niet meer direkt met een byte liet corresponderen, maar dat men een rijtje van vier bytes overeen liet komen met een rijtje van vier pixels. De 32 bits waaruit zo'n rijtje van vier bytes bestaat zijn op een vrij ingewikkelde, maar zeer efficiënte manier verdeeld over de pixels.
Van elke byte zijn drie bits gereserveerd voor het samenstellen van een basiskleur. Met deze twaalf bits is het mogelijk 4096 basiskleuren te maken. De vijf overblijvende bits in elke, byte bevatten de helderheid van de basiskleur voor het bij die byte behorende pixel. Er zijn 32 verschillende helderheden mogelijk, de waarde O levert een donkere kleur op, de waarde 31 een heldere. Deze helderheden zijn voor elk pixel onafhankelijk in te stellen.
Theoretisch zijn er met 4096 basiskleuren en 32 helderheden 131072 verschillende kleuren te mengen. Een aantal van deze kleuren blijken echter hetzelfde te zijn, waardoor er van die 131072 kleuren dan ook 'slechts' 19268 werkelijk verschillende overblijven.

De basiskleur

De opbouw van de basiskleur is, in tegenstelling tot de opbouw van de kleur van een pixel in screen 8, niet echt simpel. Vijf van de twaalf bits zijn gereserveerd voor groen, vijf anderen voor rood. Met de overige twee is een truukje uitgehaald. Ze geven niet zondermeer de hoeveelheid blauw aan.
Deze twee bits geven aan of de hoeveelheid blauw de inverse moet zijn van de hoeveelheid groen, de hoeveelheid rood of de hoeveelheid groen én rood. Wanneer beide bits O zijn, zit er geen blauwe component in de busiskleur.

Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Kleur:	Groen	Groen	Groen	Rood	Rood	Rood	Blauw	Blauw

Figuur l: Indeling van de bytes in screen 8

Bit:	7	6	5	4	3	2	1	0
1ste Byte:	Helderheid	Helderheid	Helderheid	Helderheid	Helderheid	Groen	Groen	Groen
2de Byte:	Helderheid	Helderheid	Helderheid	Helderheid	Helderheid	Blauw	Groen	Groen
3de Byte:	Helderheid	Helderheid	Helderheid	Helderheid	Helderheid	Rood	Rood	Rood
4de byte:	Helderheid	Helderheid	Helderheid	Helderheid	Helderheid	Blauw	Rood	Rood

Figuur 2: Indeling van de bytes in screen 12

Nu weten we waarvoor de verschillende bits gebruikt worden, maar nog steeds niet hoe de bytes dan precies ingedeeld zijn. En vooral die indeling is van groot belang voor mensen die zelf aan de slag willen. In figuur 2 is een overzicht te zien van een

groepje van vier bytes. Ter illustratie een paar voorbeeldjes. Om vier puur rode pixels links boven op het scherm zichtbaar te maken zijn de volgende vier VPOKE instructies voldoende:

```
VPOKE     0,&b00000000
VPOKE     l,&b00000000
VPOKE     2,&b00000111
VPOKE     3,&b00000011
```

In de eerste vijf bits van elke byte kan nog gevarieerd worden met de helderheden van de afzonderlijke pixels. Het pure blauw, wat zo lastig samengesteld lijkt te worden, blijkt ook vrij simpel te zijn:

```
VPOKE     0,&b00000000
VPOKE     1,&b00000100
VPOKE     2,&b00000000
VPOKE     3,&b00000100
```

De blauwe kleur wordt gevormd door de 2 bitjes die in byte l en 3 op de waarde l gezet worden. Doordat daarbij zowel de rode als de groene component uitgeschakeld worden, ontstaat de blauwe kleur.

Screen 11 en 10

Screen 11 zit op ongeveer dezelfde manier in elkaar als screen 12, alleen zijn er maar 4 bits gereserveerd voor de helderheden van de afzonderlijke pixels. Er zijn op screen 11 dus maar 16 variaties op de basiskleur mogelijk. Het bit dat op deze manier vrijkomt - bit 3 van elke byte -heeft een speciale functie gekregen: dat bit geeft aan of het corresponderende pixel wel of niet meedoet met de ingestelde basiskleur. Is het bit O, dan is er niets aan de hand en wordt de kleur bepaald door de basiskleur en de vier bits voor de helderheid.
Anders wordt het als bit op l staat. Dan hebben de bits voor de basiskleur alleen nog maar betekenis voor de drie andere pixels. De kleur van het pixel dat bij deze byte hoort wordt dan bepaald door het getal dat in de vier bits staat die in screen 12 voor de helderheid gereserveerd waren. Dit getal correspondeert met één van de zestien kleuren uit het kleurenpalet dat we op de MSX 2 al kenden. Hierdoor is het in screen 11 mogelijk om tekst en andere figuren 'over een digitalisatie heen' af te beelden. Dit gaat natuurlijk ten koste van een aantal kleuren, maar er blijven er altijd nog 12499 over. Net als op screen 12 leveren een aantal van de mogelijke combinaties dezelfde kleur op, waardoor het theoretische aantal van 4096 * 16 = 65536 kleuren niet gehaald wordt. Screen 10 is qua opbouw exact gelijk aan screen 11, het enige verschil zit hem in de manier waarop Basic 3.00 ermee omgaat.
Een van de grootste tegenvallers van de MSX 2+ was het feit dat Basic 3.00 niet veel met de nieuwe schermen kan doen, sterker nog, het lijkt zelfs of Basic niet goed werkt in de nieuwe schermen. Maar wat doet die Basic nou precies?

Screen 12 en 11

Als je op dit scherm lijnen of tekst plaatst, verspringen direct hele rijtjes van 4 pixels van kleur, op de meest vreemde manieren. De oorzaak hiervan ligt in het feit dat

Basic screen 12 net zo aanpakt als screen 8, er wordt gewoon een getal van O tot 255 in een byte geVPOKEd. Maar omdat de kleuren in screen 12 nu eenmaal anders samengesteld worden dan in screen 8 wordt het een zooitje op het scherm. Screen 12 is dus eigenlijk alleen maar geschikt voor digitalisaties of het maken van tekeningen met behulp van een tekenprogramma; maar zelfs het laatste is zeer moeilijk, omdat je dan nog steeds zit met de rijtjes van vier pixels. Een programma als GraphSaurus ondersteunt een groot deel van de tekenopdrachten dan ook niet in screen 12. Voor screen 11 geldt het bovenstaande ook: het scherm wordt het zelfde behandeld als screen 8, zodat ook dit scherm bij de normale Basic-tekenop-drachten dus een puinhoop wordt. Op screen 11 en 12 moeten we het in Basic
dus vooral hebben van welgemikte VPOKE's

Screen 10

Dit is eigenlijk het enige scherm dat vanuit Basic enigszins zinnig aanstuurbaar is. Dit komt doordat Basic in dit scherm het kleurenpalet van 16 kleuren aanhoudt. Zoals bij de werking van screen 11 en 10 beschreven is, kan er op scherm 10 - door het juiste bit op l te zetten - ook met de paletkleuren gewerkt worden. Met andere woorden: Basic 3.00 zet bij alle bewerkingen op screen 10 bit 3 op l en de paletkleurop de plaats waar normaal de helderheid voor dat pixel staat. Daarom lijkt dit scherm op het eerste gezicht sterk op screen 5.
Het is mogelijk om met behoud van de scherminhoud tussen screen 10 en 11 te schakelen, omdat de opbouw technisch gezien volkomen identiek is. Het enige verschil zit hem in de manier waarop Basic ermee omgaat. Zo zou je bijvoorbeeld in screen 11 dingen kunnen tekenen - gekleurde blokken bijvoorbeeld - met één of meerdere van de 12499 aldaar mogelijke kleuren en vervolgens naar screen 10 kunnen gaan om er in één of meer van de 16 paletkleuren een andere grafische voorstelling overheen te zetten, bijvoorbeeld een tekst, of een aantal cirkels.

```
10 ' SCR12-11.BAS
20 '
30 ' MSX Computer Magazine
40 ' (Listing 1)
50 ' door: David Boelee
60 '
70 INPUT "Welk (SCREEN 12) plaatje wil je omzetten ";A$
80 PRINT "Onder welke naam moet ik ";A$;" als screen 11 plaatje saven"
90 INPUT "(Return is niet saven) ";B$
100 SCREEN 12
110 BLOAD A$,S
120 LINE (0,0)-(256,212),&B11110111,BF,AND
130 IF B$="" THEN END
140 BSAVE B$,0,&HD3FF,S
150 PRINT "Klaar !"
```

Listing l

Enkele programmavoorbeeldjes

Na het lezen van de bovenstaande tekst zal duidelijk zijn, dat elk screen 12 plaatje ook op screen 11 getoond kan worden. Er geldt echter wel een voorwaarde: van alle bytes moet bit 3 op nul staan, anders wordt op screen 11 met de paletkleuren gewerkt en niet alleen maar met de basiskleuren zoals in screen 12.
Listing l laat zien hoe dat bit heel eenvoudig op O gezet kan worden, zodat elk screen

12 beeld geschikt gemaakt kan worden voor screen 11. Het enige nadeel is dat de kleuren soms iets minder vloeiend verlopen, doordat er op screen 11 zestien helderheden minder beschikbaar zijn. Maar vaak is dat ook nauwelijks te zien.
Het programma spreekt waarschijnlijk voor zich, alleen regel 90 zou even aandacht kunnen krijgen: hier wordt een gevuld blok over de tekening heen getekend met de kleur &b 11110111. Door gebruik te maken van de logische operatie 'AND' wordt bit 3 van elke byte op O gezet. Wanneer een andere kleur, bijvoorbeeld &b00000111, gebruikt wordt, worden alle helderheids-bits op O gezet. De
tekening bestaat dan alleen nog maar uit basiskleuren.

```
10 ' MENGCLR.BAS
20 '
30 ' MSX Computer Magazine
40 ' (Listing 2)
50 ' door: David Boelee
60 '
70 SCREEN 12: COLOR 0,0,0: CLS
80 VR=0: VG=33*256 :VB=66*256 :VT=132*256 :VD=99*256
90 FOR F=0 TO 3
100 FOR N=0 TO 7
110 FOR Q=0 TO 62 STEP 4
120  VPOKE Q+VR+2,N: VPOKE Q+VR+3,F
130  VPOKE Q+VG,N  : VPOKE Q+VG+1,F
140  VPOKE Q+VB,N  : VPOKE Q+VB+1,F+4
150  VPOKE Q+VD+2,N: VPOKE Q+VD+3,F+4
160  VPOKE Q+VT,N  : VPOKE Q+VT+1,F+4
170  VPOKE Q+VT+2,N: VPOKE Q+VT+3,F+4
180 NEXT Q
190 VR=VR+256: VG=VG+256: VB=VB+256: VT=VT+256: VD=VD+256: NEXT N,F
200 I=0: FOR F=0 TO 31: C=I XOR F: LINE (F*2,0)-(64,31),C*8,BF,XOR: I=F: NEXT F
210 I=0: FOR F=0 TO 31: C=I XOR F: LINE (F*2,33)-(64,64),C*8,BF,XOR: I=F: NEXT F
220 I=0: FOR F=0 TO 31: C=I XOR F: LINE (F*2,66)-(64,97),C*8,BF,XOR: I=F: NEXT F
230 I=0: FOR F=0 TO 31: C=I XOR F: LINE (F*2,99)-(64,130),C*8,BF,XOR: I=F: NEXT
F
240 I=0: FOR F=0 TO 31: C=I XOR F: LINE (F*2,132)-(64,163),C*8,BF,XOR: I=F: NEXT
 F
250 LINE INPUT A$
260 COLOR 15,4,4
270 END
```

Mengkleuren

Listing 2 behoeft iets meer aandacht. Dit programma laat zien dat er met een VPOKEje hier en een 'logisch operatietje' daar toch een vloeiend kleurverloop gerealiseerd kan worden. Het programma tekent de drie hoofdkleuren in alle kleurnuances die met de basiskleur verkregen kunnen worden op het scherm.
Al die nuances zijn op hun beurt weer te zien in 32 helderheden. Blauw wordt weergegeven in drie blokken: eerst als de inverse van rood, dan als de inverse van groen en vervolgens als de inverse van rood en groen tesamen. Op deze manier worden er theoretisch 5 * 32 * 32 = 5120 kleuren op het scherm gezet, waarvan er mogelijk een aantal hetzelfde zijn.
In de eerste regels worden alle variabelen geïnitialiseerd en de schermkleur wordt op nul gezet, zodat het scherm echt helemaal zwart is. Het volgende gedeelte van het programma lijkt vrij ingewikkeld, laat u echter niet in de luren leggen door de vele variabelen, VPOKEs en FOR-NEXT lussen. Het zit zo: zowel de vijf bits voor rood als de vijf voor groen zijn verdeeld over twee bytes, drie in de ene byte, en twee in de andere. In het gedeelte van twee bits kan maximaal een drie
- binair geschreven als 11 - staan, en in het drie bits grote deel maximaal een 7
- binair voorgesteld door 111. De FOR-NEXT lussen zorgen ervoor dat telkens 64

pixels worden geVPOKEed, in de ene byte waarde N - lopend van O tot 7 - en in de andere byte waarde F, die van O tot 3 loopt. Zodoende worden de 32 mogelijke (pure) roodtinten geVPOKEd. Hetzelfde geldt voor groen.
Bij blauw ten slotte, worden dezelfde waarden geVPOKEd als voor rood en groen, alleen door 4 op te tellen bij de VPOKE waarde wordt bit 2 van de bytes waar het blauw wordt opgeslagen op l gezet, zodat de hoeveelheid blauw de inverse wordt van de hoeveelheid rood en/of groen. Eerst wordt de inverse van de hoeveelheid rood gebruikt, vervolgens die van groen en als laatste die van rood en groen samen.
Het laatste programmaonderdeel vult de 32 verschillende helderheden in in alle blokken. Dit gebeurt met behulp van de logische operatie XOR. Dergelijke functies zijn vaak erg handig als er een beperkt aantal bits binnen een byte veranderd moet worden.

Andere opdrachten

'Allemaal wel heel erg leuk, maar wat kan ik met de andere speciale opdrachten van de 2+?', zult U zich wellicht afvragen De echt nieuwe Basic-opdrachten staan in tabel l, samen met hun syntax. De opdrachten die wat meer aandacht behoeven zullen we nu wat uitgebreider bekijken.
Een aantal opdrachten die al op de MSX 2 aanwezig waren, zijn voor de 2+ uitgebreid, zodat ze ook werken op de nieuwe schennen. We zullen deze echter niet allemaal behandelen, vooral omdat de meeste niet écht aangepast zijn. Het komt er in het kort op neer dat alle grafische opdrachten ook op de nieuwe schermen werken, maar vraag in sommige gevallen niet hoe...

SET SCROLL

SET SCROLL is een van de weinige MSX 2+ opdrachten waarvan de werking van het begin af aan eigenlijk al duidelijk was. We zullen dit commando dan ook slechts kort behandelen. Met de eerste twee parameters kan men het scherm horizontaal of vertikaal laten scrollen. De derde parameter geeft aan of de zijkant van het scherm stilgezet moet worden (1) of niet (0). Stilzetten is over het algemeen een fraaier gezicht. De laatste parameter geeft aan of de pagina's 'achter' de zichtbare pagina mee moeten scrollen (l) of niet (0)

Kanji-modes

We weten bijna allemaal wel dat we verschillende Kanji-modes kunnen aanroepen, maar wat je ermee kan doen en wat het verschil is tussen de verschillende modes is minder duidelijk. In de Kanji-modes wordt het mogelijk de gewone PRINT en LOCATE opdrachten te gebruiken in elk scherm, ook de grafische! Het is dan dus niet meer nodig
een bestand te openen om het grafische scherm te kunnen bereiken. Er zijn vier verschillende Kanji-modes, tabel 2 geeft een overzicht van de verschillen.
In screen 11 en 12 treden ook bij het gebruik van de Kanji-modes color-spill effecten op. Het ontstaan hiervan is in het bovenstaande verhaal al uitgelegd. In de Kanji-modes is het in alle schermen mogelijk alle toegestane kleuren te gebruiken. Ook in 'screen O' is hierdoor een meerkleurige tekst mogelijk! Een aantal andere opdrachten die met de Kanji-modes te maken hebben zijn in de tabel te vinden.
Listing 3 is een klein voorbeeldje voor de Kanj i-mode en voor het SET SCROLL

commando. Speel maar eens met verschillende waarden voor SET SCROLL en probeer eens wat andere Kanji-modes. Het programma heeft eigenlijk weinig
SET SCROLL <horizontaal>,<vertikaal>,<rand>,<achterl. pagina> <horizontaal> mag een waarde hebben van O tot 511, <vertikaal> mag een waarde hebben van O tot 255, <rand> en <achterl. pagina> mogen O of l zijn. Bij deze opdracht mag, net als bij onder andere COLOR met komma's worden gewerkt als een parameter onveranderd moet blijven, op voorwaarde dat er minimaal één getal in de uitdrukking moet staan. Met deze opdracht kan men het scherm laten scrollen.

CALL KANJI <modenummer>

<modenummer> kan 0 tot en met 3 bedragen.

Zet het scherm in de gewenste

Kanji-mode.

CALL CLS
Geeft een schoon scherm in de Kanji-modes. Gewoon 'CLS' werkt in de Kanji-modes niet.

CALL ANK
Ga uit de Kanji-mode.

CALL PALLETTE (<kleurnummer>,<rood>,<groen>,<blauw>) <kleurnummer> mag lopen van O tot het toegestane aantal kleuren in het scherm waarin gewerkt wordt, <rood>, <groen> en <blauw> mogen de waarden O tot en met 7 bevatten. Deze opdracht past de intensiteiten van rood, groen en blauw aan van paletnummer <kleurnummer>

SET SCROLL, CALL KANJI en _PALLETTE worden in de tekst uitgebreid besproken.

Mode 0 1 2 3 Eigenschappen Grote letters. In deze mode worden de karakters net zo weergegeven als op screen 1 , dus over de volle breedte (8 pixels) Grote letters. In deze mode worden de karakters net zo weergegeven als op screen 0, dus in de breedte vallen de laatste 2 pixels weg. Idem. als 0, maar dan met kleine letters. Idem. als 1, maar dan met kleine letters.

```
10 ' SCROLL.BAS
20 '
30 ' MSX Computer Magazine
40 ' (Listing 3)
50 ' door: David Boelee
60 '
70 SCREEN 5
80 CALL KANJI0: COLOR 15,1,1: CALL CLS
90 COPY (0,0)-(256,50) TO (0,211)
100 LOCATE 0,5: PRINT "Diagonale scroll..."
110 COLOR 4: LOCATE 2,6: PRINT " in Kanjimode 0"
120 FOR F=0 TO 255: SET SCROLL F,F,1,1: NEXT F
130 GOTO 120
```

Op regel 70 na: de COPY-instructie kopieert een stuk scherm naar het stuk videoRAM dat normaal onzichtbaar is, maar met de scrollroutines wel zichtbaar kan worden.
In dit gedeelte van het VRAM staat allerlei informatie, die er op het scherm niet echt ordelijk uitziet. Deze informatie is niet altijd nodig en kan in dit geval weggewerkt worden door middel van een COPY-instructie. Maak dit stuk VRAM echter niet leeg als U sprites heeft gedefinieerd!
Met C ALL PALETTE is het daar waar met paletkleuren wordt gewerkt mogelijk de intensiteiten van rood, groen en blauw van een kleur te veranderen. Dit commando werkt op elk scherm, alleen wordt op de schermen 8 en 12 alleen de randkleur beïnvloed.
De Kanji-modes en CALL PALETTE zitten overigens ook in MSXDOS 2 ingebouwd.

Toch meer

Het zal duidelijk zijn dat er in Basic toch wel meer met de kleuren te doen is dan je op het eerste gezicht zou denken. Het probleem is dan alleen dat je goed moet weten hoe de kleuren nou precies opgeslagen liggen in het videogeheugen. Deze kennis hopen we in dit artikel gegeven te hebben. Iedereen kan nu zelf aan de slag!
We houden ons hier op de redactie natuurlijk altijd aanbevolen voor de resultaten. We zijn namelijk van mening dat er met de 2+ schermen meer moet kunnen dat wat we tot nog toe gezien hebben. Stuur de resultaten van uw werk dus eens op, wie weet kunnen we er iets mee doen.