LETTERS op het grafisch tv-display

D. M. DE BOER

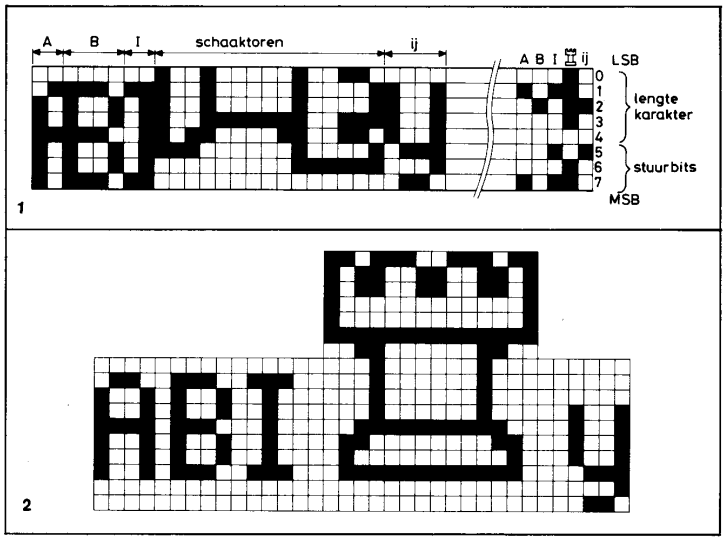
In de vorige artikelenreeks over het zelfbouw grafisch TV-display werd uitgebreid aandacht besteed aan de hardware van dit display. Ook werden twee belangrijke routines besproken, nl. het aan of uitzetten van één van de 65536 aanwezige stippen en het wissen van het beeld. Maar... een display is geen display als er geen teksten op kunnen worden weergegeven. Daarom volgt in dit artikel wellicht het belangrijkste stukje software dat voor dit display werd geschreven: Het 'tekenen' van letters, met ongekende flexibiliteit.

Hardware of Software?

Letters op het grafisch display kunnen op verschillende manieren worden gerealiseerd. Allereerst de 'klassieke' methode, een karaktergenerator met een speciaal stuk RAM waarin de karakters in ASCII-vorm zijn opgeslagen. Het voordeel van deze methode is dat het display zeer snel is, en dat gemakkelijk letters van het scherm terug gelezen kunnen worden. Nadeel van de 'hardware methode' is, dat er componenten moeten worden aangeschaft en dat de gebruiker met handen en voeten gebonden is aan de eigenschappen van de schakeling.
De software oplossing geeft een ongekende flexibiliteit voor wat betreft lettervorm, letterhoogte, letterbreedte, letterafstand, letterplaats enz. Bovendien is de oplossing goedkoper (de hele 'schakeling' past in twee EPROM's 2708). Het nadeel van software, de traagheid, lijkt mee te vallen. Met de 1 MHz 6502 kunnen ongeveer 125 karakters per seconde worden getekend. De 6800 is iets trager, de 8080 haalt ongeveer 70 karakters per seconde. We gaan er hierbij vanuit dat er voor een karakter 48 punten moeten worden getekend (8 x 6 matrix). In dit geval wordt de letter wit getekend, en de achtergrond zwart, of omgekeerd. Wanneer we het tekenen van de achtergrond achterwege laten wordt het geheel nog iets sneller. Eén nadeel blijft, het teruglezen van een karakter van het scherm is niet eenvoudig. Dit probleem is te omzeilen door de ASCII-code van de letters op het scherm in het computergeheugen op te slaan.
Vaak is het helemaal niet nodig om letters terug te lezen, de kopie is dan natuurlijk overbodig. De software oplossing biedt echter nog een aantal zeer aantrekkelijke mogelijkheden. Er is nl. automatisch een soort (zeer flexibele) programmeerbare karaktergenerator ontstaan. De karakterset kan op elke plaats in het RAM (of EPROM) staan.
De benodigde karakters voor een schaakprogramma kunnen bij het schaakprogramma worden geschreven, bij kaartspelletjes kan een kaartsymbolengenerator worden gebruikt enz. Dus voor elke specifieke toepassing kunnen de bij de toepassing behorende symbolen gewoon in het computergeheugen worden gezet. Natuurlijk kunnen ook verschillende karaktersets gelijktijdig in het geheugen staan, terwijl het programma bepaalt welke karakterset moet worden gebruikt. Doordat alle karakters slechts éénmaal op het scherm worden getekend, en daarna onveranderd blijven staan, kunnen meerdere karaktersets ook dóór elkaar worden gebruikt. Grafische karakters zijn natuurlijk niet nodig omdat dit display volledige grafische mogelijkheden bezit.
Al deze enorme mogelijkheden hebben ons doen kiezen voor de software karaktergeneratie. Natuurlijk blijft het mogelijk om naast deze software een 'normale' karaktergenerator toe te passen, en deze aan de display hardware toe te voegen. Op deze manier worden snelheid en flexibiliteit gekoppeld.

Software karaktergeneratie

Bij de karaktergeneratie hebben we een programma nodig dat karakters volgens voorgeschreven vorm op het scherm tekent. Daarbij wordt gebruik gemaakt van een tabel waarin de lettervormen liggen opgeslagen. Deze tabel kan worden beschouwd als een software karakterset. De manier waarop de letters liggen opgeslagen hangt natuurlijk af van het gebruikte programma. In het door ons ontwikkelde programma is gekozen voor een maximale vrijheid voor wat betreft karakterbreedte. Het puntjesraster waarin het karakter moet komen te staan is 8 puntjes hoog en kan 1 tot 31 puntjes breed zijn. Bij 'normale' letters zal de breedte variëren van 1 tot 5 puntjes. Wanneer een letter een verticale symmetrie-as heeft (bijvoorbeeld A) hoeft de letter maar half in de karaktergenerator te staan, het programma zorgt d.m.v. spiegeling voor de andere helft. Bovendien bestaat de mogelijkheid een karakter gekanteld in de karakterset op te bergen. We denken hierbij vooral aan schaakstukken. Een gekanteld karakter is altijd 8 puntjes breed en kan 1 tot 31 puntjes hoog zijn. Wanneer we met symmetrische figuren (verticale symmetrie-as) werken wordt het figuur 16 puntjes breed, en nog



afb. 1 Zó staan de karakters in de karaktergenerator. De laatste 5 bytes bevatten karakteristieke gegevens voor de karakters uit de karakterset.
afb. 2 Dit is het resultaat op het scherm.
afb. 3 Het effect van de 3 bits stuurcode op de letter y.

steeds tussen de 1 en 31 puntjes hoog. Op deze manier kunnen schaakstukken zeer gedetailleerd worden opgeslagen. Alleen het paard is niet symmetrisch, en moet in twee delen worden gesplitst. Op dezelfde wijze kunnen raketjes, maanlanders, poppetjes enz. als karakter in de karaktergenerator worden gezet. Een en ander is verduidelijkt in afb. 1. Hier ziet u een stukje van de karaktertabel. De bits die '0' zijn, zijn open gelaten, de bits die '1' zijn, zijn gekleurd. Achter elkaar zijn 5 verschillende situaties getekend. Normaal staan de karakters natuurlijk op alfabetische volgorde, deze volgorde is als voorbeeld gekozen. In afb. 2 ziet u hoe de verschillende karakters op het scherm verschijnen. Buiten de vorm van de letters moet het programma natuurlijk ook weten waar het karakter in de karakterset is te vinden, en hoeveel bytes het karakter in beslag neemt. Daarom moet aan de karakterset een tabel worden toegevoegd waaruit deze informatie is te halen. In deze tabel staat voor elk karakter aangegeven hoelang het karakter is, en of het karakter gespiegeld, gedraaid en/of geschoven moet worden. Op de werking van de tabel gaan we later in dit verhaal wat dieper in. Doordat alle karakters strak op elkaar staan wordt zeer efficiënt met de ruimte omgesprongen. In een normale karaktergenerator zijn *alle* letters minstens 5 hokjes breed i.v.m. de breedte van de letter 'M'. De vier bovenstaande letters zouden 20 bytes in beslag hebben genomen, nu zijn dat maar 16 bytes. De schaaktoren zou niet mogelijk zijn geweest.

Instelmogelijkheden

Zoals we hebben gezien kunnen karakters in de karaktergenerator gespiegeld, gedraaid en geschoven worden (zie afb. 1 en 2). Omdat deze instellingen per karakter verschillen moeten zij ook voor elk karakter zijn opgegeven. Er zijn echter ook instellingen welke voor een bepaalde situatie voor alle karakters geldig zijn. We denken hierbij aan vergrotingsfactoren in de hoogte en de breedte, plaats waar het karakter moet komen te staan enz. In het door RB ontwikkelde programma zijn de volgende instellingen mogelijk:

XCOR Deze geheugenplaats geeft het horizontale adres van het karakter (0-255).

YCOR Deze geheugenplaats geeft het verticale adres van het karakter (0-255).

BREED Deze geheugenplaats geeft een factor waarmee de karakterbreedte wordt vermenigvuldigd. Dus voor BREED=3 worden de letters 3 x zo breed (0-255).

HOOG Hetzelfde als breed, maar nu voor de hoogte.

KLEUR Elk bitje van deze geheugenplaats heeft een eigen betekenis.

bit 0: Wanneer dit bit '0' is, zullen de gegevens van een karakter zoals beginadres, lengte, kantelen/draaien en/of schuiven worden bepaald aan de hand van de tabel. Wanneer dit bit '1' is kunnen we deze gegevens zelf bepalen (speciale effecten!!).

bit 1: Als dit bit '0' is, wordt de XCOR verhoogd met de lengte van het karakter met daarbij opgeteld een vast aantal plaatsjes volgens geheugen 'SPATIE'. Op deze manier wordt een variabele letterafstand verkregen (weinig ruimte voor de 'i', veel ruimte voor de 'm'). Wanneer dit bit '1' is wordt de XCOR met een vast getal verhoogd (volgens LTRAF). Hierdoor komen alle letters op een vaste afstand van elkaar te staan.

bit 2: Dit bit bepaalt of de letters op een witte of zwarte achtergrond komen te staan. '1' = zwart, '0' = wit.

bit 3 en 4: Deze bits bepalen de 'kleur' van de letter volgens onderstaande tabel:

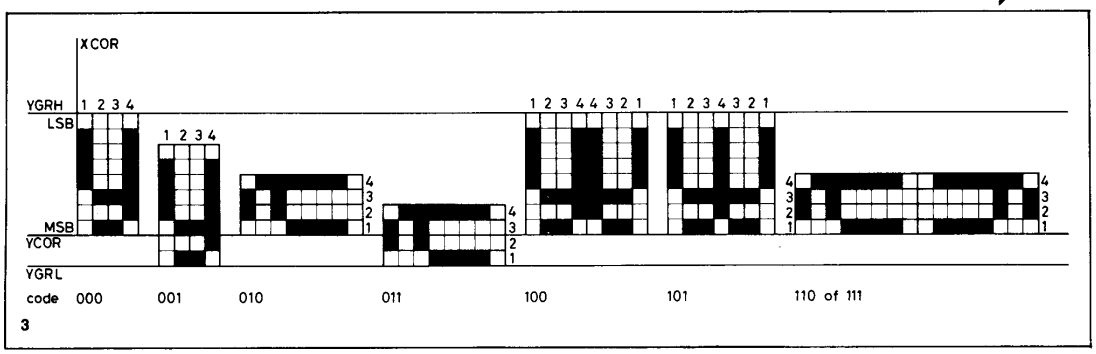
4	3	
0	0	n.v.t. ('lezen')
0	1	inverteren
1	0	wit
1	1	zwart.

bit 5: Wanneer bit 5 '1' is, wordt aan het eind van een regel automatisch een CR en LF gegenereerd (we beginnen dus op een nieuwe regel).

bit 6: Door bit 6 '0' te maken wordt de achtergrond *niet* getekend. Hierdoor hoeven per letter minder puntjes te worden getekend.

bit 7: Door bit 7 '0' te maken wordt de letter zelf niet en de achtergrond wèl getekend. Voor speciale effecten.

Door bit 6 en 7 beiden 0 te maken, gebeurt er op het scherm niets. Wel wordt door het programma beginadres en lengte van het karakter bepaald.



3

SPATIE Op deze geheugenplaats kan de afstand tussen twee letters worden aangegeven. Geldt alleen als bit 1 van 'KLEUR' nul is (variabele letter afstand).

LTRAF Op deze plaats kan een vaste letterafstand worden aangegeven (normaal = 6). Alle letters staan op een vaste afstand van elkaar. Voor LTRAF = 0 blijft XCOR ongewijzigd, en komen alle letters op dezelfde plaats op het scherm. Geldt alleen als bit van 'KLEUR' één is.

ASCII Met deze geheugenplaats kunnen we aangeven welk karakter we uit de karakterset willen halen. Wanneer bit 0 van 'KLEUR' logisch '1' is, gaan we uit van bekende gegevens ten aanzien van lengte, beginadres, kantelen/spiegelen/schuiven. In dit geval wordt niet naar 'ASCII' gekeken.

TBLHI en TBLLO Deze twee geheugenplaatsen geven het beginadres van de tabel. Omdat alle ASCII karakters onder de 20 (hex) niet afgedrukt hoeven te worden moeten we het beginadres met $20 verminderen. Voor bijvoorbeeld de ASCII code $22 wordt nu automatisch het 3e getal uit de tabel gehaald.

KARHI en KARLO Door deze 2 plaatsen wordt het 16 bits beginadres van de karakterset aangegeven.

KNTLVO bevat de x coördinaat van de beginkantlijn. Wanneer automatisch een CR en LF wordt gegenereerd (bit 5 van 'KLEUR' = 1) zal de XCOR op de waarde van KNTLVO worden gezet.

KNTLAC bevat de x coördinaat van de eindkantlijn. Zodra XCOR voorbij deze kantlijn komt zal er òf een LF en CR worden gegenereerd, òf XCOR wordt niet meer opgehoogd. Door deze kantlijn op FF te zetten schakelen we de kantlijn uit.

Bovenstaande instellingen zullen in het begin wat onoverzichtelijk lijken. Voor maximale flexibiliteit is het aantal instelmogelijkheden zeker niet overdreven. Bij normaal gebruik hoeven al deze instellingen, op ASCII na, slechts éénmaal te worden gedaan. De coördinaten die bepalen op welke plaats de letter op het scherm verschijnt worden na het tekenen van een letter steeds automatisch goed gezet voor de volgende letter. Dit automatisme is overigens uitschakelbaar. Voor bijzondere toepassingen kunnen alle instellingen naar hartelust worden gewijzigd.

Optionele instellingen

De geheugenplaatsen 'LENGTE', 'MODE', 'SUMLO' en 'SUMHI' bevatten gegevens die karakteristiek zijn voor een bepaalde letter. Bij normaal gebruik (BIT0 van 'KLEUR' = 0) worden deze gegevens door het programma uit de tabel gehaald. Bits 0...4 uit de tabel geven aan hoeveel bytes een bepaald karakter inneemt. dit gedeelte wordt in 'LENGTE' gezet. Bits 5...7 geven aan of een karakter gedraaid, gespiegeld en/of geschoven is. Deze stuurbits worden in 'MODE' gezet.

Door de lengte van alle voorgaande karakters op te tellen bij het beginadres van de karakterset bepaalt het programma op welk adres het te tekenen karakter begint. Dit adres wordt op SUMLO en SUMHI gezet.

Wanneer bit 0 van 'KLEUR' = 1 moeten de bovenstaande gegevens worden ingevoerd. Dit kan bijzonder handig zijn als er voor een bepaalde toepassing ergens in het geheugen één enkel karakter staat. De lengte van het karakter was voor de karakterset beperkt tot max. 31 bytes. Doordat de stuurbits nu apart in 'MODE' staan, kunnen we de volle 8 bits gebruiken voor de opgave van de lengte. Dus voor bijzondere karakters welke niet in de karakterset staan, maar ergens 'los' in het geheugen, is het maximale karakterformaat 8 hoog x 255 breed, of 255 hoog x 8 breed. Bij symmetrie wordt dit 255 hoog x 16 lang. Natuurlijk kunnen onafhankelijk hiervan nog verbredings- en verhogingsfactoren ingevoerd worden.

Kantelen/schuiven en spiegelen

In de tabel kunnen we voor elk karakter aangeven of het gekanteld, geschoven en/of gespiegeld moet worden. Wanneer we geen gebruik maken van de karakterset (zie 'Optionele instellingen') kunnen we deze gegevens kwijt op geheugenplaats 'MODE'. De codes zijn als volgt:

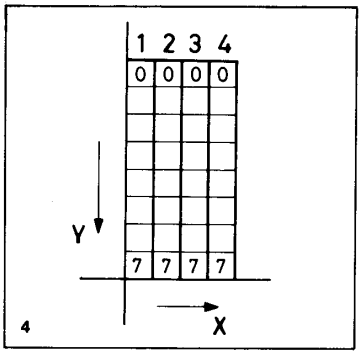
7	6	5			
1	X	0	spiegelen	midden-poot	dubbel
1	0	1	spiegelen	midden-poot	niet dubbel
X	1	X	draaien		
0	X	1	schuiven		

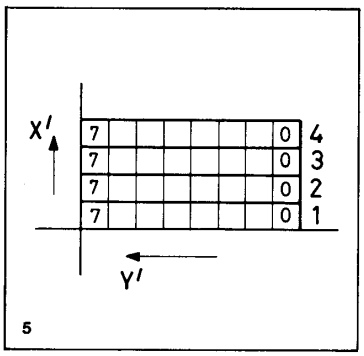
Bij symmetrische karakters hebben we dus niet de mogelijkheid tot schuiven. Bit 5 geeft nu aan of het laatste byte uit de karakterset bij spiegelen moet worden herhaald. Het effect heeft u al gezien in afb. 1 en 2 bij de letters A en I. Om het effect van de verschillende stuurcodes nog eens duidelijk te laten zien, hebben we alle mogelijke stuurbitcombinaties op de letter y los gelaten. Het resultaat ziet u in afb. 3. De normale stuurcode voor deze letter is 001 (geschoven).

Hulpvariabele

Voor de realisatie van het programma is het nodig verschillende hulpvariabelen in te voeren. In afb. 3 ziet u al twee van deze hulpvariabelen, nl. YGRH en YGRL. Wanneer de achtergrond (ook)

afb. 4 Dit is de normale richting van X en Y coördinaten.
afb. 5 Bij gedraaide karakters nemen we een gedraaid coördinatenstelsel aan, waardoor het programma vrijwel op dezelfde manier verloopt.





moet worden getekend zal het programma alle vakjes binnen deze grenzen (waar geen letters staan) met de achtergrondkleur opvullen. Op deze manier verdwijnen alle 'resten' van oude teksten. YGRH ligt 8 x de verhogingsfactor boven YCOR, en YGRL ligt 2 x de verhogingsfactor onder YCOR. De verhogingsfactor is het getal dat in 'HOOG' staat. Bij normaal gebruik zal dit 1 zijn. Bij stuurcode '000' (normale situatie afb. 3) wordt de letter zo op het scherm gezet dat een letter met het puntje links onder op de coördinaten (XCOR), (YCOR) komt te staan. Wanneer we moeten schuiven (stuurcode 001 of 011) wordt aan het begin van het programma YCOR gelijk gemaakt aan YGRL, waardoor de hele letter 2 punten zakt. Aan het eind van het programma wordt YCOR weer op z'n oorspronkelijke waarde terug gezet.
Tijdens het tekenen van de letter geven steeds de twee hulpvariabelen XCOR2 en YCOR2 aan waar het eerstvolgende puntje moet worden getekend. In afb. 4 ziet u dit in tekening gebracht. Als eerste wordt bitje 7 van byte 1 getekend. Voor elk volgend bitje wordt YCOR2 met één verlaagd (lagere Y coördinaat is hoger in het beeld). Wanneer dit proces 8 x is herhaald, wordt YCOR2 weer op de beginwaarde gezet en wordt XCOR2 met één verhoogd. Op dezelfde wijze worden de volgende bytes getekend. Het aantal bytes dat wordt getekend komt overeen met het getal dat in 'lengte' staat. Bij een gekanteld karakter willen we graag hetzelfde programma gebruiken. Dat kan door het coördinatenstelsel aan te nemen volgens afb. 5. De richting van de coördinaten is hier hetzelfde als in afb. 4 (Y coördinaat oplopend van bit 0 naar 7, X coördinaat oplopend van byte 1 naar byte 4). We zien dus dat niet alleen X en Y coördinaat zijn verwisseld, maar dat bovendien de richtingen tegengesteld zijn. De richtingsverandering is te verkrijgen door de coördinaten te inverteren. In de normale situatie maakten we XCOR2 gelijk aan XCOR en YCOR2 gelijk aan YCOR, nu moeten we XCOR2 gelijk maken aan $\overline{\text{YCOR}}$ en YCOR2 aan $\overline{\text{XCOR}}$. Bij het doorgeven van adressen aan het display moeten we coördinaten natuurlijk weer terug draaien. Wanneer nu een volgend bitje moet worden getekend zullen we net als vroeger YCOR2 met één verlagen. Doordat YCOR2 nu in wezen een geïnverteerde X coördinaat is, zal het uiteindelijke effect op het scherm een verhoogde X coördinaat zijn. Hierdoor zal het nieuwe bitje rechts van het vorige bitje worden getekend.
De symmetrie is zeer eenvoudig te realiseren. Wanneer symmetrie gewenst is wordt er bij het bereiken van het laatste te tekenen byte een 'wissel' omgezet waardoor niet het eerstvolgende byte uit de karakterset wordt gehaald, maar juist het vorige byte. Op die manier wordt (bij een karakterlengte van 4) na byte 4 weer de bytes 3, 2 en 1 getekend, waardoor het spiegeleffect ontstaat. Bij gekantelde symmetrie (code 11X) ligt de situatie iets moeilijker. We stellen de hulpcoördinaten nu zo in alsof hetzelfde karakter nog een keer strak naast het reeds getekende karakter moet komen te staan. We schuiven nu het te tekenen byte echter niet naar links, maar naar rechts. Hierdoor zal niet bit 7 maar bit 0 als eerste worden getekend. Op deze manier komt ook in dit geval het spiegelbeeld tot stand.

Volgende maand:
De flowchart en het programma.