MEER OVER DE BLITTER

Er zijn al heel wat recensies verschenen over de Blitter van ATARI. De één behandelde de globale eigenschappen van de Blitter, de andere ging wat verder en deed wat testen om snelheden te kunnen vaststellen. In dit artikel leert u wat de Blitter is, maar tevens geven we voorbeelden van het aansturen van de Blitter, zoals het detekteren, het softwarematig aan- of uitzetten en het besturen middels Line_A routines. Nu inmiddels alle Blitters aan de dealers zijn nageleverd, is het goed uw kennis weer wat op te vijzelen.

Laten we een tijdje terug gaan in de geschiedenis. Terug naar de CP/M systemen. Velen onder u kennen de schermen wel die bij CP/M sytemen hoorden. Op die schermen was alles opgebouwd uit karakters, of het nu om tekst ging of grafieken. We noemen dergelijke systemen dan ook 'karakter-georiënteerd'. Vaak gebruikten deze systemen dan ook een karaktergenerator om het scherm aan te sturen. Voor tekst een prima oplossing, maar de grafieken die meestal uit de letter 'x' werden opgebouwd, spraken niet erg tot de verbeelding.
Later kwamen de MS-DOS systemen. Deze waren eigenlijk ook karakter-georiënteerd, zij het dat men met behulp van grafische kaarten toch een grafisch systeem kon maken. Op een gegeven moment besefte men dat het grafisch weergeven door middel van vensters en dergelijke veel gebruikersvriendelijker was dan de tot dan gebruikte karakter-weergave. Vandaar dat grafisch georiënteerde systemen vanaf dat moment zeer in opmars kwamen. Voorbeelden daarvan zijn de Macintosh, de Amiga en natuurlijk de ATARI ST.

PUNTJE VOOR PUNTJE

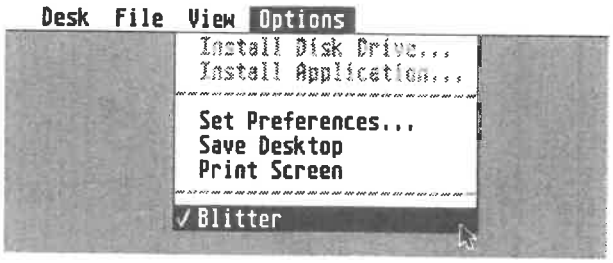
Een echt grafisch georiënteerd systeem gebruikt geen karakters of stuurkodes voor de beeldopbouw. Een dergelijk systeem heeft een videogeheugen en in dat videogeheugen geeft elk bitje (of groepje van bitjes bij kleurensystemen) een punt weer op het beeldscherm. Als we uitgaan van een monochrome systeem, dan kunnen we met 1 bit (0 of 1) 1 schermpuntje definiëren: zwart of wit. Alle tekens die u op het scherm ziet, worden bij een dergelijk systeem in het beeldschermgeheugen getekend door de bitje in de juiste volgorde te setten (1 maken) of resetten (0).
Tekenen op een grafisch systeem is eigenlijk kopiëren: het karakter wordt gekopieerd van de font-definitie (de verzameling letters en cijfers) naar het schermgeheugen. Daarom kunnen dergelijke systemen ook met meerdere typen fonts werken: je laadt gewoon een andere definitie in.
Bij CP/M systemen zat het type karakter ingebakken in de karaktergenerator en viel er dus niets te veranderen.

SNELLE HARDWARE

U begrijpt natuurlijk dat dat kopiëren bij een grafisch systeem niet vanzelf gaat. Dus moesten er routines gemaakt worden die zich bezig hielden met het kopiëren van en naar het schermgeheugen en het manipuleren van de data in dit geheugen. Zoals al eerder aangegeven, gaat het hier echter om bewerkingen met bits in plaats van komplete bytes (=8 bits). Vandaar dat men hier dan ook spreekt van Bit Block Transfer afgekort met Bit-BLT routines. Bij de ATARI zitten deze routines verwerkt in de Line_A routines.
Tot nog toe hebben we het uitsluitend over software routines gehad. Omdat schermhandelingen erg arbeidsintensief zijn, kosten Bit-BLT software-routines echter erg veel tijd van de CPU (central processing unit, de 68000 dus). Daarnaast is een CPU niet ontworpen voor handelingen met bits. Daardoor zullen de gewenste bewerkingen relatief nog meer tijd kosten.
In theorie zijn er dus snelheidsverbeteringen te behalen als we deze Bit-BLT routines weten om te zetten in een hardwareschakeling die zich uitsluitend bezig houdt met het uitvoeren van Bit-BLT operaties. Door die schakeling zo te maken dat hij zelfstandig met het schermgeheugen kan werken, sparen we CPU tijd, die anders besteed zou worden aan datatransport tussen de schakeling en het schermgeheugen. Omdat deze hardware speciaal voor Bit-BLT operaties ontwikkeld wordt, kan hij ook efficiënter met bits omgaan. ATARI heeft inmiddels, naast de software vorm, ook voor hardware gekozen: de Blitter-chip. Blitter is duidelijk afgeleid van Bit-BLT.
Een IC dat zelfstandig direkt met het geheugen van een machine kan omgaan, noemen we een DMA-device. De Blitter is zo'n DMA device. DMA staat voor Direct Memory Acces. Een klein nadeel van DMA is dat de toegang tot het geheugen via de data- en adresbus verloopt en dat het DMA-device en de CPU om beurten van deze bus gebruik moeten maken. Aan de andere kant: data-transport door de CPU zou waarschijnlijk meer tijd zou kosten.
In tegenstelling tot wat veel artikelen vermelden, is de Blitter geen vervanging van de Line_A routines. De Line_A routines doen meer dan Bit-BLT. Feitelijk gebruiken sommige Line_A routines op hun beurt Bit-BLT routines om op die manier bijvoorbeeld een figuur (polygon) te vormen. Bit-Blt routines verrichten uitsluitend manipulaties in het beeldschermgeheugen en kennen geen ingebouwde figuren of iets dergelijks. Men gebruikt deze routines juist om dergelijke figuren te maken door ze in de gewenste volgorde bewerkingen te laten verrichten. De Blitter heeft dan ook op dat gebied niet meer pijlen op zijn boog dan zijn software broeder.

BLITTER PROGRAMMEREN

De programmeur die aktief van de Blitter gebruik wil maken, heeft een aantal mogelijkheden voor aansturing. Allereerst kan hij gewoon de bestaande VDI-routines aanroepen die in elke programmeeromgeving beschikbaar zijn. Denk daarbij aan funkties om lijnen te trekken, vlakken te vullen, en dergelijke.



Daarnaast kan men gebruik maken van de Line_A routines. Men zal hiervoor al gauw uitwijken naar assembler, maar dit is niet strikt noodzakelijk. Op START disk 14 vindt u ook een voorbeeld van Line_A besturing in GFA. Als laatste mogelijkheid is er voor de zeer begaafde (assembler)-programmeur de mogelijkheid om de Blitter rechtstreeks aan te sturen. Dat wil zeggen dat men dan zelf de benodigde registers vult en de Blitter de juiste kommando's geeft. Dit vraagt erg veel inzicht van de programmeur. Voor schermroutines afwijken van Line_A door zelf routines te schrijven, heeft weinig nut daar men daar weinig snelheidsverbetering kan halen.
Een mogelijke reden om toch de Blitter-aansturing in eigen hand te houden, is het 'misbruiken' van de Blitter voor andere doeleinden zoals bijvoorbeeld een supersnelle ramdisk. Of geheugen transporteren bij soundsamplers en daardoor een hogere sample-rate en dus kwaliteit van het geluid behalen.

BLITTER STATUS

Het voordeel van gebruik van VDI en Line_A routines is dat de software niet apart aangepast hoeft te worden voor de Blitter. Als de Blitter aanwezig is en daadwerkelijk aan staat, wordt er automatisch gebruik van gemaakt. Als er geen Blitter aanwezig is, zal de software toch werken. Alleen zal de schermuitvoer aanmerkelijk trager worden.
Voor we in een programma ook maar iets met de Blitter kunnen doen, moeten we natuurlijk eerst weten of hij wel aanwezig is en, als dat het geval is, of hij wel aan staat. Zoals u misschien wel weet, heeft de gebruiker de mogelijkheid om de Blitter vanuit de desktop in het Options menu uit te schakelen. Vandaar dat een programma dat de Blitter rechtstreeks aanroept de status van de Blitter moet detekteren en hem eventueel zelf moet kunnen aanzetten. Voor dit doel heeft ATARI een extra XBIOS funktie toegevoegd aan het reeds bestaande operating system. Deze funktie kreeg als nummer $40 of zoals u wilt 64 decimaal. Door middel van deze call is de status van de Blitter op te vragen en in te stellen.
De C gebruikers onder ons kunnen deze funktie als volgt omschrijven:

```
WORD Blitmode(flag)
WORD flag;
```

Zij dienen de volgende definitie aan hun header-files toe te voegen:

```
#define Blitmode(a) xbios(64,a)
```

De aanroep gaat dan als volgt:

```
curmode = Blitmode(-1);     /* bewaar status          */
Blitmode(curmode | 1);      /* zet Blitter aan        */
do_stuff();                 /* doe wat je moet doen   */
Blitmode(curmode);          /* zet oude instelling terug */
```

Deze funktie geeft in alle gevallen een word terug, ook als deze funktie wordt uitgevoerd op het oude operating systeem, waar deze call dus niet aanwezig is. In dat geval krijgt men -32 terug, wat 'Illegal Funktion' betekent. Als deze funktie wel aanwezig is, worden er maar twee bits gebruikt van het word dat je terugkrijgt. Een overzicht:

BITBETEKENIS

0 "1" Blitter aktief, "0" Blitter uitgeschakeld
1 "1" Blitter aanwezig, "0" geen Blitter aanwezig
2-14 gereserveerd
15 altijd "0"

Als we voor de aanroep de flag de waarde -1 geven, krijgen we uitsluitend de status terug (zie voorbeeld). In het andere geval kunnen we de Blitter aan- of uitzetten door bit 0 te setten of te resetten.
Assembler programmeurs hebben meer aan onderstaande source voor het opvragen van de status.

```
move.w      #-1,-(sp)       ;vraag status op
move.w      #$40,-(sp)
trap        #14
addq        #4,sp
move.w      d0,-(sp)        ;bewaar status
or.w        #1,d0           ;zet bit 1 aan
move.w      d0,-(sp)        ;zet Blitter aan
move.w      #$40,-(sp)
trap        #14
addq        #4,sp
;
; doe wat je moet doen
;
; en omdat de oude status nog steeds op
; de stack aanwezig is,
; kunnen we hem dus direkt terugzetten
;
move.w      #$40,-(sp)      ;zet status terug
trap        #14
addq        #4,sp
```

N.B. In het voorbeeld is geen rekening gehouden met de mogelijkheid van -32 voor een systeem waar geen Blitter aanwezig is.

BLITTER SOFTWARE

Een voorbeeld van Line_A besturing van de Blitter vindt u op START schijf 14. Dit voorbeeld is in machinetaal geschreven en moet probleemloos door diverse assemblers gehaald kunnen worden. Die van DRI en GST verwerkten het in ieder geval prima. Het voorbeeld werkt uitsluitend in lage resolutie. Naast dit voorbeeld (en de geassembleerde versie) vindt u op START disk 14 diverse testprogramma's voor kleur- en zwartwit. Als u de demo's bekijkt, zult u zien dat er wel degelijk zeer goede resultaten met de Blitter mogelijk zijn.
In eerdere publikaties heeft men beweerd dat de Blitter niet goed zou zijn in het trekken van lijnen. Eén van de programma's op START disk 14 rekent genadeloos met deze fabel af. De bewuste demo trekt diverse lijnen op het scherm, herhaalt dat een aantal malen en meet de tijd die daarvoor nodig was. Het verschil tussen Blitter aan en Blitter uit is aanmerkelijk.
Wie gebruik wil maken van het Line_A-voorbeeld in GFA Basic dat op de disk staat, moet vooraf wel de nodige kennis over Line_A in huis halen. We denken daarbij aan Profi Buch en andere boeken van dat kaliber.
Voor specifieke (rechtstreekse) besturingsopdrachten aan de Blitter heeft Atari een uitgebreide (Engelse) handleiding samengesteld die echter te lang is om integraal af te drukken. START kreeg echter toestemming deze dokumentatie op START schijf 14 te zetten. Vanwege de geweldige hoeveelheid materiaal over dit onderwerp, zijn de bestanden via het programma ARC.TTP samengeperst opgeslagen.

Wilfred Kilwinger