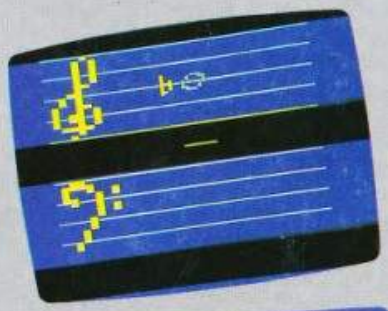
# elektuur

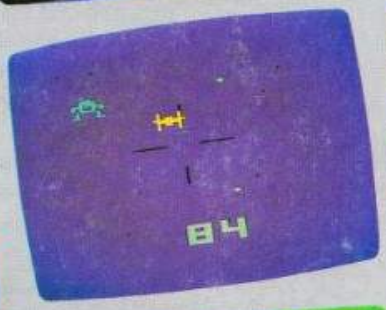
nr. 186
april 1979
f 3,50
Bfrs. 59

maandblad voor elektronica

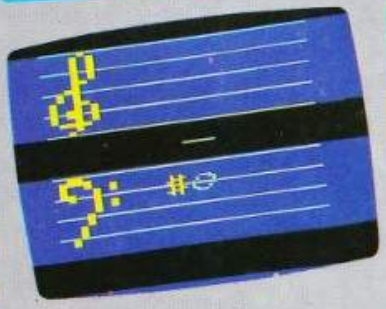
## speel-computer

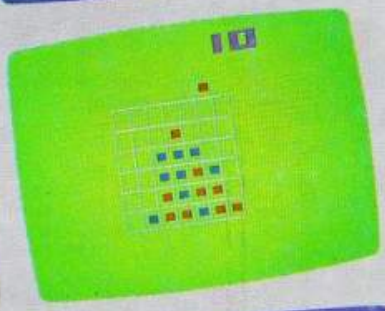
### chips in het spel





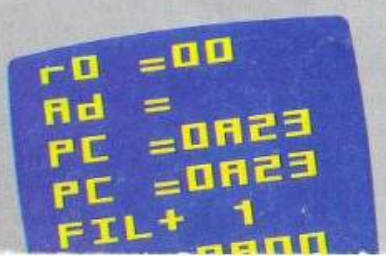


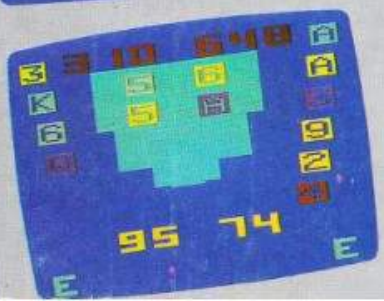


notenbalk terwijl u speelt
paardenrennen
ruimtevaart
tijdklok
vier op een rij




jagen
rekenen
21'en kaartspel
spelen met de PVI
monitor-kommando



# tv-speel-computer

## chips in het spel

**Voor veel mensen zijn 'chips' dingen die de werkgelegenheid in gevaar brengen, die je met hetzelfde scepsis moet bezien als kerncentrales en schaatsplankjes. Meer nuchter beschouwd gaat het om plakjes elektronica, die enorm grote elektronische schakelingen herbergen. Een microprocessor bijvoorbeeld.**
**Het probleem met microprocessors is dat er meer bij komt kijken dan alleen de hardware. Met name veel elektronica-amateurs worden hierdoor afgeschrikt. Ofschoon de situatie in dit opzicht de laatste tijd sterk verbetert, is de beste aanpak nog steeds om uit te gaan van een konkrete toepassing, waarin de microprocessor dezelfde funktie vervult in een schakeling als een 7490 in een tellerschakeling. Bij wijze van spreken dan, want de $\mu$P kan wel iets meer! Met als gevolg dat voor de bouw en het gebruik van de schakeling in kwestie nauwelijks of geen kennis van microprocessors is vereist.**
**Om welke schakeling gaat het in dit artikel? Een kastje met een toetsenbord en 'joystick'-potmeters waarop een grammofoon of kassetterekorder is aangesloten. Via plaat of band komt de software naar binnen. En wat krijg je dan? Een zeer 'sophisticated' tv-spellendoos, in kleur, met passende begeleidende geluiden, scorebord en een onbeperkt arsenaal aan op het scherm weer te geven vormen. Dat er een $\mu$P in de doos zit kunt u eigenlijk wel weer vergeten. Spelenderwijs kan de behoefte bestaan om zelf spelletjes te maken, te programmeren. Dat kan.**

De tv-speel-computer bevat een microprocessor, kortweg $\mu$P. Het zij zo. Niet bang zijn voor $\mu$P's, ze bijten niet! Bovendien is geen kennis vereist van de $\mu$P om dit apparaat met plezier te bouwen en te gebruiken. Aan de andere kant is er een unieke mogelijkheid om in een later stadium de $\mu$P te leren kennen, namelijk door alle mogelijkheden die hij hier biedt te benutten.
Dit artikel is bedoeld als algemene inleiding. (Elders in dit nummer vindt men een artikel met de bouwbeschrijving). De tekst is afgestemd op degenen die het apparaat willen bouwen en gebruiken in kombinatie met de via de ESS beschikbare spellen. Een meer gedetailleerde beschrijving wordt geleverd samen met de print. Deze is ook los verkrijgbaar (voor degenen die zelf dubbelzijdige printen met doorgemetalliseerde gaten kunnen maken ...). Voor de meeste lezers zal de informatie in dit artikel voldoende zijn.
Eén ding nog, alvorens we in het apparaat duiken: het gaat hier niet 'zo maar' om tv-spelletjes. Bepaald niet flauw, veel intelligenter en dus is de lol er niet gauw af. Het beste bewijs biedt het schouwspel waarbij de hele redaktie zich schaart rond het prototype, waarmee twee kollega's met elkaar in de slag zijn. Er zijn ook spelletjes mogelijk waarbij men tegen de computer speelt, zoals 'vier-op-een-rijtje', een soort boter-kaas-en-eieren met een hoog IQ. De machine, de tv-speel-computer heeft zich al bewezen als een formidabele tegenstander.

### Wat doet-ie?

Het apparaat moet de tv-beelden leveren ('spel-beelden'), die horen bij een bepaald spel. In kleur. Passende geluidseffekten verhogen de spelvreugde, dus dat willen we ook graag hebben. Een spel kan worden verloren of gewonnen: een weergave op het scherm van de score zou dus ook mooi zijn.

#### Voorwerpen

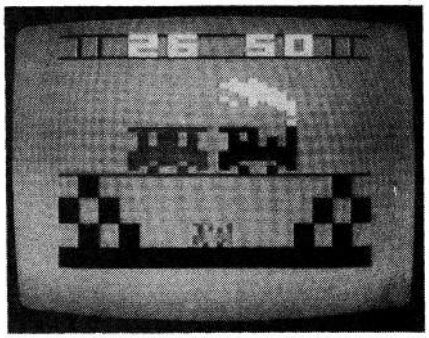
Het spelbeeld bevat 'voorwerpen'. In het grijze tv-tennisverleden bestonden deze uit vertikale strepen (de rackets) en een vierkantje (de bal). Vandaag de dag wordt wel wat meer gewenst: komplete cowboys, slagschepen, straalvliegtuigen en ga zo maar door. De tv-speel-computer bezit in dit opzicht schier eindeloze mogelijkheden. Een voorwerp wordt verkregen door een rechthoek, bestaande uit 10 bij 8 = 80 'rechthoekjes' in te vullen, of juist niet in te vullen (zie tabel 2). In figuur 1 bijvoorbeeld een lokomotief.
Een bepaald voorwerp kan worden herhaald, elders op het beeld. Deze duplikaten zijn dan onderling gelijk kwa vorm (de wijze waarop de 10 bij 8-rechthoek is ingevuld), kleur en afmetingen. Een voorwerp kan in vier maten worden weergegeven: 1:1, 2:1, 4:1 en 8:1.
De vorm van het voorwerp kan vrij snel worden veranderd (hergeprogrammeerd), zodat hetzelfde stuk 'voorwerpgeheugen' voor meerdere verschillende vormen (of afmetingen) in hetzelfde beeld gebruikt kan worden. Dit is echter meer iets voor de $\mu$P-specialisten. Voor de meeste gebruikers is het belangrijker dat ook zonder dergelijke kunstgrepen vier verschillende voorwerpen tegelijk mogelijk zijn. Bijvoorbeeld een slagschip, onderzeeër, torpedo en straalvliegtuig of (iets vreedzamer) twee voetballers, een doel (tweemaal weergegeven) en een bal.
Voor de kleur van een voorwerp heeft men de keuze uit acht mogelijkheden. De drie primaire kleuren (rood, groen en blauw) kunnen onafhankelijk gekozen worden, zodat ook de drie kombinaties van twee kleuren beschikbaar zijn (geel-oranje, blauw-groen, violet); verder zwart en wit.

#### Achtergrond

Naast voorwerpen is bij de meeste spellen een 'achtergrond' vereist. Dat kan van alles zijn: een gehele of gedeeltelijke begrenzing van het speelveld, of een arceerpatroon. In een

**Tabel 1. De tv-speel-computer in een notedop.**
- **bediening via joysticks en toetsenbord**
- **video-uitgangssignaal geschikt voor PAL-kleuren-tv (UHF- of VHF-ingang)**
- **geluidseffekten met ingebouwde luidspreker**
- **microprocessor-bestuurd en programmeerbaar voor de meest uiteenlopende spellen**
- **ingebouwde kassette-interface, voor het makkelijk inlezen of opslaan van programma's (bijvoorbeeld de via de ESS beschikbare spelprogramma's)**
- **uitgebreide monitor-software**

1




**Figuur 1. Deze foto illustreert goed wat de tv-speel-computer zo al in beeld kan brengen.**

aantal opzichten is het 'opbouwen' van een achtergrond bij de speelcomputer vergelijkbaar met het maken van een voorwerp. De achtergrond bestaat grofweg uit 160 vierkantjes, opgedeeld in 10 rijen van 16 vierkantjes. Ieder vierkantje wordt bepaald door zijn bovenrand en linker zijkant; de rechter zijkant en onderrand behoren tot aangrenzende vierkantjes. Elke rand van een vierkantje kan in beeld worden gebracht; door de linker zijkant te verbreden is het bovendien mogelijk om het hele vierkantje op te vullen.
Met dit systeem kunnen allerhande verschillende achtergrondpatronen worden opgebouwd. Een paar daarvan zijn te zien in figuur 1. De onderste rij vierkantjes is geheel ingevuld en levert een 'stevige' grens. Het schaakbord-patroon links- en rechtsonder is verkregen door vierkantjes afwisselend wel en niet te vullen. De hokjes langs de bovenrand bestaan uit de randen van de overeenkomstige vierkantjes. Diverse andere horizontale en vertikale lijnen, balken en stippen zijn mogelijk, alle variaties op hetzelfde tema.
De kleur van de 'achtergrondlijnen' en die van de 'achtergrond tussen de lijnen' (de rest van het scherm) kunnen onafhankelijk van elkaar worden gekozen. Ook hier acht kleurmogelijkheden. Er is één 'maar': hebben een voorwerp en de achtergrondlijnen dezelfde kleur, dan zal het voorwerp geheel of gedeeltelijk niet opvallen!

#### Score

Numerieke gegevens over het spelverloop willen we graag in beeld hebben. De tv-speel-computer stelt daarvoor vier cijfers ter beschikking. Daarmee kan één getal van vier cijfers worden weergegeven (bijvoorbeeld: 2650) of twee getallen van elk twee cijfers (bijvoorbeeld: 26 50). De cijfers kunnen boven (figuur 1) of onder in het beeld worden geplaatst. De kleur van de cijfers is altijd de komplementaire van de kleur van de achtergrond, zodat de stand altijd duidelijk afsteekt.

#### Geluid

De tv-speel-computer is uitgerust met een programmeerbare blokgolfgenerator, waarmee via een ingebouwd luidsprekertje met allerhande piepjes en fluitjes de saillante spelmomenten kunnen worden benadrukt.

#### Botsingen

In veel spellen hangt de score samen met 'treffers' of 'missers'. In de tv-speel-computer wordt elke botsing tussen twee voorwerpen of tussen voorwerp en achtergrond ontdekt en geregistreerd. Deze informatie hoeft niet uitsluitend te worden gebruikt voor het aanpassen van de score. Er kunnen geschikte geluids-effekten worden 'afgevuurd'. Verder kan een voorwerp van vorm veranderen, uiteen spatten of in het niet verdwijnen. De achtergrond kan veranderen. Je kunt het zo gek niet bedenken. Bijvoorbeeld in een duel tussen twee cowboys. Wordt er een getroffen dan kan men hem via voorwerpaanpassing realistisch ter aarde laten storten en vervolgens onder begeleiding van 'hemelse' muziek laten opstijgen tot hij uit het beeld verdwijnt!

#### Het geheel

Een interessant spel omvat alle reeds genoemde elementen, welke bovendien gedurende het spelverloop aan veranderingen onderhevig zijn, onder invloed van manipulaties van de speler(s) aan toetsenbord en/of joysticks. En hier blijkt pas goed hoe intelligent het apparaat is: eenmaal geprogrammeerd voor een bepaald spel zorgt de microprocessor voor de voorwerpen en de achtergrond van het spelbeeld, houdt de speler(s) via joystick en toetsenbord in de gaten en stelt botsingen vast; aan de hand van deze informatie past hij het spelbeeld aan, levert geluidseffekten en houdt de score bij.
Dat is nog niet alles. Het mooie van dit systeem is dat door een ander programma via band of plaat in te voeren een ander spel wordt verkregen, dat volslagen verschillend kan zijn van het vorige spel. Bovendien is de microprocessor voldoende slim om een geduchte tegenstander te zijn bij solospelen, dus mens tegen machine, vooropgesteld dat de spelregels niet al te gekompliceerd zijn; de basisversie beschikt niet over voldoende 'geheugenruimte' voor bijvoorbeeld schaken.

## Hoe werkt het?

In deze fase geven we een zeer ruwe schets van de werking. Later volgen nog enige details en zoals gezegd, het naadje van de kous staat in een apart verkrijgbaar geschrift.
Het blokschema staat in figuur 2. Het blok 'CPU', helemaal links, is de centra verwerkingseenheid, de 'hersen-chip'. Deze bepaalt het doen en laten van de overige blokken van figuur 2. Men moet zich voorstellen dat zo'n blok bediend wordt d.m.v. een soort kodeslot(en) (of -schakelaars). Indien de juiste kode wordt aangeboden, wordt het

**Tabel 2. Beeldopbouw/Basismogelijkheden.**

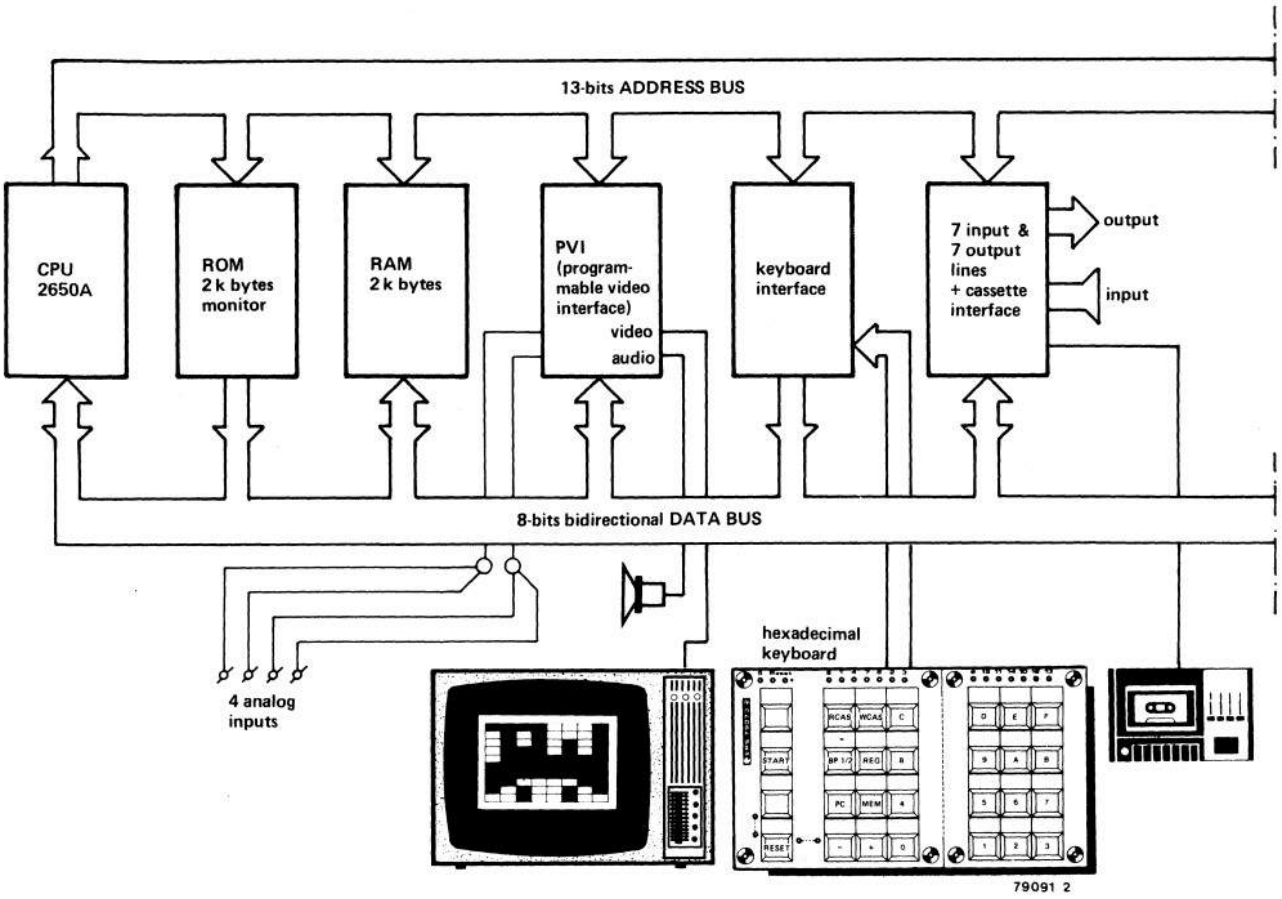
**Vier verschillende voorwerpen (die kunnen bewegen), die elk**
- **passen in een rechthoek van 8 bij 10 vierkanten.**
- **tegelijkertijd op verschillende plaatsen op het scherm kunnen worden gezet.**
- **in vier verschillende afmetingen kunnen worden weergegeven: x1, x2, x4 of x8.**
- **kunnen worden weergegeven in acht verschillende kleuren**

**Een achtergrond**
- **opgebouwd 10 rijen van 16 vierkanten; gedeelten van horizontale en vertikale lijnen kunnen al dan niet worden weergegeven, vierkanten kunnen geheel of gedeeltelijk worden ingevuld. Keuze uit acht kleuren.**

**Een scherm: kleurkeuze uit acht mogelijkheden.**

**Weergave van de score:**
- **vier cijfers boven of onder in het beeld.**
- **bruikbaar als 2x twee-cijfer-score of 1x 4-cijfer-score.**
- **in de complementaire kleur van de lijnen in de achtergrond**

2

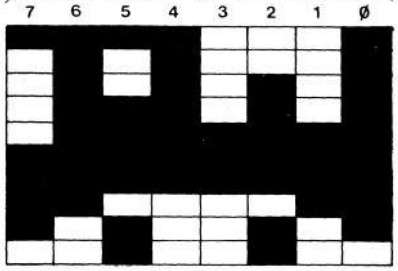



betreffende blok aktief. Zo kent ieder op zichzelf staand blok zijn eigen kode. De besturing van de blokken wordt verzorgd door de CPU. Deze biedt daartoe de juiste informatie aan op de adres-bus.
Een brein zonder geheugen is vrij hulpeloos. De speel-computer maakt gebruik van twee soorten geheugens. Het eerste geheugen is een ROM (Read Only Memory). Dit geheugen bevat de noodzakelijke basis-informatie (bijv. de wijze waarop programma's op band gelezen of geschreven moeten worden). Het tweede geheugen is een RAM (Random Access Memory). Dit geheugen verliest zijn informatie indien de voedingsspanning uitgeschakeld wordt. Het dient voor tijdelijke opslag van informatie (bijv. een programma voor een bepaald spel, of de wijzigende score-informatie).
De CPU, de ROM en de RAM kunnen de organen van de computer genoemd worden. De randapparatuur (die met de computer verbonden kan worden, ook wel 'peripherals' genoemd) bestaat uit een kleuren-tv-ontvanger, een toetsenbord (keyboard), stuurknuppels (joysticks), een luidspreker en een band- of cassetterecorder. Deze randapparatuur kan niet door de CPU bestuurd worden. D.w.z. niet zonder meer, maar wel via tussenstations, ook wel 'interfaces' genaamd.
Bekijken we nu weer het blokschema. Het vierde blok, de PVI (Programmable Video Interface), zorgt voor de signalen naar de tv-ontvanger en de luidspreker. Tevens verwerkt de PVI de van de joysticks afkomstige analoge signalen.
Het volgende blok, de keyboard-interface, dient zoals de naam al zegt als interface tussen het keyboard en de speel-computer. Een laatste blok draagt zorg voor de in- en uitgangssignalen van de band- of cassetterecorder. Een omschrijving van de funktie van de overige in- en uitgangen van dit blok wordt hier achterwege gelaten.
De 8-bit databus maakt het mogelijk dat de uitgangssignalen van een bepaald blok naar de overige blokken doorgegeven kunnen worden. Hierbij is verkeer in twee richtingen mogelijk (bidirectional). De CPU bijvoorbeeld, kan informatie ontvangen van ieder ander blok of informatie versturen naar ieder ander blok. De ROM en de keyboard-interface kunnen daarentegen alleen maar informatie geven en niet ontvangen. De wijze waarop de diverse blokken samenwerken (onder leiding van de CPU) kan het beste worden geïllustreerd aan de hand van een voorbeeld.

**Beeldvorming**

Neem nou de lokomotief van figuur 1. Wat te doen om deze op het scherm te krijgen?
De figuur op het beeldscherm wordt bepaald door de informatie welke in de PVI is opgeslagen. De PVI aksepteert echter geen informatie van de keyboard-interface, maar wel van de RAM. M.b.v. de RAM als tussenschakel kan men dus de PVI via het keyboard programmeren, waarbij de informatie in de RAM bewaard blijft.
Laten we eerst de voorwerpen beschouwen. Ze werden opgebouwd in een rechthoek van 8 (horizontaal) bij 10 (vertikaal) rechthoekjes. Bij het bepalen van hoe de rechthoek moet worden ingevuld is een vorm van kodering nodig. Nu komt wat computerjargon goed van pas: het kan gebruikt worden als een soort steno. Een bit is de kleinste eenheid van digitale informatie: wel of geen spanning op een bepaalde lijn, oftewel '1' of 'Ø' (nul). Een bepaald aantal bits, (het aantal hangt van het systeem af) heet een byte. Bij de speel-computer bestaat een byte uit 8 bits, met andere woorden:
alle informatie ('data') wordt in groepjes van 8 bits doorgegeven, in de vorm van wel of geen spanning op elke ader van een acht-aderige kabel ('bus').
Als we aan elk rechthoekje een bit toekennen en als de rechthoek bestaat uit 10 rijen van 8 vierkantjes, zijn er 10 bytes nodig om het voorwerp vast te leggen. Een ingevuld rechthoekje heeft een bitwaarde 1, een niet ingevuld een bitwaarde Ø. Als we de rechthoekjes van een rij en het byte van links naar rechts

**3**




| | binaire kode | hexa-decimale kode | (uiteindelijk) adres in PVI | tijdelijk adres in RAM |
|---|---|---|---|---|
| horizontale positie van figuur | 11110001 | F1 | 1FØØ | ØA00 |
| horizontale positie van duplikaat (buiten beeld) | Ø1Ø1ØØØ1 | 51 | 1FØ1 | ØAØ1 |
| vertikale positie van figuur | Ø1Ø1Ø1Ø1 | 55 | 1FØ2 | ØAØ2 |
| vertikale positie van duplikaat | Ø1110101 | 75 | 1FØ3 | ØAØ3 |
| | Ø1111111 | 7F | 1FØ4 | ØAØ4 |
| afmetingen van figuur | 11111111 | FF | 1FØ5 | ØAØ5 |
| kleur van figuur | 11111111 | FF | 1FØ6 | ØAØ6 |
| | 11000011 | C3 | 1FØ7 | ØAØ7 |
| | 10100101 | A5 | 1FØ8 | ØAØ8 |
| | 00100100 | 24 | 1FØ9 | ØAØ9 |
| | 10000000 | 80 | 1FØA | ØAØA |
| | 11111111 | FF | 1FØB | ØAØB |
| | Ø1ØØ1111 | 4F | 1FØC | ØAØC |
| | 11111111 | FF | 1FØD | ØAØD |
| | 11000000 | C0 | 1FCØ | ØACØ |
| | 00100000 | 20 | 1FC1 | ØAC1 |

**Figuur 2. Blokschema van de tv-speel-computer.**

**Figuur 3. Invuloefeningen!**

lezen is in figuur 3 de byte van de bovenste rij 1111ØØØ1. De bytewaarde kunnen we nog korter weergegeven om van dit gedoe met enen en nullen af te zijn. Willen we groepen van vier bits vervangen door éénn letter of cijfer (dus een byte weergegeven met twee 'karakters') dan moeten we 16 verschillende karakters nemen, omdat er met vier bits 16 verschillende mogelijkheden (kombinaties van nullen en enen) zijn. Gebruikelijk hiervoor zijn de cijfers 0 t/m 9 en de hoofdletters A t/m F, en wel als volgt:

| | | |
|---|---|---|
| Ø | ØØØØ | 0 |
| 1 | ØØØ1 | 1 |
| 2 | ØØ1Ø | 2 |
| 3 | ØØ11 | 3 |
| 4 | Ø1ØØ | 4 |
| 5 | Ø1Ø1 | 5 |
| 6 | Ø11Ø | 6 |
| 7 | Ø111 | 7 |
| 8 | 1ØØØ | 8 |
| 9 | 1ØØ1 | 9 |
| A | 1Ø1Ø | 10 |
| B | 1Ø11 | 11 |
| C | 11ØØ | 12 |
| D | 11Ø1 | 13 |
| E | 111Ø | 14 |
| F | 1111 | 15 |

Zo ontstaat de hexadecimale (zestientallige) schrijfwijze (van een byte): de byte van de bovenkant van de lokomotief van figuur 3 is F1. Volledige gegevens staan bij figuur 3.
De tien bytes per voorwerp worden eerst opgeslagen in de RAM en daarna in het geheugen van de PVI en bewaard totdat de CPU de opdracht geeft dat het 'opgeslagen' voorwerp in beeld moet worden gebracht. Het geheugen bestaat uit een aantal geheugenplaatsen, één voor iedere byte. Om te weten waar iets is opgeslagen is elke plaats van een adres voorzien.
De tien bytes van het eerste voorwerp staan in de PVI op de geheugenplaatsen 1FØØ . . . 1FØ9. Er worden 13 bits gebruikt om alle geheugenplaatsen te kunnen adresseren. Dit betekent dat er in de hexadecimale schrijfwijze vier cijfers of letters nodig zijn voor de adressering).
De volgende vier geheugenplaatsen ('lokaties' in het jargon), 1FØA . . . 1FØD, worden gebruikt om de informatie over de horizontale en vertikale positie van het voorwerp op het scherm in op te slaan, evenals de positie van een (de) duplikaat(en). De adressen 1FØE en 1FØF doen dienst als 'scratchpad memory' (kladblok). Vergeet die voorlopig maar weer.
Het tweede voorwerp wordt vastgelegd op de plaatsen 1F1Ø . . . 1F19, enzovoorts.
We noemden het al even: de positie van de voorwerpen op de beeldbuis. Dit gaat zo: het tv-scherm wordt verdeeld in 227 horizontale en 252 vertikale eenheden. De positie van het voorwerp wordt bepaald door het aantal eenheden dat de linker bovenhoek van de 8 bij 10-rechthoek in horizontaal en vertikaal opzicht is verwijderd van de linker bovenhoek van het scherm. De horizontale positie van het eerste voorwerp komt op plaats 1FØA, de vertikale op 1FØC. Van een duplikaat komt de horizontale positie te staan in 1FØB, de relatieve vertikale positie ('verschuiving' ten opzichte van het origineel) in 1FØD.
Byte 1FCØ is gereserveerd voor infomatie over de afmetingen van het voorwerp. Per voorwerp zijn twee bits nodig, dus vier verschillende mogelijkheden voor de afmetingen. Om precies te zijn: ØØ geeft de kleinste afmetingen, Ø1 een maatje groter (2x), 1Ø 4x groter en 11 acht maal zo groot.
Informatie over de kleur van de voorwerpen ligt verankerd in de bytes 1FC1 en 1FC2. Drie bits van 1FC1 bepalen van voorwerp 1 of de drie primaire kleuren al dan niet aanwezig zijn; drie andere van IFC1 idem voor voorwerp 2. Op dezelfde manier wordt byte 1FC2 gebruikt voor de kleuren van de voorwerpen 3 en 4. Twee bits van voornoemde bytes worden niet gebruikt.
Om een lang verhaal kort te maken: We weten nu welke informatie de PVI nodig heeft. Met behulp van de monitor-software (in de ROM opgeslagen) wordt deze nodige informatie opgeslagen in de RAM. De CPU kan worden opgedragen om de informatie van de RAM in de PVI te kopiëren, hetgeen het gewenste plaatsje oplevert. De informatie wordt via het toetsenbord ingetypt (zie tabel 3). Eerst wordt het geheugengedeelte met adressen van ØACØ t/m ØACA gewist. Dan wordt de bij een gewenst voorwerp horende informatie in de RAM geladen (beginnend bij ØAØØ). Dan start een kort programma dat de CPU opzadelt met het kopiëren van de betreffende RAM-inhoud naar de PVI en het gevolg van al deze digitale magie is, geloof het of niet, een plaatje op het scherm, met het ingetypte voorwerp.

#### Achtergrond

Het koderen, programmeren van de achtergrond verloopt in grote trekken hetzelfde als bij een voorwerp. De achtergrond bestaat uit 160 rechthoeken: 10 horizontale rijen van 16. Iedere rechthoek is gedefiniëerd door de boven- en de linker-rand. Totaal dus 320 zijden. Een zijde wordt geselekteerd door een bijbehorend bit 1 te maken. De 16 zijden van een rij vergen twee bytes voor de horizontale zijden en twee bytes voor de vertikale zijden. Totaal vier bytes per rij. We hadden 10 rijen, dus totaal zijn 40 bytes nodig voor alle rechthoekzijden.
Zoals figuur 4 aangeeft lopen de adressen van deze 40 bytes van 1F8Ø en 1F81, voor de boven-zijden van de bovenste rij tot 1FA6 en 1FA7 voor de

linker-zijden van de laagste rij. Een voorbeeld: laden we de bytes 1F8Ø en 1F81 elk met FF (1111 1111), dan verschijnen van de eerste rij alle bovenzijden in beeld.
Eén bit bepaalt of de bovenzijden met volle lengte of slechts als een stip in de linker-bovenhoek verschijnen. Een tweede en derde bit maken het mogelijk om de breedte van de bovenste en/of de onderste helft van de linker-zijde te kiezen, als (vertikale) lijn of als horizontale balk die de volle breedte van het vierkantje beslaat.
De volgende rij rechthoeken gaat op dezelfde manier (3 bits). Tenslotte kan met twee bits worden vastgelegd of de horizontale en vertikale zijden van de eerste twee zijden een dikte hebben van een kwart of de helft van de breedte van een vierkant, mits zij niet al door een van de andere bits op volle breedte zijn ingesteld.
Met andere woorden: met 8 bits (één byte) wordt de breedte van alle zijden van de eerste twee rijen vierkanten vastgelegd. Voor de tien rijen zijn dus vijf bytes nodig (1FA8 . . . 1FAC).
Figuur 4 geeft een overzicht van de mogelijkheden. De hexadecimale informatie links in figuur 4 geeft aan welke lijnen er verschijnen en wordt opgeslagen op de bijbehorende 40 adressen. Rechts staan de adressen en de geheugeninhoud (hexadecimaal weergegeven) van de vijf zojuist behandelde 'breedte-bytes'.
Voor de kleur van de achtergrond is het byte met adres IFC6 in de PVI gereserveerd. Van dit byte worden de drie meest linkse bits gebruikt om het al dan niet aanwezig zijn van de drie primaire kleuren voor het scherm (achtergrond tussen de lijnen) vast te leggen. Vervolgens een bit dat de achtergrondlijnen in of uitschakelt. Dit bit moet 1 zijn, wil er een achtergrond zichtbaar zijn ('background enable'). Vervolgens drie bits die de kleur van de achtergrondlijnen bepalen. Eén bit van het byte blijft ongebruikt.
Het samenspel van toetsenbord, RAM, CPU en PVI leidt ertoe dat ook nu een plaatje op het beeld verschijnt, dat naast de voorwerpen nu ook de achtergrond laat zien.

### Score

Het in beeld brengen van de score gaat zeer simpel in zijn werk. Het gaat om vier cijfers; de eerste twee staan in de PVI op de geheugenplaats met adres 1FC8, het andere stel cijfers op 1FC9. Twee bits van byte 1FC3 leggen de positie en het display-type (2 x 2 of 1 x 4) vast. Het meest rechtse bit van 1FC3 bepaalt de positie: Ø boven in het beeld, 1 onder in het beeld. Het op één na rechtse bit van 1FC3 is Ø als de score in twee maal twee cijfers moet plaatsvinden, en 1 als er sprake moet zijn van één getal van vier cijfers. De kleur is complementair aan de achtergrond kleur. Aan de afmetingen van het score-display valt verder niets te sleutelen.

**Tabel 3.**

Dit programma kan gebruikt worden bij het samenstellen van een figuur op het beeldscherm.
De informatie wordt als volgt via het keyboard ingegeven.

| TOETS | SCHERM | TOELICHTING |
|---|---|---|
| reset; start | IIII | Start monitor-programma. |
| MEM | Ad= | Gebruiker: 'Ik wil een programma in het geheugen opslaan'. Computer: 'Wat is het eerste adres?'. |
| Ø; 9; Ø; Ø | Ad= Ø9ØØ | Het eerste adres is Ø9ØØ. |
| + | Ø9ØØ xx | Geef data ('xx' is de reeds in dit adres aanwezige data) |
| Ø; 5 | Ø9ØØ Ø5 | Het eerste data-byte is Ø5. |
| C; A | Ø9Ø1 CA | Het volgende is CA. |
| Ø; 6 | Ø9Ø2 Ø6 | enz. |

Typefouten kunnen gemakkelijk gekorrigeerd worden: de —toets kan gebruikt worden om een stap terug te doen. Stel dat de eerste twee data-bytes als volgt ingegeven waren:

| | | |
|---|---|---|
| Ø; 5 | Ø9ØØ Ø5 | Tot zover niets aan de hand. |
| C; B | Ø9Ø1 CB | Fout! |
| — | Ø9ØØ Ø5 | Een stap terug. |
| + | Ø9Ø1 CB | |
| C; A | Ø9Ø1 CA | Dat moest het zijn. |
| Ø; 6 | Ø9Ø2 Ø6 | enz. |

Het komplete programma ziet er als volgt uit:

| ADRES | DATA | VERKLARING |
|---|---|---|
| ØØØØ | Ø5CA | |
| Ø9Ø2 | Ø6CA | Voorzie de PVI van data uit de RAM op de adressen ØA00 tot ØACA. |
| Ø9Ø4 | ØD4AØØ | |
| Ø9Ø7 | CD7FØØ | |
| Ø9ØA | FA78 | |
| Ø9ØC | ØC1E88 | Keer terug naar het monitor-programma indien de —toets is gebruikt. |
| Ø9ØF | 4410 | |
| Ø911 | 9979 | |
| Ø913 | 1FØØØØ | |
| *Ø916* | Ø4ØØ | Sla ØØ op in alle RAM-adressen van ØA00 tot ØACA (m.a.w. wis dit geheugengedeelte uit). |
| Ø918 | Ø5CA | |
| Ø91A | Ø6CA | |
| Ø91C | CD4AØØ | |
| Ø91F | FA7B | |
| Ø921 | Ø4FF | Onderdruk de score-uitlezing (door een 'onmogelijk' getal, bijv. FF, in de score-bytes ØAC8 en ØAC9 op te slaan). |
| Ø923 | CCØAC8 | |
| Ø926 | CCØAC9 | |
| Ø929 | Ø4Ø9 | Maak het scherm blauw (door Ø9 in byte ØAC6 op te slaan). |
| Ø92B | CCØAC6 | |
| Ø92E | Ø4Ø2 | Bepaal de afmetingen van de figuur (Ø2 = 1 : 2). Ga naar het eerste gedeelte van het programma (laad de PVI): start-adres Ø9ØØ. |
| Ø930 | CCØACØ | |
| Ø933 | 1FØ9ØØ | |

Nadat het programma tot zover is ingegeven, kan de PVI 'schoon gemaakt' worden met behulp van het 'wis'-programma dat op adres Ø916 als volgt gestart kan worden:

| TOETS | SCHERM | TOELICHTING |
|---|---|---|
| PC | PC = ØØØØ | 'Ik wil een programma laten lopen . . . |
| Ø; 9; 1; 6 | PC = Ø916 | . . . met start-adres Ø916'. |
| + | (blauw) | Het programma loopt. |
| — | PC = Ø916 | Terug naar het monitor-programma. |

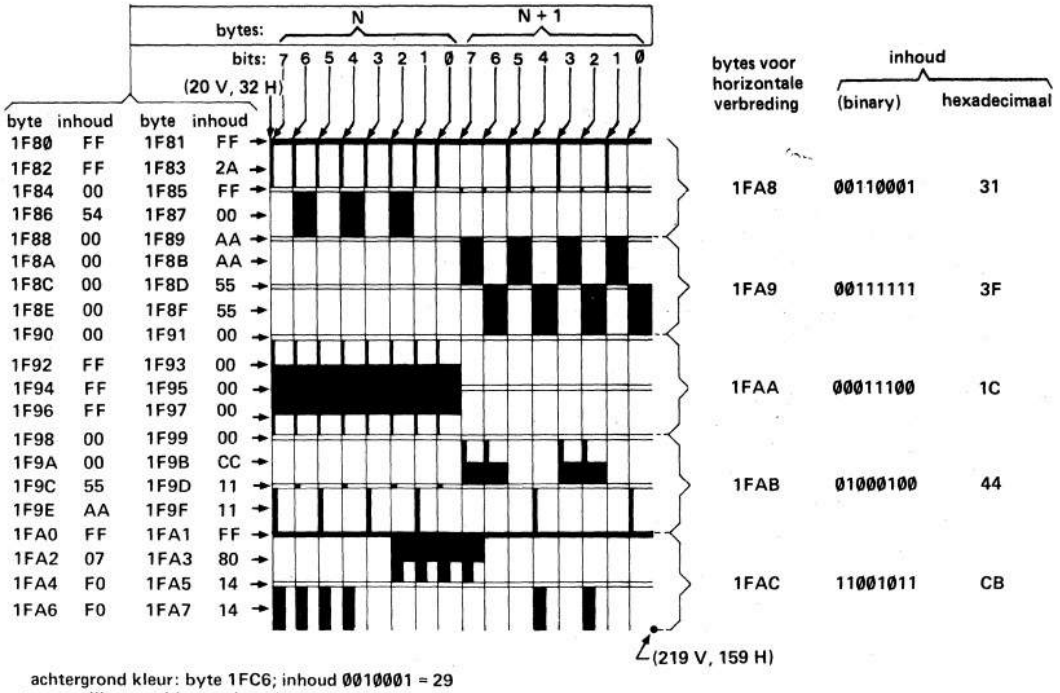
De rest van het programma kan nu opgeslagen worden door weer dezelfde procedure als voorheen te volgen: MEM, gevolgd door het eerste adres (ØA00), + en dan de data (F1, 51, enz.).

| ADRES | DATA | VERKLARING |
|---|---|---|
| ØA00 | F1 | |
| ØA01 | 51 | |
| ØA02 | 55 | |
| ØA03 | 75 | |
| ØA04 | 7F | Deze data beschrijft de vorm van de eerste figuur, de lokomotief (figuur 3). |
| ØA05 | FF | |
| ØA06 | FF | |
| ØA07 | C3 | |
| ØA08 | A5 | |
| ØA09 | 24 | |
| ØA0A | 80 | Horizontale positie van figuur. |
| ØA0B | FF | Horizontale positie van duplikaat (buiten beeld). |
| ØA0C | 4F | Vertikale positie van figuur. |
| ØA0D | FF | Vertikale verschuiving van duplikaat (buiten beeld). |

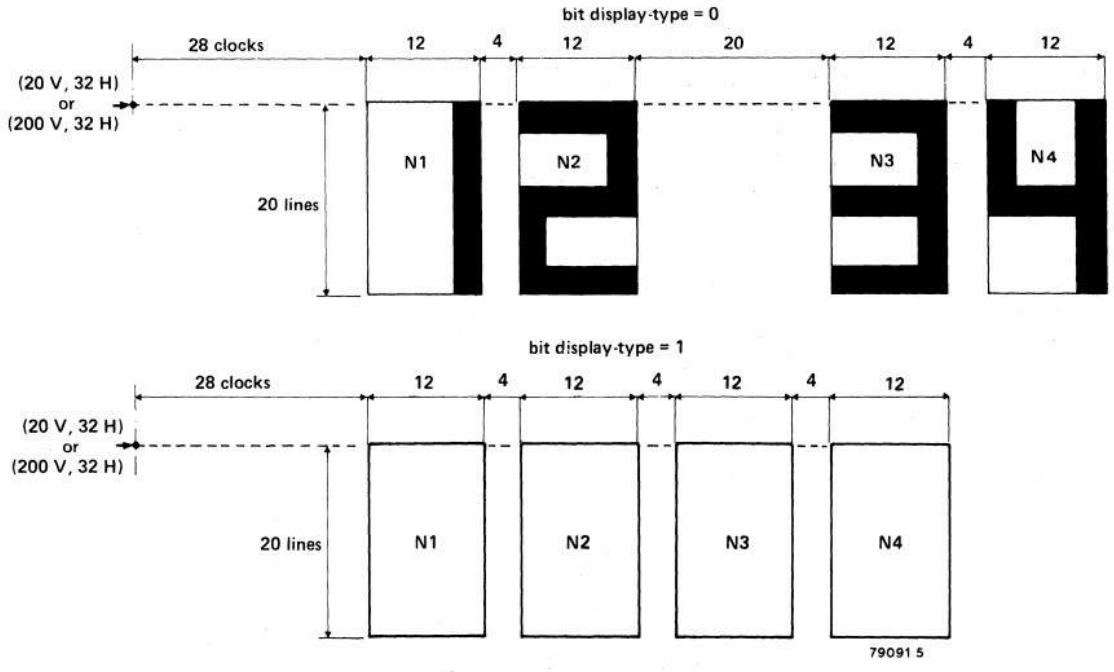
Deze figuur kan nu op het beeldscherm weergegeven worden door het laad-programma voor de PVI te starten op adres Ø9ØØ. Druk weer de volgende toetsen in: PC; Ø; 9; Ø; Ø; +.

Nadat de — en de MEM-toets ingedrukt zijn, kunnen nog drie figuren in het geheugen opgeslagen worden (beginnend op adres ØA1Ø, ØA2Ø en ØA4Ø). Ook kan er nog een achtergrond toegevoegd worden (begin-adres ØA8Ø), de afmetingen en de kleuren van de figuren kunnen worden gewijzigd (ØACØ . . . ØAC2), de achtergrond- en de beeldscherm-kleuren kunnen gekozen worden (adres ØAC6 — de achtergrond verschijnt alleen indien het tweede data-getal 8 . . . F is) en tenslotte bepaalt de data op adres ØAC7 het geluid. Het is mogelijk om met de + en —toets het geheugen stap voor stap af te vragen zonder dat de inhoud wijzigt. Indien er teveel fouten zijn ontstaan, kan het 'wis'-programma gestart worden (adres Ø916).

**4**




**5**




**Figuur 4. De achtergrond wordt opgebouwd uit een skala van horizontale en vertikale lijnen, vierkanten en 'punten'.**

**Figuur 5. De weergave van de score: twee groepen van twee cijfers of één groep van vier cijfers.**

**Botsingen**

Botsingen kunnen, indien gewenst, de score en de ten gehore gebrachte geluidseffekten beïnvloeden. Zodra er een botsing optreedt tussen een voorwerp en de achtergrond wordt een bepaald voor dit doel gereserveerd bit 1. Voor elk der vier voorwerpen is één van de vier meest linkse bits van byte 1FCA gereserveerd. Op dezelfde manier worden zes bits van byte 1FCB gebruikt om botsingen tussen voorwerpen te signaleren (bij vier voorwerpen zijn er zes verschillende botsingen tussen voorwerpen mogelijk).
De bytes 1FCA en 1FCB kunnen indien nodig worden gelezen door de CPU en worden gebruikt om de score aan te passen, de bewegingsrichting van een voorwerp te veranderen, geluidseffekten te produceren enzovoorts.

**Geluidseffekten**

Er wordt een blokgolf gegenereerd, waarvan de frekwentie afhangt van een getal, dat gelijk is aan het acht-bits getal dat is opgeslagen in byte 1FC7 in de PVI. Is het getal nul (ØØØØØØØØ), dan wordt er geen geluid geproduceerd. In alle andere gevallen is de frekwentie gelijk aan

$$f0 = \frac{7874}{n+1}\ \text{Hz},$$

waarbij n het decimale getal is dat in binaire vorm in byte 1FC7 zit. Hoe hoger dit getal, des te lager de frekwentie. Voor n = ØØØØØØØ1 is de frekwentie het hoogst: bijna 4 kHz. Voor de hoogste n, 11111111 = 255 is de frekwentie het laagst: een dikke dertig hertz.
In een later stadium kunnen andere geluidseffekten worden gerealiseerd onder gebruikmaking van bepaalde uitgangssignalen van de tv-speelkomputer, die externe geluidseffekten-generatoren 'triggeren'.

## Aanwijzingen voor het gebruik

Nadat we aan de hand van de voorgaande ruwe schets hebben gezien wat het apparaat allemaal kan en hoe dat in zijn werk gaat is de volgende stap om hem aan het werk te zetten. Vooropgesteld dat het apparaat is gebouwd, getest en afgeregeld, en op het tv-toestel is aangesloten, zoals in een ander artikel in dit nummer is beschreven, wordt het hoog tijd om met het intypen van instrukties te beginnen, via het toetsenbord (figuur 6).
De belangrijkste 'routine' (jargon voor een programma) die we moeten leren kennen is die welke het mogelijk maakt om programma's van de band of de plaat naar het RAM-geheugen van de tv-speel-computer te transporteren (het 'laden' van een programma).
Na het inschakelen van het apparaat en het indrukken van de toetsen RESET en START moet er linksonder in het scherm IIII te zien zijn. De computer is nu klaar voor programmering.
Ieder programma op band of plaat wordt vergezeld door een 'file-nummer' (file betekent hier dossier), een cijfer ongelijk aan Ø dat een bepaald programma identificeert.
Nadat dus RESET en START zijn ingedrukt wordt toets RCAS (Read CASsette, lees cassette) ingedrukt. De computer vraagt welk programma moet worden geladen. Er verschijnt op het scherm het volgende:

FIL =

Nu wordt het gewenste filenummer ingetypt, bijvoorbeeld 3.
Druk de + toets in. Het filenummer verschijnt in beeld. In ons voorbeeld:

FIL + 3

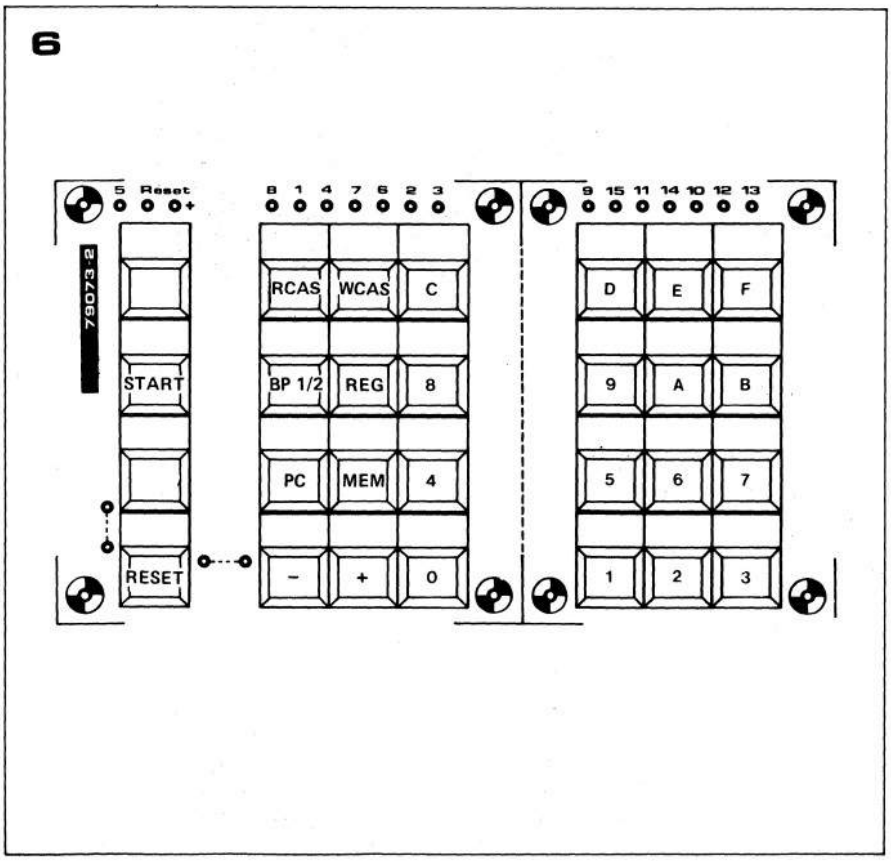
De kassette wordt gestart, de band met de programma's gaat lopen. Men hoeft niet persé aan het begin van de band te beginnen, uiteraard wel enige tijd vóór het bandgedeelte dat het gewenste programma bevat.
Het apparaat gaat nu 'op zoek' naar het juiste programma. Worden eerst andere filenummers gevonden, dan verschijnen deze achtereenvolgens in beeld, waarbij twee punten onder het plusteken snel aan en uit gaan. Is het korrekte filenummer gevonden, dan wordt het bijbehorende programma geladen. De punten onder het plusteken gaan nu langzaam aan en uit. Is het programma geheel ingeladen dan ziet men:

PC = (vier cijfers)

De vier cijfers vormen het zogenaamde startadres van het programma. Nadat ze te zien zijn wordt de +toets ingedrukt en het spelprogramma is klaar voor aktie.
Mocht er tijdens het inlezen van het programma een fout optreden, dan wordt het laden gestopt en verschijnt er een foutmelding in beeld:

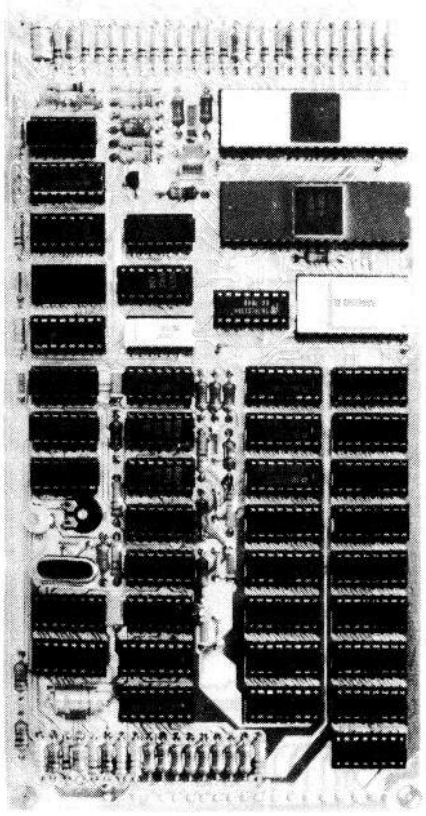
Ad = (vier karakters)

De vier karakters (letters en cijfers) hebben betrekking op het beginadres (in hexadecimale vorm) van de groep instrukties ('blok') waarbinnen de fout optrad. In zo'n geval moet de band worden teruggespoeld en de hele procedure herhaald.
Tenzij men zelf programma's gaat schrijven zijn de andere toetsen van het toetsenbord niet nodig. De toetsen staan uitgebreid beschreven in de al eerder genoemde bijlage. Ondanks dat nu een snelle oriëntatie:
-WCAS (Write CASsette), voor het schrijven van programma's op de band. Na het indrukken van de toets wil de computer weten wat het beginadres is van het programma (BEG =), vervolgens zou hij dolgraag het laatste adres willen weten (End=). Is hij op dat punt tevreden gesteld, dan vraagt hij het startadres (Sad=) en het filenummer (FIL=).
-BP1/2, voor het invoeren van maximaal twee 'break points' in het programma, een mogelijkheid om een programma op een van te voren bepaalde plaats te onderbreken, hetgeen handig is bij het 'debuggen' ('ontluizen'; jargon voor




Figuur 6. Het toetsenbord.

foutloos maken).
-REG, voor het kontroleren en eventueel wijzigen van de inhoud van de zeven 8-bit-registers en het 16-bit-status-register in de CPU.
-PC, Program Counter, programmateller, voor het bereiken van het startadres van een programma.
-MEM, de geheugentoets. Is nodig om zelfgemaakte programma's in het geheugen op te slaan. Is ook gebruikt in het programmeervoorbeeld van tabel 3.
Tot zover het toetsenbord en tot zover dit inleidend artikel. Rest ons te hopen dat iedereen net zoveel plezier beleeft van de speelse tv-speel-computer als wij nu al hebben gehad!





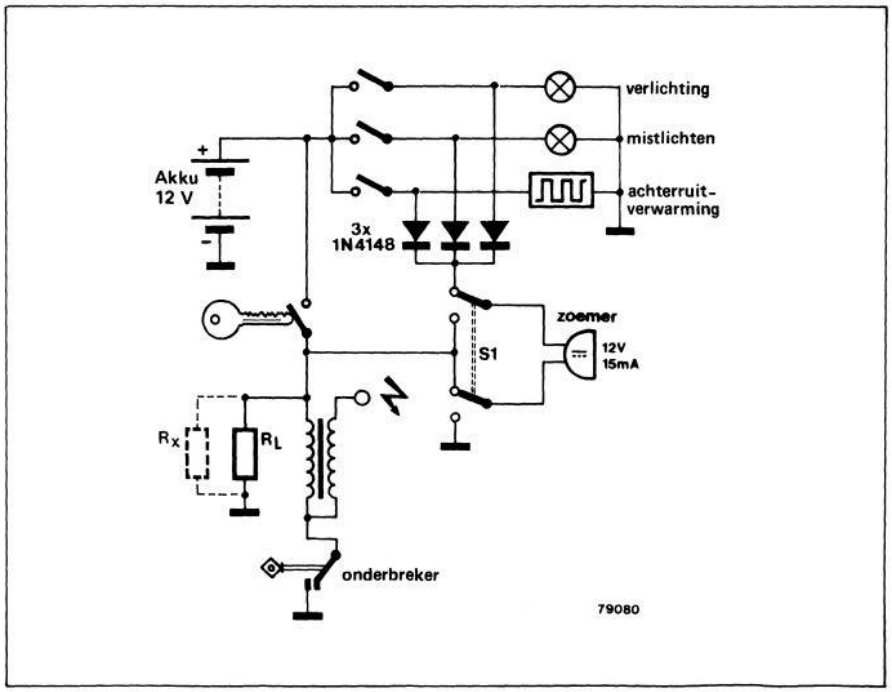


# auto uit licht uit

M. Penrose

Ontwerpen voor autolichtalarmen, die de in gedachten verzonken automobilisten bij het verlaten van hun voertuig er aan herinneren dat de lichten nog niet zijn uitgeschakeld, zijn er te kust en te keur. Een groot voordeel van het hier beschreven ontwerp is dat er geen extra komponenten in serie met de bestaande bedrading opgenomen hoeven te worden, waardoor er geen kans bestaat dat de deugdelijkheid hiervan nadelig beïnvloed wordt. Bovendien ziet het schema er allesbehalve ingewikkeld uit. De gehele schakeling bestaat slechts uit een gelijkstroomzoemertje, een dubbelpolige omschakelaar en een aantal dioden (afhankelijk van het aantal verlichtingsgroepen en/of verbruikers dat gekontroleerd dient te worden).
In bijgaand schema is aangegeven hoe bijvoorbeeld de normale wegverlichting, de mistlichten en de achterruitverwarming gekontroleerd kunnen worden.
Let wel, de schakeling geeft niet aan of genoemde zaken ook werkelijk intakt zijn! Indien schakelaar S1 in de getekende stand staat, wordt de zoemer aktief zodra de motor via het kontaktslot uitgeschakeld wordt, terwijl nog één of meer verbruikers ingeschakeld zijn.

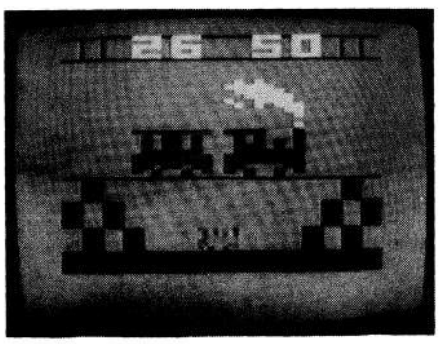
Door het uitschakelen van de desbetreffende verbruiker(s) wordt de zoemer het zwijgen opgelegd. Wil men echter toch een bepaalde verbruiker (bijv. parkeerlichten) ingeschakeld laten, dan dient S1 in de parkeerstand gezet te worden. De zoemer is nu buiten werking, totdat de motor gestart wordt. Door S1 dan om te schakelen, staat het alarm weer op scherp.
Normaliter is parallel aan het ontstekingssysteem een voldoend lage belasting ($R_L$) aanwezig (in de vorm van diverse kontrolelampjes, benzinepeilmeter e.d.), waardoor de zoemer ook funktioneert indien de onderbreker toevallig in de geopende stand staat nadat de motor tot rust is gekomen. Mocht deze belasting echter te hoogohmig zijn, dan kan er een weerstand $R_X$ (2 watt) van 100 . . . 220 Ω aan parallel geschakeld worden. Energiebesparender is natuurlijk een lampje van ongeveer 0,1W/12V (als men dat op de kop kan tikken — niet te hard natuurlijk), aangezien de weerstand hiervan vanwege het PTC-gedrag toeneemt naarmate er meer vermogen gedissipeerd wordt.



# de speelcomputer gebouwd

## eenenveertig chips op één print

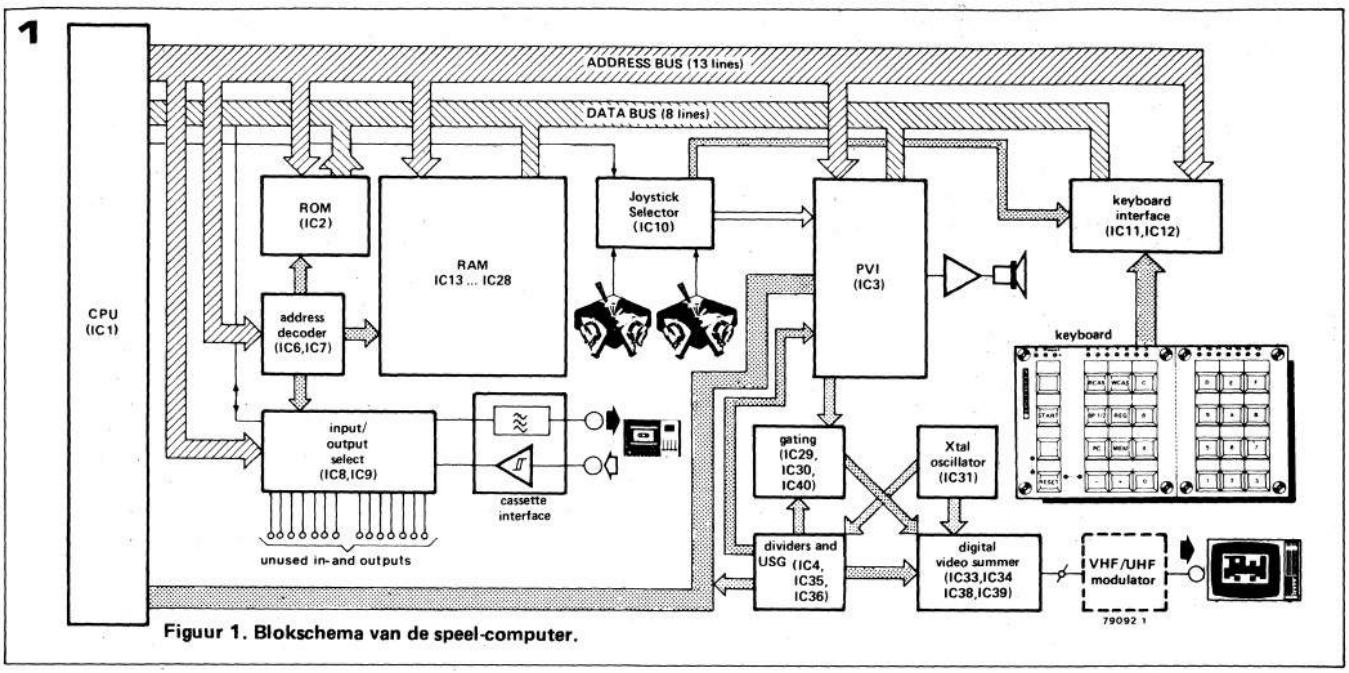
**Inleidende woorden zijn er op een andere plaats in dit blad al genoeg besteed aan het grootste projekt van deze maand: de tv-speel-computer. Uitgebreid is er beschreven hoe het kant en klare geval er uitziet, en wat het zoal presteert. Weinig woorden zijn er evenwel vuil gemaakt aan het inwendige van het kastje, laat staan over de bouw ervan. Daartoe dient dan ook dit artikel, dat zich na een oriënterende tocht door het schema bezighoudt met de konstruktie en afregeling van de speel-computer.**




Met uitzondering van de meest verknochte elektronica-liefhebber zal vrijwel iedereen die een eerste blik werpt op het gedetailleerde schema van figuur 2 een lichte neiging tot verbleken niet kunnen onderdrukken. Toch is het blokschema van figuur 1 nog niet zo'n verschrikking, en wanneer men dit heeft kunnen doorgronden beschikt men over een prima wegwijzer voor het uitgewerkte schema.
Zoals in het inleidende artikel al ter sprake is geweest, bestaat het eigenlijke 'brein' van de speel-computer uit de *microprocessor*-chip of *CPU*. Deze kan, door een aantal stuursignalen op de uit dertien verbindingen bestaande *adresbus* te zetten, alle andere onderdelen van de speel-computer sturen. De informatie-uitwisseling tussen de verschillende eenheden van de computer vindt plaats via de uit acht verbindingen bestaande *databus*; verder zijn er ook nog een aantal speciale stuursignalen die rechtstreeks van de CPU naar de verschillende eenheden gaan.
Een brein alleen begint niet veel. Het minste dat hij nodig heeft is een *geheugen*. De speel-computer heeft dan ook drie verschillende soorten geheugens ter beschikking. De eerste soort is een *read only memory* (ROM), waarin de voorgeprogrammeerde *monitor software* is opgeslagen. Voor de CPU is de ROM een soort gebruiksaanwijzing, waarin hij telkens opzoekt wat hij moet doen; de informatie in dit geheugen is vast en kan door de CPU niet veranderd worden. Naast de ROM is de speel-computer voorzien van een *random access memory* (RAM), waarin zowel informatie kan worden opgeslagen als eruit gehaald. De CPU gebruikt deze RAM voor het opslaan van het eigenlijke spelprogramma, en ook als een soort kladblok voor het tijdelijk opslaan van allerlei gegevens. Het derde geheugen is een gewone *cassette*- of *bandrecorder*, die gebruikt wordt als een soort archief, waarin zoveel spelprogramma's worden opgeslagen als men maar wil.
Welk van de drie geheugens op een gegeven moment in bedrijf is, wordt bepaald door de *adres-dekoder*, die op zijn beurt weer onder kontrole staat van de CPU. De exakte plaats in dat geheugen, waar informatie vandaan gehaald moet worden of moet worden opgeslagen, wordt bepaald door de CPU zelf. Het gewenste adres wordt door deze op de adresbus gezet.
Aangezien de meeste cassette- en bandrecorders zijn bedoeld voor het opnemen en weergeven van audio-signalen, vereist het wat overleg om ze te gebruiken voor de opslag van digitale informatie. De digitale uitgang van de computer naar de recorder moet wisselspanningsgekoppeld worden en de zeer hoogfrekwente komponenten die het digitale signaal bevat moeten er uit gefilterd worden. Omgekeerd moet het signaal van de recorder naar de computer opgepept en van allerlei ongerechtigheden ontdaan worden alvorens het als digitaal signaal bruikbaar is. Beide taken neemt de *cassette-interface* voor zijn rekening.

### Er komt nog meer bij kijken

De nu omschreven gedeelten maken deel uit van vrijwel elk computersysteem: *intelligentie* (de CPU) en *geheugen*. Tussen twee haakjes kan hier opgemerkt worden dat juist deze twee standaard-gedeelten van de computer in het hoofdschema van figuur 2 op de linker-pagina zijn ondergebracht. Met de standaardonderdelen alleen is de speel-computer echter nog niet tot veel in staat; hij moet nog op een of andere manier verbonden worden met de bedieningsorganen voor de speler(s) (toetsenbord en joy-sticks), een luidspreker en natuurlijk een televisie-toestel.




**Figuur 1. Blokschema van de speel-computer.**

Allereerst bespreken we de bedieningsorganen. De joy-sticks (stuurknuppels), zijn beide een kombinatie van twee potmeters en leveren dan ook analoge informatie. Om deze aan te passen op de digitale speel-computer, moet er een of andere vorm van analoog/digitaal-omzetting plaatsvinden. Hiervoor zorgt de *joy-stick-interface*, die in feite deel uitmaakt van de nog te bespreken PVI. Hoewel er twee joy-sticks zijn, is er slechts één dubbele interface beschikbaar en wordt er maar één joy-stick tegelijk door de computer als bedieningsorgaan beschouwd. Vandaar dat er nog een *joy-stick-selector* aanwezig is, die ervoor zorgt dat er tussen de joy-sticks kan geschakeld worden. Ook dit onderdeel staat uiteraard onder kontrole van het centrale brein, de CPU (via een aparte verbinding). Tevens is het verbonden met de *keyboard-interface*. Deze laatste zorgt, alweer onder leiding van de CPU, voor de juiste overdracht van informatie van het keyboard (toetsenbord) naar de databus.

Dit voor wat betreft de ingangen van de speel-computer. De uitgangen – naar televisietoestel en luidspreker – vergen heel wat meer. Gelukkig wordt het grootste deel van het werk gedaan door één IC, de *programmable video-interface* (PVI). Deze chip is te beschouwen als een soort slaaf-microprocessor: op kommando van de CPU neemt hij data op of geeft hij ze uit. Bovendien stelt hij vast of er zich bepaalde omstandigheden voordoen (met name het botsen van spelobjekten op het beeldscherm) en genereert hij de signalen voor het beeld en geluid. Het geluidsgedeelte is verder de eenvoud zelve: een buffertrap die een luidsprekertje stuurt.

Om het beeld te maken heeft de PVI een paar hulpschakelingen nodig. Een kristaloscillator levert het in frekwentie konstante signaal waarvan een aantal andere signalen wordt afgeleid. Via een aantal delertrappen komt er zo een clock-signaal terecht op nog zo'n chip die ons een hoop zorgen uit handen neemt: de *universal sync-generator* (USG). Deze verzorgt het hele komplex van synchronisatiesignalen die nodig zijn voor een kleuren-tv. Bovendien levert hij nog een aantal andere synchronisatiesignalen, waaronder het clock-signaal voor de microprocessor.

Een aantal uitgangssignalen van de USG gaat naar de PVI, zodat deze laatstgenoemde chip weet aan welk gedeelte van het beeld de tv-ontvanger bezig is. Op grond van die gegevens levert de PVI een aantal uitgangssignalen, die samen voor verschillende delen van het beeld de juiste kleur vastleggen, zodat achtergrond, objekten en score op de gewenste manier op het beeld verschijnen. Deze uitgangen van de PVI gaan, via een aantal poortschakelingen (*gating*) die weer door de USG gestuurd worden naar het laatste onderdeel van de eigenlijke computer: de *digital video-summer*. Uit de signalen van de kristaloscillator, de USG en de PVI stelt deze het uiteindelijke video-signaal samen. In tegenstelling tot wat de benaming zou doen denken, komt er nog wat meer bij kijken dan alleen het optellen van de ingangssignalen; er vindt ook nog frekwentiedeling, nivo-aanpassing en een aantal poortfunkties plaats.

## Wie niet waagt, die niet wint: het uitgewerkte schema

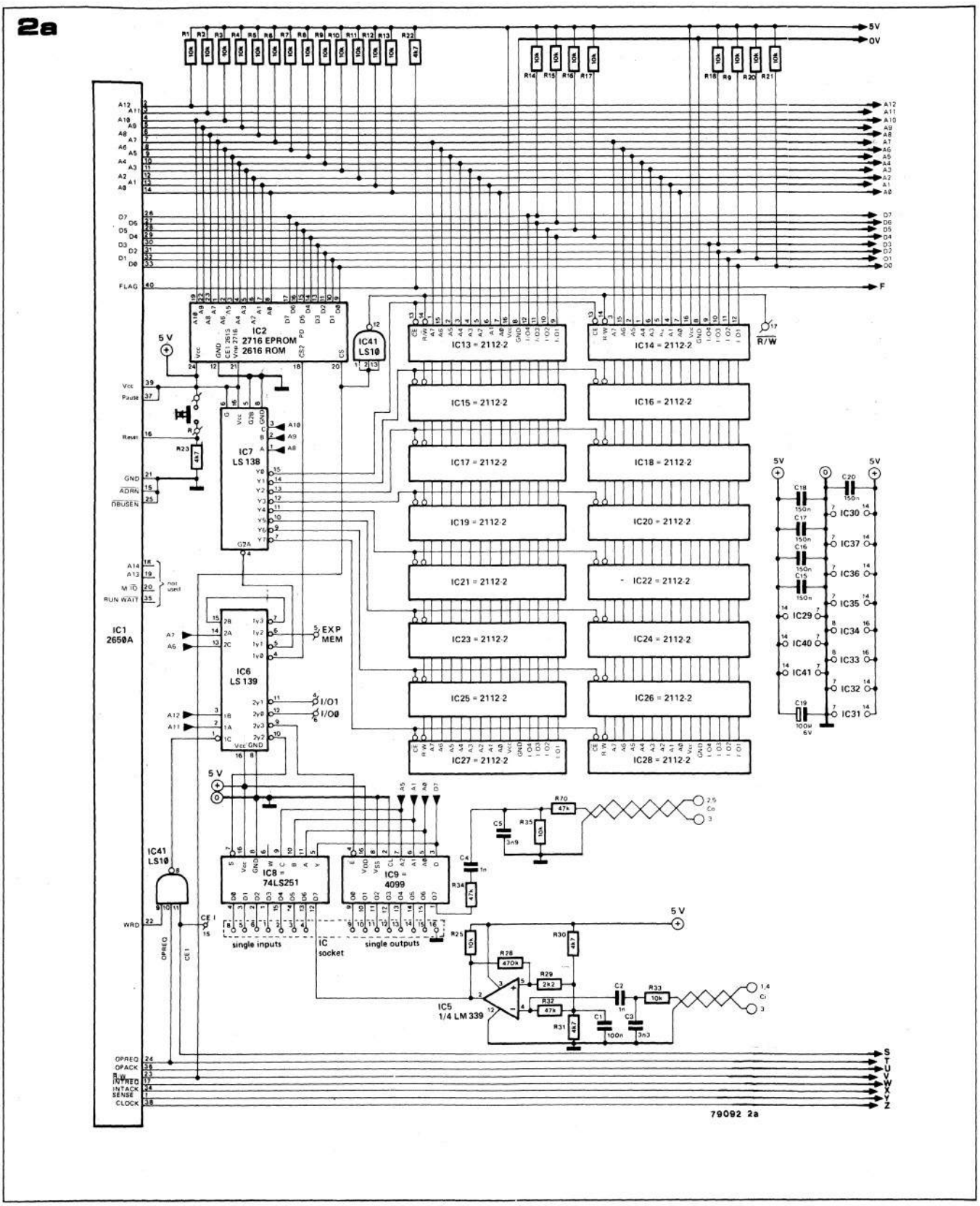
Nu we het blokschema hebben bekeken, kunnen we het erop wagen een blik te werpen op het gedetailleerde schema (figuur 2). Er is echter geen beginnen aan om ieder klein facetje uit en te na uit te pluizen. Veel houvast hebben we er al aan, dat de verschillende onderdelen uit het blokschema in het hoofdschema op dezelfde manier zijn gerangschikt.

De CPU vinden we helemaal links (IC1). Langs de bovenzijde van het hele schema lopen de adresbus en de databus. De adresdekoder voor het geheugen en de input/output-selectors (IC6 en IC7), de ROM (IC2) en de RAM (IC13 . . . IC28) spreken voor zich. Het enige punt waarop de aandacht gevestigd wordt is dat de 2112's die in de RAM gebruikt worden van het 450 ns-type (max. 500 ns) moeten zijn (of sneller). (De diverse fabrikanten van deze RAM's hanteren niet allemaal een gelijke snelheidsaanduiding.) De input/output-selectors (respektievelijk IC8 en IC9) voorzien de computer van acht seriële ingangen en acht uitgangen. In deze – voor uitbreiding vatbare – opzet van de speel-computer wordt echter van beide slechts één gebruikt; de overige zeven zijn voor latere doeleinden beschikbaar. Het uitgangsfilter van de cassette-interface bestaat uit drie weerstanden en twee kondensatoren; de ingangsbuffer is voorzien van een opamp om het signaal aan TTL-nivo aan te passen.

Tot zover de linkerhelft van de schakeling. Het is een zevenmijlslaarzentocht door de elektronica waarbij veel details blijven liggen, maar voldoende om een indruk te geven van wat er door wie gedaan wordt. Bij het EPS-printenpaket wordt meer en uitgebreider informatie geleverd.

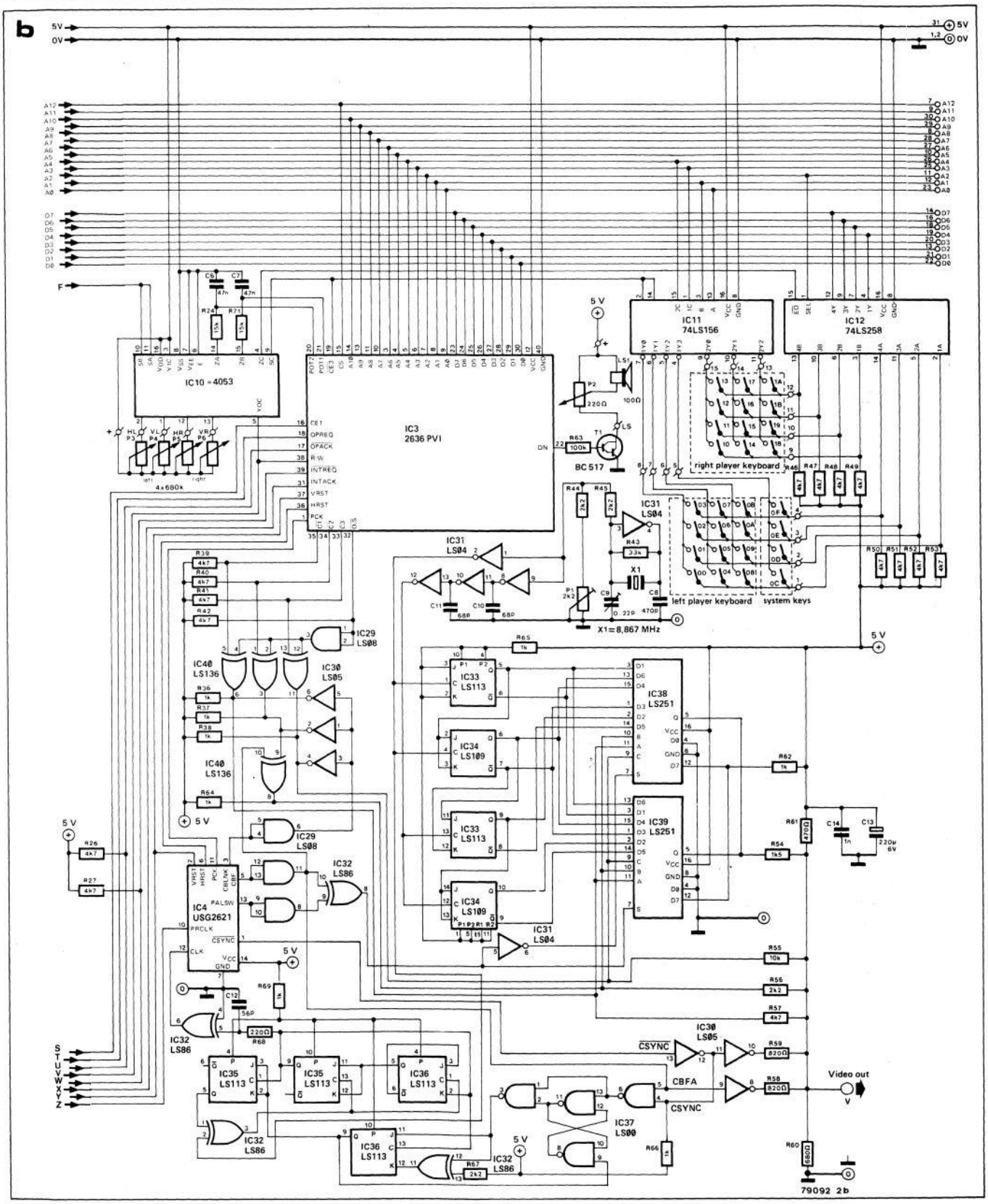
De rechterhelft van het schema is op het eerste gezicht (en niet alleen op het eerste . . .) een stuk ingewikkelder. Toch is het niet zo moeilijk aan de hand van het blokschema de belangrijkste gedeelten aan te wijzen. Het hart van dit deel is de PVI (IC3).

Naast de PVI treffen we de joy-stick-selector aan (IC10). Verder zien we de door zijn eenvoud bijna niet opvallende luidspreker-interface (T1) en het keyboard met de bijbehorende interface (IC11 en IC12). Verdere uitweiding over deze zaken valt buiten het kader van dit




**Figuur 2. Schema van de hoofdschakeling.**

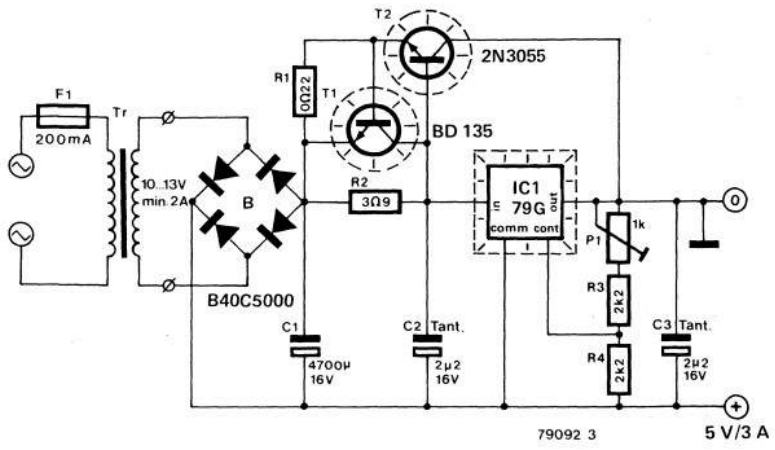
artikel. Het keyboard (of de *keyboards*, het is maar hoe je er tegenaan kijkt) zal nog bij de bouwaanwijzingen besproken worden.
De laatste loodjes wegen het zwaarst, en dat geldt ook voor de bespreking van dit schema. Schakelingen zoals die van het overgebleven deel op de rechterpagina moeten eigenlijk òf uitputtend, òf helemaal niet beschreven worden. Desondanks zullen we proberen hier een idee te geven van de gang van zaken, zonder te vervallen in een eindeloze reeks flips, flops, logische enen en nullen. Direkt beneden de PVI vinden we de kristaloscillator (IC31).
Eén van de door deze generator geleverde signalen onderneemt een lange reis door een reeks delertrappen, bestaande uit IC32, IC35 en IC36 om te eindigen als clock-signaal voor de USG (IC4). Deze USG is belangrijker dan




men uit zijn grootte in het schema zou konkluderen: de ingewikkelde synchronisatieschakelingen voor een (PAL) kleurentelevisietoestel zijn van deze chip afkomstig. Bovendien komt er een aantal referentiesignalen vandaan die nodig zijn voor de CPU en de PVI, terwijl de USG tevens zorgdraagt voor het schakelen van de video-uitgangen van de PVI. Bij dit laatste gebruikt het de diensten van een ingewikkeld netwerk van AND's, inverters en EXOR's (IC29, IC30 en IC40).
Wat er nu in het schema nog niet besproken is, is in het blokschema bij elkaar genomen als digital video-summer. Dit gedeelte (IC33, IC34, IC38, IC39 en een aantal NAND's en inverters) stelt het uiteindelijke videosignaal samen uit signalen van de kristaloscillator, het netwerk van poorten aan de uitgang van de PVI, en de USG. De juiste spanningsnivo's voor de verschillende signalen worden bepaald door de weerstanden R54 . . . R61.

De schakeling van figuur 2 is in zijn geheel ondergebracht op één print, waarover later meer. Behalve deze print zijn er nog twee schakelingen nodig: de voeding en (in de meeste gevallen) een VHF/UHF-tv-modulator.

3




**Onderdelenlijst voeding**

Weerstanden:

R1 = 0,22 Ω/3 W
R2 = 3,9 Ω
R3,R4 = 2k2
P1 = instelpotmeter 1k

Kondensatoren:

C1 = 4700 µF/16 V
C2,C3 = 2µ2/16 V tantaal

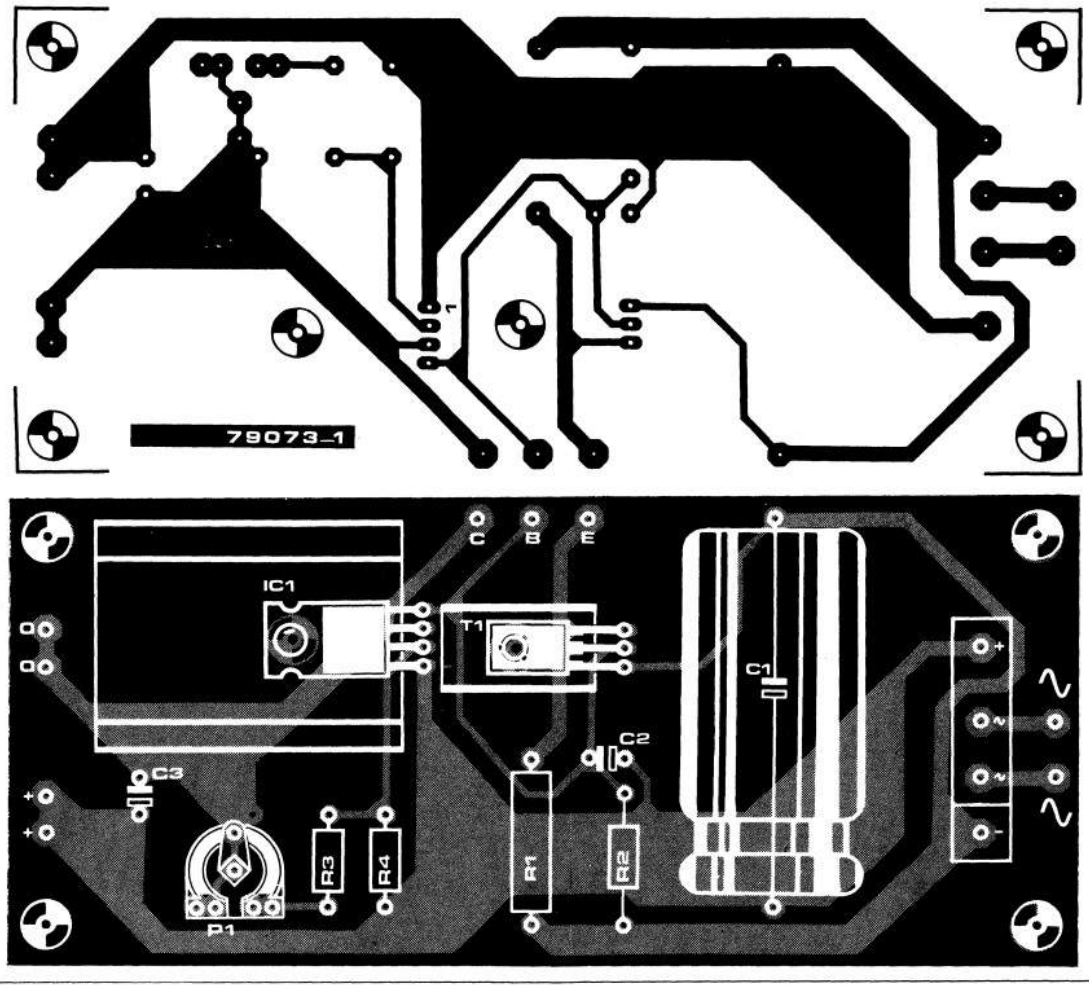
Halfgeleiders:

T1 = BD 135, BD 137, BD 139
T2 = 2N3055
IC1 = µA 79G
B1 = B40C5000

Diversen:

Tr1 = nettrafo, 10 V/2 A
F1 = smeltveiligheid 200 mA

4




## De boog kan niet altijd gespannen zijn: de voeding

Na het hoofdschema van de speelcomputer moet het schema van de voeding (figuur 3) wel als een verademing komen.

Hoewel de schakeling wat ongebruikelijk lijkt — met name voor wat betreft T1 en T2 — is het een gewone rechttoerechtaan-schakeling. Wanneer de door de voeding te leveren stroom toeneemt, zal de geïntegreerde spanningsregelaar IC1 proberen deze stroom zelf te leveren. Hierdoor zal de spanningsval over R2 groter worden, zodat T2 gaat geleiden; deze neemt dan het grootste deel van de te leveren stroom voor zijn rekening. T1 is aangebracht om de stroom door T2 ingeval van een kortsluiting tot een veilige waarde te beperken. De dissipatie in de spanningsregelaar wordt door het IC zelf binnen veilige grenzen gehouden. De print van de voeding is afgebeeld in figuur 4. Overigens is iedere andere gestabiliseerde voeding geschikt die bij 5 V een stroom kan leveren van 2 A.

## De VHF/UHF-modulator

Aan de VHF/UHF-tv-modulator gaan we hier niet teveel aandacht besteden. We hebben namelijk een zeer geschikt ontwerp ter beschikking dat al eerder in Elektuur is gepubliceerd, en wel in oktober 1978 (zie literatuur).

Het schema van de destijds uitvoerig beschreven modulator wordt hier als figuur 5 opnieuw afgedrukt, evenals de bijbehorende print (figuur 6). Voor de samenwerking met de speelcomputer, die immers ook met 5 V gevoed wordt, kan de spanningsregelaar IC1 vervallen en worden vervangen door een draadbrugje. De kans op storingen via de voedingslijn maakt men extra

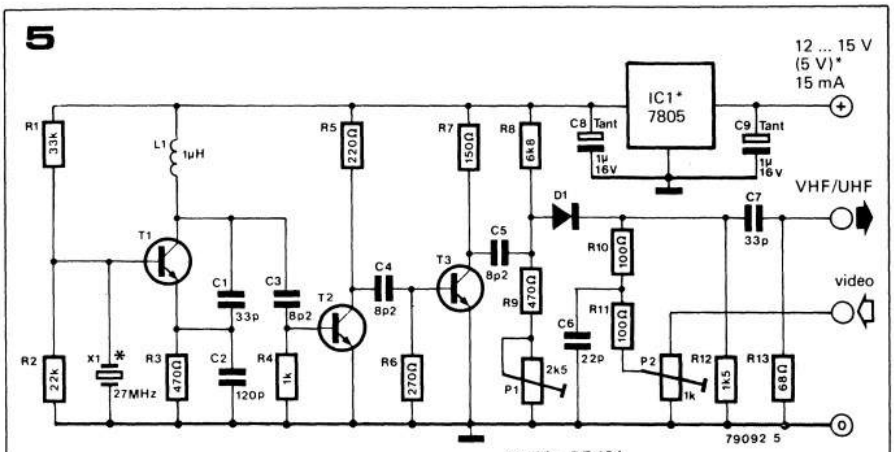
klein wanneer men in plaats van dit draadbrugje een zelfgemaakt spoeltje van een of enkele windingen (van dun montagedraad) door een ferrietkraaltje aanbrengt.
Wanneer men de speel-computer zou willen gebruiken samen met een van de vooralsnog zeldzame televisietoestellen met een video-ingang, kan de VHF/UHF-modulator eventueel vervallen, al doet men dan wel afbreuk aan de universele toepasbaarheid van de speel-computer. In dat geval kan de 'video out'-uitgang van de speel-computer rechtstreeks aan die video-ingang worden toegevoerd.

## De handen uit de mouwen

Het wordt tijd om wat opmerkingen te maken over de bouw van de speel-computer. Het apparaat bestaat uit vier printen en een aantal kleinigheden (joy-sticks, luidspreker). De printen bevatten het hoofdschema, het key-board (figuur 9), de voeding en de VHF/UHF-modulator. De onderlinge verbindingen tussen de delen van het apparaat zijn getekend in figuur 7.

De print waarop de schakeling van het hoofdschema kan worden ondergebracht (figuur 8) heeft wat toelichting nodig. Het is een zeer kompakte, dubbelzijdige print met doorgemetalliseerde gaten. Een fraai staaltje van wat de hedendaagse technologie vermag, maar helaas nog behept met de kinderziekten die kenmerkend zijn voor elke moderne techniek. De techniek bij het maken van doorgemetalliseerde gaten is nog niet zover gevorderd dat het voor 100% zeker is dat inderdaad alle gaten doorgemetalliseerd zijn (wanneer we de kosten tenminste binnen het redelijke willen houden). In de industrie heeft men met dit feit leren leven door eenvoudig de gemonteerde printen die niet blijken te werken af te keuren. De vrije-tijdselektronicus kan zich dat echter natuurlijk niet veroorloven, en het is dan ook ten zeerste aan te raden om de print *van te voren* te kontroleren op slechte doormetallisering. Bij een komplex apparaat als deze speel-computer is het achteraf opsporen van deze fout op zijn zachtst gezegd een tijdrovende onderneming. Het kontroleren kan op twee manieren gebeuren. In de eerste plaats visueel, door de print tegen het licht te

5




T1,T2 = BF 494
T3 = BFY 90
D1 = 1N4148
* zie tekst

**Onderdelenlijst VHF/UHF-tv-modulator**

Weerstanden:
R1 = 33 k
R2 = 22 k
R3,R9 = 470 Ω
R4 = 1 k
R5 = 220 Ω
R6 = 270 Ω
R7 = 150 Ω
R8 = 6k8
R10,R11 = 100 Ω
R12 = 1k5
R13 = 68 Ω
P1 = instelpotmeter 2k2 (2k5)
P2 = instelpotmeter 1k

Kondensatoren:
C1,C7 = 33 p
C2 = 120 p
C3,C4,C5 = 8p2
C6 = 22 p
C8,C9 = 1 µ/16 V tantaal

Halfgeleiders:
T1,T2 = BF 194, BF 195, BF 254, BF 255, BF 494, BF 495
T3 = BFY 90
D1 = 1N4148
IC1 kan vervallen (zie tekst)

Diversen:
L1 = 1 µH
X1 = kristal, ca. 27 MHz

**Figuur 3. De bijpassende 5 V-voeding.**

**Figuur 4. Print-layout en komponenten-opstelling van de voeding.**

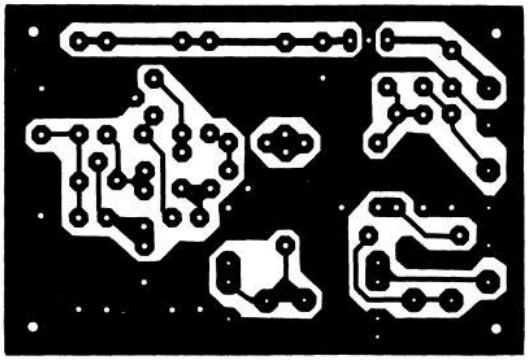
**Figuur 5. De VHF/UHF-tv-modulator, zoals hij al eerder in Elektuur stond. Voor deze toepassing kan het spanningsregel-IC vervallen.**

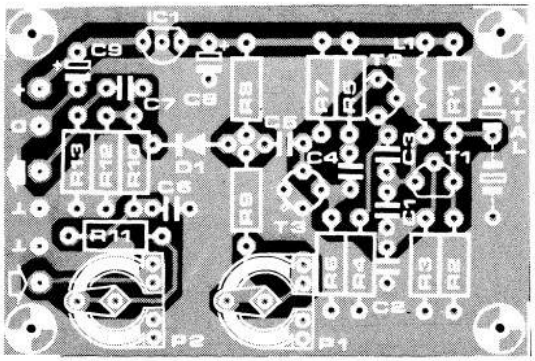
**Figuur 6. De modulatorprint. In plaats van IC1 dient een draadbrugje te worden aangebracht, of, wat nog beter is, een ferrietkraal-spoeltje (zie tekst).**

**Figuur 7. De onderlinge verbindingen tussen de verschillende delen van de speel-computer.**

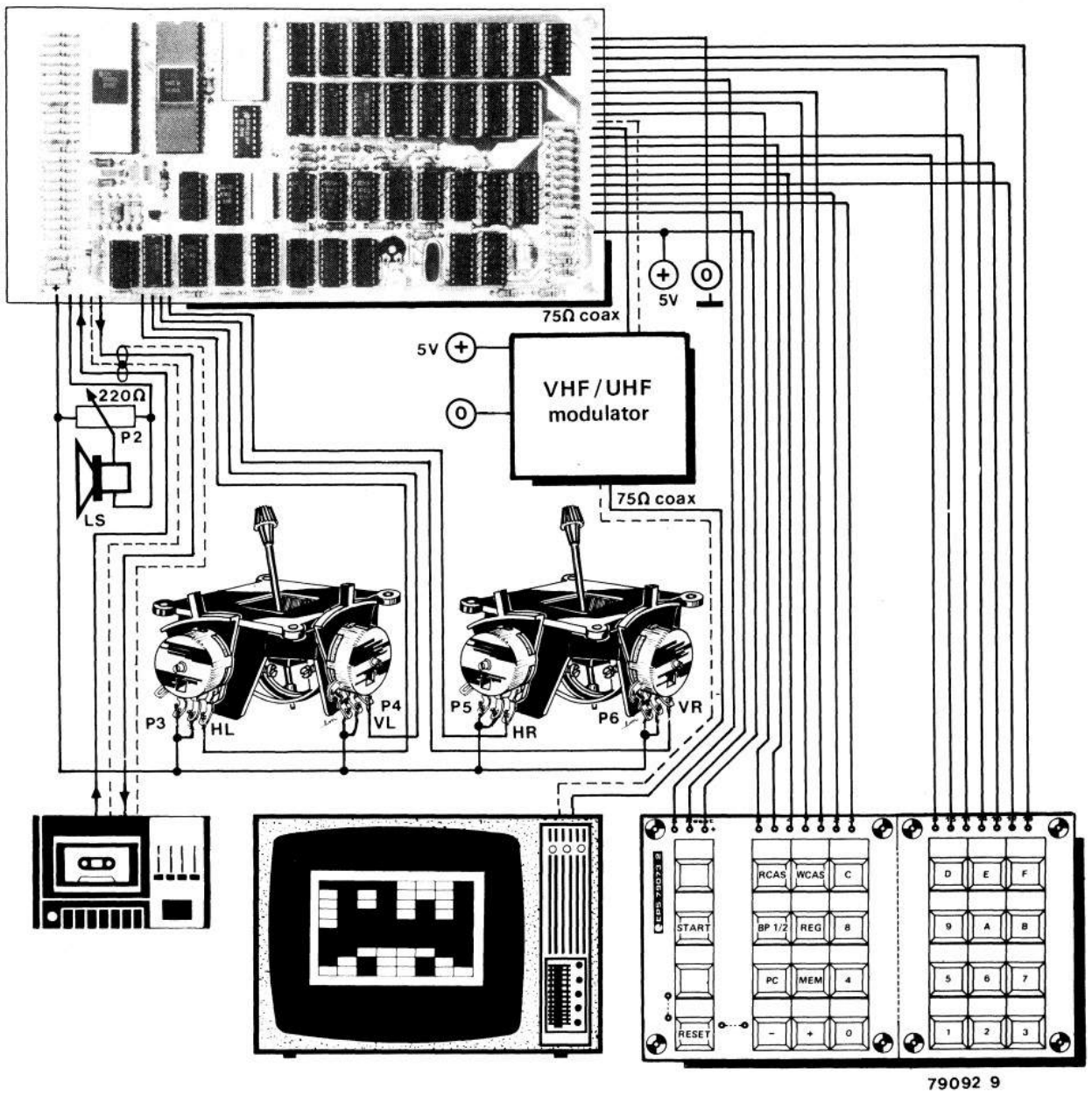
**Figuur 8. De dubbelzijdige print voor de hoofdschakeling. De gaten zijn doorgemetalliseerd. In deze afbeelding is de print verkleind tot 70% van zijn werkelijke grootte!**

6





7




79092 9

**Onderdelenlijst hoofdschakeling**

Weerstanden:

R1 . . . R21,R25,R33,R35,R55 = 10 k
R22,R23,R26,R27,R30,R31,R39 . . . R42
R46 . . . R53,R57 = 4k7
R24,R71 = 15 k
R28 = 470 k
R29,R44,R45,R56,R67 = 2k2
R32,R34,R70 = 47 k
R36,R37,R38,R62,R64,R65,R66,R69 = 1 k
R43 = 33 k
R54 = 1k5
R58,R59 = 820 Ω
R60 = 680 Ω
R61 = 470 Ω
R63 = 100 k
R68 = 220 Ω
P1 = instelpotmeter 2k2 (2k5)
P2 = potmeter 220 Ω (250 Ω)

Kondensatoren:

C1 = 100 n MKH
C2,C4,C14 = 1 n
C3 = 3n3
C5 = 3n9, evt. 3n3
C6,C7 = 47 n
C8 = 470 p
C9 = 0 . . . 22 p trimmer
C10,C11 = 68 p (ker.)
C12 = 56 p (ker.)
C13 = 220 µ/6 V
C15,C16,C17,C18,C20 = 150 n MKH
C19 = 100 µ/6 V

Halfgeleiders:

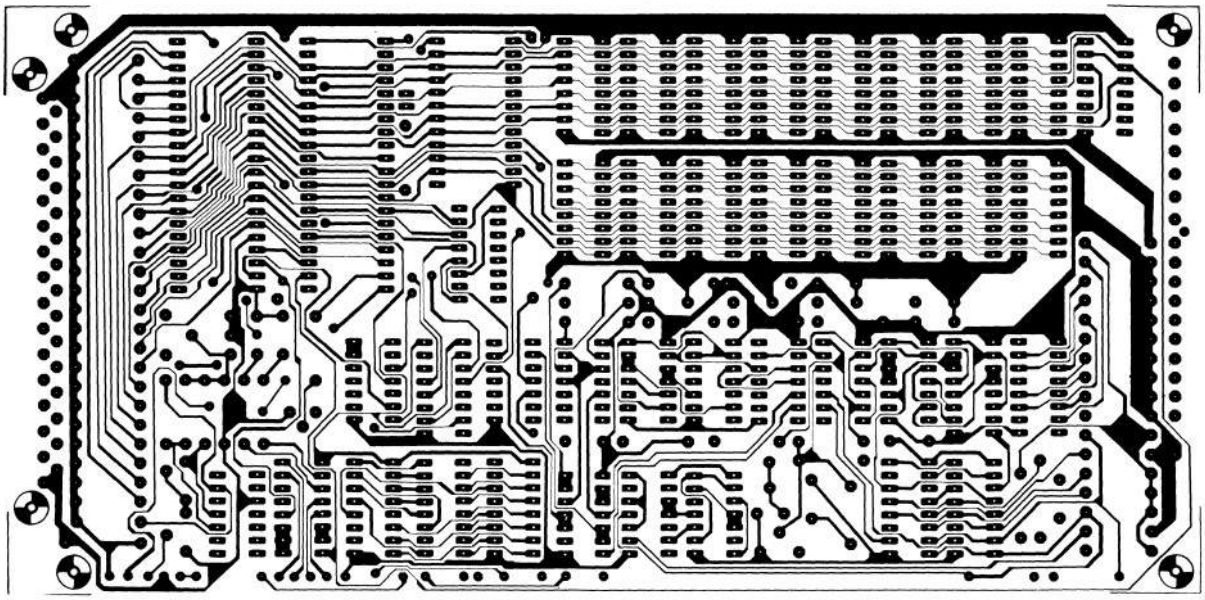
IC1 = 2650A (Signetics)
IC2 = 2616 (Signetics) met monitor-programma
IC3 = 2636 (Signetics)
IC4 = 2621 (Signetics)
IC5 = LM339 (National Semiconductor)
IC6 = 74LS139
IC7 = 74LS138
IC8,IC38,IC39 = 74LS251
IC9 = CD 4099
IC10 = CD 4053
IC11 = 74LS156
IC12 = 74LS258
IC13 . . . IC28 = MM2112–2, MM2112–4 (450 ns access time)
IC29 = 74LS08
IC30 = 74LS05
IC31 = 74LS04
IC32 = 74 LS86
IC33,IC35,IC36 = 74LS113
IC34 = 74LS109
IC37 = 74LS00
IC40 = 74LS136
IC41 = 74LS10
T1 = BC517

Diversen:

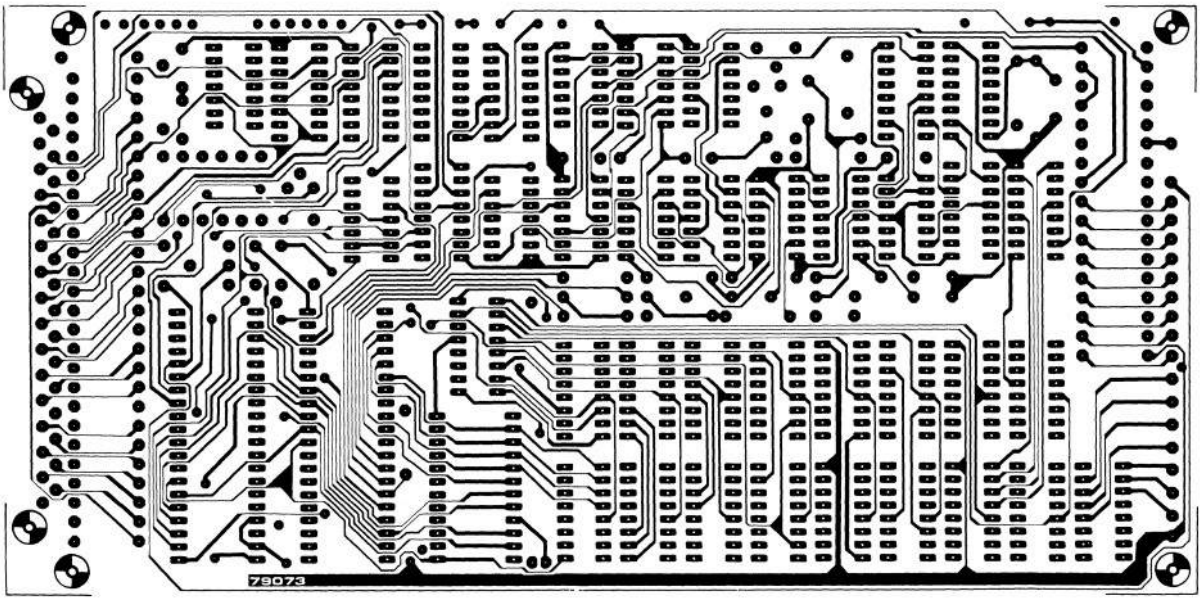
X-tal = kristal, 8,867 MHz
LS = luidspreker 100 Ω/ 500 mW
P3/P4,P5/P6 = joy-stick, potmeters = 680 K
28 x 'digitast' schakelaars voor keyboard

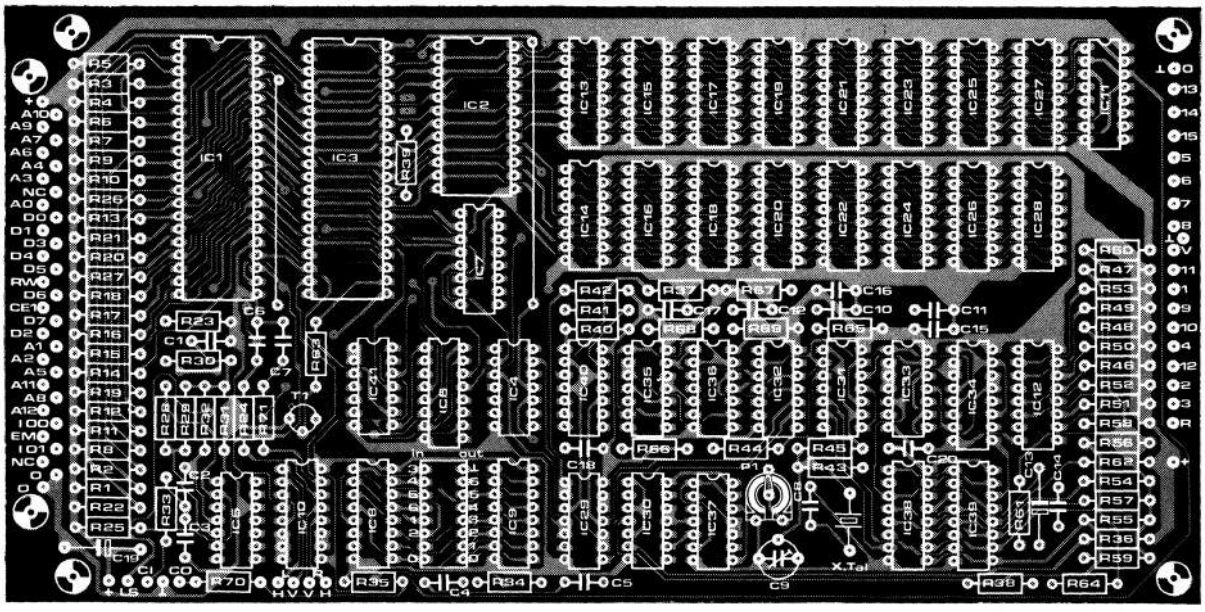
8

komponenten-zijde

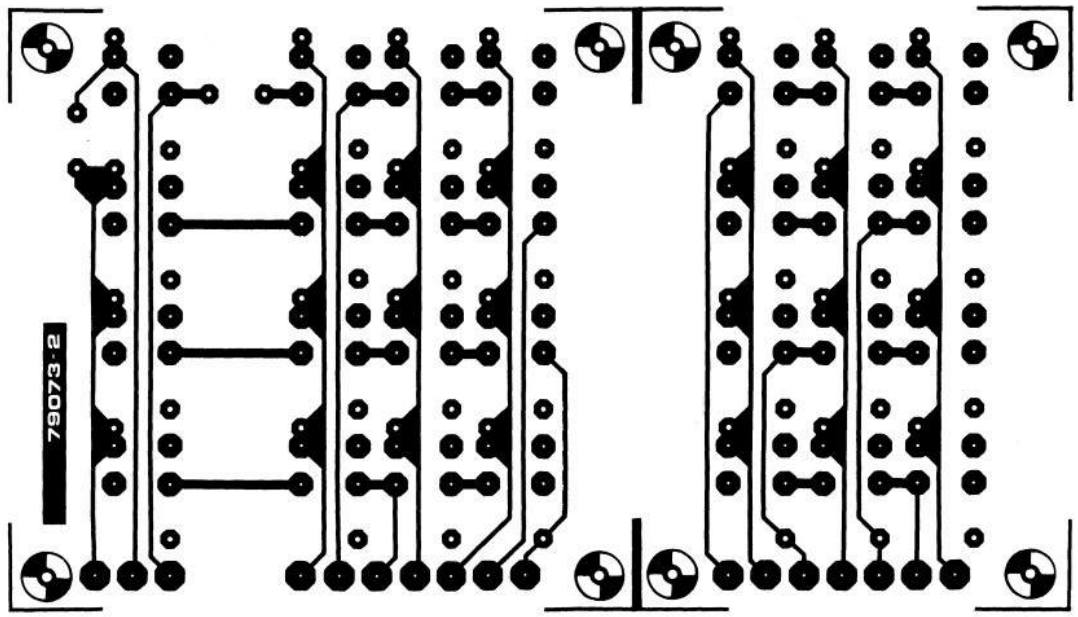


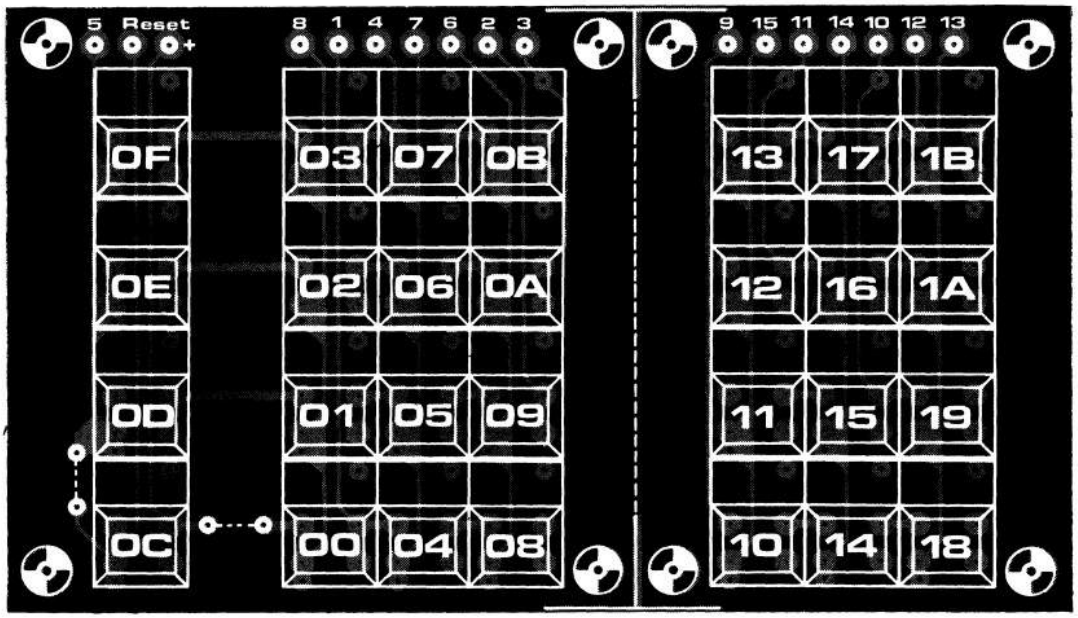
soldeer-zijde



9




bekijken. De doormetallisering moet dan duidelijk zichtbaar zijn. Extra zeker is men, wanneer elk gat afzonderlijk met een multimeter wordt gekontroleerd. Als het goed is, moet de weerstand van de ene naar de andere kant van de print nul ohm bedragen. Het is niet het leukste werk uit de elektronica, maar men behoeft er geen spijt van te hebben.

Voor de montage van de onderdelen wordt het gebruik van een goede miniatuur-soldeerbout sterk aanbevolen. Bovendien moeten er beslist goede IC-voetjes gebruikt worden; eenenveertig IC's betekent al snel achthonderd kontakten, die allemaal even belangrijk zijn. Het achteraf opsporen van een slechte soldeerverbinding of een twijfelachtig kontakt in een IC-voet is een frustrerende aangelegenheid.

De hoofdprint heeft een groot aantal in- en uitgangen, waarvan de meeste echter in deze versie van de speel-computer niet gebruikt worden (zoals bij ieder computer-systeem heeft ook deze speel-computer in principe legio mogelijkheden tot uitbreiding).

De verbindingen tussen de hoofdprint en het keyboard (zie figuur 7) zijn op beide printen duidelijk genummerd. Let wel: de op de print voor het keyboard onderbroken getekende draadbruggen moeten in dit geval *niet* aangebracht worden.

De getallen op de komponenten-opstelling van het keyboard komen overeen met de adressen van de desbetreffende toetsen. Voor het gebruik van de speel-computer verdient echter de aanduiding, zoals die gegeven is in figuur 10, op de toetsen de voorkeur, aangezien deze overeenkomen met de benaming die ze in het monitor-programma van de computer hebben gekregen. Aangezien het keyboard voor veel spelletjes uit het repertoire van de computer eigenlijk beschouwd wordt als twee kleinere keyboards – voor elke speler een – is de print-layout zodanig gemaakt dat de print naar verkiezing in twee stukken verdeeld kan worden.

Voor wat betreft de konstruktie van de voeding geldt als bijzonderheid alleen dat men de *grootst mogelijke zorgvuldigheid* dient te betrachten (voor zover dat een bijzonderheid genoemd kan worden . . .). Wanneer de voedingsspanning door een of andere

oorzaak onverhoopt veel groter mocht worden dan 5 V zouden de gevolgen voor een of meer kostbare IC's wel eens minder plezierig kunnen zijn. Vandaar dan ook dat het van het grootste belang is om de voedingsspanning (met P1 van figuur 3) op 5 V (± 5%, dus in ieder geval tussen 4,75 V en 5,25 V) af te regelen *voordat* hij met de andere onderdelen van het apparaat verbonden wordt. Een druppel zegellak (of nagellak) op de potmeter gaat niet alleen ongewenste verdraaiing tegen, maar dient tevens als waarschuwing er in het vervolg af te blijven.
De VHF/UHF-modulator (die, zoals gezegd, eventueel zou kunnen vervallen), dient terdege afgeschermd te worden, iets dat altijd met hoogfrekwent modulatoren dient te geschieden, maar met deze in het bijzonder.

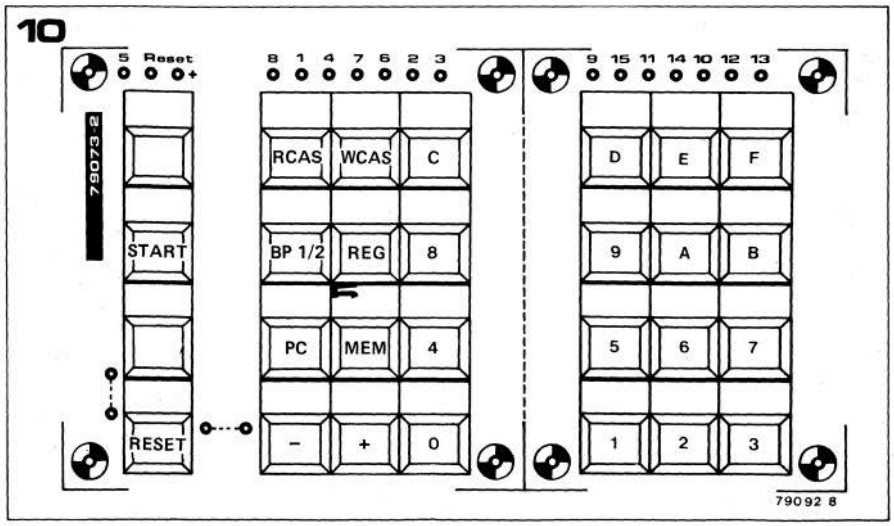
## Het eind is in zicht: de afregeling

Wanneer de printen op de juiste manier zijn opgebouwd en met elkaar verbonden is een grondige kontrole niet overdreven. Daarna kan het apparaat ingeschakeld worden, *mits men er zeker van is dat de voeding korrekt op 5 V is afgeregeld.*
Het afregelen van de schakeling is de eenvoud zelve. De eigenlijke computer bevat niet meer afregelorganen dan de VHF/UHF-modulator: twee, om precies te zijn.
*VHF/UHF-modulator.* Zet P1 in de middenstand. Een aangesloten (kleuren)televisie moet nu af te stemmen zijn op een van de harmonischen van de draaggolf. Dit is te zien aan het verdwijnen van de normaal aanwezige 'sneeuw' op het scherm. Zet P2 op maximum (helemaal rechtsom). Hiermee is de eerste, grove afregeling van de modulator voltooid. De finishing touch komt nog aan bod.
*Hoofdprint.* Druk achtereenvolgens de 'reset'- en de 'start'-toets van het keyboard in. Wanneer de computer goed afgeregeld is, moet er (althans op een kleuren-tv) een egaal blauw beeld verschijnen met vier gele letters (IIII) in de linkerbenedenhoek. Die afregeling geschiedt met de enige twee instelorganen van de hoofdprint, P1 en C9 van de kristaloscillator. De afregeling moet zodanig zijn dat het beeld op de tv zo goed mogelijk is. Daarbij kan men de volgende punten in de gaten houden:
Wanneer P1 niet goed is ingesteld zal de kristaloscillator niet werken, en zal er in het geheel geen beeld (de vier letters op de blauwe ondergrond) verschijnen. De juiste instelling van P1 is wanneer deze een beetje verder is gedraaid dan juist nodig is om een goed beeld te krijgen.
C9 beïnvloedt de frekwentie van de oscillator. Onjuiste afregeling resulteert bij kleurentelevisies in een slechte kleurweergave, of zelfs het ontbreken van kleur, en bij zwartwit-toestellen in een slecht kontrast.
*Fijnafregeling.* Wanneer men eenmaal een redelijk beeld heeft, is het eenvoudig mogelijk de speel-computer op optimale kwaliteit van dit beeld af te regelen:
Het tv-toestel kan worden afgestemd op de zijband die de beste beeldkwaliteit oplevert; wanneer er op de verkeerde zijband is afgestemd zal het beeld de neiging vertonen negatief te verschijnen. In geval van slechte vertikale synchronisatie (rollend beeld), of wanneer er interferentie van een of andere zender met het computersignaal plaatsvindt, kan de stand van P1 van de modulator een weinig veranderd worden; de tv-ontvanger dient dan overeenkomstig verstemd te worden. P2 van de modulator kan gebruikt worden om het beeldkontrast naar wens aan te passen. De kleur van het beeld wordt in de eerste plaats beïnvloed door C9 van de hoofdprint. Zowel deze, als P1 van de hoofdprint, kunnen ingesteld worden op optimale beeldkwaliteit.

## Tenslotte

De cassette-interface zal in vrijwel alle gevallen goed voldoen. Maar er is natuurlijk altijd de uitzondering die de regel bevestigt: het uitgangssignaal kan voor een bepaalde recorder te groot of te klein zijn. In dat geval moet men de waarde van R70 aanpassen. Er kunnen mogelijk problemen ontstaan wanneer de recorder te dicht bij het tv-toestel staat en daaruit allerlei ongerechtigheden oppikt. De oplossing spreekt uiteraard voor zich: plaats de recorder op een grotere afstand van de tv.
Een uitgebreid apparaat als de speelcomputer kan onmogelijk gerealiseerd worden met standaardonderdelen, en niet alle komponenten zullen dan ook op iedere straathoek verkrijgbaar zijn. Dit geldt vooral voor de ROM (IC2), die door Philips speciaal voor deze toepassing is geprogrammeerd. Verschillende belangrijke handelaren in elektronische komponenten hebben evenwel al blijk gegeven van plannen komplete onderdelenpakketten te leveren, inklusief de voorgeprogrammeerde ROM.

*Literatuur:*
*VHF/UHF-tv-modulator, Elektuur 180, oktober 1978, p. 10-47*




**Figuur 9. Print van het eventueel in tweeën te delen keyboard. De onderbroken getekende draadbruggen moeten *niet* worden aangebracht.**

**Figuur 10. Deze symbolen kunnen op de toetsen van het keyboard worden aangebracht.**

# spelen met de tv-speel-computer (2)

## alles en nog wat over het maken van software voor de speel-computer

**In het eerste deel zijn de basis-principes van de speel-computer besproken, met de belangrijkste instrukties. Nu komen de resterende instrukties aan de orde, met een aantal handige programmeertruukjes en nuttige subroutines.**
**Met deze gegevens — en wat oefening! — kunnen al interessante programma's ontwikkeld worden. Wij hebben er in elk geval heel wat in petto!**

Vorige maand zijn de "data-transport"-instrukties (load and store), de "sprong"-(Branch)- en "vergelijk"(compare en test)-mogelijkheden, en een paar "diverse"-instrukties aan de orde gekomen. Zoals uit tabellen A . . . E in dat artikel blijkt, zijn met deze mogelijkheden al interessante programma's te maken. De meer uitgebreide versie van ditzelfde programma op de nieuwe ESS-plaat maakt echter duidelijk hoeveel nieuwe mogelijkheden ontstaan indien eveneens de resterende instrukties gebruikt worden. In dit artikel beginnen we dan ook met de bespreking van de "algebraische" (arithmetic), "logische" (logical) en "schuif"(rotate)-instrukties; de "input/output"-instrukties worden in de basisversie van de speel-computer niet gebruikt.

### "Arithmetic"

Ook al zal de speel-computer niet vaak gebruikt worden om sommetjes te maken, toch zijn de "algebraische" instrukties vaak heel handig. In tabel 8 is de komplete set samengevat: optellen (add), aftrekken (substract) en "decimal adjust register" — waarover dadelijk meer.

Optellen en aftrekken gebeurt volgens de normale regels:
Ø3 + Ø5 = Ø8; 19 — Ø2 = 17; 28 + 13 = 3B; enzovoorts. De berekeningen worden uitgevoerd in 8 bit binair, waarbij negatieve getallen uitgedrukt worden in een twee-komplement. De hexadecimale berekeningen kloppen dus: 1Ø — 2 = ØD (in decimale getallen: 16 — 2 = 14). Bij iedere optelling of aftrekking worden drie bits in het "program status lower"-register geset of gereset:

- Het "carry/borrow"-bit (C): dit wordt logisch één als aanduiding van "één onthouden" bij het optellen, en logisch nul voor "één lenen" na het aftrekken. Overigens zijn deze details niet zo belangrijk: de computer weet zelf wat hij met deze gegevens aan moet bij eventueel volgende optel- of aftrek-instrukties. Wèl belangrijk is het gebruik van het "with carry"-bit (WC: bit 3 in de PSL). Als dit logisch één is, wordt de "carry"-of "borrow"-informatie gebruikt bij het optellen of aftrekken; als het "WC"-bit nul is, wordt de toestand van het "carry"-bit niet betrokken in de berekening. Vooral dit laatste blijkt in de praktijk belangrijk!
- Het "inter-digit carry"-bit (IDC): dit geeft de "carry"-of "borrow"-informatie tussen de laagste vier en de hoogste vier bits in het register. Bij hexadecimale berekeningen kan dit verwaarloosd worden, maar bij decimale berekeningen kan het van belang zijn.
- Het "overflow"-bit (OVF): aangezien grote getallen (groter dan 7F) als negatief beschouwd worden, kan er bij het optellen of aftrekken gemakkelijk iets mis gaan. Zo zal bijvoorbeeld 7Ø + 28 als resultaat 98 opleveren, maar dit kan ook gezien worden als een *negatief* getal (—68). Dergelijke dubbelzinnige resultaten zijn te herkennen aan het feit dat het "overflow"-bit logisch één is; dit bit wordt namelijk geset als het optellen van twee positieve getallen een "negatief" resultaat geeft. Ook als het aftrekken van twee negatieve getallen een "positief" resultaat

**Tabel 8**

**Arithmetic**

| omschrijving | | voorbeeld | opmerkingen |
|---|---|---|---|
| add to register zero | (ADDZ) | 81 | RØ: R1 + RØ |
| add immediate | (ADDI) | 84xx | xx = data |
| add relative | (ADDR) | 88yy | yy = sprong |
| add absolute | (ADDA) | 8Czzzz | zzzz = adres |
| subtract from register zero | (SUBZ) | A1 | RØ: = RØ - R1 |
| subtract immediate | (SUBI) | A4xx | xx = data |
| subtract relative | (SUBR) | A8yy | yy = sprong |
| subtract absolute | (SUBA) | ACzzzz | zzzz = adres |
| Decimal Adjust Register | (DAR) | 94 | |

oplevert wordt het "OVF"-bit logisch één.

Tot zover het hoofdstuk optellen en aftrekken. In de praktijk is het meestal voldoende het "WC"-bit te resetten, zodat een recht-toe-recht-aan berekening uitgevoerd wordt, zonder onverwachte "carry" of "borrow".

#### "Decimal adjust register"

Deze instruktie is een hulpmiddel bij bewerkingen waar de 8 bits gebruikt worden als twee 4 bit decimale cijfers in BCD-kode. Een volledige "gebruiksaanwijzing" is in de officiële handleiding opgenomen. Tot nu toe hebben wij deze instruktie zelf niet nodig gehad; de enige keer dat het bruikbaar had kunnen zijn (bij een aftellende tijd-display op het scherm) leek het eenvoudiger om bij iedere "Ø – F"-overgang zes af te trekken, als volgt:

```
 F70F  TMI, R7
┌9802  BCFR
│A706  SUBI, R7
└enz.
```

## "Logic"

De beschikbare "AND"-, "Inclusive OR (IOR)"- en "Exclusive OR (EOR)"-instrukties zijn in tabel 9 opgesomd. De logische bewerkingen zijn in tabel 10 weergegeven. Voor de meeste praktische toepassingen is het echter gemakkelijker om het effekt van deze instrukties op een andere manier te omschrijven:

#### "AND"

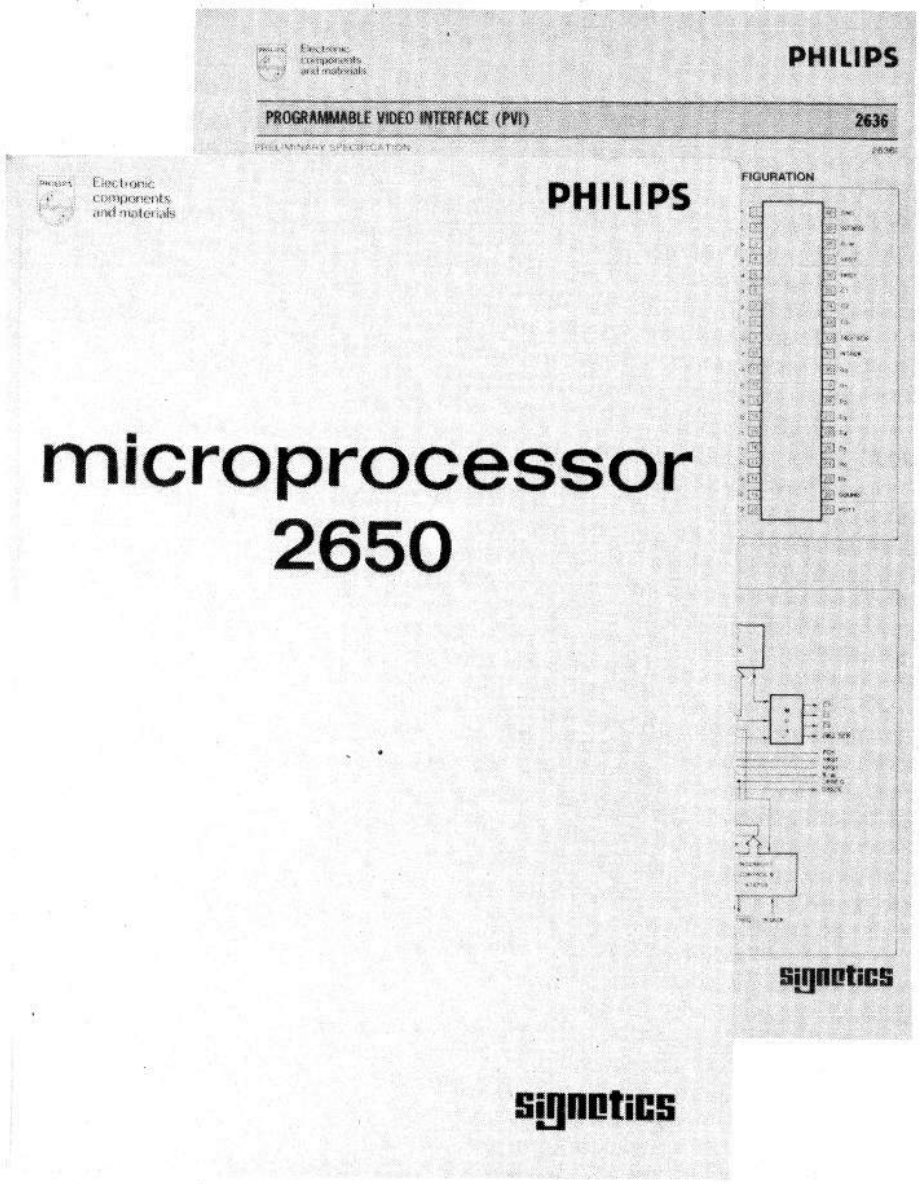
Bij een "AND"-bewerking worden twee groepen van 8 bits vergeleken; in het resultaat zijn alleen die bits logisch één die ook één waren in beide oorspronkelijke groepen. Deze instruktie is dan ook te gebruiken als bit-masker. Nemen wij bijvoorbeeld aan dat een "delay"-routine aftelt in R3, en dat de laagste drie bits in dit register gebruikt worden om de kleur van het scherm te veranderen. Dit kan als volgt:

```
Ø3       LODZ, R3
4407     ANDI, RØ
8408     ADDI, RØ
CC1FC6   STRA, RØ
```

Na het "afschermen" van de hoogste vijf bits met behulp van de "ANDI"-instruktie wordt het "background enable"-bit bij het resultaat opgeteld. De nieuwe kleurinformatie kan dan in PVI-adres 1FC6 worden ingegeven.

#### "Inclusive OR"

Ook in dit geval worden twee groepen van 8 bits vergeleken. In het resultaat zijn nu echter alle bits logisch één die in minstens één van de oorspronkelijke groepen "1" waren. Anders gezegd: alleen die bits zijn logisch nul in het eindresultaat die in beide oorspronkelijke groepen ook nul waren. Een bit-masker voor logische nullen, met andere woorden!




**Tabel 9**

**Logic**

| omschrijving | | voorbeeld | opmerkingen |
|---|---|---|---|
| AND to register zero | (ANDZ) | 41 | R ≠ RØ |
| AND immediate | (ANDI) | 44xx | xx = data |
| AND relative | (ANDR) | 48yy | yy = sprong |
| AND absolute | (ANDA) | 4Czzzz | zzzz = adres |
| Inclusive OR to register zero | (IORZ) | 61 | |
| Inclusive OR immediate | (IORI) | 64xx | xx = data |
| Inclusive OR relative | (IORR) | 68yy | yy = sprong |
| Inclusive OR absolute | (IORA) | 6Czzzz | zzzz = adres |
| Exclusive OR to register zero | (EORZ) | 21 | |
| Exclusive OR immediate | (EORI) | 24xx | xx = data |
| Exclusive OR relative | (EORR) | 28yy | yy = sprong |
| Exclusive OR absolute | (EORA) | 2Czzzz | zzzz = adres |

Zowel de "AND"- als de "IOR"-instrukties zijn ook te gebruiken als één of meer bits in een register logisch nul of één gemaakt moeten worden, zonder de overige bits te veranderen. Als in het voorbeeld hierboven de inhoud van R3 gebruikt wordt om zowel achtergrond- als schermkleur te bepalen, kan het "background enable"-bit met een "IOR"-instruktie worden toegevoegd:

```
Ø3       LODZ, R3
6408     IORI, RØ
CC1FC6   STRA, RØ
```

#### "Exclusive OR"

Nog afgezien van de "officiële" logische funktie, zijn deze instrukties ook uitstekend te gebruiken als "selektieve

inverter". Dit is het gemakkelijkste als volgt voor te stellen: één groep van 8 bits wordt opgevat als oorspronkelijke data; een tweede groep van 8 bits geeft aan wat er met deze data moet gebeuren. Als een bit in de tweede groep logisch "1" is, wordt het korresponderende bit in de eerste groep geïnverteerd; een "Ø" geeft aan dat dit bit onveranderd moet blijven. Een paar voorbeelden. Voor het gemak nemen wij aan dat de data (één van de groepen van 8 bits) in alle gevallen FF is: 1111 1111. De instruktie "EOR, FF" inverteert alle bits, met ØØ als resultaat. "EOR, CØ" inverteert alleen de eerste twee bits (CØ = 1100 0000), zodat 3F overblijft: ØØ11 1111.
Tenslotte een praktijkvoorbeeld. Zoals vorige maand beschreven, zijn bij het lezen van een kolom van het toetsenbord de laagste vier bits altijd "1". De C-toets bijvoorbeeld (kolom adres 1E8A) geeft de kode 8F. Zonodig kunnen deze ongewenste enen als volgt nul gemaakt worden:

```
ØC1E8A  LODA, RØ
240F    EORI, RØ
```

Overigens is het net zo gemakkelijk (en misschien "logischer") om in dit geval een "AND"-instruktie als bit-masker te gebruiken: "ANDI, FØ" geeft hetzelfde resultaat.

## "Rotate"

De "rotate register right"- en "rotate register left"-instrukties schuiven de data in het opgegeven register één plaats naar rechts of links. Zolang het "WC"-bit in de PSL logisch Ø is, schuift de data gewoon rond: wat er aan één kant uitkomt gaat er aan de andere kant weer in. Als het "WC"-bit geset is, wordt de zaak ingewikkelder: de "carry"- en "interdigit carry"-bits gaan nu ook een rol spelen. Gelukkig is een langdradige toelichting overbodig — alle mogelijkheden zijn in figuur 2 duidelijk weergegeven!

## Truukjes en foefjes

Nu wordt het leuk! Bij het spelen met de speel-computer — en het bestuderen van de monitor software — hebben wij een aantal nuttige programmeerfoefjes gevonden. Weliswaar hebben ervaren programmeurs ons verzekerd dat de meeste algemeen bekend zijn, maar het lijkt toch zinvol om ze even op een rij te zetten.

### "EORZ, RØ"

In machinetaal: "2Ø". De data in register Ø wordt door deze EXOR-instruktie selektief geïnverteerd, zoals bepaald door de data in register Ø. Als een bit "1" is, wordt het dus geïnverteerd; een "Ø" blijft staan. Het resultaat? ØØ in register Ø! De "normale" manier (LODI, RØ = Ø4ØØ) kost één extra geheugenplaats.

### "IORZ, RØ"

Deze instruktie (6Ø, om precies te zijn) laat de data in register Ø ongewijzigd. Toch wordt er iets met deze data gedaan — ook al verandert er niets — en dus worden de "condition code"-bits geset, afhankelijk van het teken van het getal in RØ: Ø1 voor positief, ØØ voor nul en 1Ø voor negatief.

### Vermenigvuldigen en delen

De data in een register één stap naar links schuiven komt overeen met vermenigvuldigen met twee — mits geen bit links eruit schuift of rechts erbij komt, maar dat is gemakkelijk te kontroleren. Naar rechts schuiven levert een deling door twee op. En vermenigvuldigen met drie dan? Ook niet moeilijk:

```
C1   STRZ, R1
D1   RRL, R1
81   ADDZ, R1
```

Klaar.
Het oorspronkelijke getal (in register Ø) wordt overgenomen in register 1; na vermenigvuldiging met twee in R1 wordt het weer opgeteld bij de oorspronkelijke data in RØ.

### "LODI" als geheugenplaats

In de loop van een programma is het vaak nodig om bepaalde gegevens regelmatig te veranderen. Zo kan bijvoorbeeld de horizontale plaats van een voorwerp gewijzigd worden vanaf het toetsenbord. Als nieuwe informatie in de PVI geladen is, kan ze daar rustig blijven zolang de horizontale positie niet hoeft te wijzigen. De moeilijkheid is alleen dat deze informatie niet uit de PVI terug te halen is als een nieuwe positie bepaald moet worden. De enige oplossing is om dezelfde gegevens tegelijk ergens in het "normale" geheugen op te slaan. Om een nieuwe positie te bepalen wordt dan deze data uit het tijdelijk geheugen ("scratch") gelezen, de nieuwe waarde berekend, en het resultaat wordt weer in de PVI en in het tijdelijk geheugen opgeslagen.
Tot zover niets bijzonders. Eén truukje is echter in de praktijk nuttig gebleken. Het hele programma wordt in RAM bewaard; er is dus niets op tegen om (delen van) instrukties te veranderen tijdens het uitvoeren van dit programma. Nemen we bijvoorbeeld aan dat de data in register 1 opgeteld moet worden bij de bestaande horizontale positie om de nieuwe waarde te bepalen:

```
Ø4ØØ   LODI, RØ
81     ADDZ, R1
```

**Tabel 10**

**Logische bewerkingen**

Bij de logische bewerkingen wordt elk overeenkomend paar bits in de twee opgegeven (8 bit) data-bytes behandeld volgens de onderstaande waarheidstabellen:

| | bit A (Ø . . . 7) | Bit B (Ø . . . 7) | resultaat |
|---|---|---|---|
| **AND** | Ø | Ø | Ø |
| | Ø | 1 | Ø |
| | 1 | Ø | Ø |
| | 1 | 1 | 1 |
| **IOR** | Ø | Ø | Ø |
| | Ø | 1 | 1 |
| | 1 | Ø | 1 |
| | 1 | 1 | 1 |
| **EOR** | Ø | Ø | Ø |
| | Ø | 1 | 1 |
| | 1 | Ø | 1 |
| | 1 | 1 | Ø |

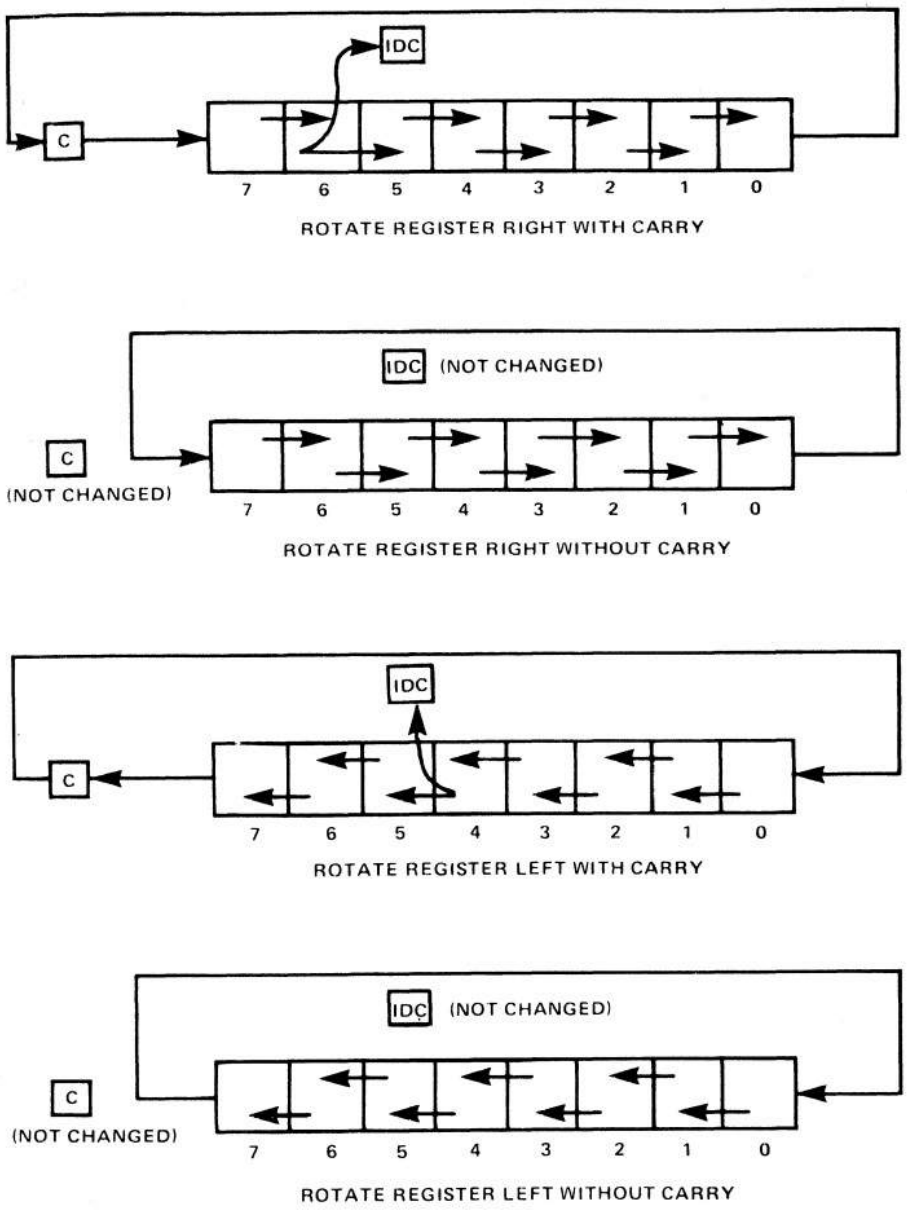
**Voorbeelden**

In deze voorbeelden wordt uitgegaan van ØF als oorspronkelijke data in register Ø.

| | | | | |
|---|---|---|---|---|
| "ANDI,RØ,33" | (4433): | data A | = ØF | = ØØØØ 1111 |
| | | data B | = 33 | = ØØ11 ØØ11 |
| | | resultaat | = Ø3 | = ØØØØ ØØ11 |
| "IORI,RØ,33" | (6433): | data A | = ØF | = ØØØØ 1111 |
| | | data B | = 33 | = ØØ11 ØØ11 |
| | | resultaat | = 3F | = ØØ11 1111 |
| "EORI,RØ,33" | (2433): | data A | = ØF | = ØØØØ 1111 |
| | | data B | = 33 | = ØØ11 ØØ11 |
| | | resultaat | = 3C | = ØØ11 11ØØ |

Deze drie logische funkties kunnen ook gezien worden als "bit masker"-bewerkingen. Na een AND-bewerking blijven alleen die bits in de oorspronkelijke data (data A) logisch "1" die aangewezen worden door een "1" in het bit-masker (data B). Bij een "IOR"-bewerking daarentegen, blijven alleen die bits in data A logisch "Ø" die overeenkomen met de nullen in data B. Met een "EOR"-instruktie tenslotte, wordt bereikt dat bits in data A geïnverteerd worden als hun "partner" in data B logisch "1" is.

**2**




het volgende: Eerst wordt de tweede byte van het gewenste absolute adres ingelezen (LODA, I-R1) en overgebracht naar adres Ø9D5 (de derde byte van de STRA instruktie). Nu wordt de bijbehorende data ingelezen (door de tweede "LODA, I-R1"-instruktie) en vervolgens geladen in de PVI op het op dat moment opgegeven adres. Dit adres is dus *niet* 1FØØ, ook al staat dat in de tabel zo aangegeven; beter was misschien "1Fxx", waarbij "xx" de adres data voorstelt die bij de eerste "LODA, I-R1"-instruktie gevonden is.

Uiteraard zijn er allerhande variaties mogelijk op ditzelfde basisprincipe. Eenmaal bedacht op de mogelijkheid om instrukties te wijzigen tijdens het uitvoeren van het programma, zullen praktijkvoorbeelden snel gevonden worden als men zelf programma's gaat maken!

## Monitor-routines

Het komplete monitor-programma is gegeven in ROM; er valt dus niets aan te wijzigen. Toch staat het bij "normale" geheugen-adressen. Er is dus niets op tegen om monitor-subroutines te gebruiken als deel van een eigen programma. In de meeste gevallen is de enige beperking dat de monitor-routine in kwestie moet eindigen met een onvoorwaardelijke "return"-instruktie: RETC, UN = 17. Bovendien moet in sommige gevallen enig voorbereidend werk gedaan worden – de juiste data in de juiste registers laden, bijvoorbeeld.

#### Toetsenbord afvragen

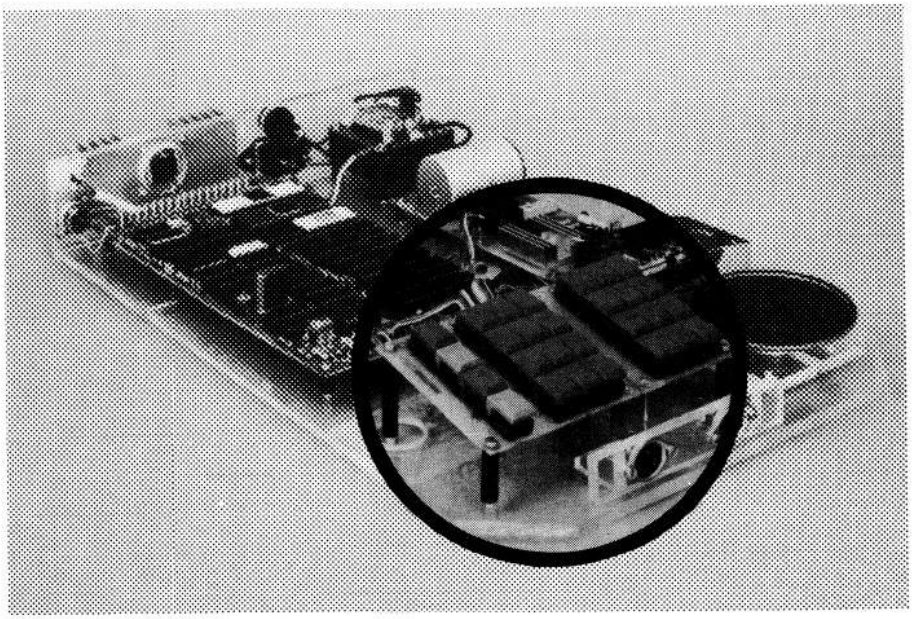
Een komplete routine om het toetsenbord af te vragen begint bij adres Ø181. Kontaktdenderonderdrukking is "ingebouwd", en er wordt gekontroleerd of per ongeluk meerdere toetsen tegelijk ingedrukt zijn. In de gegeven vorm wordt het "RS"-bit in de PSL gereset, zodat registers R1, R2 en R3 gebruikt worden. Mocht dit ongewenst zijn, kan de routine bij adres Ø183 gestart worden,

```
C87C      STRR, RØ
CC1FCA    STRA, RØ
```

Het tweede deel van de "load immediate"-instruktie wordt hier als tijdelijk geheugen gebruikt voor de bestaande plaats-informatie. Nadat hier de waarde in R1 bij opgeteld is, wordt het resultaat bewaard in de "LODI"-instruktie en vervolgens in de PVI geschreven.

#### Absoluut adres aanpassen

Dezelfde truuk kan ook gebruikt worden om een absoluut adres te wijzigen in de loop van een programma. Het testbeeld-programma op de nieuwe ESS-plaat, bijvoorbeeld, maakt hier gebruik van bij het laden van een hele reeks basisgegevens in de PVI. Dit stukje uit het programma (met een paar wijzigingen, om een leuker resultaat te krijgen!) is weergegeven in tabel 11.
Bij iedere rondgang door de lus gebeurt



Tabel 11

| | | | |
|---|---|---|---|
| 09C7 | 7620 | PPSU, II | |
| 09C9 | 056E | LODI, R1 | |
| 09CB | 0D49E2 | LODA, I-R1 | (adres) |
| 09CE | C805 | STRR, R0 | |
| 09D0 | 0D49E2 | LODA, I-R1 | (data) |
| 09D3 | CC1F00 | STRA, R0 | (09D5 = scratch) |
| 09D6 | 5973 | BRNR, R1 | |
| 09D8 | 0C1E88 | LODA, R0 | |
| 09DB | F420 | TMI, R0 | terug naar monitor als |
| 09DD | 9879 | BCFR | "PC"-toets bediend word |
| 09DF | 1F0000 | BCTA, UN | |
| 09E2 | 50 0C | data-adres | |
| 09E4 | 50 1C | data-adres | |
| 09E6 | 50 2C | data-adres | VC 1 . . . 4 |
| 09E8 | 50 4C | data-adres | |
| 09EA | FE 0D | data-adres | |
| 09EC | FE 1D | data-adres | |
| 09EE | FE 2D | data-adres | VOD 1 . . . 4 |
| 09F0 | FE 4D | data-adres | |
| 09F2 | 1A 0A | data-adres | |
| 09F4 | 3A 1A | data-adres | |
| 09F6 | 5A 2A | data-adres | HC 1 . . . 4 |
| 09F8 | 7A 4A | data-adres | |
| 09FA | AA C0 | data-adres | voorwerp-maat |
| 09FC | 09 C1 | data-adres | |
| 09FE | 09 C2 | data-adres | kleur |
| 0A00 | 19 C6 | data-adres | |
| 0A02 | 00 00 | data-adres | |
| 0A04 | 00 01 | data-adres | |
| 0A06 | 00 02 | data-adres | |
| 0A08 | 74 03 | data-adres | |
| 0A0A | 44 04 | data-adres | vorm 1 |
| 0A0C | 74 05 | data-adres | |
| 0A0E | 44 06 | data-adres | |
| 0A10 | 44 07 | data-adres | |
| 0A12 | 77 08 | data-adres | |
| 0A14 | 00 09 | data-adres | |
| 0A16 | 00 10 | data-adres | |
| 0A18 | 00 11 | data-adres | |
| 0A1A | 00 12 | data-adres | |
| 0A1C | 75 13 | data-adres | |
| 0A1E | 45 14 | data-adres | vorm 2 |
| 0A20 | 76 15 | data-adres | |
| 0A22 | 45 16 | data-adres | |
| 0A24 | 45 17 | data-adres | |
| 0A26 | 75 18 | data-adres | |
| 0A28 | 00 19 | data-adres | |
| 0A2A | 00 20 | data-adres | |
| 0A2C | 00 21 | data-adres | |
| 0A2E | 00 22 | data-adres | |
| 0A30 | 75 23 | data-adres | |
| 0A32 | 25 24 | data-adres | vorm 3 |
| 0A34 | 25 25 | data-adres | |
| 0A36 | 25 26 | data-adres | |
| 0A38 | 25 27 | data-adres | |
| 0A3A | 27 28 | data-adres | |
| 0A3C | 00 29 | data-adres | |
| 0A3E | 00 40 | data-adres | |
| 0A40 | 00 41 | data-adres | |
| 0A42 | 00 42 | data-adres | |
| 0A44 | 57 43 | data-adres | |
| 0A46 | 55 44 | data-adres | vorm 4 |
| 0A48 | 56 45 | data-adres | |
| 0A4A | 55 46 | data-adres | |
| 0A4C | 55 47 | data-adres | |
| 0A4E | 75 48 | data-adres | |
| 0A50 | 00 49 | data-adres | |

Start adres: 09C7 Terug naar monitor door de "PC"-toets te bedienen.

nadat eerst het "RS"-bit geset is en de "WC"- en "C"-bits gereset zijn.

Bij het gebruik van deze routine zijn nog twee punten van belang: de routine moet twee keer doorlopen worden (liefst bij twee opeenvolgende rasters, bijvoorbeeld met gebruikmaking van het "VRLE"-bit); bovendien moet voordat de routine de eerste keer doorlopen wordt geheugenadres 089F 00 gemaakt worden.

Een komplete routine is in tabel 12 weergegeven. Na de presets en een "wacht op VRLE"-lus volgt de eerste sprong naar de subroutine: 3F0183 = BSTA, UN. Als de subroutine uitgevoerd is, geven de twee hoogste bits in R1 de "status" aan. Als bit 6 logisch 1 is, was dit de eerste keer dat de routine doorlopen werd; het programma springt dus terug naar de "wacht op VRLE"-lus. Na de tweede rondgang is bit 6 logisch 0 en geeft bit 7 aan of er een toets ingedrukt is: "1" als één toets ingedrukt is en "0" voor geen of meer dan een toets. Handig is hierbij dat als bit 7 logisch 1 is, dit overeenkomt met een negatief getal; de "condition code" wordt dus 10.

Een verdere mogelijkheid is in deze routine niet gebruikt. Als van de data in adres 089F alleen bit 7 gereset wordt voor de eerste sprong naar de monitor-subroutine, geeft bit 5 in R1 aan of er (nog) een toets ingedrukt is.

Terug naar de routine in tabel 12. Nadat de routine twee keer doorlopen is (bij adres 0FE6, met andere woorden) geven de laagste vijf bits in R1 aan welke toets ingedrukt is. De toetsnummers zijn in figuur 3a aangegeven; links boven in elk hokje is de "monitor"-aanduiding voor de toets. Zoals boven al gesteld, is de toets-kode in R1 alleen geldig als bit 7 logisch 1 is; anders zal kode 00 verschijnen als de data in adres 089F 00 gemaakt is, of de vorige toetskode als alleen bit 7 gereset is.

Deze toetskodes kunnen voor sommige toepassingen ideaal zijn. Handig is bijvoorbeeld dat de laagste vier bits van de kode voor beide toetsenborden (zowel van linker als rechter speler) identiek zijn, terwijl het vijfde bit aangeeft welk toetsenbord bediend wordt. Toch is in sommige gevallen een andere kode handiger. De "vertaling" gebreurt in het volgende stuk van de routine (van adres 0FE6 tot 0FF5). Dit levert een nieuwe serie toetskodes op (in register 0) zoals in figuur 3b opgesomd.

Deze tweede kode heeft ook zijn voordelen. Voor de zestien "getal-toetsen" komt de data rechtstreeks overeen met het toetsnummer. Bij alle andere toetsen is bit 7 logisch 1 ("negatief" getal!); bovendien is bit 6 alleen logisch 1 voor de + en − toetsen. Op dezelfde manier komt bit 5 overeen met de "RCAS"- en "WCAS"-toetsen. Het enige nadeel is dat drie van de vier toetsen in de eerste kolom met 80 worden aangeduid, omdat zij in het monitor-programma niet gebruikt worden.

Tenslotte is in tabel 12 een kleine hulproûtine toegevoegd, vanaf adres ØFF6: "wacht tot toets losgelaten wordt". Deze routine zorgt er voor dat de hoofdroutine net zo lang herhaald wordt tot de kode 3Ø ("geen toets") gevonden wordt.

#### Een paar kleintjes

Na de uitvoerige beschrijving van de toetsenbord routine, als afwisseling nu een paar kleintjes.

*geen duplikaten*

Met de instruktie 3FØØ9E (BSTA, UN, ØØ9E) wordt de data FE als vertikale offset voor de vier duplikaten geladen op adressen 1FØD, 1F1D, 1F2D en 1F4D. Hierdoor zullen alleen de voorwerpen zelf op het scherm kunnen verschijnen, zonder duplikaten.
Eventueel kan iedere willekeurige offset in deze vier adressen geschreven worden vanuit register Ø, door de subroutine te beginnen bij adres ØØAØ.
Deze routine gebruikt alleen register Ø.

*geen voorwerpen*

Alle voorwerpen kunnen gewist worden door ØØ te laden in alle adressen van 1FØØ tot 1F4F. Hiervoor is een subroutine beschikbaar, beginnend bij adres Ø16E. Eventueel kan andere data (FF, bijvoorbeeld) vanuit RØ in al deze adressen geschreven worden door de routine te starten bij adres Ø16F.
Gebruikte registers: RØ en R2.

*getal splitsen*

De 8 bits in een register kunnen als een getal van twee hexadecimale cijfers voorgesteld worden. Soms is het handig om deze twee cijfers (de twee groepen van vier bits) afzonderlijk beschikbaar te hebben. Een subroutine, beginnend bij adres Ø35E, splitst de data in register 1. Als de oorspronkelijke data in dit register voorgesteld wordt als XY, dan blijft na uitvoeren van deze routine ØY in R1 en staat ØX in RØ.

#### tekst op het scherm

Een grote groep kleine routines hebben betrekking op de teksten die op het scherm moeten verschijnen. Ofschoon sommige van deze subroutines ook voor andere doeleinden te gebruiken zijn, is het handiger om ze als één groep te beschrijven.

**Tabel 12**

| | | | | |
|---|---|---|---|---|
| ØFDØ | 2Ø | EORZ, RØ | presets | toetsenbord afvragen en toetskodes vertalen |
| ØFD1 | CCØ89F | STRA, RØ | | |
| ØFD4 | 7712 | PPSL, RS, COM | | |
| ØFD6 | 75Ø9 | CPSL, WC, C | | |
| ØFD8 | ØCIFCB | LODA, RØ | wacht op "VRLE" | |
| ØFDB | F44Ø | TMI, RØ | | |
| ØFDD | 9879 | BCFR | | |
| ØFDF | 3FØ183 | BSTA, UN | "keyscan eerste ronde? | |
| ØFE2 | F54Ø | TMI, R1 | | |
| ØFE4 | 1872 | BCTR | | |
| ØFE6 | Ø1 | LODZ, R1 | "vertaal" toets-kode | |
| ØFE7 | 1ÂØ5 | BCTR | | |
| ØFE9 | Ø43Ø | LODI, RØ | | |
| ØFEB | 751Ø | CPSL, RS | | |
| ØFED | 17 | RETC, UN | | |
| ØFEE | 451F | ANDI, R1 | | |
| ØFFØ | ØD6122 | LODA, I/R1 | | |
| ØFF3 | 751Ø | CPSL, RS | | |
| ØFF5 | 17 | RETC, UN | | |
| ØFF6 | 3B58 | BSTR, UN | wacht tot toets losgelaten wordt | |
| ØFF8 | F43Ø | TMI, RØ | | |
| ØFFA | 987A | BCFR | | |
| ØFFC | 17 | RETC, UN | | |

Gebruike registers: RØ, R1', R2', R3'.

Subroutine nivo's: 2 voor toetsenbord afvragen (startadres: ØFDØ);
3 voor wacht tot toets losgelaten wordt (startadres: ØFF6).

*PVI voorbereiden*

Deze subroutine (startadres: Ø161) is bedoeld om de basisgegevens voor een tekst-display in de PVI te laden:

- voorwerpen maat 2 (AA in 1FCØ);
- juiste kleur (gele voorwerpen, blauw scherm);
- ØØ in 1FC3 (score form/pos);
- geluid uit;
- geen score (AA in 1FC8 en 1FC9)
- geen voorwerpen (ØØ in 1FØØ . . . . 1F4F).

De voorwerp-positie-informatie wordt dus ØØ bij deze routine! Bovendien valt op dat de achtergrond informatie **niet** verloren gaat; de achtergrond wordt alleen "onzichtbaar" gemaakt door het dezelfde kleur te geven als het scherm.
Gebruikte registers: RØ, R1, R2.

*tekst data*

Als een tekst op het scherm moet verschijnen, moet uiteraard een behoorlijke hoeveelheid gegevens als "vorm-informatie" voor de voorwerpen in de PVI geschreven worden. Gelukkig zijn een groot aantal karakters voorgeprogrammeerd in de monitorsoftware (zie tabel 13). De eerste 28 (tot en met het maal-teken) zijn bewust geprogrammeerd; de rest zijn "toevalstreffers". Een komplete uitlezing van alle beschikbare karakters en andere vormen wordt verzorgd door één van de routines onder file 2 op de nieuwe ESS-plaat (ESS 006).
Om één tekstregel op het scherm te krijgen, moeten de kodes uit tabel 13 eerst geladen worden in adressen Ø89Ø . . . Ø897; acht karakters per regel dus. Eventuele spaties moeten met kode 17 aangegeven worden. Soms kan het handig zijn om eerst acht spaties te laden en daarna pas de één of twee gewenste karakters. Ook hiervoor is een subroutine beschikbaar, beginnend bij adres Ø2D9; hierbij worden de registers RØ en R2 gebruikt.

Tijd voor een praktijkvoorbeeld. Het programma in tabel 14 (afgeleid van tabel 7 in deel 1) schrijft alle goed bruikbare karakters op het scherm.
Na de gebruikelijke "interrupt inhibit"-instruktie is de eerste stap het voorbereiden van de PVI, zoals boven beschreven: 3FØ161. Nu worden R3 en R1 geladen, respektievelijk met het totale aantal

**3**

**a**

| systeem-toetsen | linker speler | | | rechter speler | | |
|---|---|---|---|---|---|---|
| UC ØF | RCAS Ø3 | WCAS Ø7 | C ØB | D 13 | E 17 | F 1B |
| STRT ØE | BP Ø2 | REG Ø6 | 8 ØA | 9 12 | A 16 | B 1A |
| LC ØD | PC Ø1 | MEM Ø5 | 4 Ø9 | 5 11 | 6 15 | 7 19 |
| RESET ØC* | – ØØ | + Ø4 | Ø Ø8 | 1 1Ø | 2 14 | 3 18 |

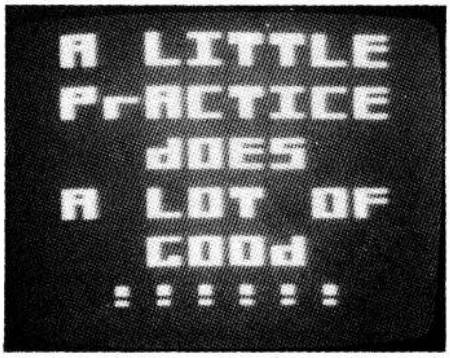
* Deze kode wordt alleen gevonden als deze toets bedraad is als deel van het normale toetsenbord — niet als het gebruikt wordt als reset-toets, zoals voorgesteld in het oorspronkelijke bedradingsplan.

**b**

| systeem-toetsen | linker speler | | | rechter speler | | |
|---|---|---|---|---|---|---|
| UC 8Ø | RCAS 9Ø | WCAS 93 | C ØC | D ØD | E ØE | F ØF |
| STRT 8A | BP 84 | REG 87 | 8 Ø8 | 9 Ø9 | A ØA | B ØB |
| LC 8Ø | PC 8D | MEM 81 | 4 Ø4 | 5 Ø5 | 6 Ø6 | 7 Ø7 |
| RESET 8Ø* | – CØ | + EØ | Ø ØØ | 1 Ø1 | 2 Ø2 | 3 Ø3 |

3Ø = geen toets
* zie opmerking onder figuur 3a.

karakters voor het display (42 = 2A) en het aantal karakters per regel (Ø7). De gewenste karakter kodes worden vanaf adres Ø93Ø ingegeven.
Hierna is de "acht spaties"-routine opgenomen (3FØ2D9). Weliswaar is dat hier nauwelijks zinvol — er worden al zeven karakters op iedere regel geschreven, en één extra spatie zou ook geen probleem zijn — maar het illustreert de mogelijkheden. De nu volgende lus (Ø9ØC . . . Ø912) brengt de kodes voor de eerste regel (beginnend bij adres Ø953) over naar de "message line scratch" (de eerder genoemde adressen vanaf Ø89Ø).
Nu komen we bij de volgende subroutine:




*"Mline" laden*

Deze routine (start adres: Ø2ØE) vertaalt de kodes in de "message line scratch" naar de bijbehorende vorm-informatie voor de vier voorwerpen en slaat deze gegevens op in een "display scratch": adresbereik Ø8ØØ . . . Ø88F, voor maximaal zes regels tekst.
Aangezien deze routine alle vier registers gebruikt, zou de informatie in R3 verloren gaan. Een oplossing zou zijn om de "LODI"-instruktie op adres Ø9Ø7 als tijdelijk geheugen te gebruiken, zoals boven beschreven. In dit voorbeeld is een andere oplossing gekozen: de tweede register-groep wordt gekozen vóór de sprong naar deze subroutine.
Vervolgens wordt nagegaan of alle karakters, voor alle zes regels, geladen zijn. Zo lang dit nog niet het geval is, springt het programma naar adres Ø927.
De volgende subroutine is aan de beurt:

*scroll*

Oftewel: nieuwe regel. Nog exakter zou zijn om deze subroutine (vanaf adres Ø2CF) aan te geven als "nieuwe regel en acht spaties in message line scratch".
Er gebeurt namelijk het volgende:

- alle vorm informatie voor de voorwerpen wordt één regel opgeschoven in de "display scratch": de zesde regel wordt de vijfde, de vijfde wordt de vierde, enz.; de eerste (bovenste) regel gaat verloren.
- de kode voor een spatie wordt in de acht "message line scratch" geheugenplaatsen geschreven.

Ook voor deze routine wordt in dit programmaatje de tweede register-groep gekozen. Volledig overbodig in dit geval; de routine gebruikt registers RØ, R1 en R2, terwijl alleen de data in R3 bewaard hoeft te blijven.
Vervolgens springt het programma terug naar adres Ø9Ø7, om de volgende regel te laden.
Als alle zes regels geladen zijn, wordt de sprong bij adres Ø91B niet uitgevoerd: de data in R3 is ØØ geworden. Het programma wordt nu op een ongebruikelijke manier beëindigd:

- wacht tot de + toets losgelaten wordt. Het programma wordt met deze toets gestart. De microprocessor is dermate snel dat hij het programmaatje afgewerkt zal hebben vóór de toets weer losgelaten is!
- terug naar monitor op adres ØØ38. Hierdoor wordt bereikt dat de monitor de tekst op het scherm zet, zonder eerst zelf een eigen berichtje te schrijven!

In de meeste gevallen zal deze gemakkelijke oplossing niet mogelijk zijn; het eigen programma moet dóór lopen, terwijl toch de tekst op het scherm moet verschijnen. Ook hiervoor is er een

**Tabel 13**

**karakter**

| karakter | kode | karakter | kode | karakter | kode | karakter | kode |
|---|---|---|---|---|---|---|---|
| 0 | ØØ | A | ØA | P | 14 | ? | 5F |
| 1 | Ø1 | b | ØB | r | 15 | . . | 8A |
| 2 | Ø2 | C | ØC | = | 16 | n (1) | AA |
| 3 | Ø3 | d | ØD | spatie | 17 | I | BB |
| 4 | Ø4 | E | ØE | + | 18 | T | BC |
| 5 | Ø5 | F | ØF | — | 19 | I | DF |
| 6 | Ø6 | G | 1Ø | : | 1A | : (2) | E6 |
| 7 | Ø7 | L | 11 | x | 1B | · | F7 |
| 8 | Ø8 | I | 12 | | | ! (3) | A2 |
| 9 | Ø9 | n | 13 | | | | |

Opmerkingen:
(1) deze n is iets groter dan de "officiële" versie (kode 13) en past beter tussen hoofdletters.
(2) dit is een grotere dubbele punt dan die bij kode 1A.
(3) het uitroepteken is eigenlijk te klein, maar een beter is er niet . . .
(4) de Ø (kode ØØ) is te gebruiken als letter O; zo is ook de 5 uitstekend bruikbaar als S, en een 2 is nog te herkennen als Z.

**Tabel 14**

```
0900    7620      PPSU, II
0902    3F0161    BSTA, UN      PVI voorbereiden
0905    072A      LODI, R3
0907    0507      LODI, R1
0909    3F02D9    BSTA, UN      8 spaties
090C    0F4930    LODA, I-R3    messline-data
090F    CD4890    STRA, I-R1
0912    5978      BRNR, R1
0914    7710      PPSL, RS
0916    3F020E    BSTA, UN      Mline laden
0919    7510      CPSL, RS
091B    5B0A      BRNR, R3
091D    0C1E89    LODA, R0      wacht tot "+" toets
0920    F410      TMI, R0       losgelaten wordt
0922    1879      BCTR
0924    1F0038    BCTA, UN      terug naar monitor
0927    7710      PPSL, RS
0929    3F02CF    BSTA, UN      nieuwe regel
092C    7510      CPSL, RS
092E    1B57      BCTR, UN

0930    5F A2 17 8A 17 E6 F7    zesde regel
0937    02 16 17 18 19 1A 1B    vijfde regel
093E    AA 13 00 14 15 05 BC    vierde regel
0945    0E 0F 10 12 DF 11 BB    derde regel      DATA
094C    07 08 09 0A 0B 0C 0D    tweede regel
0953    00 01 02 03 04 05 06    eerste regel
```

Start-adres: 0900

**Tabel 15**

– verander de instruktie bij adres 0924 in 1F095A (was 1F0038);
– voeg het volgend stukje programma toe:

```
095A   ┌─►0C1FCB   LODA, R0   ┐
095D   │  F440     TMI, R0    │ wacht op "VRLE"
095F   │  9879     BCFR       ┘
0961   │  0C1E88   LODA, R0   ┐ terug naar monitor
0964   │  F420     TMI, R0    │ als "PC"-toets
0966   │  1C0000   BCTA       ┘ bediend wordt
0969   │  7702     PPSL, COM  ┐
096B   │  3F0055   BCTA, UN   │ 6 tekstregels
096E   └──1B6A     BCTR, UN   ┘ op het scherm
```

**Tabel 16**

```
0900          1F0958   BCTA, UN
0903          B480     TPSU, sense  } alleen einde-raster-
0905          16       RETC         } interrupts
0906          B440     TPSU, flag   ┐
0908        ┌─1808     BCTR         │
090A        │ 7640     PPSU, flag   │ "flag" bij ieder
090C        │ 20       EORZ, R0     │ raster (re-)setten.
090D        │ CC089F   STRA, R0     │ toetsenbord
0910        │ ┌1B02    BCTR, UN     │ afvragen
0912        └►│7440    CPSU, flag   │
0914          └►3F0181 BSTA, UN     ┘
0917          9A38     BCFR         (geen toets)
0919          01       LODZ, R1     ┐
091A          451F     ANDI, R1     │ vertaal toetskode
091C          0D6122   LODA, I/R1   │
091F          E4E0     COMI, R0     │ voor "+" toets:
0921        ┌─182E     BCTR         │ sprong
0923        │ F480     TMI, R0      │ terug naar monitor
0925        │ 1C0000   BCTA         │ voor linker toetsen
0928        │ C804     STRR, R0     │
092A        │ 3F02CF   BCTA, UN     │ nieuwe regel
092D        │ 0400     LODI, R0     │
092F        │ D0       RRL, R0      ┐
0930        │ D0       RRL, R0      │ 8 x R0
0931        │ D0       RRL, R0      ┘
0932        │ 0608     LODI, R2     ┐
0934        │ 82       ADDZ, R2     │
0935        │ C1       STRZ, R1     │
0936        │►0D4961   LODA, I-R1   │ Mline laden
0939        │ CE4890   STRA, I-R2   │
093C        │ 5A78     BRNR, R2     │
093E        │ 3F020E   BSTA, UN     ┘
0941        │►0C1E8A   LODA, R0     ┐
0944        │ 6C1E8C   IORA, R0     │ wacht tot toets
0947        │ 6C1E8D   IORA, R0     │ losgelaten
094A        │ 6C1E8E   IORA, R0     │ wordt
094D        │ 44F0     ANDI, R0     │
094F        │ 9870     BCFR         ┘
0951        └►3F0055   BSTA, UN     tekst op scherm
0954          ►7420    CPSU, II     ┐ wacht op interrupts
0956          └1B7C    BCTR, UN     ┘

0958          7620     PPSU, II     ┐
095A          3F0161   BSTA, UN     │ PVI voorbereiden
095D          7702     PPSL, COM    │ en "COM"-bit setten
095F          1B73     BCTR, UN     ┘

0961   05 BC 0A 15 BC 17 17 17 data 0
0969   0B 0E 10 12 AA 17 17 17 data 1
0971   0A AA 0F 0A AA 10 17 17 data 2
0979   0D 0E 0B 56 BC 17 17 17 data 3
0981   0E AA 0D 17 17 17 17 17 data 4
0989   0E 12 AA 0D 0E 17 17 17 data 5
0991   0E AA 0D 0E 17 17 17 17 data 6
0999   0F 12 AA 17 17 17 17 17 data 7
09A1   0F 56 AA 17 17 17 17 17 data 8
09A9   11 00 11 17 17 17 17 17 data 9
09B1   05 14 0A 05 05 17 17 17 data A
09B9   15 12 10 00 11 0A 0D 0E data B
09C1   AA 12 0C 0E 17 17 17 17 data C
09C9   0A 0A 15 0D 12 10 17 17 data D
09D1   AA 0E BC BC 17 17 17 17 data E
09D9   10 0E AA BC 12 11 0E 17 data F
```

subroutine beschikbaar:

*zes regels op het scherm*

De zes regels op het scherm zijn elk opgebouwd met alle vier voorwerpen; regels 2 . . . 6 zijn in feite duplikaten. Om de gewenste tekst op het scherm te krijgen, moeten voor iedere regel de juiste voorwerp-vorm-gegevens op het juiste moment vanuit het geheugen naar de PVI overgebracht worden.
Dit wordt bereikt met een monitor-subroutine, beginnend bij adres 0055; hierbij worden de registers R0, R1 en R2 gebruikt. Om een korrekt display te krijgen, moet tevoren het "COM"-bit in de PSL geset worden (instruktie: 7702 = PPSL, COM). Bovendien moet bij het begin van ieder raster naar deze routine gesprongen worden. De subroutine wordt pas beëindigd nadat de zesde regel geschreven is. Dit houdt in dat alle verdere routines in het hoofdprogramma alleen uitgevoerd kunnen worden vlak vóór of tijdens het einde van het raster.
Als voorbeeld kan het programma van tabel 14 verder worden uitgebreid volgens tabel 15. Alle tekst-routines zijn nu in het programma verwerkt. Het nadeel blijkt echter als de "PC"-toets gebruikt wordt om naar de monitor terug te springen – er volgt dan: nieuwe regel; "message line" opnieuw laden; weer een nieuwe regel; en tenslotte "PC =" toevoegen. Alles bij elkaar wordt het display er niet beter op . . .

## Interrupts

Vorige maand nog kon ons advies ten aanzien van de interrupt-mogelijkheid in twee woorden samengevat worden: niet gebruiken. Toch hebben we dit advies zelf niet ter harte genomen: getuige het "ruimtegevecht"-programma op de nieuwe ESS-plaat!
Niet dat wij nu experts zouden zijn op dit gebied, maar in elk geval hebben wij nu wat ervaring opgedaan. Een paar tips kunnen wij dan ook geven.

**interrupts kiezen**

De PVI genereert een "interrupt request"-signaal telkens als een voorwerp (of duplikaat) op het scherm gezet is, en bij het einde van een raster. Als het interrupt-inhibit-bit in de PSU logisch 0 is, zal elk van deze interrupts hetzelfde effekt hebben: het interrupt-inhibit-bit wordt geset, het lopende programma wordt onderbroken en het programma dat bij adres 0903 begint wordt als subroutine uitgevoerd.
Stel nu dat alleen het einde-raster-interrupt voor het programma van belang is; alle andere interrupts moeten dus genegeerd worden. Dit is niet zo moeilijk: het "sense"-bit in de PSU is alleen logisch 1 bij het einde van een raster. De interrupt subroutine bij adres 0903 kan dus als volgt beginnen:

```
0903   B480   TPSU, sense
0905   36     RETE
```

Als het "sense"-bit niet "1" is, levert de "TPSU"-instruktie de "condition code"

10 op. De "return met interrupt enable"-instruktie (RETE) wordt dan uitgevoerd: de interrupt-routine wordt beëindigd. Alleen als het "sense"-bit logisch 1 is wordt de rest van de routine uitgevoerd. Meestal, tenminste, want er zit een addertje onder het gras, maar daar komen wij dadelijk op terug.
Een uitvoerige uitsplitsing van de interruptsignalen is uiteraard ook mogelijk. Het eerder genoemde ruimtegevecht-programma begint als volgt:

```
0900  1F090B  BCTA, UN (naar hoofdprogramma)
0903  B480    TPSU, sense
0905  1C0A10  BCTA (naar einde-raster-routine)
0908  1F09D5  BCTA, UN (naar voorwerp-klaar-routine)
090B  7620    PPSU, II (begin van hoofdprogramma)
```

Als in dit geval het "sense"-bit geset is, wordt de voorwaardelijke sprong naar adres 0A10 uitgevoerd. In het andere geval wordt deze sprong-instruktie genegeerd en volgt de onvoorwaardelijke sprong naar adres 09D5. Op dit adres begint de voorwerp-klaar-interrupt-routine met een volgende check:

```
09D5  0C1FCA  LODA, R0   voorwerp 3
09D8  F402    TMI, R0    klaar?
09DA  36      RETE       zo niet: return.
```

Het eindresultaat is dat slechts twee soorten interrupts worden uitgevoerd: einde-raster en voorwerp 3 (of duplikaat 3) klaar. Alle andere voorwerp-klaar-interrupts worden genegeerd.
Bij het testen van dit programma stak het bovenvermelde addertje zijn kop op: soms werd de einde-raster-routine niet uitgevoerd. De oorzaak bleek te zijn dat een "voorwerp 3 klaar"-interrupt vlak vóór het einde van het raster de betreffende routine startte; deze was lang genoeg om het raster-einde-interrupt-signaal te maskeren. In dit geval was de oplossing eenvoudig: door een geschikte keuze van vertikale offsets is er voor gezorgd dat voorwerp 3 nooit vlak voor het einde van het raster geschreven kan worden.

**Interrupt enable**

Een nadere beschouwing van het boven gegeven programma voorbeeld (van adres 0900 tot 090B) levert een verrassing op: het hoofdprogramma begint met het setten van het interrupt-inhibit-bit! Interrupt-signalen worden dus genegeerd. De interrupt-routines worden nooit uitgevoerd! Of toch?
Het zal duidelijk zijn dat het interrupt-inhibit-bit ergens in het hoofdprogramma gereset moet worden. Dat gebeurt ook, **nadat** allerhand voorbereidend werk gedaan is (basisgegevens in de PVI, tussengeheugens in het programma laden, enz.). Dan volgen bij adres 09D1 deze twee instrukties:

```
09D1  7420  CPSU, II    wacht op
09D3  1B7C  BCTR, UN    interrupts
```

De processor blijft in deze lus rondlopen tot een interrupt-signaal verschijnt. De interrupt-routine wordt dan uitgevoerd (waarbij automatisch het interrupt-inhibit-bit gereset wordt). Deze routine wordt afgesloten met een "return"-instruktie, waardoor de processor terugkeert in de wachtlus. Binnen deze lus wordt het interrupt-bit gereset, dus zowel "normale" return-instrukties (17) als "return met interrupt enable"-instrukties (37) zijn bruikbaar.

Het gebruik van interrupts is in tabel 16 uitgewerkt. Hetzelfde resultaat was weliswaar ook zonder deze mogelijkheid te bereiken, maar het gaat om het idee!
De data vanaf adres 0961 komt overeen met zestien woorden, één voor elk van de "getal"-toetsen. Voor het samenstellen van andere woorden wordt verwezen naar tabel 13. Per woord kunnen maximaal 8 letters gegeven worden; bij kortere woorden moet de rest van de regel opgevuld worden met spaties (kode 17).

## Stuurknuppels

Tot het laatst bewaard, omdat wij er weinig ervaring mee hebben . . . Het basis-principe is vrij eenvoudig, dat wel.
Twee adressen in de PVI, 1FCC en 1FCD, korresponderen met respektievelijk de linker en rechter stuurknuppel. Als de "flag" geset is, worden de vertikale posities van de beide stuurknuppels gelezen en de resultaten verschijnen in de bijbehorende adressen. De horizontale posities worden afgetast als de "flag" niet geset is. Let wel: de data in de twee PVI-adressen is slechts geldig bij het einde van elk raster, met andere woorden, als het "sense"-bit logisch 1 is. Een lage waarde in adres 1FCC of 1FCD komt overeen met "omhoog" of "naar rechts", afhankelijk van de stand van de "flag" gedurende het voorafgaande raster (toen de feitelijke AD-omzetting plaatsvond).
De hoogste en laagste waarden die bereikt kunnen worden zijn voor iedere stuurknuppel verschillend. Jammer genoeg! Het is hierdoor niet gemakkelijk om een programma te maken dat voor alle stuurknuppels zonder meer geschikt is. Het ruimtegevecht-programma op de nieuwe ESS-plaat bevat zelfs een stuurknuppel-routine . . . maar het is geblok-

**Tabel 17.**

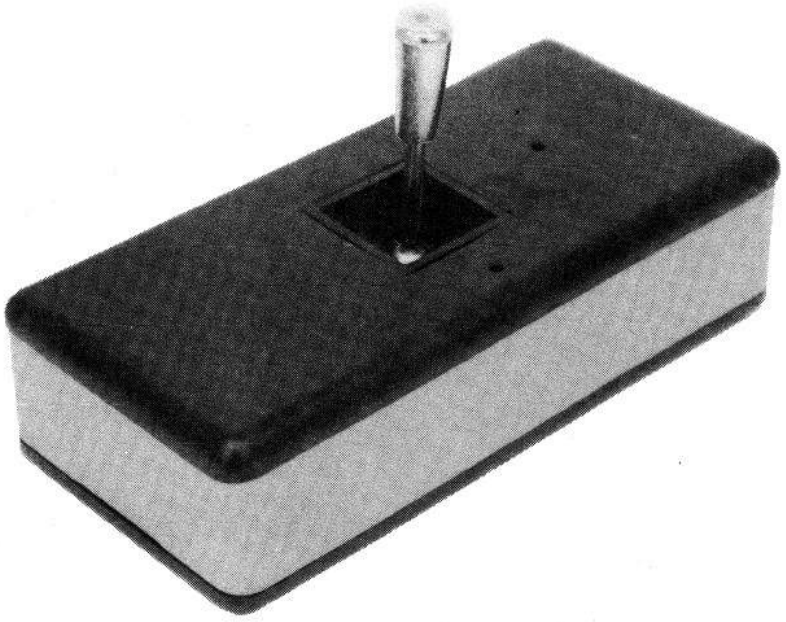
```
0900   1F0990   BCTA, UN
0903   B480     TPSU, sense     | alleen einde-raster-interrupts
0905   16       RETC
0906   B440     TPSU, flag      |
0908   1804     BCTR            | "flag" beurtelings setten
090A   7640     PPSU, flag      | en resetten
090C   1B02     BCTR, UN        |
090E   7440     CPSU, flag      |
0910   0D1FCC   LODA, R1        |
0913   0E1FCD   LODA, R2        | stuurknuppel-data uitlezen
0916   C90B     STRR, R1        | en bewaren
0918   CE095C   STRA, R1        |
091B   3F0055   BSTA, UN        | tekst op scherm
091E   0702     LODI, R3
0920   0602     LODI, R2
0922   0500     LODI, R1        | stuurknuppel-data! (1FCC)
0924   B440     TPSU, flag
0926   1802     BCTR
0928   0604     LODI, R2
092A   0418     LODI, R0        |
092C   CC096D   STRA, R0        |
092F   04E0     LODI, R0        |
0931   CC0984   STRA, R0        | presets voor subroutine
0934   04CD     LODI, R0        |
0936   CC0985   STRA, R0        |
0939   0E4963   LODA, I-R2      |
093C   CC0987   STRA, R0        |
093F   CC098A   STRA, R0        |
0942   3F035E   BSTA, UN        | register splitsen
0945   3F0967   BSTA, UN        |
0948   0498     LODI, R0        |
094A   CC096D   STRA, R0        |
094D   040E     LODI, R0        |
094F   CC0984   STRA, R0        | presets voor subroutine
0952   046D     LODI, R0        |
0954   CC0985   STRA, R0        |
0957   01       LODZ, R1        |
0958   3F0967   BSTA, UN        |
095B   0500     LODI, R1        | stuurknuppel-data! (1FCD)
095D   FB4B     BDRR, R3        |
095F   7420     CPSU, II        | wacht op interrupts
0961   1B7C     BCTR, UN        |

0963   89 71 41 29             | adres-data
```

Noot:
bij de adressen 096D, 0983 en 0985 kan steeds naar keuze een van beide alternatieven ingegeven worden. Het programma verandert deze instrukties zelf!
Start-adres: 0900.

```
0967    7710      PPSL, RS
0969    0700      LODI, R3
096B    F401      TMI, R0       } R3 presetten
096D    1802/9802 BCTR/BCFR     }
096F    0701      LODI, R3      }
0971    440E      ANDI, R0      }
0973    C2        STRZ, R2      }
0974    D2        RRL, R2       } 3 x R0
0975    82        ADDZ, R2      }
0976    0506      LODI, R1      }
0978    81        ADDZ, R1      } R1, R2 presetten
0979    C2        STRZ, R2      }
097A    0E4278    LODA, I-R2
097D    5B04      BRNR, R3
097F    D0        RRL, R0
0980    D0        RRL, R0
0981    D0        RRL, R0
0982    D0        RRL, R0
0983    44E0/440E ANDI, R0
0985    CD6829/   STRA/IORA, I/R1
        6D6829
0988    CD6829    STRA, I/R1
098B    F96D      BDRR, R1
098D    7510      CPSL, R5
098F    17        RETC, UN
0990    7620      PPSU, II
0992    3F0161    BSTA, UN      PVI voorbereiden
0995    04CC      LODI, R0      }
0997    C80F      STRR, R0      } adres-preset
0999    0702      LODI, R3      }
099B    0610      LODI, R2      }
099D    0508      LODI, R1      }
099F    7710      PPSL, RS      }
09A1    3F02CF    BSTA, UN      } nieuwe regel
09A4    7510      CPSL, RS      }
09A6    0E49CC    LODA, I-R2    }
09A9    CD4890    STRA, I-R1    } messline-data
09AC    5978      BRNR, R1      }
09AE    04C4      LODI, R0      }
09B0    C876      STRR, R0      } adres-preset
09B2    7710      PPSL, RS      }
09B4    3F020E    BSTA, UN      } Mline laden
09B7    7510      CPSL, RS      }
09B9    0504      LODI, R1      }
09BB    5A62      BRNR, R2      }
09BD    FB5C      BDRR, R3      }
09BF    7702      PPSL, COM     }
09C1    1F095F    BCTA, UN      }

09C4    01 0F 0C 0D             }
09C8    01 0F 0C 0C             } data voor
09CC    0F 11 0A 10 17 00 0F 0F } hoofdtekst
09D4    0F 11 0A 10 17 00 AA 17 }
```



keerd! De bijgeleverde informatie geeft aan hoe het weer in gebruik te nemen is. Dit is natuurlijk een verre van ideale situatie. Maar . . . er is een oplossing! Met het programma in tabel 17 kunnen stuurknuppels getest en "gekalibreerd" worden. De twee PVI-adressen worden bij het einde van ieder raster uitgelezen, waarbij de "flag" per raster beurtelings geset en gereset is. Dit levert vier getallen op, die als volgt op het scherm verschijnen:

| | |
|---|---|
| FLAG On | (= horizontaal) |
| 1FCC 75 | (= linker speler) |
| 1FCD AD | (= rechter speler) |
| FLAGG OFF | (= vertikaal) |
| 1FCC 11 | (= linker speler) |
| 1FCD 83 | (= rechter speler) |

De getallen in dit voorbeeld (75, AD, 11, 83) zijn willekeurig gekozen, ze hebben dus geen speciale betekenis.

Als de stuurknuppels bedraad zijn zoals in het oorspronkelijke artikel aangegeven, moet adres 1FCC overkomen met de linker stuurknuppel. "Flag on" korrespondeert met vertikale beweging; lage data-waarden horen bij "naar boven" en "naar rechts". Nu een verzoek. Als die lezers die een set stuurknuppels hebben ons laten weten welke resultaten zij vinden (zowel in de middenstand van de stuurknuppels als in de diverse uiterste standen) kunnen wij een beeld krijgen van de toleranties. Het is ook interessant te weten welke getallen verschijnen zonder aangesloten stuurknuppels – bij onze prototypen is dat steeds ØD. Met deze informatie moet het mogelijk zijn om een "universele" stuurknuppel routine uit te werken. Hetgeen goed van pas kan komen!

## Tenslotte

Dat was het dan, voorlopig . . . Vrijwel al onze ervaringen tot nu toe zijn in deze artikelen beschreven. Mochten wij nog meer truukjes vinden, laten wij het weten. Intussen hopen wij dat u nu voldoende informatie hebt om interessante programma's te maken! ◀

## Drukfout

*In de informatie bij de eerste ESS-plaat voor de speel-computer staat dat de snelheid van het "surround"-spelletje verandert als de data in adres ØD02 gewijzigd wordt. Fout! Het moet adres ØD20 zijn.*

# elektuur μP-systemen

**Tot nu toe zijn er in Elektuur drie microprocessorsystemen gepubliceerd. Zeker voor de beginnende amateur, die zich op microprocessor-gebied nog moet oriënteren, kan dat verwarrend werken. Dit artikel zal hopelijk de weg wijzen aan iedereen die één van de Elektuur-systemen wil bouwen.**

De drie systemen, die we hier nader zullen bespreken, zijn in volgorde van publikatie het SC/MP-systeem, de speelcomputer en de junior-computer. Hoewel het niet de bedoeling is in dit overzichtsartikel de mogelijkheden van microprocessors in het algemeen te behandelen, zal er toch zo af en toe bij wijze van voorbeeld een toepassing worden genoemd. Het is dus zeker niet zo, dat de systemen *uitsluitend* te gebruiken zouden zijn voor de genoemde toepassingen.

Op de eerste plaats is er het SC/MP-systeem. Dit is een uit modules opgebouwd geheel dat zijn naam ontleent aan de toegepaste microprocessor: de SC/MP (spreek uit skemp). Dit is een processor van het merk National met het typenummer INS 8060.

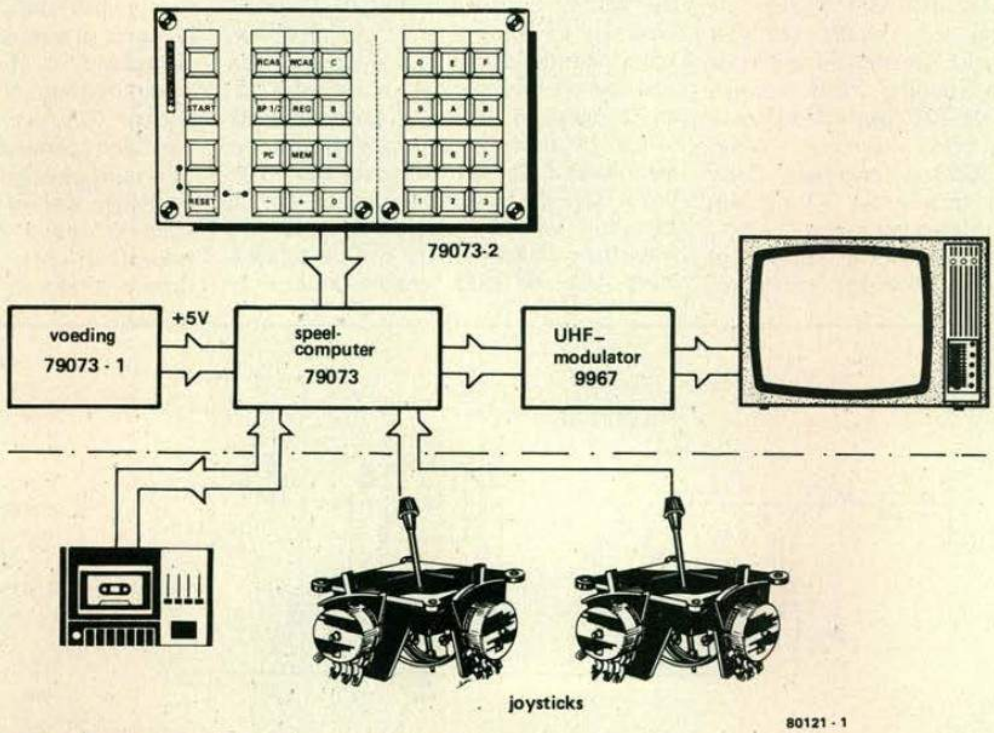
Het voornaamste kenmerk van het SC/MP-systeem is de modulaire opbouw. Dit houdt in dat er gebruik wordt gemaakt van een aantal printplaten van Eurokaart-formaat (10 x 16 cm groot) die onderling zijn verbonden via een gemeenschappelijke bus. Deze bus bestaat uit niets anders dan een aantal doorverbindingen: de punten 1 van alle eurokaarten worden hierdoor met elkaar verbonden, alle punten 2 worden met elkaar verbonden, enzovoort.

Door de modulaire opbouw is het geheel zeer flexibel. Het kleinst mogelijke systeem bestaat uit twee kaarten. Uitbreiding is zeer gemakkelijk mogelijk door meer kaarten in de bus te steken. Deze uitbreidingen kunnen niet alleen uit meer geheugen (toevoegen van RAM en/of ROM), maar ook uit bijvoorbeeld een metaalfoliedrukker bestaan.

De beide andere systemen zijn veel kleiner van opzet. De speelcomputer is weliswaar voorbereid om enige uitbreiding mogelijk te maken, maar wanneer er in de toekomst aan een uitbreiding wordt gedacht, dan zal dit hoogstens wat geheugenuitbreiding zijn. Omdat de speelcomputer maar voor één soort karwei is ontworpen, is uitbreiding ook niet nodig.

De junior-computer is, net als trouwens de speelcomputer, in zijn geheel op één print opgebouwd (exklusief de voeding). Er is geprobeerd een zo goedkoop en klein mogelijk geheel te verkrijgen, maar met behoud van alle "echte" microprocessor-eigenschappen. Via een konnektor op de print kan de junior-computer aan de SC/MP-bus en daarmee dus ook aan alle uitbreidingen van het SC/MP-systeem worden gekoppeld. In

1



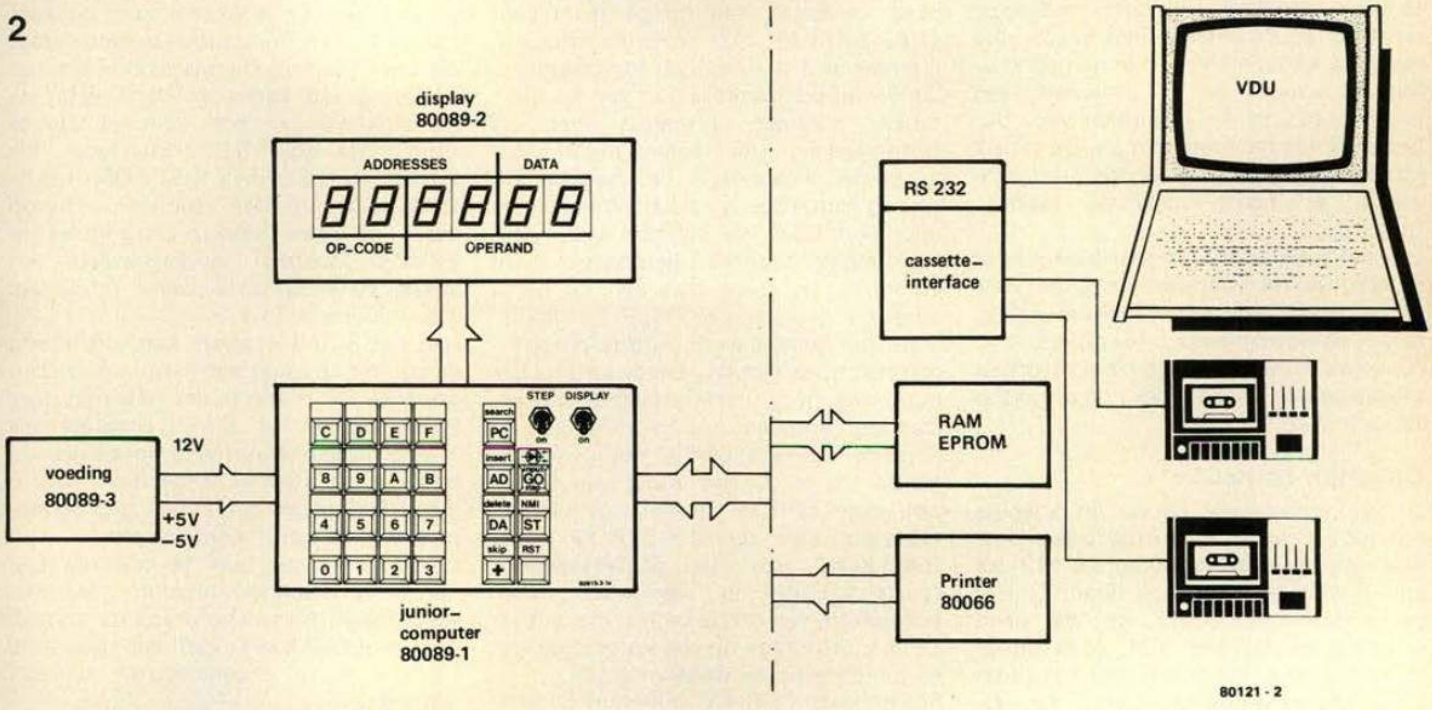

**Figuur 1. Opbouw van de speelcomputer; kleiner kan niet, groter hoeft niet.**

dat geval is er dus eigenlijk sprake van een SC/MP-systeem met een andere processor.
Het buitenbeentje van ons drietal is de speelcomputer. De speelcomputer is ontworpen om direkt kleurentelevisiebeelden te genereren. Deze beelden die bewegen kunnen en van grootte, vorm en kleur kunnen veranderen, worden geheel door het programma bepaald. Dit is dus eigenlijk een luxe TV-spel, waarbij het bovendien mogelijk is zelf spelletjes te bedenken en die vervolgens om te zetten in een programma. De hardware (dit is de computer zelf) is hier ook op aangepast: er zijn twee toetsenbordjes (voor twee spelers) met 12 toetsen en 4 gemeenschappelijke toetsen. Verder is hij voorzien van een ingang voor twee joysticks ("stuurknuppels") en is er een luidspreker ingebouwd voor speciale geluidseffekten. De programma's kunnen snel worden gewisseld met behulp van een kassetterecorder die de programma's op tape zet, waarna ze zo vaak als nodig afgespeeld kunnen worden. De nadruk ligt hier dus op gebruik door het hele gezin (spelletjes), terwijl de heer of vrouw des huizes kan programmeren.
Zowel de junior-computer als het SC/MP-systeem zijn voor algemener gebruik gedacht. Behalve konkrete opgaven (zoals sturing van de verwarming bijvoorbeeld) kunnen ook hiermee spelletjes worden gespeeld (maar niet op TV). Daarnaast bezitten beide machines mogelijkheden om er programma's mee te ontwikkelen (deze zijn al in het standaard monitorprogramma opgenomen) en kunnen ze ook voor hogere programmeertalen worden gebruikt. Zo is er voor het SC/MP-systeem al een Tiny-BASIC beschikbaar.
Alle kommando's worden bij het SC/MP-systeem gegeven via een terminal. Dit is een apart apparaat dat bestaat uit een toetsenbord (meestal zo'n 60 toetsen) waarmee behalve de cijfers 0 tot en met 9 ook het gehele alfabet en enkele zogenaamde kontrolekarakters kunnen worden overgebracht. Eventuele resultaten vanuit de microcomputer worden vervolgens zichtbaar gemaakt met behulp van een drukker of, wat veel goedkoper is, op een TV-scherm. Om met het SC/MP-systeem te kunnen werken is de terminal dus onontbeerlijk. Een geschikte terminal is beschreven in onze uitgaven van november en december 1978.
Het grote voordeel van een terminal is, dat dan in een hogere programmeertaal met de computer kan worden gekommuniceerd. In zo'n taal worden immers "normale" woorden gebruikt en het is dus noodzakelijk over een toetsenbord met het hele alfabet te beschikken.
De junior-computer doet het een stuk eenvoudiger, en daardoor bovendien goedkoper. Er wordt gebruik gemaakt van 23 toetsen. Resultaten verschijnen op 6 zeven-segment-displays.

## De microprocessor

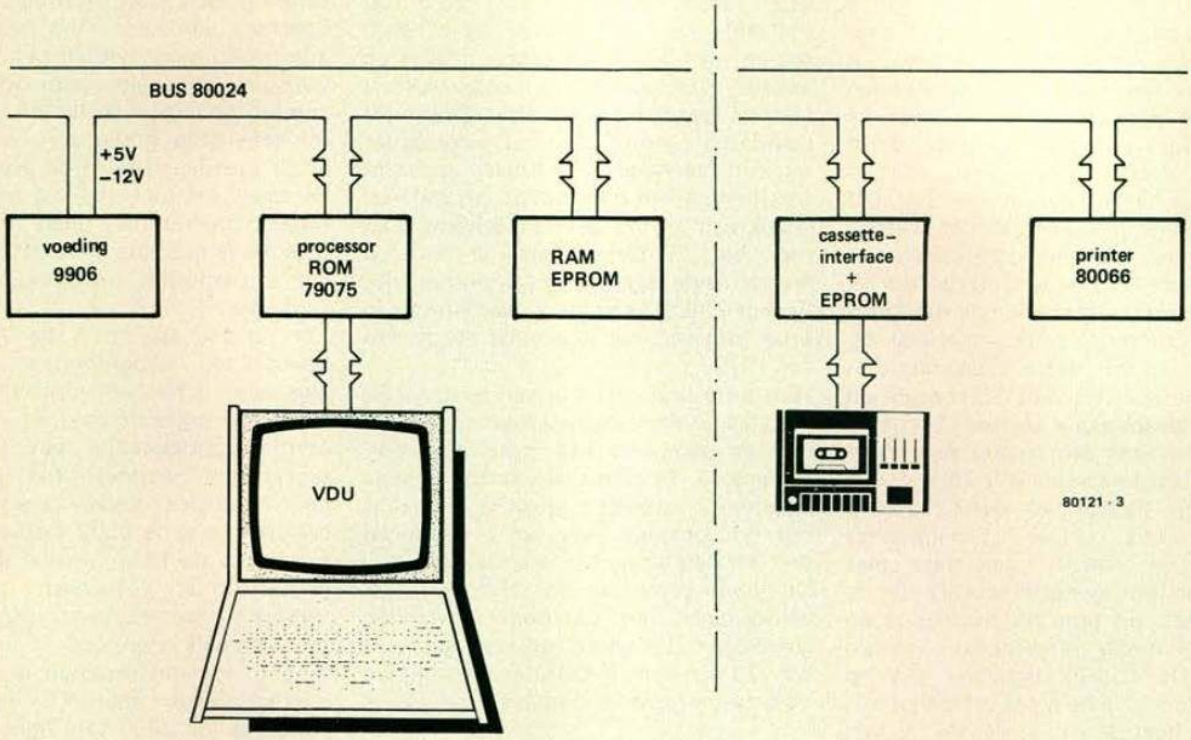
Er is nog een aspekt dat voor- en tegenstanders met rode koppen tegenover elkaar laat staan: de keuze van de microprocessor. De microprocessor vormt het hart van het systeem (de microcomputer) en bepaalt voor een groot deel de mogelijkheden en de snelheid waarmee alle klusjes worden afgehandeld. Op het eerste gezicht lijkt het aantrekkelijk een microprocessor te kiezen die zo veel mogelijk kan en die zo snel mogelijk is. Daar staan echter ook enkele zaken tegenover. Een microprocessor met honderden instrukties is veel lastiger te programmeren, want de ervaring leert dat de programmeur bij voorkeur alle instrukties zo'n beetje in zijn hoofd moet hebben. Wat betreft de snelheid: een processor die snel is zonder dat er ook snelle (en dus duurdere) geheugens nodig zijn, is natuurlijk altijd prettig, maar in de praktijk komt de traagheid toch pas tot uiting bij de hogere programmeertalen of bij geweldige matematische berekeningen. Maar het is natuurlijk wel leuk, zo'n snelle processor.
Een ander aspekt is de beschikbare hoeveelheid programma's. Over het algemeen is het wel mogelijk programma's van anderen over te nemen met relatief gemakkelijk aan te brengen wijzigingen, wanneer het programma voor dezelfde processor is geschreven. In dit opzicht is de 6502 favoriet, want in Europa is de KIM (waarin deze microprocessor is toegepast) zeer sterk verspreid en hiervoor zijn erg veel programma's in omloop.
In alle drie de systemen is een andere microprocessor gebruikt: in de speelcomputer de 2650 van Signetics, in het SC/MP-systeem de INS 8060 van National en in de junior-computer de 6502 van Rockwell. De SC/MP (8060) is van dit gezelschap de eenvoudigste en langzaamste processor. De 6502 is het meest uitgebreid en ook het snelst. Tussen deze beide uitersten in liggen de prestaties van de 2650.
Aangezien de relatieve traagheid van de SC/MP-processor door sommigen als een gemis wordt gezien, is er in de handel voor het SC/MP-systeem ook een processorkaart met als processor een Z-80 verkrijgbaar. Deze laatste aktiviteit staat echter (nog) geheel los van Elektuur. Tot slot geven we nog een

2




**Figuur 2. Het gedeelte links van de stippellijn is de blokschematische voorstelling van de junior-computer zonder uitbreiding. De modules rechts kunnen naar behoefte worden toegevoegd.**

3




**Figuur 3. Het SC/MP-systeem is direkt uitgerust met een terminal. Door op de processor-print een voorgeprogrammeerde ROM met Tiny BASIC te plaatsen, kan in BASIC worden gewerkt. Rechts van de stippellijn zijn de mogelijke uitbreidingen aangegeven.**

kort gehouden beschrijving van de opbouw van ieder systeem en – vooral – welke kombinaties er mogelijk zijn. Wil men zich meer verdiepen in de technische achtergronden, dan moeten we verwijzen naar de desbetreffende artikelen.

## De speelcomputer

Om te beginnen: de speelcomputer. Deze bestaat uit een hoofdprint met toetsenbord (zie figuur 1), een voeding (5 V) en meestal een UHF-modulator om hem op de antenne-ingang van elke normale kleuren-TV te kunnen aansluiten. Daarnaast is het gewenst een kassetterecorder te gebruiken voor het bewaren van de programma's. De gehele kassette-interface is al op de print aanwezig, er hoeft niets te worden toegevoegd.

Om het programmeren gemakkelijker te maken, is de monitor uitgerust met uitgebreide debug(foutzoek)-mogelijkheden waaronder twee breakpoints.

Als extra kunnen twee joysticks worden toegepast om de spelletjes nog attraktiever te maken.

## De junior-computer

De junior-computer bestaat in principe ook uit één print. Natuurlijk is daarnaast ook nog een voeding nodig die +12, +5 en –5 volt levert (zie ook figuur 2). Het toetsenbord is direkt op de print geplaatst omdat hier niet, zoals bij de speelcomputer, het toetsenbord gesplitst hoeft te worden bij het spelen. Resultaten verschijnen op 6 zeven-segment-displays. Deze displays zijn op een klein hulpprintje gemonteerd dat bij voorkeur schuin op de hoofdprint wordt gesoldeerd. Het geheel van hoofdprint, display-print en voeding vormt het kleinst mogelijke geheel. Daarnaast is uitbreiding mogelijk met een kassette-interface voor twee kassetterecorders (er kan er natuurlijk ook maar één worden gebruikt).

Verder is het via de konnektor aan de smalle zijde van de print mogelijk een verbinding met de SC/MP-bus te maken. Dit is bijv. van belang wanneer men meer geheugen dan op de print past (1 K EPROM met het monitorprogramma en 1 K RAM) wil toevoegen.

De junior-computer is met een komfortabele monitor uitgerust met als bijzonderheid de zogenaamde hex-assembler. Hiermee is het mogelijk bij spronginstrukties symbolische namen te gebruiken voor de plaatsen waar naar toe moet worden gesprongen. De computer berekent dan zelf de juiste adressen.

Voor het grotere werk (hogere programmeertalen, assembler, enzovoort) is het noodzakelijk een terminal aan te sluiten. Dat kan eveneens via de print van de kassette-interface. Om de nodige intelligentie toe te voegen, moet dan echter ook meer EPROM worden toegevoegd. Hiervoor kan de 8 K-EPROM- + 8 K RAM-kaart van het SC/MP-systeem (zoals we die in september zullen publiceren) worden gebruikt, die dan via de aansluitpunten op een korte zijde van de junior-print wordt aangesloten.

Samengevat: Het minimumsysteem werkt in hex-kode, met de getallen 0 t/m 9 en vervolgens met A, B, C, D, E en F. Wil men programma's bewaren op tape, dan moet de kassette-interface-print worden toegevoegd. Wil men met de beschikbare editor, assembler, disassembler, of in hogere programmeertalen (BASIC is beschikbaar) werken, dan moet bovendien extra EPROM en/of RAM worden toegevoegd.

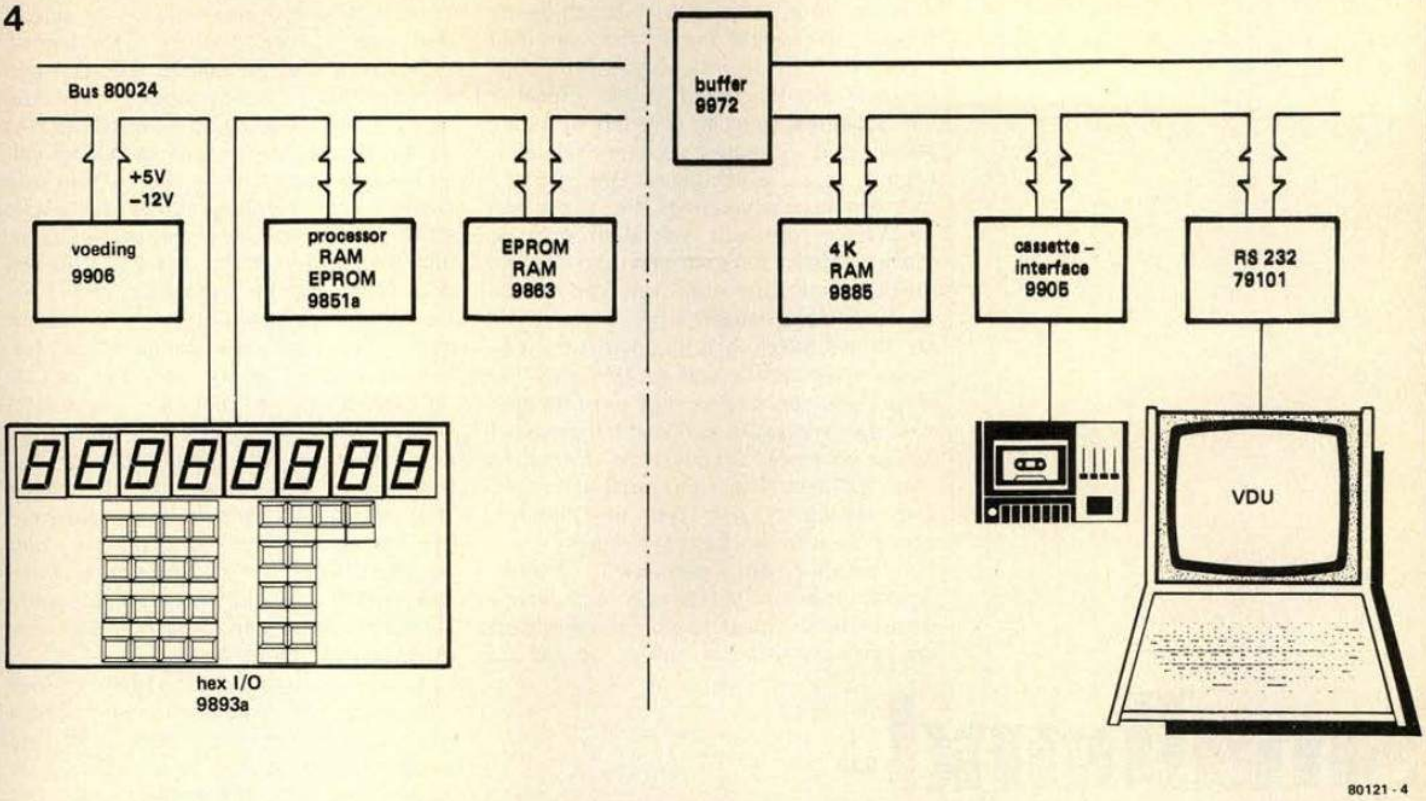
## Het SC/MP-systeem

Tenslotte het SC/MP-systeem (zie figuur 3). Omdat dit geheel modulair is opgebouwd zijn er meerdere konfiguraties mogelijk. Een minimumsysteem bestaat uit twee kaarten: De processorkaart met buffering van adres- en databus en de mogelijkheid om een terminal aan te sluiten (de z.g. RS 232 interface). Als tweede kaart kan de 8 K-EPROM- + 8 K-RAM-kaart worden gekozen. Hierop moet dan het monitorprogramma in EPROM worden ondergebracht en zoveel RAM als men wenst (minimaal 1 K, maximaal 8 K).

Met het SC/MP-systeem kan uitsluitend in kombinatie met een terminal worden gewerkt. Dit houdt in dat zelfs met deze twee kaarten al BASIC-programma's kunnen lopen. Daartoe is op de processorkaart een IC-voet aangebracht waarin een ROM (door de fabriek geprogrammeerd geheugen) wordt geplaatst. Om programma's op tape te bewaren kan een kassette-interfaceprint worden toegevoegd. Met behulp van de metaalfolie-interface-kaart kan een gedrukte listing van programma's worden gemaakt.

Welke voedingsspanningen nodig zijn, hangt af van de gebruikte EPROM's. De 8 K-EPROM- + 8 K-RAM-kaart gebruikt

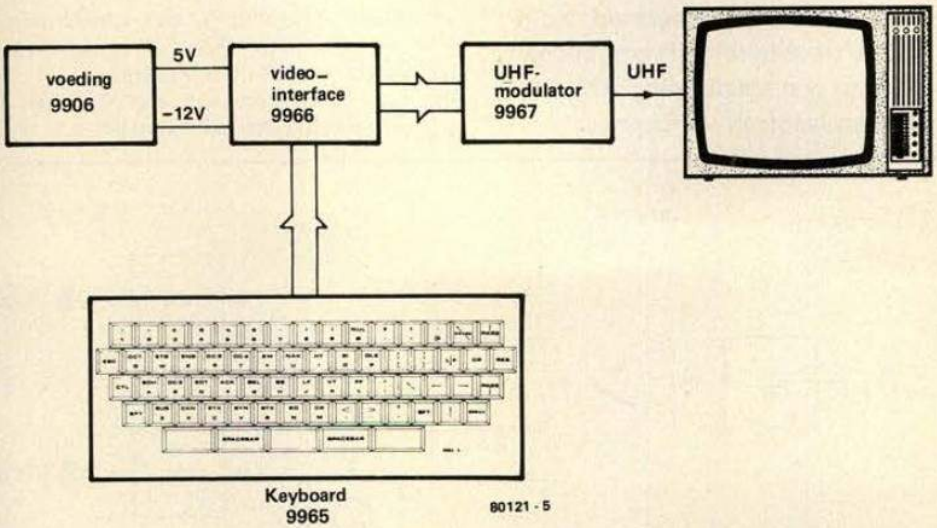
4




**Figuur 4. Het wat kleiner (en met andere kaarten!) opgezette SC/MP-systeem zoals beschreven in het boek "SC/MP, μP voor zelfbouw, deel 1" (Uitg. Elektuur).**

2716 EPROM's die 5 volt nodig hebben. Op de kassette-interface-kaart is plaats voor EPROM's van het type 5204 en die hebben +5 en –12 volt nodig. In plaats van de 2716 kan ook de 2708 worden toegepast. Dit heeft tot gevolg, dat voor de 8 K-print nog twee voedingsspanningen (+12 V en –5 V) aan de voeding moeten worden toegevoegd en dat de geheugenkapaciteit wordt gehalveerd (4 K). De bestaande voeding voor het systeem (behalve de trafo) is ondergebracht op de busprint en levert +5 volt en –12 volt.

5




**Figuur 5. Een terminal kan ook met Elektuur-printen worden gebouwd. Hiermee kunnen 16 regels met ieder 64 karakters via een gewone TV zichtbaar worden gemaakt.**

Behalve de hier genoemde modules zijn er ook nog enkele kaarten verkrijgbaar die gebaseerd zijn op gebruik in een wat kleiner systeem dat kommuniceert met de buitenwereld via een toetsenbordje en 8 zeven-segment-displays (zie figuur 4). De opzet gaat dan echter sterk overeenkomen met die van de juniorcomputer in zijn basisvorm en is daarom voor de SC/MP iets minder aantrekkelijk. Met behulp van deze kaarten kan als volgt een geheel worden samengesteld:

De processorkaart + uitbreidingskaart vormen de eigenlijke computer. De data-lijnen zijn niet gebufferd, waardoor het niet zonder meer mogelijk is het systeem een willekeurige grootte te geven. Wil men toch een groot systeem opbouwen, dan kan een databusbuffer worden toegevoegd (EPS-print 9972). De gebruiker beschikt over 1½ K EPROM die in beslag wordt genomen door het monitorprogramma) en 1 K RAM. Uitbreiding van RAM is mogelijk met de 4 K-RAM-kaart.

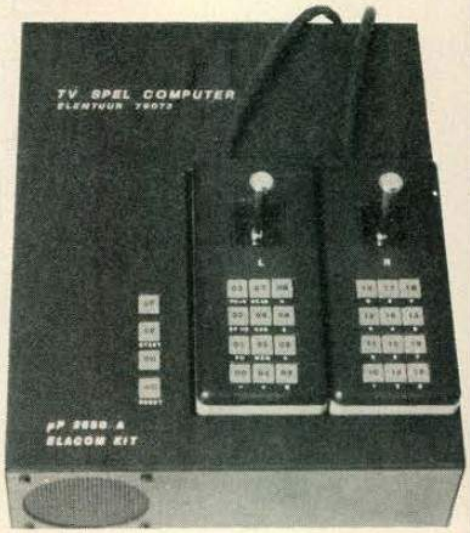
Het ingeven van data en het uitlezen van de processor-informatie geschiedt met 26 druktoetsen en 8 zeven-segment-displays die onder zijn gebracht op de hex-I/O print. Om een kassetterecorder aan te kunnen sluiten is nog een klein hulpprintje nodig (EPS 9905).

Aansluiten van een terminal is wèl mogelijk, er is ook dan weer een hulpprintje (EPS 79101, interface voor μP) nodig en de I/O-print is dan overbodig geworden. Een en ander is in figuur 3 nog schematisch weergegeven. Dit systeem is beschreven in het boek "SC/MP, μP voor zelfbouw, deel 1".

# TV-speel-computer-tips

## belofte maakt schuld

**In het artikel "spelen met de TV-speel-computer (2)" uit het novembernummer van 1979 vroegen we de lezers om hun ervaringen met de stuurknuppels, en beloofden hierop terug te komen als er voldoende gegevens waren.**
**Verschillende interessante reakties kwamen binnen, vaak met alternatieve ideeën en kommentaar. Voldoende informatie om nu ook alle andere geïnteresseerde lezers op de hoogte te brengen.**




We vallen maar meteen met de deur in huis en beginnen met de stuurknuppels. De reakties en informatie van de lezers waren, op zijn zachtst uitgedrukt, nogal uiteenlopend. Ongelukkigerwijze was er ook nog een foutje geslopen in het testprogramma (tabel 17 in het novembernummer): de instruktie op adres ØØ97A moet ØE427B zijn in plaats van ØE4278. Zolang dit niet is gekorrigeerd staat er alleen maar onzin op het scherm.
Een aantal lezers had deze fout echter al ontdekt of een alternatief programma gemaakt, zodat toch heel wat "stuurknuppeldata" binnenkwam. En wat een variatie! De gevonden laagste waarden lagen tussen Ø5 en 28 en de hoogste tussen 25 en FA. De middenstand kon alles zijn tussen 15 en 7E. Wat doe je nou als het minimum bij de een groter is dan het maximum bij de ander? Het enige resultaat dat, zoals verwacht, hetzelfde bleef was de waarde zonder aangesloten stuurknuppel: ØD.
Toch geloven we dat we een voor iedereen bevredigende oplossing hebben. Deze is gebaseerd op twee konklusies uit de hierboven vermelde resultaten:

- Als stuurknuppels worden gebruikt, is een (automatische) kalibratie noodzakelijk.
- Indien mogelijk, kunnen de stuurknuppels het beste gebruikt worden als vier-weg-schakelaar (signaal voor omhoog, omlaag, links of rechts).

Het verkrijgen van waarden die korresponderen met alle mogelijke standen is bijna onmogelijk, tenminste voor zover deze kompatibel moeten zijn met andere computers. Voor persoonlijke programma's is dat natuurlijk geen probleem.
Voordat we onze oplossing geven nog iets anders. Zoals verschillende lezers al opmerkten is onze "definitie" van de stuurknuppelaansluitingen niet ideaal. Verder korrespondeert deze niet met die in het programma band 1 op de ESS003 plaat, en ook niet met de aansluitingen bij een kommerciële "TV-speelcomputer", die werkt met dezelfde CPU en PVI. Daarom is besloten de volgende "standaard" in te voeren (zie figuur 1):

- Linker stuurknuppel = adres 1FCC, rechter stuurknuppel = 1FCD.
- Horizontale beweging = "flag off", vertikale beweging = "flag on".
- Lage data-waarde = links of omhoog, hoge data-waarde = rechts of omlaag.

Hieruit volgt dat er toch enig soldeerwerk verricht moet worden om een bestaande TV-speel-computer op deze standaard aan te passen. Gelukkig niet veel. Tabel 17 uit het novembernummer (met de bovengenoemde wijziging) geeft een geschikte testprocedure.
Nu dan onze "oplossing": Een automatische routine voor kalibratie en voor "stuurknuppel afvragen" die kan worden opgenomen in ieder willekeurig programma waarbij men stuurknuppels gebruikt. De komplete routine is afgedrukt in tabel 1. Zoals hier gegeven, begint de eigenlijke kalibratieroutine op adres ØF94. Een programma kan daardoor op twee manieren worden gestart: 1FØF94 (BCTA,UN) of 3FØF94 (BSTA,UN). In het laatste geval wordt de kalibratieroutine beeindigd op adres ØFAF met 16, CØ, CØ. In het eerste geval kan een sprong naar elk gewenst adres worden ingevoegd door middel van 1EXXXX op het laatstgenoemde adres. In beide gevallen wordt de kalibratieroutine eenmaal doorlopen aan het begin van het programma. Hierbij wordt aangenomen dat de stuurknuppels in de middenstand staan; de schakelpunten worden dan berekend ten opzichte van deze positie en opgeslagen

Tabel 1.

```
ØF8Ø    Ø881      LODR,RØ,Ind   }
ØF82   ┌ØC1FCB    LODA,RØ       }  subroutine:
ØF85   │ F44Ø     TMI,RØ        }  wacht op VRLE
ØF87   └ 9879     BCFR          }
ØF89     17       RETC,UN       }

ØF8A     C1       STRZ,R1       }
ØF8B     51       RRR,R1        }
ØF8C     51       RRR, R1       }
ØF8D     453F     ANDI,R1       }  subroutine:
ØF8F     A1       SUBZ,R1       }  berekenen van grenzen
ØF9Ø     C2       STRZ,R2       }
ØF91     81       ADDZ,R1       }
ØF92     81       ADDZ,R1       }
ØF93     17       RETC,UN       }

KALIBRATIE-ROUTINE
ØF94     7660     PPSU,II/Flag  }  (flag on = vertikaal)
ØF96     7518     CPSL,RS/WC    }
ØF98   ┌ 3B66     BSTR,UN       }  wis VRLE,
ØF9A   │ 3B66     BSTR,UN       }  een raster wachten
ØF9C   │ Ø8BØ     LODR,RØ,Ind   }  1FCC = links
ØF9E   │ ØBB1     LODR,R3,Ind   }  1FCD = rechts
ØFAØ   │ 3B68     BSTR,UN       }
ØFA2   │ C81D     STRR,RØ       }  onder- en bovengrens voor links
ØFA4   │ CA1A     STRR,R2       }  berekenen en opslaan
ØFA6   │ Ø3       LODZ,R3       }
ØFA7   │ 3B61     BSTR,UN       }  onder- en bovengrens voor rechts
ØFA9   │ C818     STRR,RØ       }  berekenen en opslaan
ØFAB   │ CA15     STRR,R2       }
ØFAD   │ B44Ø     TPSU,flag     }
ØFAF   │ 16       RETC          }  terug als alle grenzen zijn
ØFBØ   │ CØ,CØ    2xNOP         }  bepaald. Opm.: absolute sprong
ØFB2   │ 744Ø     CPSU,flag     }  met 1Exxxx
ØFB4   │ Ø5Ø4     LODI,R1       }  (flag off = horizontaal)
ØFB6   │┌ØD4FCØ   LODA,I-R1     }
ØFB9   ││CD6FC4    STRA,I/R1     }  verschuif data
ØFBC   │└ 5978     BRNR,R1       }
ØFBE   └ 1B58      BCTR,UN       }

                laag hoog  laag hoog
ØFCØ            ØØ  ØØ     ØØ  ØØ  ← hor.   }
ØFC4            ØØ  ØØ     ØØ  ØØ  ← vert.  }  data-grenzen
                links      rechts

SUBROUTINE: STUURKNUPPEL AFVRAGEN
ØFC8     77Ø2     PPSL,COM      }  (of 7712 = PPSL,RS/COM)
ØFCA     2Ø       EORZ,RØ       }
ØFCB     C1       STRZ,R1       }
ØFCC     C2       STRZ,R2       }  R1, R2 wissen en
ØFCD     ØC1FCC   LODA,RØ       }  stuurknuppel-data laden
ØFDØ     ØF1FCD   LODA,R3       }
ØFD3     B44Ø     TPSU,flag     }
ØFD5   ┌ 98Ø2     BCFR          }  preset voor R1, indien vertikaal
ØFD7   │ Ø5Ø4     LODI,R1       }  Opm.: soms is 1802 = BCTR nodig
ØFD9   └→ED2FBF   COMA,I+R1     }
ØFDC   ┌ 9AØ2     BCFR          }
ØFDE   │ A6Ø1     SUBI,R2       }  resultaat linker
ØFEØ   └→ED2FBF   COMA,I+R1     }  stuurknuppel in R2
ØFE3   ┌ 99Ø2     BCFR          }
ØFE5   │ 86Ø1     ADDI,R2       }
ØFE7   └→Ø3       LODZ,R3       }
ØFE8     Ø7ØØ     LODI,R3       }
ØFEA     ED2FBF   COMA,I+R1     }
ØFED   ┌ 9AØ2     BCFR          }  resultaat rechter
ØFEF   │ A7Ø1     SUBI,R3       }  stuurknuppel in R3
ØFF1   └→ED2FBF   COMA,I+R1     }
ØFF4   ┌ 99Ø2     BCFR          }
ØFF6   │ 87Ø1     ADDI,R3       }
ØFF8   └→17       RETC,UN       }
ØFF9     CØ                     }
  ,      ,                      }  plaats voor 751Ø = CPSL,RS
  ,      ,                      }  en/of CExxxx,
ØFFF     CØ                     }  CFxxxx voor data opslag.
```

Tabel 1. De kalibratie- en afvraagroutine voor de stuurknuppels.

in adres ØFCØ en verder.

Even een opmerking tussendoor. De subroutine "wacht op VRLE" (die begint op adres ØF8Ø) kan ook nuttig zijn op andere plaatsen in het hoofdprogramma.

Na de kalibratie van de stuurknuppels wordt teruggegaan naar het hoofdprogramma. Op elke gewenste plaats in dit programma kunnen de stuurknuppels "gelezen" worden door naar de subroutine te springen die begint op adres ØFC8. Voor een goede werking moet deze sprong naar de subroutine plaats vinden aan het einde van het raster, bijvoorbeeld na een "wacht op VRLE"-lus. Afhankelijk van het toegepaste programma kunnen verschillende variaties van deze routine nuttig zijn:

- door de instruktie op adres ØFC8 te veranderen in 7712 worden de hoogste registers gekozen (om zo de data in R1 . . . R3 te beschermen).
- vanaf adres ØFF8 zijn de volgende toevoegingen mogelijk: óf het resetten van de registers (751Ø = CPSL, RS), óf het opslaan van de gevonden data (R2 bevat de data van de linker stuurknuppel en R3 de data van de rechter stuurknuppel).
- De instruktie op adres ØFD5 is afhankelijk van de plaats waar de "flag" wordt geset of gereset. Om zowel de horizontale als vertikale stuurknuppelpositie af te vragen moet de flag beurtelings worden geset en gereset bij het begin van opeenvolgende rasters. De hier gegeven routine neemt aan dat de flag wordt veranderd *nadat* de stuurknuppel-afvraag-routine is doorlopen. In sommige gevallen kan het echter wenselijk zijn eerst de flag te veranderen; de instruktie op adres ØFD5 moet dan 18Ø2 worden.
- Indien gewenst, kan de hele routine op een andere plaats in het geheugen worden gezet. Aangezien de meeste instrukties gebruik maken van relatieve adressering kunnen deze onveranderd blijven. De enige uitzonderingen zijn de absoluut geïndexeerde instrukties op de adressen ØFB6, ØFB9, ØFD9, ØFEØ, ØFEA en ØFF1. Deze zijn afhankelijk van de dataplaatsen die nu op adres ØFCØ en verder staan.

Tabel 2 geeft een eenvoudig demonstratieprogramma dat laat zien hoe deze routine werkt. Nadat de twee programma's zijn geladen (tabel 1 en 2) wordt het hoofdprogramma gestart op adres Ø9ØØ. De eerste twee voorwerpen, in dit geval de tekens "PC" en "=", springen nu naar het midden van het beeld ("Ø9ØØ" blijft in de rechter bovenhoek van het scherm). De plaats van de twee voorwerpen kan daarna worden veranderd door het bedienen van respektievelijk linker en rechter stuurknuppel.

We hopen dat de hier gegeven informatie voldoende is voor de lezers die hun eigen programma's maken. Voor alle andere lezers is alleen de mededeling van belang dat alle toekomstige programma's van de "Elektuur Software Service" goed zullen werken op hun computer,

tenminste indien de stuurknuppels zijn aangesloten zoals in dit artikel is beschreven.

## Interrupt

Een veelbesproken onderwerp bij de lezerreakties was de interrupt-mogelijkheid. Van de heer Norman ontvingen we een lange brief met de volgende tips:
"Bij het gebruiken van interrupts geeft u de metode om een lus in het hoofdprogramma te leggen en verder alles aan de interrupt routine(s) over te laten. Dit is verspilling van processortijd en ik verdeel het "werk" dan ook liever, bijvoorbeeld objektbeweging en botsingsdetektie voor de interrupt routines en puntentelling, geluiden, toetsen afvragen enzovoorts door het hoofdprogramma.
Om zowel hoofdprogramma als interrupt te laten lopen is het noodzakelijk dat registers en condition codes niet "botsen" en u beschrijft dit niet in details.
Als bijvoorbeeld de interruptroutine de bovenste registergroep gebruikt en het hoofdprogramma de onderste, dan kan een typische interrupt routine als volgt beginnen:

```
7710      PPSL,RS
CC08FE    STRA,R0
13        SPSL
CC08FF    STRA,R0
```

en eindigen met:

```
0C08FF    LODA,R0
93        LPSL
0C08FE    LODA,R0
7510      CPSL,RS
37        RETE,UN
```

Het is nodig dat PSL bewaard blijft, anders gaat het hoofdprogramma beslissingen nemen aan de hand van condition codes die door de interrupt routine zijn geset!"

Klopt. Hoewel, een andere lezer wees er op dat bovenstaande routine niet helemaal korrekt is: *na* het opnieuw wegschrijven van de PSL-data wordt R0 weer geladen, zodat de condition code verandert!
De volgende routine schijnt het wel goed te doen:
Begin de interrupt routine met:

```
7710      PPSL,RS
CC09F1    STRA,R0
13        SPSL
CC09F3    STRA,R0
24 EF     EORI,R0
CC09F5    STRA,R0
```

en eindig, bijvoorbeeld op 09F0, met:

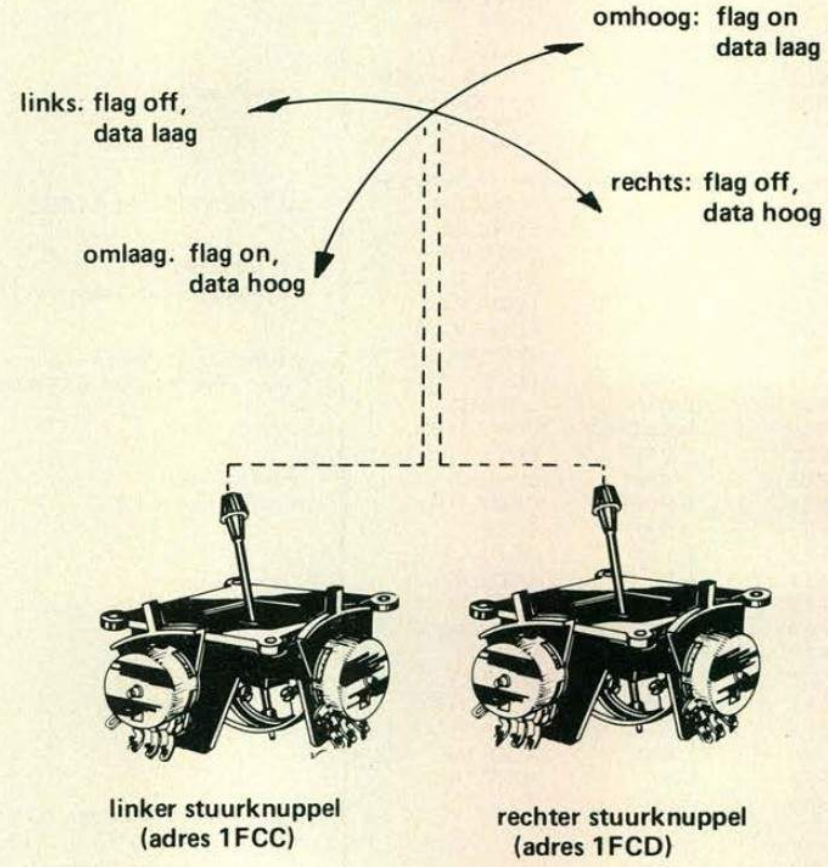
```
09F0      04XX      LODI,R0
09F2      77XX      PPSL
09F4      75XX      CPSL (RS
                      inbegrepen!)
09F6      37        RETE,UN
```

Het is duidelijk dat de drie absolute adressen in de "save" routine (09F1, 09F3 en 09F5 in dit voorbeeld) afhankelijk zijn van de plaats van de "restore"

**Tabel 2**

```
STUURKNUPPEL DEMONSTRATIEROUTINE
0900      3F0F94    BSTA,UN     kalibratieroutine
0903  ->  7440      CPSU,flag   horizontaal
0905      3F0F82    BSTA,UN     wacht op VRLE
0908      3F0FC8    BSTA,UN     stuurknuppel-data
090B      0828      LODR,R0
090D      82        ADDZ,R2     horizontale positie
090E      C825      STRR,R0     voorwerp 1
0910      CC1F0A    STRA,R0
0913      0821      LODR,R0
0915      83        ADDZ,R3     horizontale positie
0916      C81E      STRR,R0     voorwerp 2
0918      CC1F1A    STRA,R0
091B      7640      PPSU,flag   vertikaal
091D      3F0F82    BSTA,UN     wacht op VRLE
0920      3F0FCA    BSTA,UN     stuurknuppel-data
0923      0812      LODR,R0
0925      82        ADDZ,R2     vertikale positie
0926      C80F      STRR,R0     voorwerp 1
0928      CC1F0C    STRA,R0
092B      080B      LODR,R0
092D      83        ADDZ,R3     vertikale positie
092E      C808      STRR,R0     voorwerp 2
0930      CC1F1C    STRA,R0
0933      1B4E      BCTR,UN
0935      44 55 77 71           positie-data
```

1




**Figuur 1. Deze tekening illustreert de nieuwe "stuurknuppel-standaard" zoals die in de tekst wordt beschreven.**

bytes die hierboven als "XX" zijn aangegeven.

## PVI-punten

Een ander onderwerp dat nogal wat kommentaar losmaakte was de PVI. Twee punten werden vaak genoemd:

- Uit de bij de print geleverde dokumentatie blijkt dat er verschillende adressen in de PVI beschikbaar zijn als "kladblok". Tot nu toe hebben we ze zelf nooit gebruikt, maar verschillende lezers hebben er op gewezen dat deze op dezelfde manier kunnen worden gebruikt als de gewone RAM.
- Ook staat in de dokumentatie dat het "I/O and control" veld in feite vier keer herhaald wordt:

1FCØ . . . 1FCD, 1FDØ . . . 1FDD, 1FEØ . . . 1FED en 1FFØ . . . 1FFD.

Dit is vooral belangrijk voor de data op adressen 1FCA en 1FCB (botsingen, VRLE, etc.). Deze bytes worden gewist bij het lezen, wat erg lastig kan zijn. Een lezer wees er echter op dat bij het lezen van bijvoorbeeld 1FCA alleen dit byte wordt gewist en niet 1FDA, 1FEA of 1FFA. Dit betekent dat voor elk voorwerp een ander adres gebruikt kan worden om de data te lezen, indien nodig, zonder de informatie aan te tasten die later voor een van de andere voorwerpen nodig is. Heel nuttig!

## Vragen en fouten

Er wordt ons vaak gevraagd waarom in sommige programma's Ø4 wordt weggeschreven op adres 1E8Ø. Dat was nieuw voor ons, maar we hebben het antwoord gevonden. Er bestaat een kommerciële versie van de TV-speelcomputer met dezelfde CPU en PVI. Als daar Ø4 op adres 1E8Ø staat, worden de geluidseffekten via de TV-ontvanger weergegeven. We weten niet hoe dat precies gebeurt – bij ons apparaat gaat dat in elk geval niet – maar misschien kan iemand ons helpen?

Een andere, regelmatig terugkerende vraag was de aansluiting van weerstand R58. In het schema is deze weerstand (zoals het hoort) verbonden met de video-uitgang. Oplettende lezers hebben echter ontdekt dat deze weerstand op de print aan de voedingsspanning hangt. Toen we deze fout ontdekten werden onmiddelijk enkele prototypes getest om te zien wat het gevolg hiervan was. Tot onze grote verbazing – en opluchting – was er geen enkel verschil in het beeld. Daarom werd dit nog niet vermeld.

Sommige lezers hebben nog problemen met de "vertikale offset" voor de duplikaten. Misschien is niet goed duidelijk gemaakt dat een vertikale offset "FF" wordt gebruikt als "min 1": de afstand tussen de duplikaten wordt nul. Om de duplikaten helemaal te laten verdwijnen moet een offset "FE" worden ingevoerd.

Tenslotte nog de opmerking dat in het oorspronkelijke artikel werd vermeld dat alleen file-nummers van 1 tot 9 zijn toegestaan. Dit is niet nodig; elk nummer van 1 . . . F kan worden gebruikt.

De heer Norman ontdekte ook nog een fout: "De aansluitingen van IC6 in het schema op bladzijde 58 van Elektuur nummer 186 zijn foutief aangegeven. De A7-aansluiting moet naar pin 13 en A6 naar pin 14. De aanduidingen bij pin 13 en 15 zijn verwisseld. Pin 15 is de 2C-input en pin 13 de 2B-input."

## Een nuttige tip

Hebt u al eens geprobeerd zelf een programma te maken, om er dan na het proefdraaien achter te komen dat u een paar belangrijke stappen vergeten hebt? Nou, u bent zeker niet de enige!

Het tussenvoegen van de nodige stappen kan gebeuren door drie van de oorspronkelijke instruktie-bytes te vervangen door een onvoorwaardelijke sprong naar een lege geheugenplaats. Daar worden de zojuist weggelaten instruktie-bytes neergezet en de vergeten stappen, waarna weer terug wordt gegaan naar het programma. Dit systeem werkt goed, maar een mooie oplossing is het bepaald niet. Lezers die bijvoorbeeld het ruimtegevecht-programma op de ESS 006-plaat hebben "ontcijferd" weten hoe het eindresultaat er uit ziet: een grote wirwar.

De enige oplossing die dan over blijft is het opschuiven van de rest van het programma. Helaas betekent dit dat alle volgende programmastappen (vaak bladzijden lang) met de hand moeten worden ingetikt op de nieuwe adressen. Dit vervelende en uiterst minutieuze karweitje kan men ook laten uitvoeren door de computer, door gebruik te maken van een zogenaamde "block transfer" routine.

Het principe is vrij eenvoudig. Stel dat er een extra "store absolute" instruktie (3 bytes) moet worden toegevoegd op adres ØA00. De rest van het programma, laten we zeggen van ØA00 tot ØAFE, moet dan drie plaatsen opschuiven in het geheugen. Dit gaat als volgt:

```
Ø8CØ      Ø5FF      LODI,R1
Ø8C2  ┌─→ ØD4AØØ    LODA,I-R1
Ø8C5  │   CD6AØ3    STRA, I/R1
Ø8C8  └── 5978      BRNR,R1
Ø8CA      1FØØØØ    BCTA, UN
```

Door het starten van dit programma op adres Ø8CØ worden alle instruktie-bytes drie adresplaatsen opgeschoven: een voor een, van boven af. In de praktijk doet de computer dit zo snel dat het beeld op het scherm nauwelijks flikkert. De enige veranderingen die daarna nog met de hand moeten worden ingegeven zijn de plaatsen die absolute adres-instrukties bevatten die verwijzen naar het programmadeel dat is opgeschoven en elk relatief adres dat "over de toevoeging heen springt".

## Nieuwe programma's

In de software-service is nu een cassette opgenomen met speelcomputer-programma's. Deze cassette, nummer 007 in de ESS, bevat 15 programma's waarvan enkele reeds verschenen zijn op de ESS-platen 003 en 006. Hiervoor was namelijk nog genoeg plaats op de cassette en in het algemeen geeft een cassette minder problemen bij het inlezen van programma's dan een plaat. Op de ESS 007-cassette staan de volgende programma's:

| File | Titel | auteur |
|---|---|---|
| 1 | Mastermind | H. Tröndle |
| 2 | Codebreaker | K. Reilly/P. Williams |
| 3 | Reversi | H. Tröndle |
| 4 | Amazone | P. Holmes |
| 5 | Space shoot-out | P. Holmes |
| 6 | 4-in-a-row, small | |
| 7 | 4-in-a-row, large | H. Tröndle |
| 8 | Jackpot | P. Holmes |
| 9 | Surround | |
| A | Shapes | P. Williams |
| B | Piano | |
| C | PVI programming | P. Holmes |
| D | Disassembler | M. Saliger/P. Holmes |
| E | Test patterns | P. Holmes |
| F | Lotto numbers | M. Saliger |

## En in de toekomst . . .

Er zijn nog stapels ideeën, maar de uitwerking kost veel tijd. Er zijn toch zeker wel lezers met goede ideeën? We zijn graag bereid te helpen bij problemen die zich bij de realisering voordoen. Programma's zijn altijd welkom, hoe meer hoe beter.

Wat de hardware betreft hebben we ook nog plannen. Een geheugenuitbreiding is tot nu toe niet nodig geweest, maar we hebben zo'n schakeling klaar en getest. Als er voldoende belangstelling is kunnen we een print ontwerpen. Ook een getallengenerator is eenvoudig te maken, met enkele IC's is dat zo opgebouwd. In principe is het zo dat u het maar hoeft te zeggen en wij maken het. Alleen moet hier een grote "maar" achter. We kunnen namelijk geen kostbare Elektuurpagina's gebruiken voor schakelingen die maar voor een of twee lezers interessant zijn. Daarom stellen we reakties van geïnteresseerde lezers zeer op prijs: Als er genoeg belangstelling is voor een uitbreiding hebben we een goede reden ermee door te gaan. De keus is aan u!

# een kort programma

Van de heer Saliger ontvingen wij een kort programma voor het automatisch doorlopen van bestaande software. Na de nodige inkorting ontstond het hier afgebeelde resultaat. De gebruiksaanwijzing is eenvoudig. Het eerste adres van het programma(gedeelte) dat op het scherm moet verschijnen wordt in adres Ø8CØ gezet, waarna de routine kan worden gestart op adres Ø8C2. De adressen en instrukties zullen nu achtereenvolgens op het beeld verschijnen. Het beeld kan worden stilgezet door indrukken van de bovenste kommandotoets en weer gestart door middel van de start-toets. De snelheid van het schuiven wordt bepaald door de inhoud van adres 1F9C (Ø2, Ø4, Ø8 of 1Ø).
Denk er wel aan dat de eerste instrukties na een serie data-waarden door de computer verkeerd geïnterpreteerd kunnen worden. Bij het doorlopen van lange tabellen kan het handig zijn de instruktie op adres Ø8FØ te veranderen in Ø6Ø4, Ø6Ø8 of Ø6ØC.
Terugkeren naar het monitorprogramma geschiedt door indrukken van de reset-toets. De start-toets mag in deze stand *niet* ingedrukt worden, anders wordt het programma vanaf adres 1F8Ø gewist!

```
Ø8CØ    ØØØØ      2 bytes voor startadres
Ø8C2    7620      PPSU,II
Ø8C4    Ø502      LODI,R1
Ø8C6  ↱ØD48CØ     LODA,I-R1   ┐
Ø8C9  │ CD68A4    STRA,I/R1   ├ verplaats startadres
Ø8CC  └ 5978      BRNR,R1     ┘
Ø8CE    3FØ2CF    BSTA,UN     nieuwe regel
Ø8D1    3FØ42B    BSTA,UN
Ø8D4    Ø7Ø6      LODI,R3     ┐
Ø8D6  ↱CF488A     STRA,I-R3   ├ adres naar MLINE
Ø8D9  └ 5B7B      BRNR,R3     ┘ en nieuwe regel
Ø8DB    3BF2      BSTR,UN,Ind.(Ø2CF)
Ø8DD    ØE88A4    LODA,Ind,R2 ┐
Ø8EØ    F6Ø8      TMI,R2      │
Ø8E2  ┌ 18ØC      BCTR        │
Ø8E4  │ F6AØ      TMI,R2      │
Ø8E6  ┌ 18Ø6      BCTR        │
Ø8E8  │ 264Ø      EORI,R2     ├ 1-, 2- of 3-byte
Ø8EA  │ F654      TMI,R2      │ instruktie?
Ø8EC  ┌ 18Ø2      BCTR        │ Resultaat in R2
Ø8EE  ↳ 8604      ADDI,R2     │ als Ø1, Ø2 of Ø3
Ø8FØ  ↳ 46ØC      ANDI,R2     │
Ø8F2    52        RRR,R2      │
Ø8F3    52        RRR,R2      │
Ø8F4    1F1F8Ø    BSTA,UN     ┘
(Ø8F7)

1F8Ø  ↱ 87Ø4      ADDI,R3     ┐
1F82  │ ØD88A4    LODA,R1,Ind │
1F85  │ 3FØ354    BSTA,UN     ├ 1, 2 of 3 databytes
1F88  │ 3FØ39F    BSTA,UN     │ op het scherm;
1F8B  └ FA73      BDRR,R2     │ adres inkrementeren
1F8D    3FØ2ØE    BSTA,UN     ┘
1F9Ø  ↱ ØC1FCB    LODA,RØ     ┐
1F93  │ DØ        RRL,RØ      ├ wacht op VRLE;
1F94  └ 9A7A      BCFR        │ 6 tekstregels op het scherm
1F96    77Ø2      PPSL,COM    │
1F98    3FØØ55    BSTA,UN     ┘
1F9B    87Ø4      ADDI,R3     snelheid
1F9D  └ ØC1E8B    LODA,RØ     ┐
1FAØ  ┌ 9AØ2      BCFR        ├ stop als bovenste
1FA2  │ 67Ø1      IORI,R3     ┘ kommandotoets ingedrukt wordt
1FA4  ↳ DØ        RRL,RØ      ┐
1FA5  ┌ 9AØ2      BCFR        ├ start als start-toets
1FA7  │ 47FE      ANDI,R3     ┘ ingedrukt wordt
1FA9  ↳ 5B65      BRNR,R3     vertraging of stop
1FAB    1F08CE    BSTA,UN     herhaal
```

Lezers die dit programma willen gebruiken om de monitor-software te onderzoeken moeten er rekening mee houden dat deze data bevat op de volgende adressen:

ØØØ6 . . . ØØØ9
ØØAD . . ØØBC
Ø122 . . . Ø13D
Ø177 . . . Ø18Ø
Ø27B . . . Ø2CE
Ø2F5 . . . Ø31C
Ø537 . . . Ø53F

Het RAM "kladblok" begint op adres ØØØØ. De startadressen voor de hoofdroutines zijn:

- Initiate : ØØ23
- "Reg" : Ø3BØ
- "Mem" : Ø4ØC
- "BK" : Ø4A9 en Ø594/Ø5B2
- "PC" : Ø5ØE
- "Wcas" : Ø5E8
- "Rcas" ; Ø758

# het lek van elektuur

### Lichtautomaat met dimmer

In de opdruk van de print en op de foto's (okt. 1979, blz. 10-48) staat de triac verkeerd; de metalen zijde dient naar P1 toe te wijzen. Ondanks deze fout bleek de schakeling een test te doorstaan. Op de lange duur heeft de verkeerde aansluiting toch fatale gevolgen voor de triac.

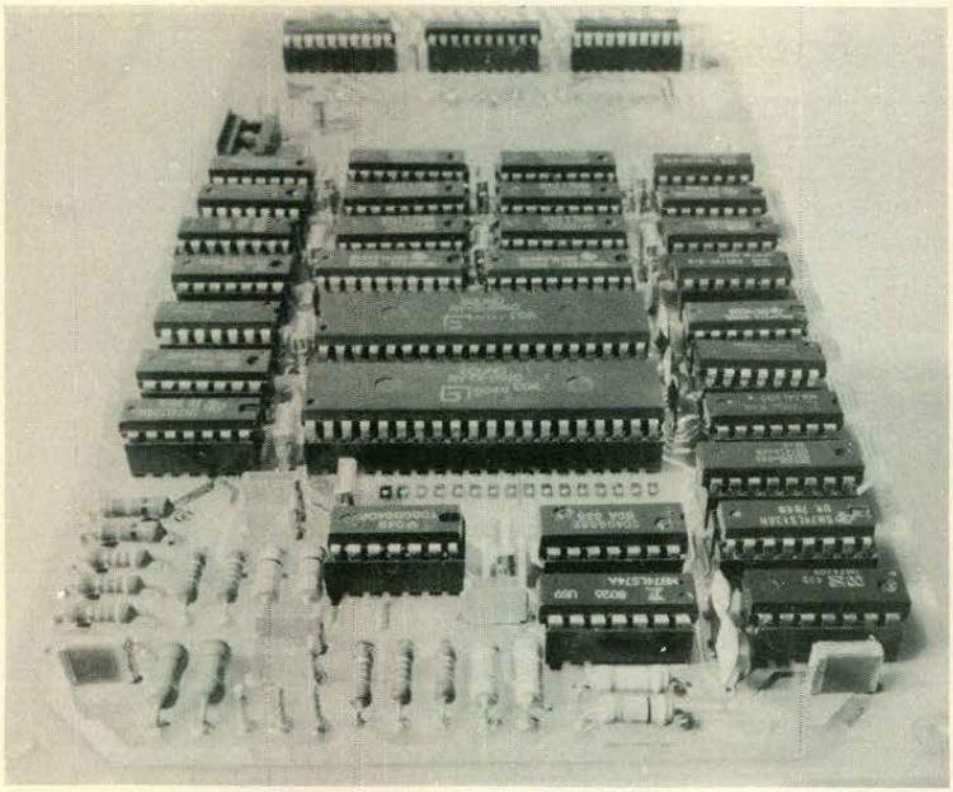
### Digitale hartslagmonitor

In het schema van de hartslagmonitor (juli/aug. '80, blz. 800) zijn de aansluitpennen 3 en 10 van IC7 met elkaar verbonden. Dat is onjuist; de verbinding dient te liggen tussen de pennen 6 en 10. Gelukkig is deze fout niet op de print aanwezig.

# mikromarkt

* De L290 (f/U-konverter), de L291 (D/A-konverter) en de L292 (switch mode driver) van SGS-ATES vormen gezamenlijk een kompleet positioneringssysteem voor DC-motoren. De kommando's worden verkregen uit een microprocessor (bijv. Z80).
* De digitalker van National Semiconductor bestaat uit een spraakprocessor-chip en een spraak-ROM, en kan na aansluiting van een eenvoudig filter via een versterker en een speaker al een stemgeluid van een uitstekende kwaliteit produceren. Het programmeren van de tekst neemt de fabrikant voor zijn rekening.
* De LM1851 is een funktioneel IC van National Semiconductor, waarmee met slechts enkele extra komponenten een aardlekschakelaar gevormd kan worden.
* Het timer-IC WD-55 van Western Digital Corporation vormt in kombinatie met een keyboard en een display een dokatimer die via opto-couplers en triacs de verlichting in de vergroter en de dokaverlichting kan schakelen.
* National Semiconductor introduceert de INS8073: een revolutionaire 8-bits single-chip microprocessor die BASIC verstaat (2,5 K NSC Tiny BASIC aan boord).
* De MSM5832 van OKI is een real time klok/kalender-IC voor gebruik in bus-georiënteerde microprocessorsystemen en levert 4 bits data over sekonden, minuten, uren, dag van de week, datum, maand en jaar. Data access vindt plaats via een 4 bits adres.

mikromarkt



# uitbreiding speelcomputer

## 3K geheugenruimte erbij en geluidseffekten!

**De basisuitvoering van de speelcomputer heeft bijna 2K programmageheugen. Dat bleek voldoende voor flink wat interessante spelletjes, zonder dat de programma's kompleet uit de hand gingen lopen — zulks met het oog op de Wet van Parkinson: "Computer programma's groeien totdat alle beschikbare geheugenruimte is gevuld". Ondertussen zal echter het fanatiekste deel van de speelcomputer-fan's al lang het punt hebben bereikt waarop een wat groter geheugenbereik niet onwelkom is. Kijken we bijvoorbeeld naar de meeste van de 30(!) nieuwe spelletjes die binnenkort via de ESS op band beschikbaar worden. We vonden derhalve de tijd rijp worden voor een uitbreidingsprint. Met deze print wordt de geheugenruimte zo goed als verdrievoudigd, terwijl er bovendien twee programmeerbare geluidseffektgenerators op konden worden ondergebracht.**
**We beperken ons in dit artikel tot een beknopte beschrijving van "hoe het werkt" en "hoe je het bouwt". Meer bijzonderheden, alsmede een heleboel interessante informatie over de speelcomputer, kan men vinden in een boek dat wij zeer binnenkort over dit onderwerp zullen uitgeven.**
**De uitbreidingsprint bezit nog een interessant pluspuntje: er is namelijk een mogelijkheid geschapen tot het inpluggen van cassettes die bedoeld zijn voor kommerciële apparaten.**

Alvorens we de uitbreidingsschakeling onder de loep gaan nemen, eerst een paar dringende verzoeken aan alle ijverige programmeurs. In de eerste plaats dient men in gedachten te houden dat andere speelcomputer-bezitters wellicht (nog) niet over extra geheugenruimte beschikken. Organiseer het spel, indien mogelijk, daarom zó dat een basisversie ervan ook op een basisuitvoering van de computer kan draaien en een uitgebreide versie alleen op een computer met extra geheugen. De spelen "doolhof" en "memory" van de nieuwe ESS-cassette zijn hier voorbeelden van. In beide gevallen gaat een sprong naar "initialiseer voor extra geheugen" als volgt in zijn werk:

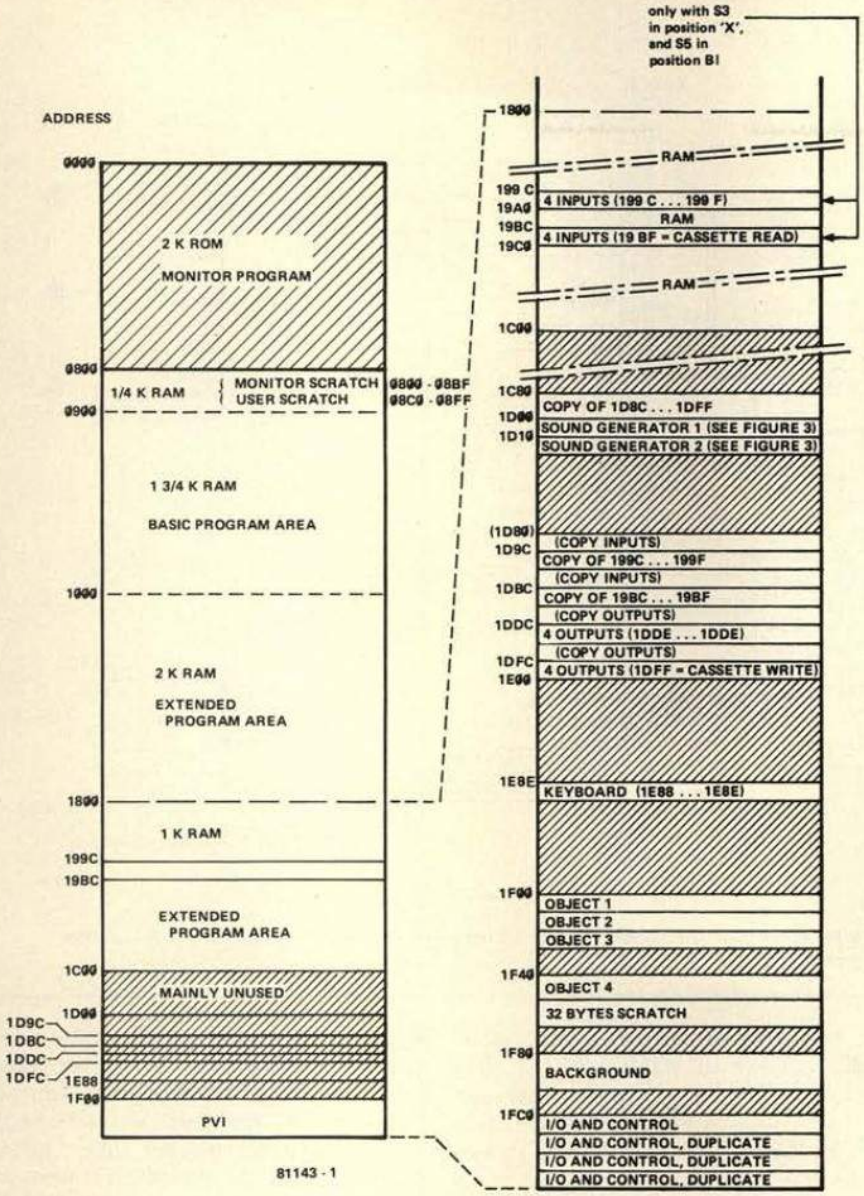
```
ØC1ØØØ LODA, RØ
       BAFC BSFR, N, 1ØØØ
```

Als geen extra geheugen aanwezig is, zal de data op 1ØØØ gelijk zijn aan FF (negatief) en zal er niets gebeuren; in het andere geval zullen via een subroutine op 1ØØØ enkele instrukties in het basisprogramma worden aangepast om de programma-uitbreidingen te omvatten. Bezitters van een basisversie zullen slechts merken dat de load-routine stopt bij "AD = 1ØØØ"; zij kunnen het programma echter starten bij PC = ØØØØ. Een ander verzoek aan de H.H. programmeurs betreft de joysticks. Hou asjeblieft goed in de gaten dat er geen twee joysticks gelijk zijn! Kalibratie is en blijft noodzakelijk.

### De uitgebreide versie

De nieuwe extra's kunnen het beste worden geïllustreerd aan de hand van een geheugen-"plattegrond" als getekend in figuur 1. In de basisuitvoering neemt de monitor de eerste 2K geheugen in beslag, gevolgd door 2K RAM (1¾K voor de gebruiker). Het adresbereik vanaf 1ØØØ werd in feite niet gebruikt — het bevatte slechts een paar input/output-lijnen en de PVI. Nu wordt er nog eens 3K RAM toegevoegd, van 1ØØØ tot 1BFF, terwijl er twee programmeerbare geluidseffektgenerators (programmable sound generators — PSG's) komen op de adressen 1DØØ tot 1D1F.
Het gebied vanaf 18ØØ ziet er tamelijk verward uit, gedeeltelijk als gevolg van een klein ongerechtigheidje in de oorspronkelijke monitor ROM. Het adres "read cassette" bevindt zich namelijk op 19BF, in plaats van op 1DBF waar het eigenlijk hoort. Aangezien de adressen in dit bereik niet volledig gedekodeerd waren, maakte dat toen geen verschil. Nu wordt de zaak er echter wat gekom-

1



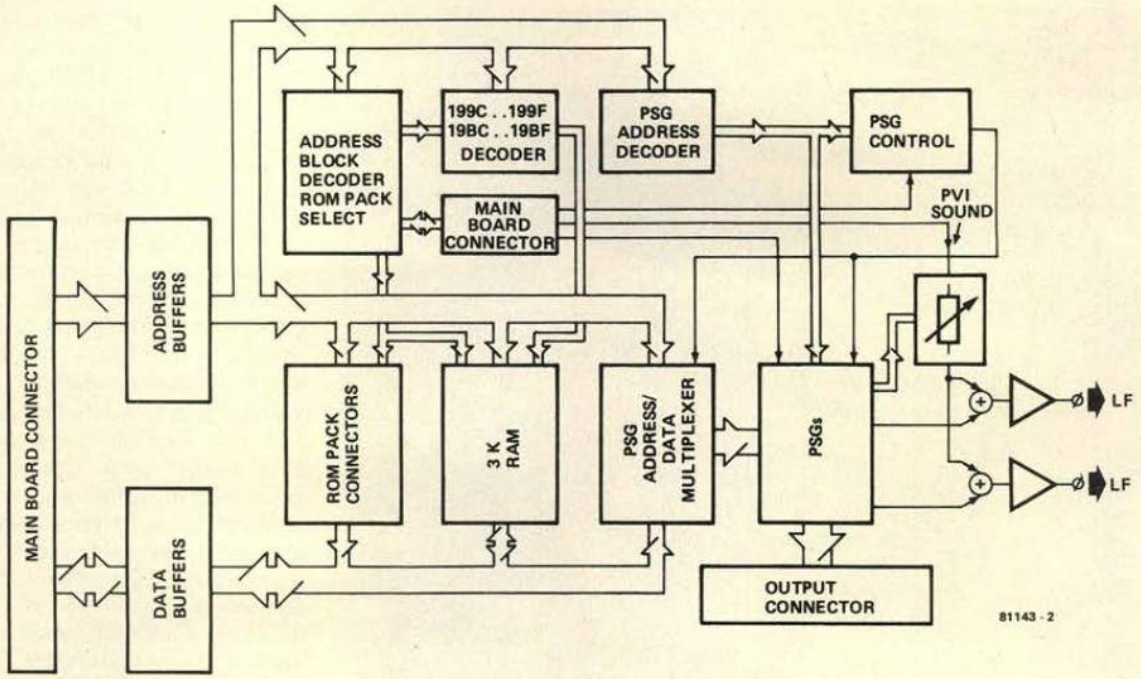

**Figuur 1. Geheugenkaart van de uitgebreide speelcomputer. Het laatste gedeelte, van 1800 tot 1FFF is rechts vergroot weergegeven.**

pliceerder door. Er moet een mogelijkheid worden gekreëerd om de RAM even buitenspel te zetten op 199C . . . . 199F en 19BC . . . 19BF. Dat gebeurt door S3 in stand "X" en S5 in stand "B" te zetten. In het rechter gedeelte van figuur 1 zien we een wat gedetailleerdere weergave van het "moeilijke" stuk uit de geheugenplattegrond. Eerst de RAM met de twee bovengenoemde "input"-bereiken; dan even niets; dan de geluidsgenerators op 1D00, gevolgd door enkele input- en output-bereiken (inklusief het inlezen van en uitschrijven op cassette). Op 1E8E zien we het keyboard; en tenslotte vanaf 1F00 de PVI. Eenieder is uiteraard vrij om eigen ideeën toe te voegen. Een hardware toevalsgenerator op 1D20? Een explosiegeluidsgenerator op 1E80? U zegt het maar. Het kan allemaal, onder één voorwaarde: ze moeten niet worden gebruikt in programma's waarvan men wil dat wij ze in de ESS-service opnemen!

## De uitbreidingsprint

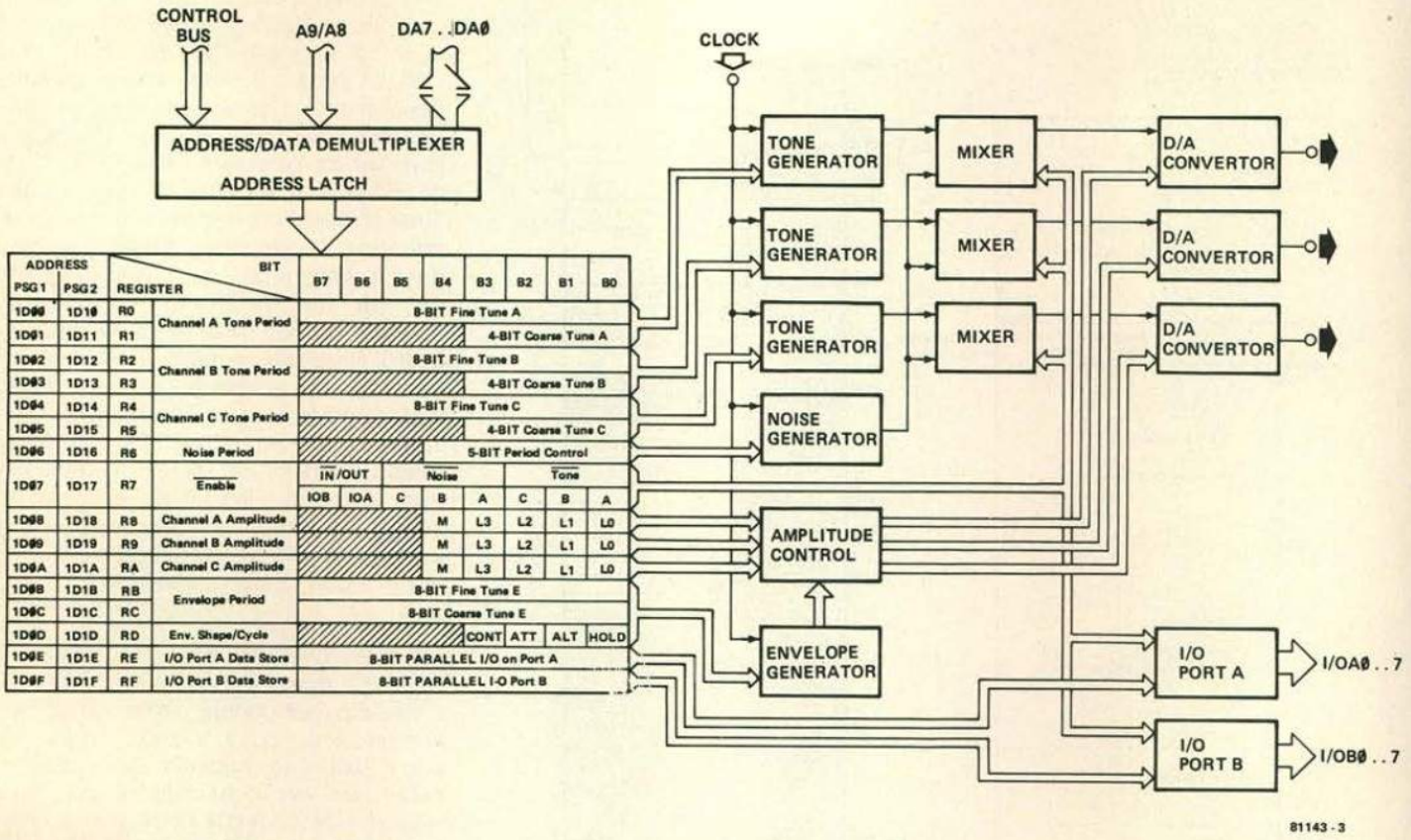
Figuur 2 toont een blokschema van de uitbreidingsschakeling. Niet al te gekompliceerd, zoals u ziet. Gezien de extra belasting, moeten de adres- en data-lijnen van de hoofdprint gebufferd worden. De onderste helft van de tekening geeft de feitelijke uitbreidingen weer: ROM "pack connectors" voor kommerciële speelcassettes; RAM-uitbreiding; programmeerbare geluidsgenerators (PSG's) met bijbehorende audio-uitgangen. Daarboven bevinden zich de stuurschakelingen. de "address block decoder" met schakelaars om kommerciële ROM-speelcassettes te kiezen; de dekoder voor adressen welke korresponderen met de (foutieve) ingangsblokken; de PSG adresdekoder en stuurschakeling. In het midden van figuur 2 is nog een konnektor naar de

2




**Figuur 2. Blokschema van de uitbreidingsprint.**

3




| ADDRESS PSG1 | ADDRESS PSG2 | REGISTER | BIT B7 B6 B5 B4 B3 B2 B1 B0 |
|---|---|---|---|
| 1D00 | 1D10 | R0 | Channel A Tone Period — 8-BIT Fine Tune A |
| 1D01 | 1D11 | R1 | Channel A Tone Period — 4-BIT Coarse Tune A |
| 1D02 | 1D12 | R2 | Channel B Tone Period — 8-BIT Fine Tune B |
| 1D03 | 1D13 | R3 | Channel B Tone Period — 4-BIT Coarse Tune B |
| 1D04 | 1D14 | R4 | Channel C Tone Period — 8-BIT Fine Tune C |
| 1D05 | 1D15 | R5 | Channel C Tone Period — 4-BIT Coarse Tune C |
| 1D06 | 1D16 | R6 | Noise Period — 5-BIT Period Control |
| 1D07 | 1D17 | R7 | Enable — IN/OUT: IOB IOA; Noise: C B A; Tone: C B A |
| 1D08 | 1D18 | R8 | Channel A Amplitude — M L3 L2 L1 L0 |
| 1D09 | 1D19 | R9 | Channel B Amplitude — M L3 L2 L1 L0 |
| 1D0A | 1D1A | RA | Channel C Amplitude — M L3 L2 L1 L0 |
| 1D0B | 1D1B | RB | Envelope Period — 8-BIT Fine Tune E |
| 1D0C | 1D1C | RC | Envelope Period — 8-BIT Coarse Tune E |
| 1D0D | 1D1D | RD | Env. Shape/Cycle — CONT ATT ALT HOLD |
| 1D0E | 1D1E | RE | I/O Port A Data Store — 8-BIT PARALLEL I/O on Port A |
| 1D0F | 1D1F | RF | I/O Port B Data Store — 8-BIT PARALLEL I-O Port B |

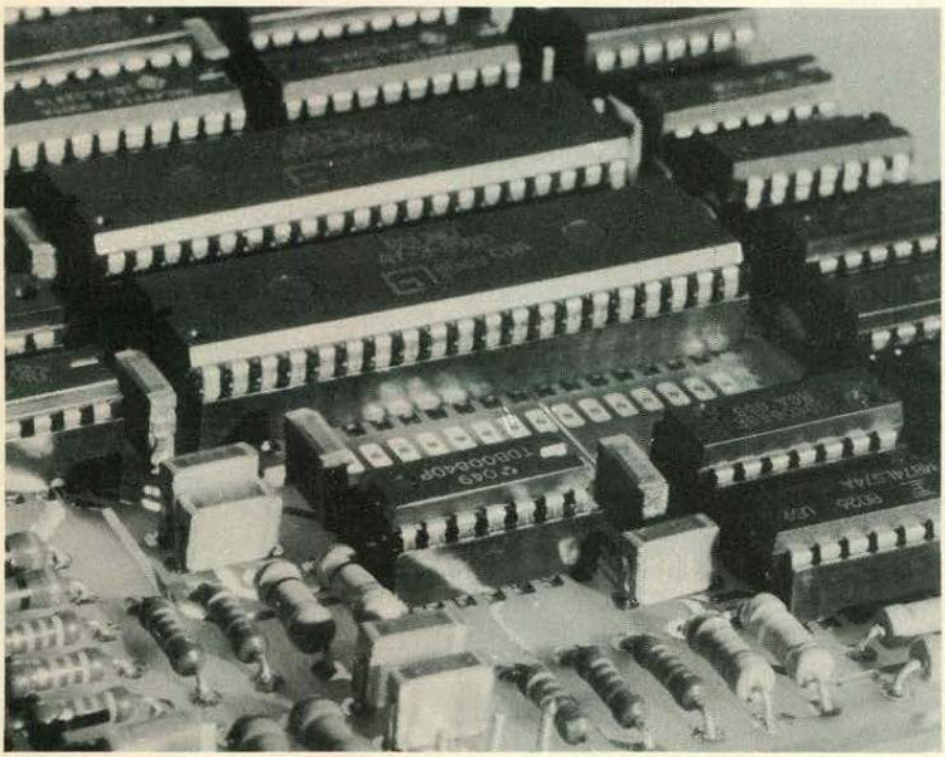
**Figuur 3. Blokschema van het interieur van een "programmable sound generator"-IC. De 16 registers sturen drie toongenerators, een ruisgenerator, een omhullende generator, mixers en uitgangspoorten.**

hoofdprint te zien, welke enkele "rare" signalen transporteert als "clock" en "PVI audio output".

## De PSG's

De programmeerbare geluidsgenerators vormen het meest opvallende deel van de uitbreidingsprint. Die IC's zitten aardig gekompliceerd in elkaar, zoals figuur 3 laat zien. Elk IC bevat 16 registers, korresponderend met de adressen 1D00 . . . 1D0F of 1D10 . . . 1D1F. Die registers worden geadresseerd door een in het IC aanwezige address/data demultiplexer — niet zo gunstig in ons geval, omdat het betekent dat we, zoals in figuur 2 te zien, een externe address/data multiplexer moesten toevoegen!

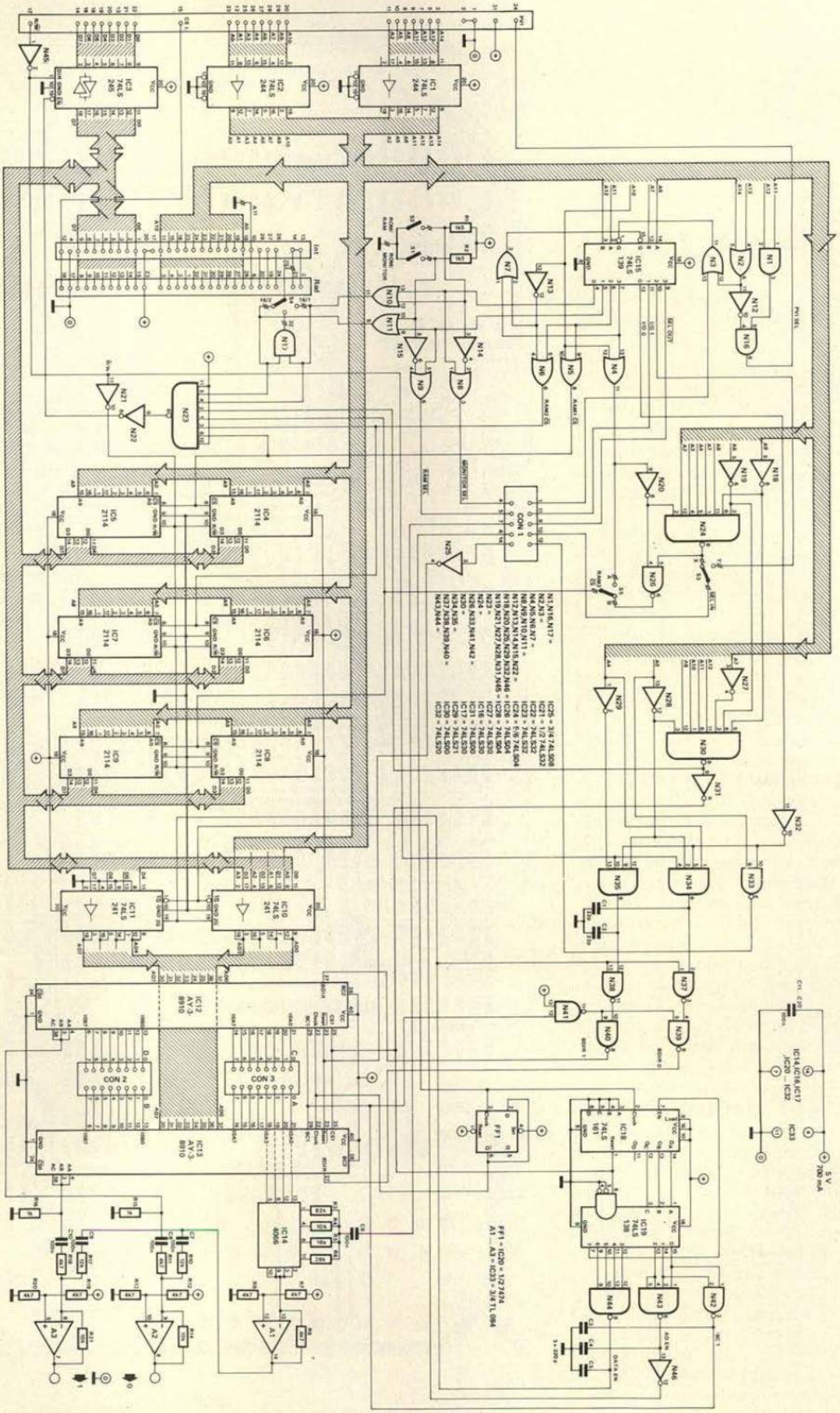
De in de registers opgeslagen data stuurt drie toongenerators, een ruisgenerator alsmede enkele mixers; een omhullende (envelope) generator met bijbehorende D/A-converters; en twee input/output-poorten. Om met het laatste te beginnen: normaliter is elke input/output-poort 8 bits breed, en kan naar wens wel of niet worden toegepast — onafhankelijk van de rest van de PSG — mits geaktiveerd via het korresponderende bit in R7. In de hier beschreven toepassing kunnen ze echter uitsluitend als uitgang worden gebruikt. Daarnaast worden de vier minst signifikante bits van poort A in PSG1 (adres 1D0E) gebruikt voor amplituderegeling van de PVI-geluidsuitgang.



De grondfrekwentie van elke toongenerator kan door het hele audiobereik worden gevarieerd — het TV-speelcomputer-boek bevat een tabel voor het kiezen van een willekeurige noot in een acht-oktaafs-bereik. Ook wat de "grondfrekwentie" van de ruisgenerator betreft is er keuze mogelijk. De gewenste uitgangen kunnen afzonderlijk worden geaktiveerd. De uitgangsamplitude kan voor elk kanaal worden ingesteld, of geregeld door de omhullende generator; in beide gevallen wordt het resultaat bepaald door een 4-bit D/A-converter, overeenkomend met 16 amplitude-nivo's. De omhullende generator kan worden ingesteld op "single-shot"-effekten, zoals explosies, of op periodieke amplitude-effekten zoals tremolo of machinegeweer-salvo's.

Hou er rekening mee dat het de-aktiveren van een toon- of ruisgenerator met

4



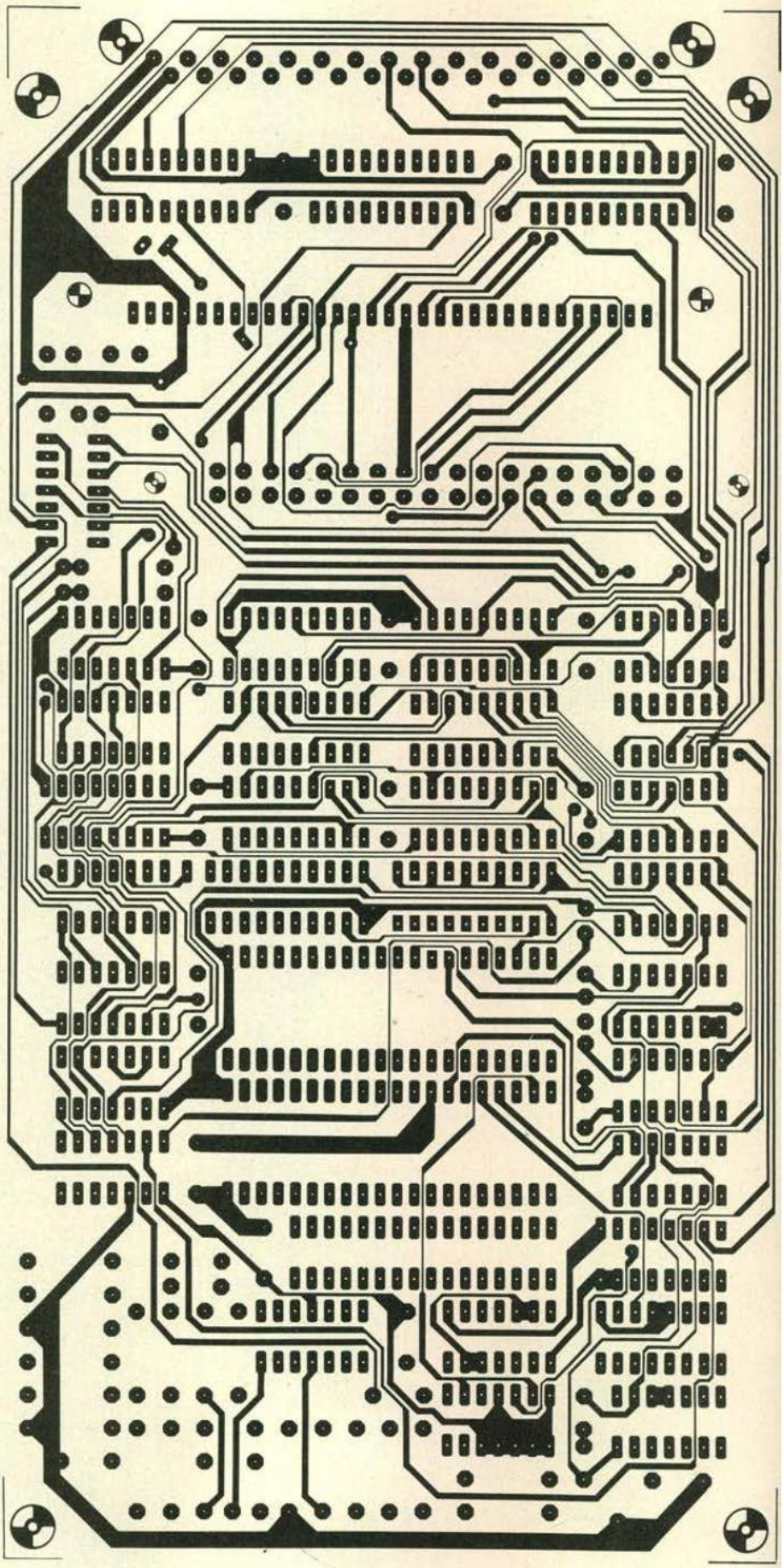
**Figuur 4. Kompleet schema van de uitbreidingsprint. De uitbreiding is op twee manieren verbonden met de hoofdprint: via de 31-pens konnektor links, en een 14-pens konnektor in het midden van de tekening.**

behulp van R7 (adres 1DØ7 of 1D17) niet voldoende is om hem volledig "dood" te maken. Absolute stilte wordt uitsluitend verkregen, als het amplitudenivo van alle uitgangen op nul wordt gezet door ØØ in de registers R8 . . . RA op te slaan. Dit ontdekten wij ook pas na de nodige frustratie . . .
Al met al is er nu een indrukwekkende serie geluidseffekten beschikbaar. De oorspronkelijke fabrieksapplikatie geeft een "explosie", "geweerschot", "Europese sirene", "laser geluidseffekten", "fluitende bom", "wolvengehuil" en "race-auto". In minder dan geen tijd hadden wij daar al een rammelende ketting en verschillende polyfone trouwmarsen aan toegevoegd. Ontzettend geinig!

## De bouw

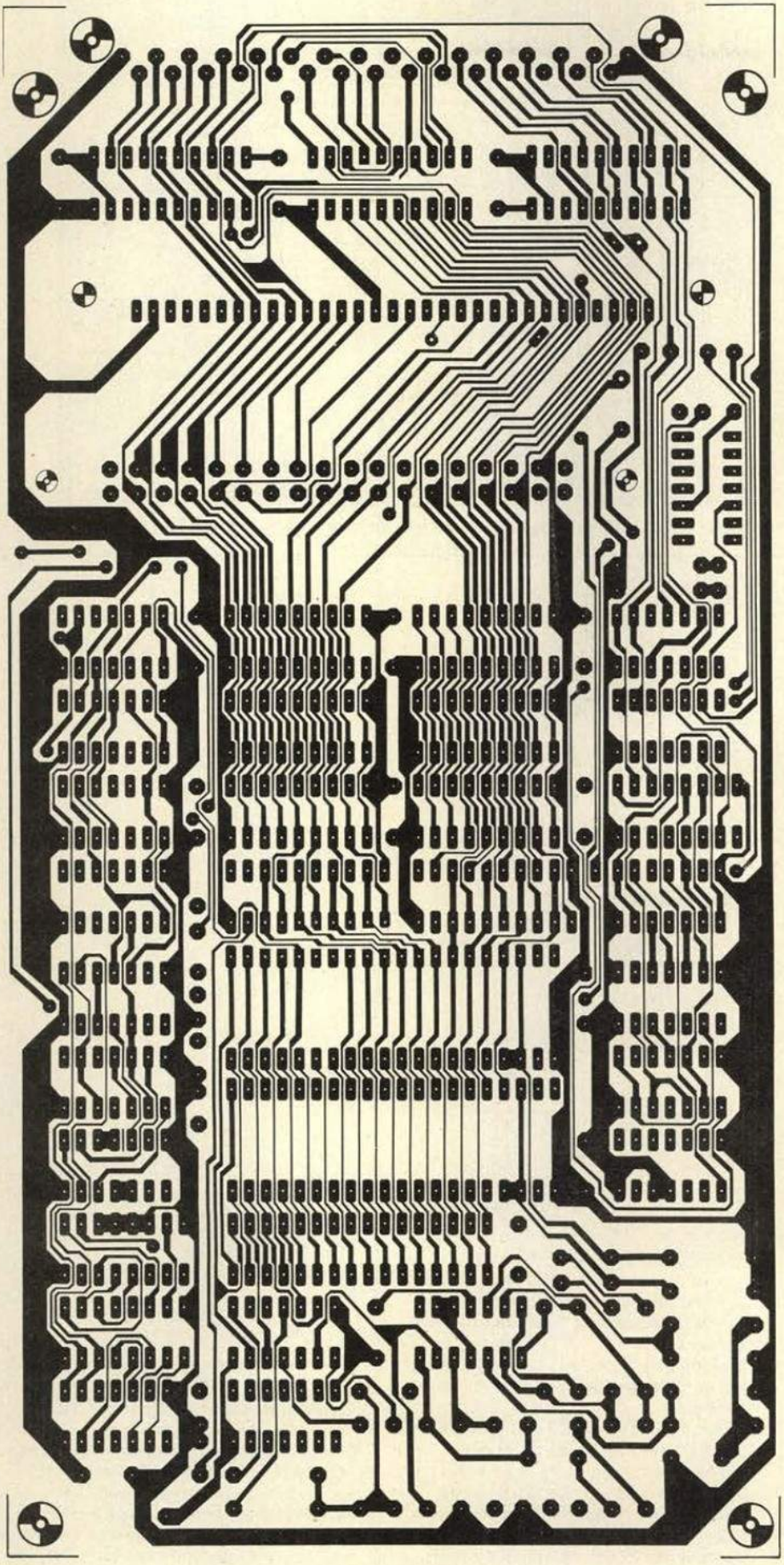
Figuur 4 geeft het komplete schema. We gaan hier verder niet op dit schema in, maar volstaan met te zeggen dat het exakt overeenkomt met het blokschema van figuur 2. Verdere bijzonderheden zijn te vinden in het speelcomputerboek.
In figuur 5 is de print afgebeeld – wegens ruimtegebrek verkleind tot 70% van de ware grootte – en in figuur 6 is het bedradingsschema weergegeven. Te zien is dat er twee verbindingspunten zijn naar de hoofdprint: de 32-polige konnektor en een 14-polige DIL-konnektor die met een 16-pens (!) versie op de hoofdprint wordt verbonden. Laatstgenoemde konnektor wordt gemonteerd op de voor IC6 bedoelde ruimte; dat bewuste IC, een 74LS139, is nu ondergebracht op de uitbreidingsprint en heet daar IC15.
Er zijn twee nieuwe audio-uitgangen, namelijk één van elke PSG; het PVI-geluid wordt bij beide uitgangen bijgemengd. Ofschoon dit de mogelijkheid biedt tot het kreëren van "stereo"-geluidseffekten, vonden wij het eenvoudiger om het geheel te kombineren tot een enkele audio-uitgang, met behulp van een stel mengweerstanden van 4k7. Gek eigenlijk – we bieden een mogelijkheid en gebruiken die dan zelf niet. Maar ja, niet iedereen zal er hetzelfde over denken als wij . . .
Tenslotte nog iets over de voeding. De oorspronkelijke uitvoering zal helaas soms een weinig "under-powered" blijken. De uitbreidingsprint wordt aardig warm en dat kost nu eenmaal stroom. De voeding moet in staat zijn om ongeveer 2 A te leveren zonder het benauwd te krijgen. Als u dezelfde fout hebt gemaakt als wij – namelijk de oorspronkelijke voeding iets te krap hebt bemeten – dan zijn enkele modifikaties gewenst. De vermogenstransistor moet van een voldoende groot koellichaam worden voorzien, liefst buiten de kast gemonteerd. Het kan nodig zijn om de bruggelijkrichter eveneens met een koellichaam uit te rusten. De beste manier om vast te stellen of een onderdeel te heet wordt, is om het met een natte vinger aan te raken: als het sist, is het te heet!

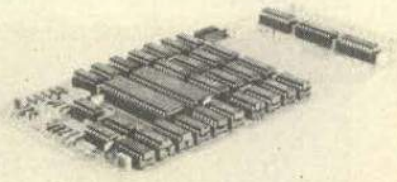
5a



Figuur 5a+b. Koper-layouts van de dubbelzijdige uitbreidingsprint (5a: komponentenzijde, 5b: soldeerzijde).

5b





**Onderdelenlijst**

Weerstanden:
R1,R2 = 1k5
R3 = 82 k
R4,R14,R21 = 10 k
R5 = 18 k
R6 = 39 k
R7 . . . R9, R11 . . . R13, R18 . . . R20 = 4k7
R10,R17 = 12 k
R15,R16 = 1 k

Kondensators:
C1,C2 = 22 p
C3 . . . C5 = 220 p
C6 . . . C26 = 100 n

Halfgeleiders:
IC1,IC2 = 74LS244
IC3 = 74LS245
IC4 . . . IC9 = 2114
IC10,IC11 = 74LS241
IC12,IC13 = AY-3-8910
IC14 = 4066
IC15 = 74LS139 (was IC6)
IC16,IC17 = 74LS30
IC18 = 74LS161
IC19 = 74LS138
IC20 = 74LS74
IC21 . . . IC23 = 74LS32
IC24,IC26,IC28 = 74LS04
IC25 = 74LS08
IC27 = 74LS30
IC29 = 74LS21
IC30,IC31 = 74LS00
IC32 = 74LS20
IC33 = TL 084

Schakelaars:
S1,S2,S5 = enkelpolig om
S3 = enkelpolig om of draadbrug
S4 = enkelpolig drie-standen