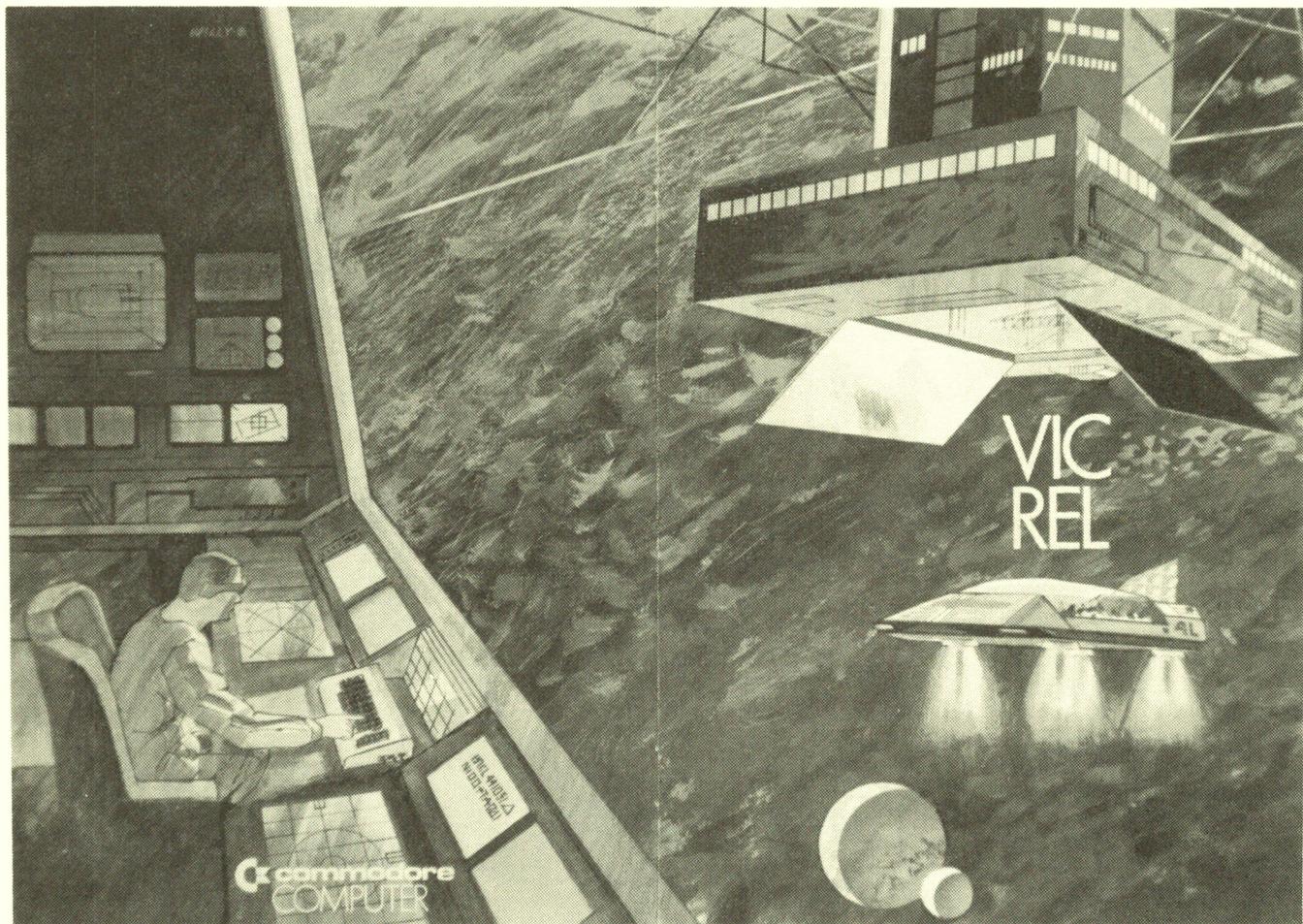


VIC-PRIMEURS

JAARGANG 2

No. 3



- * **Commodore VIC-1020 uitbreidingsbord**
- * **Nieuwe ROM-spelen**
- * **Prijsvraag**
- * **Introduktie machinetaal**
- * **Tips en programma's**

INHOUDSOPGAVE :

	blz.
AAN DE LEZER.....	3
NIEUW VIC REL.....	5
SPELEN OP DE VIC-20, EN HOE.....	7
De favoriete TIEN.....	12
HOGE-RESOLUTIE OP DE VIC-20 deel 3.....	13
VIC TIP VIC-1515 PRINTER TIP.....	16
ASSEMBLER vervolg.....	16
VIC TIP SPECIALE PRINTER TEKENS.....	19
Inleiding machinetaal.....	19
DIVERSE.....	29
DE HOOGSTE SCORE.....	30
BASICODE WIJZIGING.....	31
PROGRAMMALISINGS.....	31
FLOPPY DISK HULPPROGRAMMA.....	36
NIEUW Het VIC-1020 moederbord.....	37
PROGRAMMA PRIJSVRAAG.....	40
VIC-1211A SUPER EXPANDER.....	43
De VIC gebruikersclubs.....	47
VIC TIP VIC MERGE.....	48
VICKIES.....	49
PRIJSLIJST.....	50
DEALERLIJST.....	51

DE VOLKSWAGEN WERD OOK EERST UITGELACHEN



The Commodore VIC-20 is a home computer released in 1980. It was one of the first successful 8-bit home computers, featuring a built-in VIC chip for its video output. The illustration shows the VIC-20's distinctive keyboard unit and a separate television set displaying a small screen with a grid pattern, which was typical for early computer displays.

commodore
COMPUTER

En de VolksComputer? Dat zal de VIC-20 Volks-
Computer niet overkomen! De VIC-20 is dè eerste volwaar-
dige micro waarmee je het 'computeren'

Aan de lezer

Noch het woordenboek, noch de encyclopedie kennen de term **videospel**, en dat terwijl het het snelst groeiende produkt ter wereld is. In deze vierde editie gaan wij dan ook uitgebreid in op de nu leverbare spellen voor de VIC-20 in nederland. De hoogste scores op de diverse spellen gemeten door een amerikaans tijdschrift nemen we ook op, dan weet je gelijk wat je te doen staat.

Naarmate de VIC-20 meer gebruikers gaat tellen, merk je dat de interesse naar het programmeren in machinetaal steeds belangrijker gaat worden. Een uitgebreid engels artikel uit de reference guide voor het gebruik van machinetaal is in deze uitgave opgenomen en wij denken een welkome aanvulling op **VIC-revealed**.

In een vorige editie hebben we al het een en ander besproken met betrekking tot uitbreiding van geheugen d.m.v. ram/rompacks en het ARFON moederbord. Vanaf 1 juli heeft ook COMMODORE zijn eigen moederbord en een recensie vindt u op een van de volgende pagina's.

Voor alle nieuwkomers nog even dit.

VIC-primeurs is een tweemaandelijks tijdschrift en wordt speciaal samengesteld voor de eigenaren van de VIC-20 computer. Het heeft ook tot doel de VIC gebruikers clubs, die in het land worden opgezet, te ondersteunen.

In **VIC-primeurs** worden allerlei zaken over de VIC-20 gepubliceerd, in de vorm van artikelen door deskundigen, informatie van en over dealers en natuurlijk niet in de laatste plaats de bijdragen van de lezers zelf.

Het adres voor inzendingen is :
Aan de VIC-gebruikersclub
Postbus 12972
1100 AZ AMSTERDAM

VIC-primeurs geeft ook informatie uit het buitenland. Deze informatie wordt vergaard uit alle inzendingen, die wij van alle COMMODORE bedrijven (ruim twee honderd over de hele wereld), ontvangen. Tevens zullen de lezers exclusieve aanbiedingen van nieuws, literatuur en software voor de VIC-20 uit andere landen krijgen.

In nederland zijn al veel dealers en VIC gebruikers bezig om hard- en software voor de VIC te ontwikkelen. In een aantal rubrieken zal hier uitgebreid aandacht aan worden besteed. De stroom van informatie uit de afgelopen weken is in deze editie verwerkt en daarvoor onze hartelijke dank. Misschien denk je dat jouw ontwikkelingen of programma's niet de moeite waard zijn voor de PRIMEURS. Stuur ze toch in. Want veel lezers zijn beginners en van elkaar kunnen we veel leren, maar vooral goede ideeën opdoen. Dus graag, alle inzendingen zijn welkom.

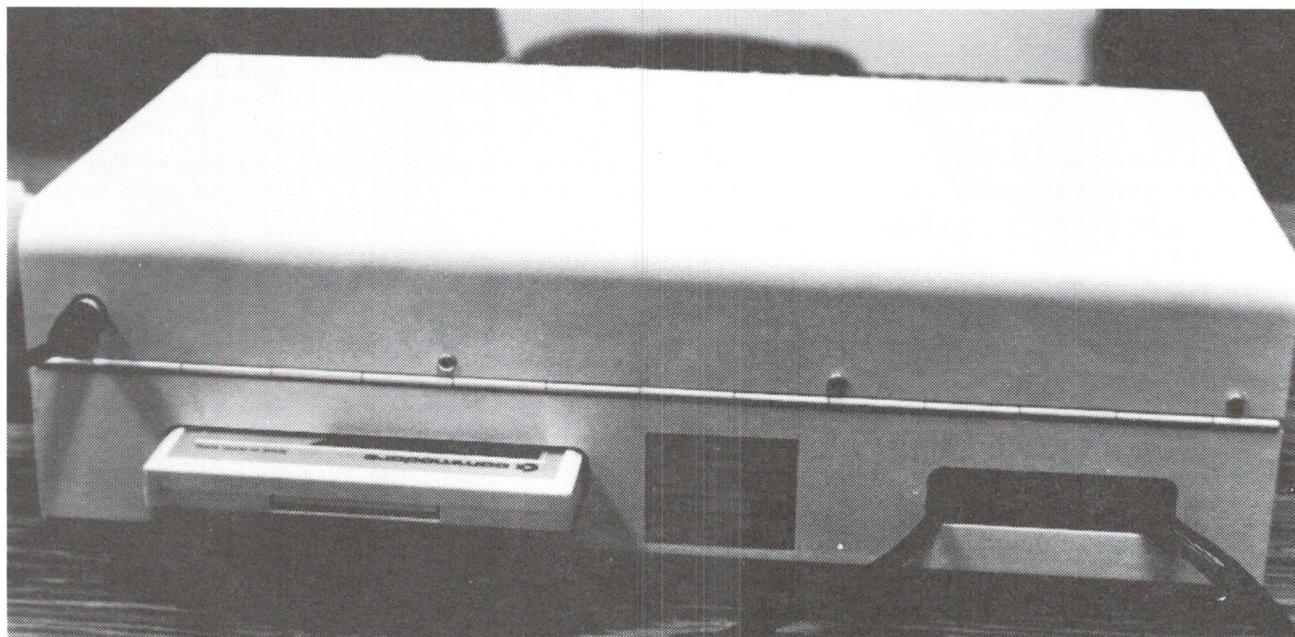
Wederom zouden we VIC eigenaren willen aansporen om lid te worden van de VIC-gebruikersclub. VIC-primeurs , het tijdschrift van de club , is echt onmisbaar als u alles uit de VIC-20 wilt halen , wat eruit te halen valt.Bij HANDIC ontvangt men veel vragen , die reeds in de PRIMEURS zijn beantwoord. Hierdoor ontdekken ook veel nieuwe beginners VIC-primeurs. Wilt u dus meepraten met de andere 2000 leden wordt dan lid en ontvang elke twee maanden VIC-primeurs door overmaking van Fl. 50,- op Postgiro 2930100 , t.n.v. HANDIC BENELUX BV.

Postbus 213
1850 AE HEILOO

onder vermelding van VICCLUB lidmaatschap en uw naam , adres , postcode en woonplaats .

Veel plezier met deze primeurs en tot volgende keer .

HOI .



Nieuw

VIC REL

Een nieuw stukje hardware voor de VIC-20 is de VIC REL welke het mogelijk maakt om op een veilige manier met de IN/UIT-gangen aan de USER-poort te kunnen experimenteren. Met deze cassette kunt U op een eenvoudige wijze bijvoorbeeld de volgende systemen besturen: inbraak alarm, garage deuren, deursloten, verwarmingselementen, lampen, enz. De cassette is opgebouwd uit 6 relais uitgangen en 2 ingangen met optocouplers.

Aan de indicatie LED's is het niveau van de uitgangen af te lezen. Is een uitgangsrelais bekrachtigd dan licht het overeenkomstige rode LEDje op.

Brandt bijvoorbeeld rode LED 1 dan zijn pin 1 en pin 2 doorverbonden (zie afbeelding VIC REL).

De 2 ingangen kunnen a.h.w. "zien" of er een schakelaar dicht of open is. Is de schakelaar op bijvoorbeeld ingang 1 dicht dan licht groene LED 1 op.

Alle aansluitingen op de VIC REL cassette kunnen dus geschieden zonder dat U er bang voor hoeft te zijn dat U een 6522 I/O chip opblaast.

Bij de VIC REL wordt een duidelijke engelstalige handleiding geleverd. Hierin wordt ook besproken welke registers aangesproken dienen te worden.

Dit wordt gedaan d.m.v. een testprogrammaatje welke achtereenvolgens de uitgangen 1 t/m 6 aan- en uitschakelt. Hieronder volgt een toepassingsvoorbeeld uit de handleiding. Het gaat hier om het automatisch openen en sluiten van een deur en het in- en uitschakelen van het licht.

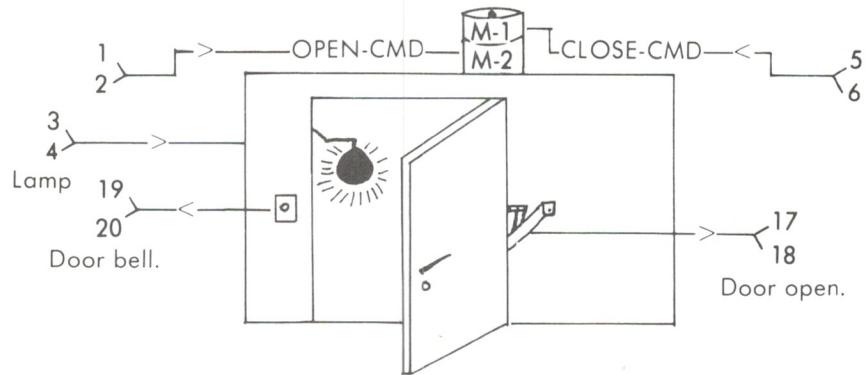
Ook het programma hiervoor staat hieronder als voorbeeld vermeld.

VIC REL is verkrijgbaar via de Handic dealers.



Bovenaanzicht

VIC REL



```

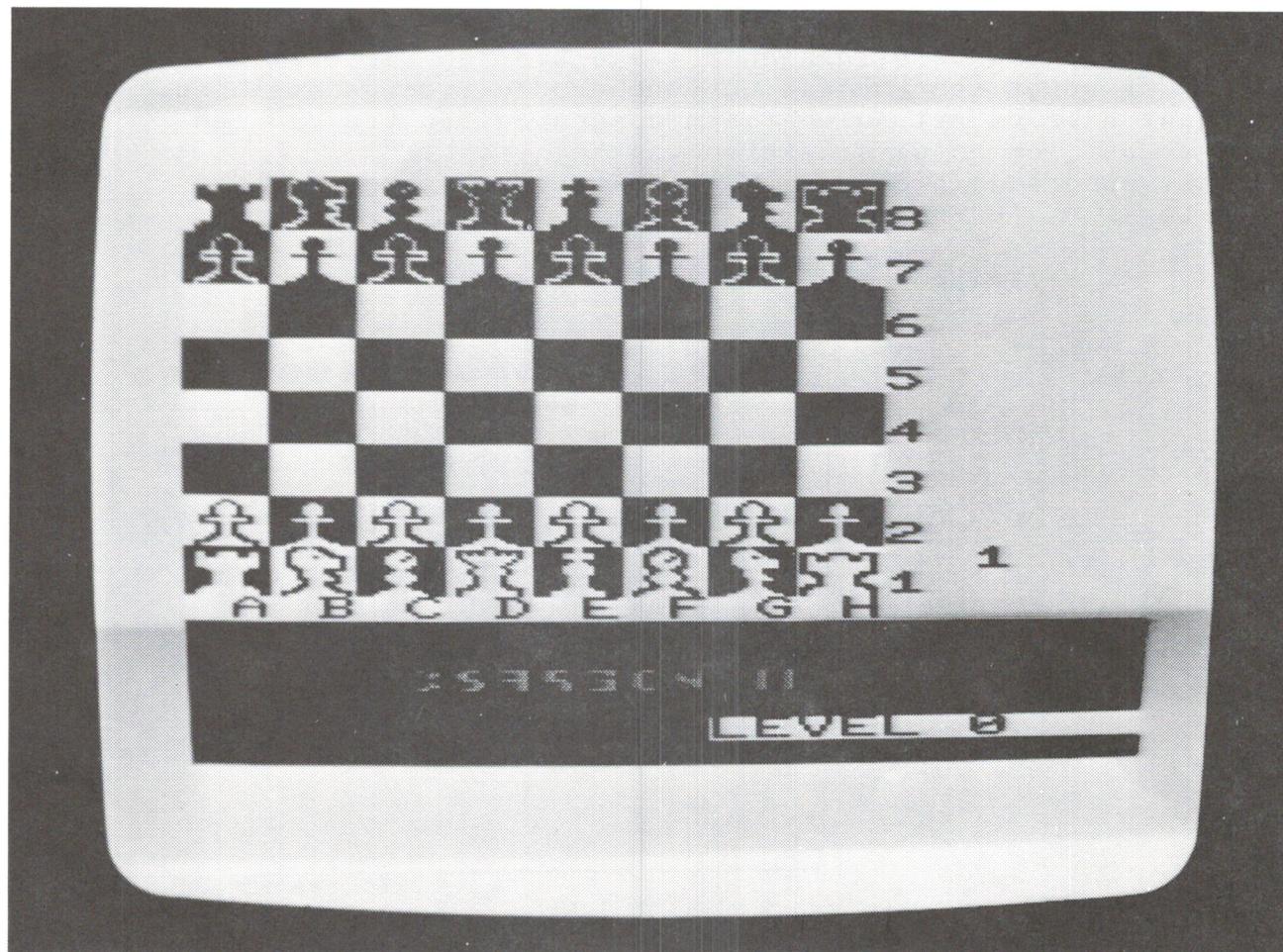
10 POKE 37138,63 :REG=37136
20 A = PEEK (REG) AND 128
30 IF A <> 0 THEN 20
40 POKE REG,PEEK(REG) OR 1
50 B = PEEK (REG) AND 64
60 IF B <> 0 THEN 50
70 POKE (REG),PEEK(REG)AND
(63-1)
80 FORT = 1 TO 800 : NEXT T
90 POKE (REG),PEEK(REG) OR 2
100 POKE (REG), PEEK(REG) OR 4
110 FORT1=1 TO 1000 : NEXT T1
120 POKE REG,PEEK(REG)AND
(63-4)
200 GOTO 20

```

Initiation
 Read inlet two (door bell button)
 If not pressed read again
 Start door motor R1=1
 Read inlet one
 Door not fully open yet
 Open! switch off motor R1=0

Wait a few seconds.
 Switch on the light in the hall. R2=1
 Close the door, leave the light on.
 Wait a few seconds
 Closed! Stop motor. R3=0

Be ready for the next guest.



SPELEN OP DE VIC-20, EN HOE

Laten we voorop stellen, dat er maar twee videospelgiganten zijn op deze bol en dat zijn BALLY/MIDWAY en ATARI. Beiden strijden al jaren om de eer (voor het geld hoeven ze het niet meer te doen) de grootste te zijn. BALLY heeft met spellen als SPACE INVADERS (origineel japans), PAC-MAN, OMEGA RACE, BLITZ en GALAXIAN nog altijd de touwtjes in handen voor wat de amusementshallen betreft. ATARI is verreweg de grootste producent van spelcomputers, maar dan ook met iedere grote titel, die je je maar kunt bedenken. Hoe het ook zijn mag, COMMODORE en dus de VIC-20 zijn de grote winnaars.

Door de aankoop van MOS-technology in 1981 is COMMODORE eigenaar geworden van de oh zo belangrijke chip: de 6502. En dat is dezelfde chip waarmee de ATARI computers worden gestuurd. COMMODORE volgt de ontwikkeling op de voet. Tevens kocht men BALLY/MIDWAY om zodoende te zorgen, dat men veel goede spellen zou kunnen produceren voor de VIC-20 en dat blijkt inmiddels wel.

Een andere tak van (computer)spellen zijn natuurlijk de bordspellen. De belangrijkste daarin zijn schaken, dammen, othello en go. Schaken hebben we inmiddels al op de VIC. Zo ook OTHELLO. Gezien de geavanceerde mogelijkheden van de VIC-20, die een pre zijn op de traditionele spelcomputers, lag het voor de

hand dat ook succesvolle avontuur bordspellen, zoals die van Scott Adams voor de VIC zouden worden ontwikkeld. Dit is inmiddels al gebeurd en zullen in de loop van augustus in ons land worden uitgegeven. Meer hierover in de volgende primeurs.

De laatste tak is logischerwijs de reeds ontwikkelde spellen voor mikrocomputers. AIR-TRAFFIC CONTROLLER en FLIGHT SIMULATION alsmede Adventure- en wargames zijn hiervan goede voorbeelden. Hier in ons eigen land zijn we al in een ver gevorderd stadium met een 16 K en 28 K ADVENTURE spel, alsook een tweetal wargames. In engeland o.a. wordt een simpele TRAFFIC CONTROLLER aangeboden. Een betere wordt op het ogenblik ontwikkelt in amerika door SEGA software en zal in september daar worden uitgegeven.

Het is dus nogal wat, dat de computermarkt ons biedt. Om de keus enigsinds te vereenvoudigen, volgen hier een aantal recensies van spellen voor de VIC-20. De favoriete TIEN vind je elders in dit blad.

En het begon allemaal met AVENGER.

Je vraagt je misschien af, waarom HANDIC als rechthebbende het spel niet onder zijn eigenlijke naam SPACE INVADERS heeft uitgebracht. Welnu, dat heeft te maken met het reeds bestaande kontrakt, dat ATARI

met BALLY had afgesloten, zodat ATARI de naam exclusief kon gebruiken. Wel moest men zelf het programma ontwikkelen. Commodore daarintegen kan de naam dus niet gebruiken, echter wel het programma. Een conclusie, die jezelf wel getrokken zult hebben, als je het spel vergelijkt met die uit een amusementshal. Hetzelfde zal zich ook nog voordoen met PAC-MAN, wat door HANDIC wordt uitgegeven als JELLY MONSTERS. Maar daar later meer over. In AVENGER hebben een vijftal verschillende ruimtewezens het plan gevat de aarde in te nemen en zich daar verder te kultiveren. Jouw taak is om dit plan om zeep te helpen, door d.m.v. een drietal bewegende en gecomputeriserende bunkers deze monsters neer te halen. Denk nu niet, dat doe ik wel even, want dan vergis je je. Deze ruimtewezens bewegen namelijk in een rustig tempo van links naar rechts en zakken langzaam richtingjawel, de aarde. Als je nu aardig onderweg bent om het geschut gericht te besturen en je schiet inderdaad een aantal wezen weg, dan plotseling verhoogd hun snelheid. Dit gebeurt viermaal per golf. Ja, en je raad het al. Stel je bent snel genoeg en je schiet een familie ruimtewezens weg, dan volgen al spoedig de volgende aanverwanten. Toch niet zo gemakkelijk dus. De computer houdt bovendien de hoogste score bij. Iedere keer, dat een ruimtewesen wordt weggeschoten ontvang je punten. Hoe hoger het ruimtewesen, hoe hoger de score. De hoogste score in een keer, behaal je door het

opperhoofd, die in een blauw ruimteschip van links naar rechts vliegt, neer te halen.

Enige tactische tips.

Schiet zoveel mogelijk rijen weg. Hoe meer volledige rijen je kunt wegschieten, hoe verder de ruimtewezens moeten open. Bovendien kunnen zij dan op beperktere schaal bommen loslaten.

Zodra je door hebt, wanneer de snelheid vergroot wordt, tracht je het wegschieten uit te stellen, waardoor je meer gelegenheid hebt om een of meer opperhoofden weg te schieten. De score verhoogt dan aanzienlijk.

Spelinhoud:	7
(aanstekelijk)	
Graphics:	8
Behendigheid:	8
Levensduur:	7

MAN VAN HET JAAR:PAC-MAN

Jelly Monsters werd in Engeland voor het eerst aan een geselecteerd publiek getoond en is samen met ASTEROIDS het meest gespeelde spel van dit moment. In Nederland verwacht HANDIC het spel met de FIRATO te kunnen uitgeven.

Ik denk dat JELLY MONSTERS onder de categorie zenuwslopend mag vallen.

Het spel start in het land van de spoken, alwaar energiedropjes voor het oprapen liggen en jouw opdracht is dan ook dit spokenland van dropjes schoon te vegen (happen). Maar liefst vier spoken zullen je echter op andere gedachten proberen te

brengen. Zowel de spoken als jij kunnen overal bewegen. Alleen het spokenhuis is voor de spoken toegankelijk. Alhoewel zij daar alleen in beginnen en er daarna niet meer terug kunnen keren. De vier uithoeken van dit miezerige land kennen grote energiedroppen, die na gegeten te zijn PACMAN instaat stellen de vier spoken, waarvan ik iedere keer weer de namen vergeet, op te peuzelen. De reincarnatie in het spokenhuis is heel snel, waarna de spoken weer voltallig erop los gaan. Boven- en onderin het beeld bevindt zich de ontsnappingskamer, waardoor je tijdelijk de spoken kunt ontlopen. Verdwin je dus boven uit het beeld, dan verschijn je weer onderaan in het spokenland. Wees echter voorzichtig op afwachtende spookjes. Onder het spookhuis zul je incidenteel een energietraktatie kunnen ophappen, die bovendien een aardig bijverdienste aan de score toevoegt. Zodra het spokenland van energiedroppen is schoongehapt, eindigt het spel niet, maar vult het spokenland zich opnieuw met dropjes. Een rage in de amusementshallen en levensechte belevenis op de VIC-20

Spelinhou: 9
Graphics: 7
Behendigheid: 9
Levensduur: 8
(ook voor kinderen)

In Spin in SPIDERS FROM MARS

Dit is tot op heden zonder twijfel het beste spel verkrijgbaar op de VIC-20 (zowel de mensen bij HANDIC als de gebruikersclubleden waren het daarover eens). Het is een fascinerend spel met uitzonderlijk goede graphics. Er wordt zelfs gefluisterd dat COMMODORE Amerika de rechten zou hebben overgenomen en zeker niet onterecht. Het spel is ontwikkeld door Peter Fokos en wordt in nederland door Computerworld geimporteerd. Eindelijk kun je je verplaatsen in de wereld van de kleine ERIK van Godfried Bomans, die in een droom zich geplaatst ziet tussen de vriendelijke maar ook harde insektenwereld. Alleen hier zijn de insekten allemaal hard en veelal dodelijk. Als vlieg (en wat voor een) moet je spinnen, vliegen, vlinders en speciaal de dodelijke wespen neer steken. Dit gebeurt fictief door angels af te schieten in de richting van een insect. Ieder insect op zijn beurt heeft weer een verdediging. Vlinders vliegen gewoon tegen je op. Zo ook de spinnen, die aan een rag naar beneden klimmen. Zodra een spin geland is kun je echter ook in zijn web verstrikt raken. Alleen door dan een angel af te vuren, kun je tijdelijk door dit web heen vliegen. Andere vliegen laten doelzoekende bommen los, die tien tot vijftien seconden blijven rond zweven. En dan is er nog de wesp, die eigenlijk het meest agressief aanvalt. Zijn

angel vliegt als een vuurpijl door de lucht, waardoor je echt alle behendigheid nodig hebt om zijn dodelijke steek te ontwijken. Om het nog wat moeilijker te maken zal het beest zich incidenteel verschuilen achter een vlieg of een vlinder. Wees dus wakker als je aan dit spel begint en zorg dat je zo snel mogelijk de wesp doodt. Zodra je groeit in niveau zullen er andere dieren met aparte eigenschappen zich aankondigen en tot groene ergenis zullen meerdere wespen rondvliegen. Niets wordt dus aan het toeval overgelaten. Een keiharde tien.

Spelinhoud:	10
(verslavend)	
Graphics:	9
Behendigheid:	8
Levensduur:	9

SCHAKEN OP HOOG NIVEAU: SARGON II

Mijn grootste probleem in het recenseren van de diverse spellen was wel die ene schaakkassette. Want alhoewel iedere meester en grootmeester een boek heeft geschreven over ditzelfde onderwerp, is dit eigenlijk niet de bedoeling van deze rubriek en bovendien ben ik daartoe niet kapabel genoeg. De enige oplossing die ik kon vinden was dan ook om een meer of min gelijkwaardig produkt tegenover de SARGON II te zetten, maar ook dit bleek niet echt eerlijk te zijn. Want hoewel de ATARI schaakkassette tot de vijf beste ter wereld behoort, bleek hij op niveau 6 (het een na hoogste niveau) niet in staat om de SARGON II

redelijk partij te geven. Met de SARGON op niveau 3 stond de ATARI na 20 zetten al op een loper en toren achterstand. Wel moet ik daaraan verbinden, dat de ATARI pas echt op niveau gaat spelen in de hoogste stand, maar iedere zet duurt dan ruim tien uur. Ik moet deze recensie dan ook beperken tot de test resultaten van de microchess kampioenschappen onlangs in Amerika gehouden, waarin de SARGON II bij de eerste drie eindigde. Rest ons nog te melden dat en passant en rokeren tot de mogelijkheden behoort en dat naast het bord (kontinue in beeld) men de vorige drie zetten altijd terug kan zien. Voor diegenen die een bepaalde situatie willen bestuderen of willen na spelen bestaat er de mogelijkheid om een bepaalde set-up te plegen. De graphics zijn van een dermate kwaliteit, dat discussies omtrent wie, wat, hoe, waar niet kunnen ontstaan. Een van de beste voor een belachelijk lage prijs.

DE RATTENVANGER VAN HANDIC

Van alle RAM/ROM cassettes geniet RADAR RATRACE mijn persoonlijke voorkeur. Een muis, drie ratten en een aantal muisstille katten. De muis ben jij en de rest wordt bestuurd door de computer. Een gelijke strijd, vind je niet. Welnu voor fl. 98,- bevindt je je in een doolhof waarin tien stukken kaas en bijna evenzo veel katten en ratten ronddolen om door jou te worden opgepeuzeld en om jou op te peuzelen. Het moeilijkste is wel, dat je vrijwel konstant naar twee verschillende schermen moet kijken. Allereerst is daar

het radarscherm, waarop je precie kunt zien waar de ratten, de katten, het kaas en natuurlijk jijzelf zich bevindt en het scherm, waar je maar een gedeelte van het scherm ziet, en waar je jouw muis doorheen moet leiden.

Bepaald geen eenvoudige klus. Het spel kan zowel met keyboard als met een joystick gespeeld worden. De muis kan overal lopen, maar kiest zelf een richting zodra hij in een muur wordt geleidt. De ratten zullen trachten om hem in een hoek op te sluiten, waarna hij simpelweg kan worden opgepeuzeld. Wel heeft de muis een afleidingswolk (sterren), die d.m.v. de rode knop op de joystick kan worden vrijgegeven. Zodra een rat in zo'n wolk terecht komt raakt hij alle gevoel voor richting kwijt en kan de muis ontsnappen. Sommige stukken kaas hebben tweemaal een bepaalde waarde, waardoor de score (want daar gaat het uiteindelijk om) aanmerkelijk verhoogd kan worden.

Ook houdt de computer de hoogste score in beeld, mits natuurlijk de computer niet na ieder spel wordt uitgezet. Uiteindelijk is het dan de bedoeling om in een bepaalde tijd alle stukjes kaas op te eten. Zodra de tijd echter is verstrekken gaan de ratten sneller dan de muis bewegen en dan is het natuurlijk snel afgelopen. Helaas is het mij, noch de verschillende medespelers, gelukt om het hele bord schoon te vegen, dus ik kan nog weinig over het vervolg vertellen.

Om toch vooral te zorgen dat ons zenuwstelsel voldoende te verwerken krijgt, heeft men het programma gekompleteert met een opbeurend melodietje, waar je na

vijf keer spelen AGGRESSIEF VAN WORDT. Om daarom toch vooral te proberen uit de gekke-boerderij te blijven kun je misschien beter het geluid wat kalmeren.

Een spel met een repeterende werking. Je kunt er echt niet van afblijven.

Spelinhoud:	7
Graphics:	7
Behendigheid:	9
Levensduur:	8

EEN WOLK VOL WONDEREN

Dit spel is vreselijk snel en ontwikkelt met een originele frisheid, die zijn weerga nog moet vinden. Evenals SPIDERS FROM MARS is CLOUDBURST afkomstig van Peter Fokos. Een man, die aan het eind van '82 de originaliteitsprijs van HANDIC en/of COMMODORE zou moeten ontvangen. Zijn spellen zijn een perfecte aanwinst op de VIC-20 en moeten met alle superlatieven behangen worden.

Het laserkanon staat tot jouw beschikking. De overwaaiende wolken vertonen een verassing en jij weet in het begin niet wat je overkomt. Al met al voldoende voedsel om weer een rijk spel op de markt te gooien.

Ook met dit spel kun je weer allerlei komische opmerkingen plaatsen en uitgebreid ingaan op wat er inhoudelijk gebeurt. Doe me echter een plezier en ondervindt waarom ik dit spel niet te beschrijven vindt. Vast en zeker zul je ver na zessen de winkel uitgeschopt worden, want zodra je aan CLOUDBURST begint, kun je het niet langer laten. Taktisch is het wel belangrijk dat je in ieder geval naar een

kant van het speelveld uitwijkt. Laat de bommen rustig gaan (wel ontwijken graag) en concentreer alle aandacht op de witte karren uit de blauwe hemel. Bovendien hoe meer wolken je kunt wegschieten des te minder kans krijgen de karren om neer te dalen en je dus aan te vallen. Schieten kun je in ieder geval naar alle kanten. Maar nogmaals de karren zijn verschrikkelijk snel.

Doen hoor, loop een VIC winkel binnen en kijk hoelang je er vanaf kan blijven. Een gegarandeerd succes.

Spelinhoud:	6
(Een unieke variant)	
Graphics:	8
Behendigheid:	9
Levensduur:	8

VEEL SPEELPLEZIER.
Nicky Moeken.

De favoriete TIEN

1. SPIDERS OF MARS --- Audiogenic
2. CLOUDBURST --- Audiogenic
3. ALIEN --- Microsave
4. RADAR RATRACE --- Commodore
5. DEFENDA --- LLamasoft
6. SPACE STORM --- Rabbit
7. GALAXY --- Microsave
8. OTHELLO/RENAISSANCE --- Microsave/Audiogenic
9. INVADER FALL --- Vic pack
10. SARGON CHESS II --- Commodore



Hoge-resolutie op de VIC-20

DEEL 3

Programma opzet geschikt voor iedere geheugengrootte.

In het vorige deel van deze Hoge-resolutie serie hebben we gezien dat het ook mogelijk is hoge-resolutie en zelfdefinieerbare karakters te gebruiken met 8K en/of 16K RAM uitbreiding. Helaas werkten deze programma's dan weer niet op een standaard 5K of een VIC-20 met 3K RAM uitbreiding. In deze aflevering gaan we een programma opzet bekijken die het mogelijk maakt een programma te schrijven dat voor alle geheugenversies geschikt is. De inhoud van de voorgaande delen wordt bekend voorondersteld. Tot slot geven we een programmavoorbeeld hiervan voor zelfdefinieerbare karakters, dat ook een eenvoudige vorm van animatie laat zien.

Nu eerste het setup gedeelte van het programma. We bepalen eerst met behulp van de 'zero page' locatie 44 (het begin van BASIC), met wat voor een geheugen we te doen hebben.

Er zijn 4 mogelijkheden :

X=18 : Er is een 8K of 16K RAM uitbreiding aanwezig en u heeft niet de twee POKE opdrachten gegeven voor het verplaatsen van het BASIC programma gedeelte. Het programma doet dit nu voor u maar moet daarna opnieuw ingeladen worden.

X=28 : U heeft dit nu wel van te voren gedaan of het programma heeft het zelf hiervoor gedaan (zie X=18). Nu kunnen de variabele bepaald worden. Hier gaan we verder op in.

X>16 : Onbekende locatie. Waarschijnlijk afkomstig van voorafgaand ander programma. Het beste kan de computer geRESET worden (SYS 64802).

X<=16: We hebben een 5K of 8K VIC-20. Eerst moet nu de TOP van het programmageheugen verplaatst worden (locatie 56) om daar plaats voor de tekens te reserveren. Dit is afhankelijk van CG (zie ook regel 10). Als CG=0, zoals in het voorbeeld, gebruiken we maar ruimte voor 64 definieerbare karakters (en er blijft dus meer ruimte over voor het programma). Is CG ongelijk aan 0, dan wordt er ruimte gemaakt voor alle 256 tekens. Als we met 8K of 16K werken is er altijd ruimte voor 256 karakters. Hieronder een tabel met beschikbare programmaruimte, let wel dat ook het setup programma hier al afgetrokken is. Het is dus de netto programma ruimte.

5K	CG=0	2739 BYTES
5K	CG=1	691 BYTES
8K	CG=0	5811 BYTES
8K	CG=1	3763 BYTES
13K	CG=0/1	8883 BYTES
21K	CG=0/1	17075 BYTES

Het volgende gedeelte bepaalt de variabele. De variabelen X, CG en I hebben we verder niet nodig dus die kunnen als dat nodig is weer gebruikt worden. De andere variabele hebben de volgende betekenis :

CA : aantal te definieren karakters (0-CA).

CS : wordt gebruikt in POKE 36869,CS om
in de grafische mode te komen.

CT : Eerste positie beeldscherm geheugen.

CK : Eerste positie kleuren geheugen.

CB : Citroen brandewijn of
eeste locatie van karaktergenerator.

Het programmavoorbeeld laat zien hoe deze variabelen te gebruiken zijn.

Ze hebben de volgende waarde als:

5/8 K CG=0

X= 28

CA= 63

CS= 255

CT= 7680

CB= 7168

CK= 38400

I= 512

5/8 K CG=1

X= 20

CA= 255

CS= 253

CT= 7680

CB= 5120

CK= 38400

I= 2048

8/16 K CG=0/1

X= 28

CG= 0

CA= 255

CS= 205

CT= 4096

CB= 5120

CK= 37888

I= 2048



Ons programma voorbeeld van deze keer (regel 1000-5060) maakt gebruik van de methode een aantal 'lege' karakters op het scherm te zetten (regel 1050) en daar dan een, veranderd, patroon in te zetten. Dit eigenlijke veranderen gebeurt in regel 2040. Het programma geeft natuurlijk slechts het basis idee van animatie weer zonder dit verder uitgewerkt te hebben. Het laat wel duidelijk het gebruik van de diverse variabelen uit ons SETUP programma zien.

```

10 X=PEEK(44):CG=0:PRINT"PROGRAMMA SETUP"
20 IFX=18THENPRINT"LAAD PROGRAMMA OPNIEUW":POKE44,28:POKE7168,0:NEW
30 IFX=28THENCA=255:CS=205:CT=4096:CB=5120:CK=37888:GOT090
40 IFX<>16THENPRINT"GEHEUGEN ONJUIST":STOP
50 IF CG=0THENPOKE56,28:GOT070
60 POKE56,20
70 CLR
80 X=PEEK(56):CA=63+((28-X)/4)*96:CS=253+(X-20)/4:CT=7680:CB=256*X:CK=38400
90 FORI=0TOCA*8+7:POKEI+CB,PEEK(I+32768):NEXT
1000 REM HOOFDPROGRAMMA
1010 POKE 36869,CS:PRINT":":POKE36879,185:DIMA(26)
1020 FORI=CBTOCB+23*8:POKEI,0:NEXT
1050 FORI=0TO22:POKECT+10+(22*I),I:POKECK+10+(22*I),0:NEXT
1060 X=CB+23*8
1070 LA=25:GOSUB 2000
1080 LA=7:GOSUB 2000
1090 FORI=CBTOCB+LA:POKEI,0:NEXT
1900 FORI=1TO1000:NEXT:SYS 65234
1999 REM SUBROUTINE
2000 FORI=0TOLA:READA:A(I)=A:NEXT
2020 FORI=0TOLA:POKEX-LA+I,A(I):NEXT
2040 FORII=X-LA+1TOCBSTEP-2:FORI=0TOLA:POKEII+I,A(I):NEXT:NEXT
2060 RETURN
4999 REM DATA
5000 DATA 16,16,56,40,56,124,124,124,124,108,124,124,124,108,124,124,56,56,124
5050 DATA 124,84,40,0,40,40,0,16
5060 DATA 16,56,84,146,186,16,0,0

```

VIC TIP

VIC-1515 PRINTER TIP

Normaal geeft de printer een stukje open regel tussen de tekst om alles goed leesbaar te houden. Dit is niet zo prettig als je op de printer gebruik wilt maken van de grafische tekens die net als op het beeldscherm netjes moeten aansluiten. Het onderstaande programma laat zien hoe dit te doen is, het spreekt voor zichzelf dacht ik.

```
10 OPEN 1,4
20 B$=CHR$(15)
30 E$=CHR$(8)
40 PRINT#1
50 PRINT#1,"      ████      ████"
55 PRINT#1,"      ████      ████";E$
60 PRINT#1
70 PRINT#1,B$;"      ████      ████";E$
75 PRINT#1,B$;"      ████      ████";E$
80 PRINT#1,B$;
99 END
```

█████ █████

█████ ████████

ASSEMBLER vervolg

In de vorige aflevering van VIC PRIMEURS hadden we van de heer van Westerhoven een beschrijving van zijn ASSEMBLER afgedrukt. Voor dit nummer ontvingen wij van hem een aanvulling op deze ASSEMBLER en een aanvulling in de vorm van een nieuw programma, nl. DEBUG. Verder heeft de heer van Westerhoven zich bereid verklaard een 'eendaagse workschap' machinetaalprogrammering te organiseren, in principe op een zondag. Mochten leden hierin geïnteresseerd zijn dan kunnen ze dat even schriftelijk doorgeven aan de VIC GEBRUIKERSCLUB.

UITVERDING VOOR ASSEMBLER-1

Hieronder worden voor de assembler die beschreven is in VIC-primeurs nr. 2 1982, enkele wijzigingen gegeven die de mogelijkheden van deze assembler zullen vergroten.

1. De regel van de verwerkingsteller van programma 8 altijd als eerste regel op het scherm:
8165 PRINT "cithwacht tot:"; PL; "-"
2. In programma 7 de waarde van de onbekende operatiekode printen:
7070 T(1)=VAL(LEFT\$(NA\$1)): IF T(1)=0 THEN T(1)=1:
IN\$=STR\$(LN): PRINT "cursor lx->; : GOTO 7150
3. Nieuwe tabelfunctie met deze mogelijkheden:
TBL FAD 2 byte tabelregel via adreslabel @AD
TBL CAD 1 byte tabelregel via adreslabel @AD
TBL KON 1 byte tabelregel via constantenlabel KON
TBL O32 1 byte tabelregel via decimaal getal van 3 cijfers
4. In de geconditioneerde sprongopdrachten de mogelijkheid om het aantal bytes door de assembler te laten berekenen:
bv.: BNE 1AD

```

8105 IF MID$(NA$,T,1)="↑" AND LEFT$(NA$,3)<>"BIT" AND
LEFT$(NA$,1)="B" THEN LN=LN-1
8375 IF LEFT$(NA$,1)="B" AND LEFT$(NA$,3)<>"BIT" AND T=3
THEN T=9: T(2)=T(2)-LN-2
8377 IF T(2)<0 THEN T(2)=256+T(2)

```

5. Een check op het inlezen van programma 6 en een verify voor het programma dat geschreven is met programma 5:
5060 CLOSE 1: PRINT "rewind for verify";:
INPUT IN\$: GOTO 6010
6030 INPUT#1 IN\$:
6035 IF ST AND 48 THEN PRINT "error": CLOSE 1 : GOTO 1040
6037 IF PR=6 THEN P\$(LN)=IN\$
6040 IF IN\$="↑↑↑" THEN P\$(LN)="" : GOTO 6060
6. Verhoging van het versienummer:
1010 GS=37888 : PS=4096 : REM assembler-2

Na deze wijzigingen is de naam van de assembler: ASSEMBLER-2

Systeembeschrijving voor DEBUG

Dit programma is ontwikkeld om op een eenvoudige en geriefelijke manier een systeemroutine te kunnen testen. Het bestaat uit 2 delen t.w. een basicprogramma en een systeem-

pag 18 routine. Voor deze systeemroutine kan het gebruikte RAM pagina direct voor basic start gebruikt worden. Zie schema hiernaast.

Deze programmakombinatie kan als een zelf-

standige eenheid functioneren, maar het is ook mogelijk hem aan 'assembler-2' te koppelen via de aansprong van progr. 11 van de as-

sembler. Dit laatste heeft tot voordeel dat programma's van de assembler en de functies van DEBUG door elkaar gebruikt kunnen worden.

Met DEBUG kan op iedere gewenste instructie in de machinecode van de 6502 mikroprocessor gestopt worden. Dit kan door het adres van een andere woorden. Verder moet in gedachten gehouden worden dat het eerste byte van een instuktie als 'break'adres in te geven.

Met ander woorden, het breakpoint moet altijd op het adres van een operatiekode zijn. Verder moet in het niveau waarin het breakpoint ligt groter of gelijk moet zijn aan het niveau waarin met DEBUG gestart wordt voor de test. Als alleen gewerkt wordt met -, = en ; blijft de inhoud van de registers X, Y, A en P intact en wordt steeds weer deze waarden gebruikt voor de volgende testfase. In DEBUG kan gekozen worden uit:

VRIJ RAM gebied	debug syst.	debug basic
-----------------	-------------	-------------

```

A(waarde)      ; nieuwe inhoud voor akku
X(waarde)      ; nieuwe inhoud voor reg. X
Y(waarde)      ; nieuwe inhoud voor reg. Y
S              ; nieuwe inhoud voor reg. Y
V              ; zet flag S
D              ; zet flag V
C              ; zet flag D
2              ; zet flag 2
C              ; zet flag C
-(breakpointadres) ; break'adres I
=(breakpointadres) ; break'adres II
W(adres)       ; schrijf 'waarde' in locatie
R(adres)       ; lees aantal locaties
G(adres)       ; start test vanaf 'adres'
L              ; 1. ga terug naar assembler
                ; 2. ga naar begin DEBUG en
                ; clear alle variabelen
                ; 3. lees nu DEBUG-basicprogramma in (basic start is nog steeds 22)
                ; 4. verplaats basic start naar pagina 21 met:
POKE 44,21

```

Om systeemroutine en DEBUG samen als één programma op de band te save'n kan als volgt gewerkt worden:

1. Stel basic start is gelegen op pagina 22 (zie locatie 44)
2. Lees assembler-2 in assembler de DEBUG-systeemroutine op pagina 21
3. Lees nu DEBUG-basicprogramma in (basic start is nog steeds 22)
4. Verplaats basic start naar pagina 21 met:
5. Daarna SAVE "debug",1,1

Vanaf dit moment moet DEBUG vanaf de band ingelezen worden met basic start 'pagina 22'. Het systeemdeel valt dan automatisch voor basic start en het basic deel er achter.

Systeemroutine_BREAKPOINT

DEBUG

```

5632      ; (start basic)-256
        TBL 000      ; save accumulator
        ↑SA          ; save register X
        ↑SX          ; save statusregister P
        ↑SP          ; save stackpointer SP
        ↑SS          ; save breakpointkode
        ↑BK          ; returnadres in routine
        ↑AD          ; breakpointadres L
        ↑BL          ; breakpointadres H
        ↑LDA         ; laad Y
        ↑LDS         ; laad X
        ↑LDA         ; laad Y
        ↑PHA         ; laad P
        ↑PLP         ; save SP
        ↑STX         ; 100 → SF
        ↑LDX         ; laad breakvector H
        ↑STA         ; laad breakvector L
        ↑LDA         ; laad A
        ↑INC         ; sprong naar systeemroutine
        ↑CLI         ; terug van systeemroutine
        ↑PLA         ; save Y
        ↑STA         ; save X
        ↑PLA         ; save A
        ↑STA         ; save P
        ↑PLA         ; save breakpointadres L
        ↑STA         ; save breakpointadres H
        ↑PLA         ; herstel SP
        ↑LDX         ; return naar basic
        ↑TXS
        ↑RTS

```

```

11000 S=0 : V=0 : D=0 : Z=0 : C=0 : B1=0 : V1=0 : B2=0 : V2=0 :
      B=(PEEK(44)-1)*256
      PRINT "-A2;" ; INPUT NAS$ : PRINT "cursor_lxi"; TAB(13); : IF LEN(NAS$)<2
      THEN 11180
      IF T(1)<0 THEN 11180
      IF NAS$="A" THEN T(2)=0 : GOTO 11190
      IF NAS$="Y" THEN T(2)=1 : GOTO 11190
      IF NAS$="N" THEN T(2)=2 : GOTO 11190
      IF T(1)>65535 THEN 11180
      IF NAS$="L" THEN 11180
      IF NAS$="W" THEN 11180
      IF NAS$="=" THEN 11180
      IF NAS$="G" THEN 11180
      IF NAS$="R" THEN INPUT T(3) : GOTD 11210
      IF NAS$="S" THEN S=228 : GOTD 11450
      IF NAS$="V" THEN V=64 : GOTD 11450
      IF NAS$="D" THEN D=8 : GOTD 11450
      IF NAS$="Z" THEN Z=2 : GOTD 11450
      IF NAS$="C" THEN C=1 : GOTD 11450
      PRINT "#?*" : GOTD 11010
      IF T(1)>255 THEN 11180
      POKE B+(2),T(1) : PRINT : GOTD 11010
      PRINT TAB(6); T(1) : PEEK(T(2)) : IF T(3)<1 THEN 11010
      T(3)=T(3)-1 : T(2)=T(1)+1 : GOTD 11210
      INT(T(3)) : IF T(3)>255 OR T(3)<0 THEN PRINT "cursor_lxi"
      MAE(13) : GOTD 11180
      POKE T(1),T(3) : GOTD 11010
      B1=T(1) : PEEK : GOTD 11010
      B2=T(1) : PEEK : GOTD 11010
      B1=0 AND B2=0 THEN PRINT : PRINT TAB(6) ; "zet -=" ;
      GOTO 11010
      T=B+PEEK(B+6) : IF S+V+D+Z>0 THEN POKE B+3, S+V+D+Z+C
      IF B1<>0 THEN V1=PEEK(B1) : POKE B1,PEEK(B+5)
      IF B2<>0 THEN V2=PEEK(B2) : POKE B2,PEEK(B+5)
      POKE T, INT(T(1)/256) : POKE T-1,PEEK(T)*256+T(1)
      SYS B+9
      IF B1<>0 THEN POKE B1,V1
      IF B2<>0 THEN POKE B2,V2
      IF B1<>0 THEN PRINT "clh"
      IF B2<>0 THEN PRINT "clh"
      IF B1<>0 THEN T=-128 : PRINT "S" ;
      IF B2<>0 THEN T=-128 : PRINT "V" ;
      IF B1<>0 THEN T=-32 : PRINT "B" ;
      IF B2<>0 THEN T=-32 : PRINT "Y" ;
      IF B1<>0 THEN T=-8 : PRINT "D" ;
      IF B2<>0 THEN T=-4 : PRINT "I" ;
      IF B1<>0 THEN T=-2 : PRINT "Z" ;
      IF T<>0 THEN PRINT "C" ;
      PRINT "ERK" ;
      "cursor_lxi" : GOTD 11000
      PRINT : GOTD 11010

```

VIC TIP

SPECIALE PRINTER TEKENS

Als u een listing op het scherm of in het bijzonder een van de printer bekijkt zult u vast en zeker de speciale tekens hebben gezien die daarin (soms veelvuldig) voorkomen. Het onderstaande lijstje moet u helpen deze tekens gemakkelijker over te nemen. In de eerste kolom staat de CTRL of SHIFT toets die u eventueel tegelijkertijd met de in de tweede kolom vermelde toets moet indrukken. CR U/D staat voor de toets die de cursor naar boven/beneden beweegt, CR L/R is de toets daarnaast (cursor links/rechts) geheel rechts onder op het toestenbord.

En in de laatste kolom vindt u het teken dat op het scherm verschijnt of dat u in de printer-lising vindt. Om het teken te krijgen moet de cursor zich wel na een quote (dubbel aanhalingssteken) bevinden anders wordt het direct uitgevoerd.

CTRL	1	"█"
CTRL	2	"█"
CTRL	3	"█"
CTRL	4	"█"
CTRL	5	"█"
CTRL	6	"█"
CTRL	7	"█"
CTRL	8	"█"
CTRL	9	"█"
CTRL	0	"█"
SHIFT	CR U/D	"█"
SHIFT	CR U/D	"█"
	CR L/R	"█"
SHIFT	CR L/R	"█"
SHIFT	HOME	"█"
	CLR	"█"
	DEL	"█"

Inleiding_machinetaal

Op de volgende pagina's vindt u een overdruk uit de amerikaanse 'REFERENCE GUIDE' over machinetaal programmeren op de VIC-20. Hoewel natuurlijk in het engels lijkt het gedeelte ons voldoende interessant om in deze VIC PRIMEURS op te nemen.

INTRODUCTION TO MACHINE LANGUAGE

MCS6501-MCS6505 MICROPROCESSOR	INSTRUCTION SET – ALPHABETIC SEQUENCE
ADC Add Memory to Accumulator with Carry AND "AND" Memory with Accumulator ASL Shift Left One Bit (Memory or Accumulator)	JSR Jump to New Location Saving Return Address LDA Load Accumulator with Memory LDX Load Index X with Memory LDY Load Index Y with Memory LSR Shift Right One Bit (Memory or Accumulator)
BCC Branch on Carry Clear BCS Branch on Carry Set BEQ Branch on Result Zero BIT Test Bits in Memory with Accumulator BMI Branch on Result Minus BNE Branch on Result not Zero BPL Branch on Result Plus BRK Force Break BVC Branch on Overflow Clear BVS Branch on Overflow Set	NOP No Operation ORA "OR" Memory with Accumulator PHA Push Accumulator on Stack PHP Push Processor Status on Stack PLA Pull Accumulator from Stack PLP Pull Processor Status from Stack
CLC Clear Carry Flag CLD Clear Decimal Mode CLI Clear Interrupt Disable Bit CLV Clear Overflow Flag CMP Compare Memory and Accumulator CPX Compare Memory and Index X CPY Compare Memory and Index Y	ROL Rotate One Bit Left (Memory or Accumulator) ROR Rotate One Bit Right (Memory or Accumulator) RTI Return from Interrupt RTS Return from Subroutine
DEC Decrement Memory by One DEX Decrement Index X by One DEY Decrement Index Y by One EOR "Exclusive-Or" Memory with Accumulator	SBC Subtract Memory from Accumulator with Borrow SEC Set Carry Flag SED Set Decimal Mode SEI Set Interrupt Disable Status STA Store Accumulator in Memory STX Store Index X in Memory STY Store Index Y in Memory
INC Increment Memory by One INX Increment Index X by One INY Increment Index Y by One JMP Jump to New Location	TAX Transfer Accumulator to Index X TAY Transfer Accumulator to Index Y TSX Transfer Stack Pointer to Index X TXA Transfer Index X to Accumulator TXS Transfer Index X to Stack Pointer TYA Transfer Index Y to Accumulator

WHAT IS MACHINE LANGUAGE?

At the heart of every microcomputer, there is a central microprocessor, a very special microchip which is the "brain" of the computer. The VIC 20's microprocessor is the 6502 chip. Every microprocessor understands its own language of instructions, and these instructions are called the machine language instructions of that chip. To put it more precisely, machine language is the ONLY programming language that your VIC 20 really understands. It is the native language of the machine.

If machine language is the only language that the VIC 20 understands, then how does it understand the VIC BASIC programming language? If VIC BASIC is not the machine language of the VIC 20, what makes the VIC 20 understand VIC BASIC instructions such as PRINT and GOTO?

To answer this question, we must first see what happens to your VIC 20 when you turn it on. How does your computer know what to do when it is first turned on? Well, apart from the microprocessor which is the brain of the VIC 20, there is a huge machine language program which is "burnt" into a special type of memory called ROM that cannot be changed, and does not get erased when the VIC 20 is turned off, unlike a program that you put into the VIC's RAM. This huge program is in two parts, one taking care of the BASIC language, and the other called the "operating system."

The operating system is in charge of "organizing" all the memory in your machine for various tasks, looks at what characters you type on the keyboard and puts them onto the screen, and a whole number of other functions. The operating system can be thought of as the "intelligence and personality" of the VIC 20 (or any computer for that matter). So when you turn on your VIC 20, the operating system takes control of your machine, and after it has done its housework, it then says:

READY.
*

The operating system of the VIC 20 then allows you to type on the keyboard, and use the built-in "screen editor" on the VIC 20. The screen editor allows you to move the cursor, DELet, INSert, etc., and is, in fact, only one part of the operating system that is built-in for your convenience.

All of the commands that are available in VIC BASIC are simply recognized by another huge machine language program built into your VIC 20. This huge program "RUN's" the appropriate piece of machine language depending on which VIC BASIC command is being executed. This program is called the "BASIC interpreter," because it interprets each command, one by one, unless it encounters a command it does not understand, and then the familiar message appears:

?SYNTAX
ERROR
READY.
*

WHAT DOES MACHINE CODE LOOK LIKE?

You should be familiar with the PEEK, and POKE commands in the CBM BASIC language for changing memory locations. You will probably have used them for graphics on the screen, and for sound effects. The memory locations will have been 36874, 36875, 36876, 36877, 36878 for sound effects. This memory location number is known as the "address" of a memory location. If you can imagine the memory in the VIC 20 as a street of houses, the number on the door is, of course, the address. Now we will look at which parts of the street are used for which purpose.

SIMPLE MEMORY MAP OF THE VIC 20

Address	Description
0	Start of memory.
to 1023	Memory used by BASIC and the operating system.
1024 to 4095	This is a gap in memory for a 3K memory expansion module
4096 to 65535	8K VIC BASIC Interpreter.
57344 to 57343	8K VIC KERNEL OPERATING SYSTEM
37888 to 38399	Character color control table in expanded VIC 20
38400 to 38911	Character color control table.
38912 to 40959	Unused.
40960 to 49151	Expansion ROM.
49152 to 57343	8K VIC BASIC Interpreter.
57344 to 65535	8K VIC KERNEL OPERATING SYSTEM

Don't worry if you don't understand what the description of each part of memory means. This will become clear from other parts of this manual.

Machine language programs consist of instructions which may or may not have operands (parameters) associated with them. Each instruction takes up one memory location, and any operand will be contained in one or two locations following the instruction.

In your BASIC programs, words like PRINT, and GOTO do, in fact, only take up one memory location, rather than one for each character of the word. The contents of the location that represents a particular BASIC keyword is called a "token." In machine language,

7680 to 8185	This is the screen memory.
32768 to 36863	This is a gap in memory for memory expansion.
36864 to 36879	Character representations.
37136 to 37167	The VIC chip registers.
37888 to 38399	Input and output chip registers.
38400 to 38911	Character color control table.
38912 to 40959	Unused.
40960 to 49151	Expansion ROM.
49152 to 57343	8K VIC BASIC Interpreter.
57344 to 65535	8K VIC KERNEL OPERATING SYSTEM

there are different tokens for different instructions, which also take up just one byte (memory location = byte). Machine language instructions are very simple, i.e., each individual instruction cannot achieve a great deal. Machine language instructions either change the contents of a memory location, or change one of the internal registers (special storage locations) inside the microprocessor. The internal registers form the very basis of machine language.

REGISTERS INSIDE THE 6502 MICROPROCESSOR

THE ACCUMULATOR—This is THE most important register in the microprocessor. Various machine language instructions allow you to copy the contents of a memory location into the accumulator, or copy the contents of the accumulator into a memory location, or modify the contents of the accumulator or some other register directly, without affecting any memory. Also, the accumulator is the only register that has instructions to perform math on it.

THE X INDEX REGISTER—There are instructions to do nearly all of the transformations you can do to the accumulator, and other instructions to do things that only the X register can do. Again, various machine language instructions allow you to copy the contents of a memory location into the X register, or copy the contents of the X register into a memory location, or modify the contents of the X, or some other register directly, without affecting any memory.

THE Y INDEX REGISTER—There are instructions to do nearly all of the transformations you can do to the accumulator, and the X register, and other instructions to do things that only the Y register can do. Again, various machine language instructions allow you to copy the contents of a memory location into the Y register, or copy the contents of the Y register into a memory location, or modify the contents of the Y, or some other register directly, without affecting any memory.

THE STATUS REGISTER—This register consists of eight "flags" (a flag = something that indicates that something has, or has not, occurred).

THE PROGRAM COUNTER—This contains the address of the current machine language instruction being executed. Since the

operating system is always "RUN"ning in the VIC 20 (or, for that matter, any computer), the program counter is always changing. It could only be stopped by halting the microprocessor in some way.

The Stack Pointer—This register contains the location of the first empty place on the stack. The stack is used for temporary storage by machine language programs, and by the computer.

THE TOOLS AVAILABLE; GETTING READY . . .

How Can You Write Machine Language Programs?

Since machine language programs reside in memory, and there is no facility in your VIC 20 for writing and editing machine language programs, you must use either a program to do this, or write for yourself a BASIC program that "allows" you to write machine language.

Most commonly used to write machine language programs are "assemblers." These packages allow you to write machine language instructions in a standardized "mnemonic" format, which makes the machine language program a good deal more readable than a stream of numbers. To recap: A program that allows you to write machine language programs in mnemonic format is called an "assembler," and also, a program that displays a machine language program in mnemonic format is called a "disassembler." Available for your VIC 20 is a machine language monitor cartridge (with assembler/disassembler, etc.) made by Commodore.

VICMon

The VICMon cartridge available from your local dealer is a program that allows you to escape from the world of VIC BASIC, into the land of machine language. It can display the contents of the internal registers in the 6502 microprocessor, and it allows you to display portions of memory, and change them on the screen, using the screen editor. It also has a built-in assembler and disassembler, and many other features that allow you to write and edit machine language programs easily.

You don't HAVE to use an assembler to write machine language, but the task is considerably easier with it. If you wish to write machine language programs, it is advised strongly that you buy an assembler of some sort. Without an assembler you will probably have to "POKE" the machine language program into memory, which, if you value your sanity, is totally inadvisable. This manual

will give examples in the format that VICMon uses from now on. Nearly all assembler formats are the same, therefore the machine language examples shown will almost certainly be compatible with any assembler other than the one incorporated in VICMon.

Hexadecimal Notation

This is a notation which most machine language programmers refer to when referring to a number or address in a machine language program.

Some assemblers let you refer to addresses and numbers in decimal (base 10), binary (base 2), or even octal (base 8) as well as hexadecimal (or just "hex" as most people say). These assemblers do the conversions for you.

Hexadecimal will probably seem a little hard to grasp at first, but like most things it doesn't take long (with practice) to master it. By looking at decimal (base 10) numbers, you will see that each digit in that number ranges between zero and a number equal to the base less one, i.e., > 9 . THIS IS TRUE OF ALL NUMBER BASES.

Binary (base 2) numbers have digits ranging from zero to one (which is one less than the base). Similarly hexadecimal numbers should have digits ranging from zero to fifteen, but we do not have any single digit figures for the numbers ten to fifteen, so the first six letters of the alphabet are used instead:

DECIMAL	HEXADECIMAL	BINARY
0	—	00000000
1	1	00000001
2	2	00000010
3	3	00000011
4	4	00000100
5	5	00000101
6	6	00000110
7	7	00000111
8	8	00001000
9	9	00001001
10	A	00001010
11	B	00001011
12	C	00001100
13	D	00001101
14	E	00001110
15	F	00001111
16	—	00010000

If that's confusing, let's try to look at it another way:

Example of how a base 10 (decimal number) is constructed.

Base raised by 3 2 1 0
increasing powers.. 10 10 10 10

Equals:..... 1000 100 10 1

Consider 4569 (base10) 4 5 6 9 = $4 \times 1000 + 5 \times 100 + 6 \times 10 + 9$

Example of how a base 16 (hexadecimal number) is constructed.

Base raised by 3 2 1 0
increasing powers.. 16 16 16 16

Equals:..... 4096 256 16 1

Consider 11D9 (base16) 1 1 D 9 = $1 * 4096 + 1 * 256 + 13 * 16 + 9$

Therefore 4569(Base10) = 11D9(Base16)

The range for addressable memory locations is 0 – 65535 (as was stated earlier). This range is therefore 0 – FFFF in hexadecimal notation.

Usually hexadecimal numbers are prefixed with a dollar sign, to distinguish them from decimal numbers. Let's look at some "hex" numbers, using VICMon by displaying the contents of some memory. VICMon shows you:

```
B* PC SR AC XR YR SP
: 0401 32 04 5E 00 F6 (these may be different)
```

Then if you type in:

```
.M 0000 0020 (and press RETURN ).
```

You will see rows of 6 hex numbers. The first 4 digit one is the address of the first byte of memory being shown in that row, and the other five numbers are the actual contents of the memory locations beginning at that start address.

You should endeavor to learn to "think" in hexadecimal. This is not difficult, since there is no need to think in decimal. For example, if it is said that a particular value is stored at \$14ED instead of 5357, this shouldn't cause any headaches.

YOUR FIRST MACHINE LANGUAGE INSTRUCTION

"LDA"—Load the Accumulator

In 6502 assembly language, mnemonics are always three characters. LDA represents "load accumulator with . . ." and what the accumulator should be loaded with is decided by the parameter(s) associated with that instruction. The assembler knows which token is represented by each mnemonic, and when it "assembles" an instruction, it simply puts into memory (at whatever address has been specified), the token and what parameters are given. Some assemblers give error messages, or warnings when the user has tried to assemble something that either the assembler or the 6502 microprocessor cannot do.

If we put a "#" symbol in front of the parameter associated with the instruction, this means that we wish the register specified in the instruction to be loaded with the "value" after the "#". For example:—

```
LDA #$05
```

This instruction will put \$05 (decimal 5) into the accumulator register. The assembler will put into the specified address for this instruction, \$A9 (which is the token for this particular instruction, in this mode), and it will put \$05 into the next location after the location containing the instruction (\$A9).

If the parameter to be used by an instruction has "#" before it, i.e., the parameter is a "value," rather than the contents of a memory location, or another register, the instruction is said to be in the "immediate" mode. To put this into perspective, let us compare this with another mode.

If we want to put the contents of memory location \$102E into the accumulator, we are using the "absolute" mode of instruction:

```
LDA $102E
```

The assembler can distinguish between the two different modes because the latter does not have a "#" before the parameter. The 6502 microprocessor can distinguish between the *immediate mode* and the *absolute mode* of the LDA instruction because they have slightly different tokens. LDA (immediate) has \$A9 (as stated previously), and LDA (absolute) has \$AD.

The mnemonic representing an instruction usually implies what it does. For instance, if we consider another instruction, "LDX," what do you think this does?

If you said "load the X register with . . .", go to the top of the class.

If you didn't, then don't worry; learning machine language does take patience, and cannot be accomplished in a day.

The various internal registers can be thought of as special memory locations, because they too can hold one byte of information. It is not necessary for us to explain the binary numbering system (base 2) since it follows the same rules as outlined for hexadecimal and decimal outlined previously, but one "bit" is one binary digit and eight bits make up one byte.

The maximum number that can be contained in a byte is the largest number that an eight digit binary number can be. This number is 11111111 (binary), which equals \$FF (hexadecimal), which equals 255(decimal). You have probably wondered why only numbers from zero to two hundred and fifty-five could be put into a memory location. If you try POKE 7680,260 (which is a BASIC statement that "says":—"Put the number two hundred and sixty into memory location seven thousand, six hundred and eighty." The BASIC interpreter knows that only numbers 0 – 255 can be put in a memory location, and your VIC 20 will reply with:

```
?ILLEGAL QUANTITY  
ERROR  
READY.
```

If the limit of one byte is \$FF (hex), how is the address parameter in the absolute instruction "LDA \$102E" expressed in memory?

Well, it is expressed in two bytes (it won't fit into one, of course). The lower (rightmost) two digits of the hexadecimal address form the "low byte" of the address, and the upper (leftmost) two digits form the "high byte."

The 6502 requires any address to be specified with its low byte first, and then the high byte. This means that the instruction "LDA \$102E" is represented in memory by the three consecutive values:

```
$AD, $2E, $10
```

We need to know one more instruction before we can write our first program. That instruction is "BRK." For a full explanation of this instruction, refer to M.O.S 6502 Programming Manual. You can think of it as the "END" instruction in machine language.

If we write a program with VICMon and put the BRK instruction at the end, the program will return to VICMon when it is finished. This might not happen if there is a mistake in your program, or if the BRK instruction is never reached (just like an "END" statement in BASIC may never get executed, and thus if the VIC 20 didn't have a STOP key, you wouldn't be able to abort your BASIC programs!)

WRITING YOUR FIRST PROGRAM

If you have used the POKE statement in BASIC to put characters onto the screen, you will be aware that the character codes for POKEing are different to CBM ASCII character values. For example, if you enter:

PRINT ASC("A") (and press <RETURN>)

The VIC 20 will respond with:

65
READY.
*

However, to put an "A" onto the screen by POKEing, the code is 1. Since the screen memory starts at 7680 (decimal), or 4096 if you have 8K or more of expansion memory, by entering:

<CLR> (To clear the screen)
POKE 7680,1 (and <RETURN>) (NOTE: POKE 4096,1 on a VIC 20 with 8K or more of expansion memory)

The "P" in the POKE statement should now be an "A." We will now do this in machine language. Type the following in VICMon:
(Your cursor should be *flashing alongside a ":" right now.*)
.A 1400 LDA #\$01 (and press <RETURN>)

The VIC will prompt you with:

.A 1402 *

Type:

.A 1402 STA \$1E00 (or STA \$1000 on a VIC 20 with 8K or more of expansion memory)

The STA instruction stores the contents of the accumulator in a specified memory location. The VIC will now prompt you with:
.A 1405 *

Now enter:

.A 1405 BRK

Clear the screen, and type:

G 1400

The G should turn into an "A" if you have done everything correctly. You have now written your first machine language program! Its purpose is to store one character, the letter A, in the first byte of screen memory.

ADDRESSING MODES

ZERO PAGE

As shown earlier, absolute addresses are expressed in terms of a high order and a low order byte. The high order byte is often referred to as the page of memory. For example, the address \$1637 is in page \$16 (22), and \$0277 is in page \$02 (2). There is, however, a special mode of addressing known as "zero page" addressing and it is, as the name implies, associated with the addressing of memory locations in page zero. These addresses have a high order byte of zero. The zero page mode of addressing only expects one byte to describe the address, rather than two when using an absolute address, which saves speed and time. This mode tells the microprocessor to assume that the high order address is zero. Therefore zero page addressing can reference memory locations whose addresses are between \$0000, and \$00FF.

THE STACK

The 6502 microprocessor (like almost all others) has what is known as a "stack." This is used both by the programmer and the microprocessor to temporarily remember things, and to remember the order of events. The GOSUB statement in BASIC, which allows the programmer to call a 'subroutine,' must remember where it is being called from. When the RETURN statement is executed in the subroutine, the BASIC interpreter "knows" where to go back in order to continue executing. When a GOSUB statement is encountered in a program by the BASIC interpreter, the BASIC interpreter "pushes" its current position onto the stack before going to do the subroutine, and when a RETURN is executed, the interpreter "pulls" from the stack the information that tells it where it was before the subroutine call was made, so that it may continue as if nothing had happened. The interpreter uses instructions like PHP which will push the contents of the accumulator onto the stack, and PLA (the inverse) which will pull a value off the stack into the accumulator. The status register can also be pushed and pulled with the PHP, and PLP respectively.

The stack is 256 bytes long, and is located in page one of memory. It is therefore from \$0100 to \$01FF. It is organized

backwards in memory, i.e., the first position in the stack is at \$01FF, and the last is at \$0100. Another register in the 6502 microprocessor that hasn't been mentioned yet is called the "stack pointer," and it always points at the next available location in the stack. When something is pushed onto the stack, it is placed where the stack pointer points to, and the stack pointer is moved down to the next position (decremented). When something is pulled off the stack, the stack pointer is incremented, and the byte pointed to by the stack pointer (at \$0100 offset by the contents of the stack pointer) is placed into the specified register.

Up to this point, we have covered immediate, zero page, and absolute mode instructions. We have also covered (but have not stated) the "implied" mode, which means that the instruction itself tells what registers/flags/memory the instruction is referring to. The examples we have seen are PHA, PLA, PHP, and PLP, which refer to stack processing and the accumulator and status registers.

The X register will be referred to as X from now on, and similarly with A – accumulator, Y – Y index register, S – stack pointer, and P – processor status).

INDEXING

Indexing plays an extremely important part in the running of the 6502 microprocessor. It can be defined as "creating an actual address from a base address plus the contents of either the X or Y index registers."

For example, if X contains \$05, and the microprocessor executes an LDA instruction in the "absolute X indexed mode" with base address, e.g., \$9000, then the actual location that is loaded into the A register is \$9000 + \$05 = \$9005. The mnemonic format of an absolute indexed instruction is the same as an absolute instruction except a "X" or "Y" denoting the index is added to the address, e.g.:

```
LDA $9000,X
```

INDIRECT INDEXED ADDRESSING

This mode allows the program to choose a memory location from 256 adjacent locations. The address of the lowest location is stored in zero page, and the value in the Y register is added to that address to choose the final address.

For example, we will place a \$45 in location \$01, and a \$1E in location \$02. We will use the instruction to load the accumulator in the indirect indexed mode, specifying zero page address \$01 as the

location where the address to be used is held. Then the actual address will be comprised of:

```
low address byte = contents of $01 = $45
high address byte = contents of $02 = $1E
Y register = $10
```

The actual address = \$1E45 + Y = \$1E55

If you think of indexed addressing like delivering junk mail through a post office, here is the principle for *indirect indexed addressing*:

We will deliver the letters to all the houses on the block starting at \$1E00 Memory St. and continuing for 256 houses. Here is the equivalent program for VICMOn:

• A 1200 LDA #\$00	load low order actual base address
• A 1202 STA \$01	set the low byte of the first indirect address
	set the low byte of the second address
	load high order indirect address
	set the high byte of the first indirect address
	load the second address's high byte
	set the high byte of the second address
	set the indirect index (Y)
	66 is the value of our "letter" to the first "block"
• A 1204 STA \$FE	store the "letter" in the house on the first "block"
• A 1206 LDA #\$1E	0A is the value of our second "letter"
• A 1208 STA \$02	store the "letter" in the house on the second "block"
• A 120A LDA #\$96	add 1 to index
• A 120C STA \$FF	branch back & send next letter
• A 120E LDY #\$00	return to VICMOn when done
• A 1210 LDA #66	sends the "letter"—fills the top of the screen with blue & red lines!
• A 1212 STA (\$01),Y	
• A 1214 LDA #\$0A	
• A 1216 STA (\$FE),Y	
• A 1218 INY	
• A 1219 BNE \$1210	
• A 121B BRK	
• G 1200	

INDEXED INDIRECT ADDRESSING

This mode allows the program to choose an address from a table in page zero. Since page zero space is limited to 256 bytes, this is a mode that isn't used too often.

This mode only works with the X register. It is like *indirect indexed*, except that the zero page location is indexed, rather than an address stored in zero page. Therefore, the address stored in

page zero is the actual address because the index has already been used in the indirection.

Let us fill location \$05 with \$45, and location \$06 with \$1E. If the instruction to load the accumulator in the indexed indirect mode is executed and the specified zero page address is \$01, then the actual address will be comprised of:

```
low order = contents of ($01 + X)
high order = contents of ($02 + X)
X register = $04
```

Thus the actual address will be in = \$01 + X = \$05

Therefore, the actual address will be the indirect address contained in \$05 and \$06 which is \$1E45

This is like sending a mailing to a specific list of addresses. We will store a list in zero page, and send the "letter" only to those in the list. Suppose the list of addresses starts at \$00. Here is a program to send a "letter" to one of the addresses:

```
LDA #$00           —load low order actual base address
STA $06             —set the low byte of the indirect address
LDA #$16           —load high order indirect address
STA $07             —set the high byte of the indirect address
LDX #$06           —set the indirect index (X)
LDA ($00,X)         —load indirectly indexed by X.
```

BRANCHES AND TESTING

Another very important principle in machine language is the ability to test, and detect certain conditions, in a similar fashion to the "IF...THEN" structure in VIC BASIC.

The various "flags" in the status register are affected by different instructions in different ways. For example, there is a flag that is set when an instruction has caused a zero result, and is reset when a result is not zero.

```
LDA #$00
```

This instruction will cause "the zero result flag" to be set, because the instruction has resulted in the accumulator containing a zero.

There is a set of instructions that will, given a particular condition, "branch" to another part of the program. An example of a branch instruction is "BEQ", which means "branch if result equal to zero." The branch instructions will "branch" if the condition is true, and if not, the program will continue onto the next instruction, as if nothing had occurred. The branch instructions branch not by the result of

the previous instruction(s), but by internally examining the status register.

As was just mentioned, there is a "zero result" flag in the status register. The "BEQ" instruction branches if the "zero result" flag (known as "Z") is set. Every branch instruction has an opposite branch instruction. The BEQ instruction has an opposite instruction "BNE" ("branch on result NOT equal to zero," i.e., "Z" not set). The index registers have a number of associated instructions which modify their contents. For example, the "INX" instruction will "increment the X index register." If the X register contained \$FF before it was incremented (the maximum number the X register can contain), it will "wrap around" back to zero. If we wanted a program to continue to do something until we had performed the increment of the X index that pushed it around to zero, we could use the BNE instruction to continue "looping" around, until X became zero. Apart from INX, there is "DEX", which will decrement the X index register. If it is zero, it will wrap around to \$FF. Similarly, there are "INY" and "DEY" for the Y index register.

But what if a program didn't want to wait until X or Y had reached (or not reached) zero? Well there are comparison instructions, "CPX" and "CPY", which allow the machine language programmer to test the index registers with specified values, or even the contents of memory locations. If we wanted to see if the X register contained \$40, we would use the instruction:

```
CPX #$40           compare X with the "value" $40.
BEQ (some other      branch to somewhere else in the program, if
part of the          this condition is "true."
program)
```

The compare and branch instructions play a major part in any machine language program.

The operand specified in a branch instruction when using VICMon is the address of the part of the program the branch should go to, if taken. However, the operand is only, in fact, an "offset" from where the program currently is, to the address specified. This offset is just one byte, and therefore the range that a branch instruction can branch to is limited from 128 bytes backward, to 127 bytes forward; this is a total range of 255 bytes, which is, of course, the maximum range of values one byte can contain. VICMon will tell you if you branch out of range, by refusing to "assemble" that instruction. It is unlikely that you will be doing such huge branches for quite a while anyway. For nearly all situations this is adequate anyway. The branch is a "quick" instruction by machine language standards because of this "offset" principle as opposed to an

the previous instruction(s), but by internally examining the status register.

absolute address. VICMon allows you to type in an absolute address, and it calculates the correct offset. This is just one of the "comforts" of using an assembler.

Subroutines

In machine language (in the same way as using BASIC), you can call subroutines. The instruction to call a subroutine is "JSR" (jump to subroutine), followed by the specified absolute address. Incorporated in the operating system is a machine language subroutine that will PRINT a character to the screen. The CBM ASCII code of the character should be in the accumulator before calling the subroutine. The address of this subroutine is \$FFD2. Therefore, to print "HI" to the screen, the following program should be entered:

- A 1400 LDA #\$48 load the CBM ASCII code of "H"
- A 1402 JSR \$FFD2 print it
- A 1405 LDA #\$49 load the CBM ASCII code of "I"
- A 1407 JSR \$FFD2 print that too
- A 140A LDA #\$0D print a carriage return as well
- A 140C JSR \$FFD2 return to VICMon.
- G 140F BRK will print "HI" and return to VICMon

The "PRINT a character" routine we have just used is part of the KERNAL "jump table." The instruction similar to GOTO in BASIC, is "JMP," which means "jump to the specified absolute address." The KERNAL is a long list of "standardized" subroutines that control ALL input and output of the VIC 20. Each entry in the KERNAL JMP's to a subroutine in the operating system. This "jump table" resides at \$FF84 to \$FFFF in the operating system. A full explanation of the KERNAL is in the "KERNAL REFERENCE SECTION" in this manual, but certain routines will be used here to show how easy, and effective, the KERNAL is.

We will now use these new principles in another program which will help you to put these instructions into context:

This program will display the alphabet using a KERNAL routine. The only new instruction introduced here is TXA "transfer the contents of the X index register, into the accumulator."

- A 1400 LDX #\$41 X = CBM ASCII of "A".
- A 1402 TXA A = X.
- A 1403 JSR \$FFD2 print character.
- A 1406 INX bump count.
- A 1407 CPX #\$51 have we gone past "Z" ?

- A 1409 BNE \$1402 no—go back and do more.
- A 140B BRK yes—return to VICMon.

To see the VIC print the alphabet, type the familiar command:

G 1400

The comments that are beside the program explain the program flow, and logic. If you are writing a program, write it on paper first, and test it in small parts if possible.

DIVERSE DIVERSE

DIVERSE

INFO BOEK

Bij de VICKIES van de vorige keer ontbrak het adres van INFO BOEK ZEELAND. Voor informatie over de software kunt u kontakt opnemen met :

INFO BOEK ZEELAND, POSTBUS 12, 4356 ZH Oostkapelle.

Hier kunt u ook informatie krijgen over drie nieuwe programma's die INFO BOEK aan zijn lijst van programma's heeft toegevoegd.

Dit zijn :

- Debiteuren rekening en bewaking.
- Factureringsprogramma.
- Database.

PROGRAMMA BEURS

De rubriek PROGRAMMA BEURS komt deze keer, vanwege herorganisatie van dit gebeuren, te vervallen. Wel worden leden verzocht als zij ideeen hebben over onze programma beurs deze aan de redactie te sturen zodat we deze in de nieuwe opzet kunnen meenemen. We hopen op jullie medewerking.

VIC COMPUTING

Een groot deel van de leden van de VIC GEBRUIKERS club hebben bij hun eerste nummer ook een aflevering van VIC COMPUTING, een engels tijdschrift, ontvangen met de mededeling dat de volgende nummers van dit blad ook toegezonden zouden worden. Helaas voordat wij alle leden van een exemplaar hebben kunnen voorzien, is VIC COMPUTING overgenomen door een onafhankelijk bedrijf, sterker nog het bestaat niet eens meer in zijn originele vorm. Het is dus, tot onze spijt, niet meer mogelijk dit blad rechtstreeks van COMMODORE te betrekken. Op dit moment zijn er nog onderhandelingen gaande met de nieuwe uitgever. Wij zullen echter pas in de volgende VIC PRIMEURS kunnen meedelen, hoe deze onderhandeling zijn afgelopen en wat voor oplossing wij hebben kunnen vinden.

DE HOOGSTE SCORE.

Het is waanzinnig te weten, dat zoveel mensen wereldrecords van videospellen verbeteren, zonder er daadwerkelijk bij stil te staan hoeveel uren (soms zelfs dagen of weken) men moet oefenen om de geavanceerde spellen onder de knie te krijgen. De aandacht, die hier door de verschillende media aan besteed wordt is relatief zeer laag, en in nederland niet eens aanwezig. Toch zal het niet lang meer duren of ook wij worden wakker en zullen ontdekken dat een nieuwe industrie is opgestaan. De industrie van SPACE INVADERS, PACMAN, DEFENDER en de DONKEY KONG.

Iedere mikro heeft wel de mogelijkheid om meerdere van deze spellen in de huiskamer te spelen en dat zijn er inmiddels toch al wat. De scores, die in de huiskamer gerealiseerd zijn, blijken vele malen hoger dan de onderstaande scores, die uit de amusementshallen zijn geregistreerd.

Goed, stroop je mouwen op en maak wat uren vrij.

Amusementshaltitel	VIC-20 Titel	Hoogste score
DEFENDER	DEFENDA	33.000.013.200
DONKEY KONG	APEMAN	270.000
PAC-MAN	JELLY MONSTERS	5.579.350
GALAXIAN	BATTLE STAR	134.900
SPACE INVADERS	AVENGER	33.000.000
BERZERK	AMOK	59.770
MOUSETRAP	RADAR RATRACE	652.039
ASTEROIDS	verschillende	20.307.890
ALIEN	ALIEN	64.980

Zo, je ziet de amerikanen (daar komen deze scores vandaan) hebben niet stil gezeten.

Bij de bovenvermelde lijst willen we graag ook de nederlandse scores gaan bijhouden. Sterker nog, mocht je een van de bovenstaande scores verbeteren, maak hier dan een foto van en win het gloednieuwe COMMODORE moederbord ter waarde van FL. 648,-. Stuur deze foto naar: HANDIC BENELUX B.V.

POSTBUS 213
1850 AE HEILOO

Met de volgende editie van VIC primeurs zullen we de namen van de nationale recordhouders gaan publiceren en prijzen uitreiken op verbeteringen van de verschillende scores. Stuur dus nu al jouw hoogste score in naar het bovenstaande adres.

BASICODE WIJZIGING

Hieronder de wijziging van twee programmaregels van het BASICODE LAAD programma. Dit programma gaf soms problemen. De wijzigingen moet u dus aanbrengen in de listing op blz. 43 van VIC PRIMEURS 1 van dit jaar.

```
560 DATA"22D000A0D26H149208D26H1A922C920D005CD26H1D90A2027H1EAEEAEAE201
```

```
590 DATA"8D8D7A02A9048506605359530000000018C960900318E91F60","*****"
```

Programmalistings

Van de heer C. KET uit GOUDA ontvingen wij twee programma's. De eeste is een spelprogramma voor twee spelers en werkt op iedere geheugengrootte. In het programma zelf bevindt zich de gebruiksaanwijzing. De naam van het spel is 'MANCALA'. Het tweede programma is uitsluitend voor de floppy disk gebruikers. Het is niet alleen een verbeterde versie wat betreft uitlezing van het programma 'DISPLAY T&S' van de systeem diskette. Er is ook een fout uitgehaald die goede werking van het programma onmogelijk maakte. Let op REGEL 400, hier waren de parameters van "B-P" onjuist in de orginele listing. Op deze manier een zeer bruikbaar programma voor floppy disk gebruikers. Daar het programma 'uit de ruillijst' deze keer vervallen is stelt de redactie het ROM-spel beschikbaar voor de heer KET.


```

800 J=12:FOR I=7TO12:BP=B(I):BF=INT(BP/10):IF BF=0 THEN BF=-16
810 POKESC+69+3*I, BP+48:BP=BP-INT(BP/10)*10
820 POKESC+91+3*I, BP+48:J=J-1:NEXTI
830 FOR I=1TO6:BP=B(I):BF=INT(BP/10):IF BF=0 THEN BF=-16
840 POKESC+175+3*I, BP+48:BP=BP-INT(BP/10)*10
850 POKESC+197+3*I, BP+48:NEXTI
860 RETURN
900 FOR I=1TO12:T(I)=B(I):NEXT
910 T(M)=0:LP=M
920 FOR I=1TOB(M):LP=LP+1:IF LP=13 THEN LP=1
930 IF LP=M THEN LP=LP+1
940 IF LP=13 THEN LP=1
950 T(LP)=T(LP)+1:NEXTI
960 IF LP>(PL+1)*6 THEN 1020
970 IF LP<PL*6+1 THEN 1020
980 IF T(LP)<20 RT(LP)>3 THEN 1020
990 SC(PL)=SC(PL)+T(LP):T(LP)=0
1000 LP=LP-1:IF LP=0 THEN LP=12
1010 GOTO 960
1020 IF SC(PL)<=0 THEN 1050
1030 PRINT"*****";:IF PL=1 THEN PRINT"*****";
1040 PRINT"*****MID$(STR$(SC(PL)),2)
1050 RETURN
1100 PRINT"*****" EINDE SPEL!"
1110 FOR W=1TO5000:NEXT
1120 PL=0:IF SC(1)>SC(0)THEN PL=1
1130 PRINT"*****HET SPEL IS AFGELOPEN!"
1140 IF SC(1)=SC(2)THEN PRINT"het is een gelijk spel geworden (beiden"SC(0)" en "SC(1)
":GOTO 1160
1150 PRINT"PROFICIAT, ";N$(PL);":!":PRINT"JIJ HEBT GEWONNEN MET"SC(PL)"TEGEN"SC(1
-PL)"!! *****"
1160 PRINT"*****":END
2000 POKE36869,PEEK(36869)OR2
2010 PRINT"IT SPEL WORDT DOOR 2 SPELERS GESPEELD."
2020 PRINT"ELKE SPELER HEEFT ZES TUINTJES."
2030 PRINT"AN HET BEGIN VAN HET SPEL LIGGEN IN ALLE TUINTJES VIER ZAADJES."
2040 PRINT"FM DE BEURT MOET ELKE SPELER VANUIT EEN VAN ZIJN TUINTJES DE ZAAD-"
2050 PRINT"JES VERDELLEN OVER ALLETUINTJES (IN EEN RICH-TING TEGEN DE KLOK)."
2060 PRINT"OMT MEN BIJ HET ROND-GAAN WEER LANGS HET TUINTJE VAN WAARUIT"
2070 PRINT"WERD BEGONNEN, DAN WORDT DIT TUINTJE NU OVERGESLAGEN.
2080 GOSUB3000
2090 PRINT"OMT HET LAATSTE ZAAD-JE IN EEN TUINTJE VAN DE TEGENSTANDER WAARIN"
2100 PRINT"DAN 2 OF 3 ZAADJES LIGGEN, DAN WORDT ER GESCOORD !"
2110 PRINT"JET AANTAL PUNTEN IS GELIJK AAN HET AANTAL ZAADJES VAN HET LAAT-"
2120 PRINT"STE TUINTJE EN DE VOORGRANDE TUINTJES INDIENDAARIN STEEDS AANSLUI-TEN
D";
2130 PRINT" OOK 2 OF 3 ZAAD-JES LIGGEN.!!"
2140 PRINT"K HOOP, DAT JE HET NUBEGRIPPT !"
2150 GOSUB3000
2160 POKE36869,PEEK(36869)AND253:RETURN
3000 PRINT"***** EEN TOETS IN"
3010 GETX$:IF X$="" THEN 3010
3020 RETURN

```



Programmalisting

```
100 REM*****  
110 REM* DISPLAY ANY TRACK & SECTOR *  
120 REM* ON THE VIC TO THE SCREEN *  
130 REM* OR THE VIC PRINTER *  
140 REM*****  
150 PRINT"100"  
160 PRINT"DISPLAY BLOCK CONTENTS";  
165 PRINT"  
170 REM*****  
180 REM* SET PROGRAM CONSTANT *  
190 REM*****  
200 SP$="" :NL$=CHR$(0) :HX$="0123456789ABCDEF"  
210 FS$="" :FOR I=64T095:FS$=FS$+" "+CHR$(I)+" ":"NEXTI  
220 SS$="" :FOR I=192T0223:SS$=SS$+" "+CHR$(I)+" ":"NEXTI  
240 DIM A$(15),NB(1)  
251 D$="0"  
253 PRINT"SCREEN OR PRINTER"  
254 GET JJ$: IF JJ$=" " THEN 254  
255 IF JJ$="S" THEN PRINT"SCREEN": GOT0265  
256 IF JJ$="P" THEN PRINT"PRINTER": GOT0260  
258 GOT0254  
260 OPEN4,4,0  
265 OPEN15,8,15,"I"+D$: GOSUB650  
270 OPEN2,8,2,"#": GOSUB650  
280 REM*****  
290 REM* LOAD TRACK AND SECTOR *  
300 REM* INTO DISK BUFFER *  
310 REM*****  
320 FOR I=1T050: GET A$: NEXT  
322 INPUT"TRACK "; T$: T=VAL(T$): IFT=0 THEN 330  
325 INPUT"SECTOR "; S$: S=VAL(S$)  
330 IFT=0 OR T>35 THEN PRINT#15,"I"D$: CLOSE2:CLOSE4:CLOSE15:PRINT"END": EN  
D  
340 IF JJ$="S" THEN PRINT"TRACK "T" SECTOR "S"  
341 IF JJ$="P" THEN PRINT#4: PRINT#4,"TRACK "T" SECTOR "S: PRINT#4  
350 PRINT#15,"U1:2,"D$: T:S: GOSUB650  
360 REM*****  
370 REM* READ BYTE 0 OF DISK BUFFER *  
390 REM*****  
400 PRINT#15,"B-P:2,0"  
410 PRINT#15,"M-R"CHR$(0)CHR$(6)  
420 GET#2,A$(0): IF A$(0)="" THEN A$(0)=NL$  
425 K=1: NB(0)=ASC(A$(0))  
430 IF JJ$="P" THEN 460  
431 REM*****  
432 REM* READ & CRT DISPLAY *  
433 REM* REST OF THE DISK BUFFER *  
434 REM*****  
438 FOR J=0T063: IF J=32 THEN GOSUB710: IF Z$="N" THEN J=80: GOT0458  
440 FOR I=KTO3  
442 GET#2,A$(I): IF A$(I)="" THEN A$(I)=NL$  
444 IF K=1 AND I<2 THEN NB(1)=ASC(A$(I))  
446 NEXT I: K=0
```

```

448 A$="" : B$="" : N=J*4:GOSUB790:A$=A$+" "
450 FORI=0TO3:N=ASC(A$(I)):GOSUB790
452 C$=A$(I):GOSUB850:B$=B$+C$
454 NEXTI:PRINTA$" "+B$
455 NEKTJ:GOTO571
456 REM*****N*****N*****N*****N*****N*****N*****N*****
457 REM* READ & PRINTER DISPLAY *
458 REM*****N*****N*****N*****N*****N*****N*****N*****
459 FORJ=0TO15
460 FORI=KTO15
461 GET#2,A$(I):IFA$(I)=="THENA$(I)=NL$
462 IFK=1ANDI<2THENNB(1)=ASC(A$(I))
463 NEXTI:K=0
464 A$="" : B$="" : N=J*16:GOSUB790:A$=A$+" "
465 FORI=0TO15:N=ASC(A$(I)):GOSUB790:IFZ$="N"THENJ=40:GOTO571
466 C$=A$(I):GOSUB850:B$=B$+C$
467 NEXTI
468 PRINT#4,A$B$
469 NEKTJ
470 REM*****N*****N*****N*****N*****N*****N*****N*****
471 REM* NEXT TRACK AND SECTOR *
472 REM* NEXT TRACK AND SECTOR *
473 REM*****N*****N*****N*****N*****N*****N*****N*****
474 PRINT"NEXT TRACK AND SECTOR IS "+MID$(STR$(NB(0)),2)+": "+MID$(STR$(NB(1)),2)+""
475 PRINT"DO YOU WANT NEXT TRACK AND SECTOR (Y/N)""
476 GETZ$:IFZ$=="N"THEN590
477 IFZ$="Y"THENT=NB(0):S=NB(1):GOTO330
478 IFZ$="N"THEN320
479 GOTO590
480 REM*****N*****N*****N*****N*****N*****N*****N*****
481 REM* SUBROUTINES *
482 REM*****N*****N*****N*****N*****N*****N*****N*****
483 REM* ERROR ROUTINE *
484 REM*****N*****N*****N*****N*****N*****N*****N*****
485 INPUT#15,EN,EM$,ET,ES:IFEN=0THENRETURN
486 PRINT"DISK ERROR:"EN,EM$,ET,ES
487 END
488 REM*****N*****N*****N*****N*****N*****N*****N*****
489 REM* SCREEN CONTINUE MSG *
490 REM*****N*****N*****N*****N*****N*****N*****N*****
491 PRINT"CONTINUE(Y/N)""
492 FORCL=1TO10:GETZ$:NEXT
493 GETZ$:IFZ$=="N"THEN750
494 IFZ$="Y"THENRETURN
495 IFZ$<>"Y"THEN750
496 PRINT"DISKTRACK"+T+" SECTOR"+S+""
497 REM*****N*****N*****N*****N*****N*****N*****N*****
498 REM* DISK BYTE TO HEX PRINT *
499 REM*****N*****N*****N*****N*****N*****N*****N*****
500 A1=INT(N/16):A$=A$+MID$(HX$,A1+1,1)
501 A2=INT(N-16*A1):A$=A$+MID$(HX$,A2+1,1)
502 A$=A$+SP$:RETURN
503 REM*****N*****N*****N*****N*****N*****N*****N*****
504 REM* DISK BYTE TO ASC DISPLAY *
505 REM* CHARACTER *
506 REM*****N*****N*****N*****N*****N*****N*****N*****
507 IFASC(C$)<32THENC$=" ":"RETURN
508 IFASC(C$)<128ORASC(C$)>159THENRETURN
509 C$=MID$(SS$,3*(ASC(C$)-127),3):RETURN

```

FLOPPY-DISK_HULPPROGRAMMA

De programma listing die hieronder afgedrukt staat bevat twee hulpprogramma's

Het eerste gedeelte (regel 10-1299) is een programma wat het mogelijk maakt 3 sequentiele write files te openen. Het is niet mogelijk 5 files tegelijkertijd te openen, zoals waarschijnlijk foutief in de handleiding vermeld staat. Als u 3 write- (schrijf-) files tegelijkertijd geopend wilt hebben gaat dat niet zonder meer. Dit programma laat zien hoe het wel kan. Het geeft meteen een voorbeeld en het leest ook het fout-kanaal af (regel 7000-7100). Het tweede gedeelte van dit programma laat zien hoe fouten met het SAVEn van programma's voorkomen kunnen worden en het bespaart typewerk. RUN 10000 zorgt ervoor dat het programma netjes naar disk wordt geschreven, over de oude versie heen (het werken met het 'at' of 'a met een rondje' teken levert met het DOS van de disk nogals eens problemen op, zodat deze methode de voorkeur heeft). Het programma is ingezonden door de heer STOMP uit Sassenheim, met uiteraard veel dank voor het uitgebreide gepuzzel dat aan dit programma vooraf is gegaan.

```
10 PRINT"DISK TEST"
20 OPEN15,8,15,"I":PRINT"I":GOSUB7100
30 PR$="A":N=2:M$="W":GOSUB1000
40 PR$="B":N=3:M$="W":GOSUB1000
50 PR$="C":N=5:M$="W":GOSUB1000
55 CLOSE2:CLOSE3:CLOSE5
60 PR$="A":N=2:M$="R":GOSUB1200
70 PR$="B":N=3:M$="R":GOSUB1200
80 PR$="C":N=5:M$="R":GOSUB1200
90 CLOSE2:CLOSE3:CLOSE5
99 CLOSE15:QUIT010000
1000 PRINTPR$:PRINT#15,"S:"+PR$:GOSUB7100
1010 OPENN,8,N,PR$+":S,"+M$:PRINT"O ";M$):GOSUB7100
1020 CLOSEN
1040 GOSUB1100
1050 RETURN
1100 OPENN,8,N,PR$+":S,R":PRINT"O R":GOSUB7100
1110 FORR=1TO10
1120 PRINT#N,STR$(R):GOSUB7000:PRINTR
1130 NEXT
1140 RETURN
1200 PRINTPR$:
1210 OPENN,8,N,PR$+":S,"+M$:PRINT"O ";M$):GOSUB7100:E=0
1220 INPUT#N,I$:IFST=64THENE=1
1230 GOSUB7000
1240 PRINTI$:
1250 IFE1THEN1220
1260 RETURN
7000 INPUT#15,EN,EM$,ET,ES:IFEN<20THENRETURN
7010 PRINTEN:EM$:ET:ES:STOP:RETURN
7100 INPUT#15,EN,EM$,ET,ES:PRINTEN:EM$:ET:ES:IFEN<20THENRETURN
7110 STOP:RETURN
9999 OPEN15,8,15:GOSUB7100:CLOSE15:END
10000 PR$="DISK TEST":PRINTPR$:OPEN15,8,15,"I":PRINT"I":GOSUB7100
10010 PRINT#15,"S:"+PR$:GOSUB7100
10020 SAVE PR$,8:GOSUB7100
10030 VERIFY PR$,8:GOSUB7100
10040 CLOSE15:END
```

Nieuw

Het VIC-1020 moederbord.

Er bestaan veel verschillende benamingen voor de VIC-1020 zoals uitbreidingsbord of moederbord, maar als je hem gezien hebt dan is het meer dan zo maar een moederbord.

Het is een heel complete kast met daarin een print welke plaats biedt aan 6 RAM/ROM packs.

De VIC kan er op eenvoudige wijze ingeschoven worden via de geheugenuitbreidingsbus. Wel moet er op gelet worden dat zowel de VIC-20 als het moederbord uitgeschakeld zijn. Is de VIC erin geschoven tot de voorkant van de computer gelijk is met de rand van de kast en de kap is gesloten, dan zijn uw VIC en de VIC-1020 een heel mooi geheel geworden.

Maar nu iets meer over de konstuktie.

Zoals al uit het voorgaande blijkt, wordt het moederbord compleet met een bovenplaat geleverd. Deze bovenplaat is d.m.v. een scharnier bevestigd aan de kast zodat dit een mooi en robuust sluitend geheel is. Achterin de bovenplaat is een plaats gemaakt om de modulator te kunnen bevestigen. Deze wordt in een beugel ingeklemd.

In de achterkant van de bovenplaat zit een gaatje waaruit de coaxiale aansluiting van de modulator komt, welke met het kabeltje naar de TV gaat.

De print met de 6 uitbreidingsbussen heeft in de kast plaats voor 5 RAM/ROM packs en er is tevens aan de achterkant een zesde aansluitbus

Nieuw



welke heel gemakkelijk met een gesloten kast gebruikt kan worden.

Verder zitten er op de print de buffer IC's en een spanningsstabilisatie IC om de RAM/ROM packs van spanning te kunnen voorzien.

Om zowel de VIC-20 als het moederbord van spanning te voorzien wordt er gebruik gemaakt van de bij de computer geleverde transformator.

Het verschil is echter dat de plug van de transformator niet in de VIC maar in het moederbord gestoken wordt. Uit het moederbord komt een kort kabeltje welke dan weer in de VIC gestoken dient te worden zodat deze ook van spanning wordt voorzien.

Misschien bestaat er enige twijfel of de transformator dit allemaal wel zou kunnen voeden, maar na enige tijd met het gehele systeem gewerkt te hebben vervalt deze twijfel geheel. De transformator wordt nauwelijks warmer dan wanneer deze alleen met de computer zelf gebruikt wordt.

Tot zover laten we even de uitvoering van het moederbord rusten en gaan we verder in op de uiteindelijke toepassing van het moederbord.

Het moederbord breidt in zekere zin het aantal geheugenuitbreidingsbussen uit van 1 op een standaard VIC-20 naar 6 met de VIC-20 en de VIC-1020.

Door nu de beschikking te hebben over deze 6 uitbreidingsbussen kunt U uw computer nog veel "krachtiger" maken door verschillende RAM/ROM packs tegelijkertijd te gebruiken. Eerst kon U bijvoorbeeld het RAM geheugen van de VIC uitbreiden tot 19 K door de grootste RAM uitbreiding (16 K) te gebruiken, maar nu met het moederbord kan de VIC tot zelfs 32 K RAM uitgebreid worden. Dit is mogelijk door gebruik te maken van een 3 K, een 8 K en een 16 K RAM pack.

Opgemerkt dient te worden dat de VIC bij het inschakelen dan "28159 bytes free" aangeeft. Dit is het beschikbare RAM geheugen dat voor Basic-programma's gebruikt kan worden.

Het 3 K gebied, dat door de Basic overgeslagen wordt, kan gebruikt worden voor bijvoorbeeld machinetaalsubroutines.

Alle RAM/ROM packs, m.u.v. 8 K RAM, hebben een vaste plaats in het adresbereik van de VIC-20. Bij de 8 K RAM is het mogelijk het adresbereik d.m.v. dipswitches te veranderen (zie gebruiksaanwijzing VIC-1110). Het is niet mogelijk om twee packs met dezelfde of gedeeltelijk dezelfde adresplaats tegelijkertijd te gebruiken.

Om een overzicht te geven van de adresplaatsen van de RAM/ROM packs volgt hieronder een tabel.

A	I	B	C	I	D	E
0400-0FFF	I			I		
2000-5FFF	I	6000-6FFF	7000-7FFF	I	A000-AFFF	B000-BFFF
3K RAM	I	Mach.Mon.	Prog.Aid	I	H-Res	IEEE 488
	I			I	Sup.Exp.	Interface
8K RAM	I			I		
of	I-of een extra 8K RAM--			I-----of 1 spel-----		
16K RAM	I			I		
of	I			I		
2* 8K RAM	I			I		

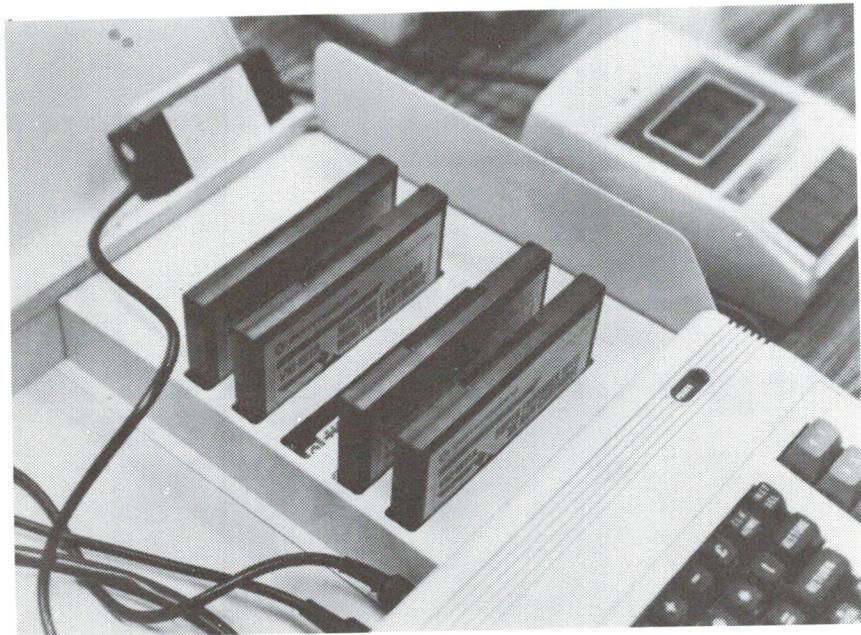
Samenvattend kunnen we stellen dat het VIC-1020 moederbord de mogelijkheid geeft om de VIC-20 maximaal uit te breiden.

Verder geeft de kast de mogelijkheid om er zelfs een monitor of TV op te plaatsen en ook bij voorbeeld de cassettereorder 1530.

Zeer goed doordacht is de zesde uitbreidingsbus aan de achterkant van het moederbord. Hierdoor is het mogelijk om met een gesloten deksel waarop bij voorbeeld een kleine TV staat een RAM/ROM pack te plaatsen of te verwijderen zoals een spelletje of de IEEE 488 interface.

De VIC-20 heeft er mijns inziens een hele mooie uitbreidingsmogelijkheid bijgekregen welke tevens zeer doeltreffend is.

Gerard Reis.



Programma prijsvraag

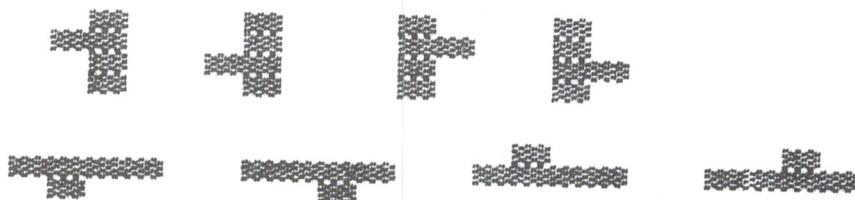
In het vorige nummer van 'VIC PRIMEURS' stond helaas geen prijsvraag, vandaar deze keer dubbel op. Dit keer geen twee moeilijkheids nivo's zoals vorige keer, vooral ook omdat veruit de meeste inzenders het moeilijke gedeelte hadden opgelost. Het eerste deel van de prijsvraag is echt voor de puzzelaars onder u, voor het tweede deel komt wat meer inzicht in de 'binnenkant' van de VIC-20 kijken met de extra voorwaarde dat er een uitbreiding op de VIC noodzakelijk is (de VIC-1211A superexpander) en een printer. Om toch zoveel mogelijk de gelegenheid te geven hem op te lossen verdubbelen we de inzend termijn zodat er volop tijd is (bijvoorbeeld om samen met iemand die de printer heeft of de VIC-1211A uitbreiding het programma te schrijven). Ook hebben wij het lijstje met voorwaarden (dat de vorige keer ontbrak) toegevoegd.

PRIJSVRAAG DEZE MAAND

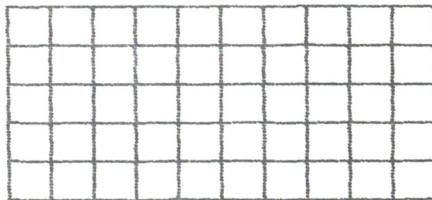
De hoofdrol in de opgave is een zogenaamde 'penta-kubus', een voorwerp bestaande uit vijf blokje, zoals hieronder staat afgebeeld :



We kunnen dit voorwerp op acht verschillende manieren neerleggen. Hoewel de schaal wel niet geheel juist is staan die acht posities hieronder afgebeeld :



Als we een vlak nemen van 5×10 vakjes kunnen we dit dus bedekken met 10 van deze penta-kubussen. Het vlak staat hier afgebeeld :



Nu de opgave van de prijsvraag :

Schrijf een programma voor de VIC-20 die minstens 1 oplossing 'bereikt' om de 10 penta-kubussen te verdelen over het vlak van 5×10 zodat dat geheel bedekt is. Het programma moet dus zowel de ligging als de positie van elk van de penta-kubussen geven. De oplossing mag niet in DATA of als variabelen in het programma voorkomen. Als het programma meer dan 1 oplossing geeft zal snelheid waarmee de eerste oplossing gevonden wordt doorslag gevend zijn in vergelijking tot ander programma. Het programma moet geschikt zijn voor een VIC-20 met 3K RAM uitbreiding of een standaard VIC-20.

PRIJS :

VIC-1212 programmers-aid of VIC rom spel NAAR KEUZE !!

De EXTRA PRIJSVRAAG.

Zoals eerder vermeld is er een VIC-1211A en VIC-1515 printer voor nodig om deze prijsvraag op te lossen. De opgave is :

Schrijf een PROGRAMMA SUBROUTINE die een beeld opgebouwd door de VIC-1211A superexpander afdrukt op de printer (in high-resolution mode). Het programma moet ook voorzien zijn van een voorbeeld-gedeelte die de subroutine demonstreert. De snelheid waarmee het programma dit doet vormt natuurlijk een belangrijk onderdeel van de beoordeling. De programma subroutine moet zowel werken voor de GRAPHIC 1, GRAPHIC 2 als GRAPHIC 3 mode. Daar er een aantal van de VIC-1211A uitbreidingen geleverd zijn zonder gebruiksaanwijzing, volgt na de lijst van voorwaarde het gedeelte van de handleiding die het grafische werken met de uitbreiding beschrijft.

PRIJS :

!! VIC-1110 8K RAM GEHEUGEN UITBREIDING !!

De voorwaarden voor beide prijsvragen :

1. De oplossing moet worden ingezonden op een cassette. De ingezonden cassettes worden nadat de uitslag bekend is teruggestuurd.
2. De inzender gaat ermee accord dat het programma mag worden afdrukkt in een aflevering van VIC PRIMEURS
3. Alleen leden van de VIC-gebruikers club komen in aanmerking voor deze prijsvraag.
4. Bij de inzending moet worden vermeld :
naam
adres
postcode & woonplaats
telefoon nummer
lid nummer
programma naam
geheugen grootte waarvoor programma geschikt is
bijzondere aanwijzingen voor gebruik

Uw naam en woonplaats moeten in een REM of PRINT in het programma voorkomen.

5. Sluitings datum prijsvraag van de maand:
11 september 1982
- Sluitings datum EXTRA prijsvraag :
6 november 1982

6. Opsturen aan :

Programma prijsvraag
VIC-gebruikersclub
POSTBUS 12972
1100 AZ Amsterdam



VIC-1211A SUPER EXPANDER

Om iedereen voor de extra prijsvraag een gelijke start te geven en om de VIC gebruikers die niet in het bezit zijn van de VIC-1211A uitbreiding een indruk te geven van de mogelijkheden drukken we hieronder een gedeelte af van de gebruiksaanwijzing van deze uitbreiding.

Het is het gedeelte dat het grafische mogelijkheden beschrijft. Laat u niet verwarren door de 1024 bij 1024 matrix, de Super Expander verhoogt niet het oplossend vermogen van de VIC-20, wel de mogelijkheden om hiervan gebruik te maken. De verdeling van de punten wordt bepaald door de GRAPHIC mode.

Naast de hieronder vernoemde mogelijkheden geeft de VIC-1211A nog meer mogelijkheden. We noemen het gebruik van de functietoetsen in 'direct mode', muziek maken met PRINT statements en opdrachten om joystick, paddle, de kleur van een bepaald punt of de inhoud van de geluids-registers af te lezen.

OPSTELLEN VAN GRAFIEKEN MET SUPER EXPANDER

3.1 Inleiding

In deel drie worden de nieuwe opdrachten om grafieken te maken beschreven die door de S.E. worden aangeboden. De opdrachten stellen u in staat punten uit te zetten, vormen te omlijnen, tekst in te voeren en in één van de 15 kleuren te "schilderen", zonder locaties in het geheugen te moeten berekenen. Deze opdrachten worden net als enig andere BASIC-opdracht ingevoerd en kunnen in directe of indirecte mode gebruikt worden. De manier waarop de VIC de door u geselecteerde kleuren in kleurenregisters vasthoudt wordt eveneens uitgelegd. De opdrachten worden gelist in de volgorde waarin ze gebruikt zijn om een programma te schrijven, zoals die in deel zes van deze handleiding.

3.2 Scherm coördinaten

Met het doel grafieken op het scherm uit te zetten is deze verdeeld in een 1024 bij 1024 matrix. Een matrix is een voudig een rooster waarin een bepaalde lokatie wordt "aangesproken" door aan zijn coördinaten te refereren. Een coördinaat is een nummer dat gebruikt wordt om de afstand vast te stellen van een punt op een rooster tot het begin- of 0-punt. Twee nummers zijn daarbij aanwezig, één die aangeeft hoever het punt verwijderd is van het begin op de horizontale as en de andere geeft de afstand aan in vertikale richting. De horizontale as wordt de x-as genoemd en de vertikale de y-as. De eerste coördinatie die in alle plotting commando's wordt vastgesteld is de X-coördinaat. In de S.E. is dit de positie van een punt op het

scherm, die betrekking heeft op de linker kant van het scherm. De tweede coördinaat is de Y-coördinaat. Deze geeft aan hoever naar beneden vanaf de bovenzijde van het scherm de punt zich bevindt. Het begin of nulpunt van het "screen"rooster in de S.E. bevindt zich in de hoek links boven in het scherm. De coördinaten mogen een waarde aannemen die ligt tussen 0 en 1023.

3.3 Kleur-registers

Wanneer de VIC wordt uitgerust met de S.E. cassette, dan worden er vier kleurenregisters toegevoegd. (Een register is een speciale geheugenlocatie die wordt gebruikt om een waarde vast te houden). Een getal van 0 tot 15 is in het register opgeslagen en wordt gebruikt om de kleur aan te geven.

De volgende waarden zijn gekoppeld aan de volgende kleuren:

0	zwart
1	wit
2	rood
3	cyaan
4	paars
5	groen
6	blauw
7	geel
8	oranje
9	licht oranje
10	rose
11	licht cyaan
12	licht paars
13	licht groen
14	licht blauw
15	licht geel

Register 0 wordt aan de waarde gekopieerd die gebruikt wordt als achtergrond van het scherm (graphic-mode). Register 1 bevat de waarde van de grenskleur.

Register 2 houdt de waarde vast van de karakters die in de graphic-mode moeten worden weergegeven op het scherm (plotting).

Register 3 heeft de waarde van de hulpkleur, deze kan ook gebruikt worden om grafische vormen op te stellen. Dit register is alleen toegankelijk in de multi-kleuren mode of mixed (samengesteld) mode, zie 3.4.1.

Beginnende kleurwaarden worden in deze registers opgenomen door het commando COLOR in te voeren (zie 3.4.2) wanneer het systeem in working is gesteld dan zijn de waarden in de registers resp. 1, 3, 6 en 0.

3.4 Grafische commando's

Alle voorbeelden van commando's gebruikt om grafieken te maken, worden in de volgende voorbeelden gegeven in de vorm van een programma-voorbbeeld. U kunt het programma opbouwen en RUNnen na iedere nieuwe regel die is ingebracht. Indien u dit doet, moet u voor u de volgende regel inbrengt de RUN/STOP toets indrukken en tegelijk de RESTORE toets indrukken.

3.4.1 GRAPHIC

Formaat: GRAPHIC n
Doel: Een grafische set te keren uit het VIC werkgeheugen.

Het commando GRAPHIC kan gebruikt worden in de direct mode en in de indirect mode. Er zijn vier verschillende graphic modes, waarvan ieder een set grafieken seleert uit de VIC. GRAPHIC 0 zet de VIC in de normal-

mode. Dit commando wordt gebruikt om van de grafische mode terug te keren naar de normale tekst-display. Het wordt gewoonlijk gebruikt aan het eind van een programma met grafieken.

GRAPHIC 1 zet de grafische multicolorset uit 't VIC systeem in werking. In deze mode worden de normale punten van de VIC op 't scherm opnieuw verdeeld in vakjes van 16 punten breed en 8 punten hoog, dit geheel binnen het scherm dat is opgedeeld in 1024 bij 1024 (zie deel 3.2).

De punten kunnen op het scherm worden geplaatst door een van de vier beschikbare kleuregisters te gebruiken (zie deel 3.3). Alle S.E. kleuren kunnen op deze wijze worden gebruikt. GRAPHIC 2 activeert hoog oplossende mode van de VIC. In deze mode wordt iedere positie binnen de normale punten op het scherm weer onderverdeeld in 8 bij 8. Hierdoor geeft deze een oplossend vermogen weer dat twee keer zo groot is als die van het graphic 1 commando. Alleen zijn de kleuren waarin deze punten worden afdrukkt op het scherm beperkt tot die, welke in de registers 0 en 2 zijn opgenomen, d.w.z. de scherm- of karakterkleuren. Alleen deze kleuren, 0 t/m 7, kunnen worden gebruikt in deze mode (zie deel 3.3). Als u een kleurwaarde opgeeft die hoger ligt dan 7, in het color commando, dan zal de S.E. 8 van die waarde aftrekken en in plaats van de opgegeven kleur, de kleur gebruiken die overeenkomt met de waarde die hij heeft berekend.

De karakterkleur kan veranderd worden, door het REGION commando te gebruiken, wanneer u meer dan één kleur wenst te gebruiken om punten op het scherm te PLOTTEN (zie deel 3.4.4).

GRAPHIC 3 activeert de grafische multi-color, dan wel de hoge op-

lossings-set afhankelijk van de waarde die in het karakter kleuregister is opgenomen nadat het COLOR commando is uitgevoerd (zie deel 3.4.2). Indien de waarde van de kleur binnen de reeks van 0 tot 7 ligt, zal alle PLOTTING plaats vinden in de "high-resolution" mode. Indien het kleurnummer groter is dan 7 zullen alle punten op het scherm in de multi color mode getekend worden. Dit commando is geschikt indien u high resolution en multi color graphic op hetzelfde scherm wilt gebruiken.

Voorbeeld: Het selekteren van de grafische "high resolution" set:
Programma: 100 GRAPHIC 2
Resultaat: De high resolution graphic set binnen het VIC systeem is geactiveerd.

3.4.2 COLOR

Formaat: COLOR sc, bo, cha, au
Doel: Kleuren te koppelen aan elk van de vier kleur-registers.

COLOR wordt gebruikt om kleuren te selekteren voor het scherm, de omgrenzing, het karakter en de hulpkleur. De "screen"-kleur, samen met het instellen van de achtergrondkleur kan ook gebruikt worden om punten op het scherm uit te wissen, die gePLOT waren in een andere kleur. De hulpkleur wordt meestal gebruikt met het PAINT commando (zie deel 3.4.7). De waarde van elke kleur is opgeslagen in een van de vier kleuregisters in VIC's geheugen. Als de registers eenmaal met het COLOR commando zijn uitgerust vormt het nummer van het kleuregister de eerste parameter van alle SUPER EXPANDER commando's die gebruikt worden om grafieken te

"plotten" op het scherm. (zie deel 3.3). Wanneer het systeem in werking is gesteld bevatten de vier kleuregisters respectievelijk de waarden 1, 3, 6 en 0. De beschikbare kleuregisters voor het gebruik in multi-color en high resolution modes worden hieronder aangegeven:

<i>multi-color</i>	<i>high resolution</i>
0 schermkleur	0 schermkleur
1 grenskleur	1 niet beschikbaar
2 karakterkleur	2 karakterkleur
3 hulpkleur	3 niet beschikbaar

Zoals u uit bovenstaande lijst kunt opmaken zijn de enige beschikbare kleuregisters voor gebruik in high resolution mode die welke het scherm en de karakterkleuren bevatten. Deze beperking kan opgeheven worden door de karakterkleur met het REGION commando indien u een andere karakterkleur op het scherm nodig hebt. (zie deel 3.4.4).

Voorbeeld: Het selekteren van een wit "scherm", 'n blauwe omgrenzing (border), een zwarte karakterkleur en een rose hulpkleur:

Programma
regel: 110 COLOR 1,6,0,10

Wanneer deze programma "regel" wordt uitgevoerd, wordt een witte "scherm" omgeven door een blauwe "border" afgebeeld. Alle "graphics"-die gePLOT zijn en gebruiksmaken van 't kleuregister dat de karakterkleur bevat zullen in het zwart zijn. Indien de hulpkleur wordt gebruikt zal de "plotting" in rose zijn.

3.4.3 POINT

Formaat: POINT cr, x, y
of: POINT cr, x, y, xl, yl,...,
 xn, yn

Doel: Een punt of punten te projecteren op 't scherm in een bepaalde kleur.

In de multi color mode kan een punt geprojecteerd worden op het scherm in elke kleur die beschikbaar is in de vier kleurregisters. (zie deel 3.3). In high resolution mode kunnen punten alleen geprojecteerd worden met die kleuren die beschikbaar zijn in de scherm- of karakterkleurregisters (zie deel 3.4.2).

Het commando kan gebruikt worden om één of een aantal punten op het scherm te projecteren. Als slechts één set scherm-coördinaten gespecificeerd is in het commando, dan zal slechts een punt geprojecteerd worden.

Aan de andere kant kunt u een aantal sets met scherm-coördinaten specificeren die in dat aantal punten zal resulteren die op het scherm geprojecteerd zijn. U kunt echter slechts in één kleur "plotten" binnen één commando.

Resultaat: Wanneer deze instructie wordt uitgevoerd zullen vier zwarte punten worden geprojecteerd in de rechter bovenhoek van het VIC scherm.

3.4.4 REGION

Formaat: REGION c
Doel: Het veranderen van de karakterkleur.

REGION verandert de karakterkleur die eerder specificeerde in het COLOR commando, d.w.z. het verandert de waarde opgeslagen in de kleurregister twee (zie deel 3.4.2).

Elk van de 15 S.E. kleuren kunt u gebruiken als de parameter van dit commando in multi color mode. In high resolution mode kunnen alleen kleuren van 0 tot 7 worden gebruikt. Dit commando is vooral waardervol als u zich in een high resolution mode bevindt en afbeelding wenst te maken anders dan in de scherm- of karakterkleuren (zie deel 3.4.2).

Het commando kan gebruikt worden om één of een aantal punten op het scherm te projecteren. Als slechts één set scherm-coördinaten gespecificeerd is in het commando, dan zal slechts een punt geprojecteerd worden.

Aan de andere kant kunt u een aantal sets met scherm-coördinaten specificeren die in dat aantal punten zal resulteren die op het scherm geprojecteerd zijn. U kunt echter slechts in één kleur "plotten" binnen één commando.

Voorbeeld: Een enkele punt projecteren op het scherm in de kleur zwart:

Programma 120 POINT 2, 900, 900
regel: Als deze instructie wordt uitgevoerd zal één zwarte punt worden afgebeeld in de rechter beneden hoek van het scherm.

Voorbeeld: Vier punten te projecteren op het scherm in dezelfde kleur.

Programma 130 POINT 2, 900, 100,
 900, 300, 1000, 100,
 1000, 300

3.4.5 DRAW

Formaat: DRAW cr, x, y TO xl, yl
of: DRAW cr, x, y TO xl,
 yl TO x2, y2,...,
 DRAW cr TO x, y

Doel: Een rechte lijn te tekenen tussen twee punten op het scherm.

DRAW zal een rechte lijn tekenen in de kleur aangegeven in het gespecificeerde kleurregister (zie deel 3.4.2). Het DRAW commando kan op 3 manieren gebruikt worden:

1. U kunt één lijn hebben (DRAW) tussen twee punten op het scherm daarbij slechts één set van coördinaten specificerend. Dan, indien u dat wenst, kunt u het commando opnieuw gebruiken om een andere lijn te DRAW-en tussen andere punten op het scherm.
2. U kunt vele lijnen hebben met een DRAW commando door veel sets van coördinaten te specificeren, die gescheiden worden met de code TO. De eerste set van coördinaten geeft het beginpunt aan van de eerste lijn die op het scherm getrokken moet worden. Het eindpunt van elke lijn wordt het startpunt van de volgende lijn.
3. U kunt ook DRAW-en terwijl u alleen het eindpunt van een lijn specificeert. In dit geval zal het startpunt van de lijn het eindpunt van de laatste grafische vorm, cirkel of punt zijn.

Voorbeeld: Een enkele punt projecteren op het scherm in de kleur zwart:

Programma 140 REGION 2
regel: Wanneer deze instructie wordt uitgevoerd, wordt de kleurwaarde voor rood opgeslagen in register twee. Na commando zullen alle punten die op het scherm getekend worden en de karakterkleur gebruiken in rood getekend worden. Dit gaat door tot de karakterkleur veranderd is met een ander REGION commando.

Voorbeeld: Het tekenen van 'n rechthoek:

Programma 150 DRAW 2, 100,
 100 TO 200, 100 TO 200,
 200 TO 100, 200 TO 100,
 100

regel: Een kleine rode rechthoek zal getekend worden in de linker bovenhoek van het scherm als deze instructie wordt uitgevoerd.

Voorbeeld: Een diagonale lijn te trekken naar het midden van het scherm:

Programma 160 DRAW 2 TO 512, 512
regel: Een lijn zal getrokken worden van het punt waar VIC het trekken van de rechthoek is gestopt naar het midden van het scherm.

Voorbeeld: Een cirkelvorm op het scherm te trekken:

3.4.6 CIRCLE

Formaat: CIRCLE cr, x, y, rx, ry
of: CIRCLE cr, x, y, rx, ry,
 as, af

Doel: Een cirkelvorm op het scherm te trekken.

Opmaking: In het laatste geval, als er geen voorafgaande vorm op het scherm is getrokken, zal het startpunt van de lijn de screenlokatie 0, 0 zijn, d.w.z. de linker bovenhoek van het scherm.

Programma 160 DRAW 2 TO 512, 512
regel: Een diagonale lijn te trekken naar het midden van het scherm:

Programma 170 CIRCLE 10, 10, 10, 10
regel: De coördinaten die in het CIRCLE commando zijn bepaald specificeren de scherm-locatie van het middelpunt van een cirkel of een cirkelvorm. De volgende set van parameters geven de breedte en de hoogte van die vorm vanuit zijn middelpunt. Deze laatste set van coördinaten stelt u in staat afgeplatte cirkels, d.w.z. ellipsen, te trekken.

Voorbeeld: Vanwege de wijze waarop punten getekend zijn op het scherm is om een precieze cirkel te trekken rx niet gelijk aan ry, zoals u zou verwachten. U moet de rx parameter vermeerderen met 0,7 om rekening te houden met het feit dat het scherm rechthoekig is, hoewel de x en de y as elk hetzelfde aantal verdelingen hebben.

Om een boog van een cirkel te trekken kunt u een volgende set van parameters aan het CIRCLE commando toevoegen.

De eerste parameter is het beginpunt van de boog op de omtrek van de cirkel en de tweede parameter is het eindpunt van de boog. De eenheid die gebruikt wordt in deze parameters is een "gradian". Er gaan 100 gradians in een complete cirkel. De positie van de 0 gradian is gelijk aan de 3.00 uur stand op een klok. Gradians lopen op, met de klok mee, op de omtrek van de cirkel eindigend bij gradian 100, weer in de 3.00 uur stand.

Voorbeeld: Een cirkel trekken:

Programma regel: 165 CIRCLE 2, 512, 512,

Resultaat: Wanneer deze instructie wordt uitgevoerd zal een rode cirkel in het midden van het VIC scherm getrokken worden.

Voorbeeld: Het trekken van twee cirkelbogen.

Programma regel: 170 CIRCLE 2, 800, 300,

Resultaat: Als deze instructies worden uitgevoerd zal een stel ronde haakjes getrokken worden in rood op het VIC scherm.

register is weergegeven. Het gebied moet volledig zijn afgesloten, aangezien anders het kleuren over het hele scherm zal plaats vinden. Het gebied dat gekleurd moet worden, wordt gespecificeerd door de coördinaten van één van de punten binnen de grenzen. Elk gebied kan slechts éénmaal gekleurd worden. Indien u de GRAPHIC3 instructieset gebruikt (zie deel 3.4.1), moet een gebied dat begrensd is met een omtrek in multi color mode gekleurd worden in dezelfde mode, d.w.z. u kunt modes niet veranderen tussen het maken van de vorm en het kleuren ervan.

Voorbeeld: Het trekken van een rode cirkel en deze geel inkleuren:

Programma regel: 190 CIRCLE 2, 300, 800,

Resultaat: Wanneer dit deel van het programma wordt uitgevoerd zal een kleine cirkel in rood getrokken worden in de linker benedenhoek van het scherm en het gebied binnen de cirkel zal geel gekleurd worden.

Voorbeeld: Het trekken van twee cirkelbogen.

Programma regel: 140, 100, 40, 60

Resultaat: CHAR – TEXT DISPLAY

Voorbeeld: Het opvullen van een afgebakend gebied met een kleur.

Programma regel: PAINT cr, x, y

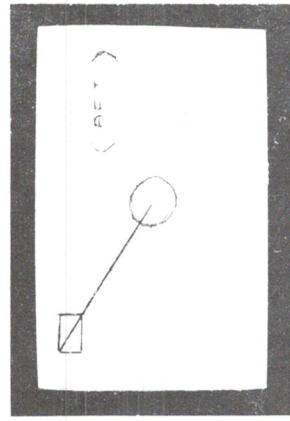
Resultaat: Het schoonmaken van een grafisch scherm

van de kleur die is opgenomen in de karakterkleurregister minder dan 8 is (zie deel 3.4.1). Dit commando is niet beschikbaar in GRAPHIC 1 multi color mode.

Voorbeeld: Het woord "ART" afbeelden in zwart op het grafische scherm:

Programma regel: 210 REGION 0

Resultaat: Wanneer dit deel van het programma wordt uitgevoerd zal het woord "ART" verschijnen in zwart tussen de eerder aangebrachte haakjes (zie figuur 3-1).



Figuur 3-1. Tekst display

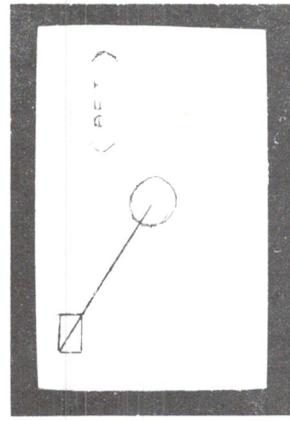
Het schoonmaken van het grafische scherm in het programma zoals u dat tot nu toe ingetyp heeft.

Het schoonmaken van de kleur die is opgenomen in de karakterkleurregister minder dan 8 is (zie deel 3.4.1). Dit commando is niet beschikbaar in GRAPHIC 1 multi color mode.

Voorbeeld: Het woord "ART" afbeelden in zwart op het grafische scherm:

Programma regel: 220 CHAR 5, 14, "ART"

Resultaat: Wanneer dit deel van het programma wordt uitgevoerd zal het woord "ART" verschijnen in zwart tussen de eerder aangebrachte haakjes (zie figuur 3-1).



Figuur 3-2. Running van het voorbeeld

Programma regel: 240 SCNCLR

Resultaat: Wanneer deze opdracht wordt uitgevoerd zal het scherm blank worden, d.w.z. een wit scherm met een blauwe begrenzing.

Figuur 3-3. Running van het voorbeeld

Programma regel: 230 FOR X = 1 TO 2500:
NEXT X
250 GRAPHIC 0: END

Als u dit programma runt zult u kennismaken met een voorbeeld van een aantal eenvoudige grafische mogelijkheden van SUPER EXPANDER.

3.4.9 SCNCLR

Formaat: SCNCLR
Doe: Het schoonmaken van een grafisch scherm

SCNCLR wordt gebruikt in de grafische "mode" om het scherm op dezelfde wijze schoon te maken als de BASIC vermelde PRINT "(SHIFT/CLR/HOME)" wordt gebruikt. Dit commando kan zowel in direkte als indirecte mode worden gebruikt.

3.4.7 PAINT (kleuren)

Formaat: PAINT cr, x, y
Doe: Het opvullen van een afgebakend gebied met een kleur.

PAINT vult een volledig gebied op met de kleur die in het bepaalde kleur-

3.4.8 CHAR – TEXT DISPLAY

Formaat: CHAR row, column,
"text"
Doe: Normale tekst af te beeldend op een grafiek-

scherm.

CHAR zal normale tekst op een grafiek scherm afbeelden beginnend bij de paalde rij en kolom. Dit commando kan gebruikt worden in de GRAPHIC 2 high resolution mode en in de GRAPHIC 3 mixed mode als de waarde

De VIC gebruikersclubs

REGIO NIEUWS

VIC gebruikersclub regio DRECHTSTREEK.

Ongeveer een half jaar geleden hebben wij een zeer bescheiden regionale vic club opgezet.

Ons doel is: Het belangeloos uitwisselen van informatie, programma's, en het ontwikkelen van kleine hardware projecten, zoals b.v. een EPROM-kaart (inmiddels ook op de markt). Ook beschikken wij over een mogelijkheid om langs fotografische weg printkaarten te maken, EPROMs te programmeren, en te wissen.

Thans bestaat deze club nog maar uit 6 leden, en wij zoeken daarom naar mensen die actief bezig willen zijn met hun computer. Een voorwaarde is dat u woonachtig moet zijn in de onmiddellijke omgeving van Dordrecht, Sliedrecht of Papendrecht. Indien u in het bezit bent van een VIC-20 en woont in een van bovengenoemde plaatsen, neen dan contact op met :

COR KLOP
P. K. Onneslaan 99
3362 VE Sliedrecht
Tel. 01840-18746 (na 18.00 uur)

Denkt u beslist niet in de trent van "dat is veel te moeilijk voor mij" of "ik heb bijna geen programma's om te ruilen", want daarvoor is deze club nu juist opgezet. De contributie kan u ook niet weerhouden, want het kost in principe niets.

Hebt u interesses, maar u woont niet in een van de genoemde plaatsen? Denk er dan eens over zelf een regionale club op te zetten. De benodigde leden zijn niet zo heel moeilijk te vinden. Stap bijvoorbeeld eens op een koopavond naar uw dealer, en wat ziet u; een winkel vol mensen die een VIC hebben of er een willen kopen. Hebt u dan een groep van ongeveer 10 mensen bij elkaar, bel mij dan eens om software etc. uit te wisselen tussen de verschillende clubs.

Cor Klop.

Aanvulling Redaktie :

Wij hopen dat er in de toekomst meer van deze regionale clubs zich in VIC PRIMEURS zullen presenteren. Misschien kunnen we zelfs wel een vaste rubriek met nieuws van de diverse club gaan opzetten ?

Wij wachten met spanning de reacties af !

VIC TIP

VIC MERGE (ingezonden door de heer M.J.P.M. ROOT)

Samen-voegen (MERGE) van programma's.

Met de volgende gegevens kan men twee (of meer) programma's samen voegen tot 1 programma.

Programma op tape zetten:

Type OPEN 1,1,1,"NAAM PROG":CMD 1:LIST (druk op RETURN).

Voor list kunt u alle commando's neer zetten.

bv. LIST SAVE het hele programma
LIST 10- SAVE het prog. vanaf regel 10 tot het einde
LIST -10 SAVE de regels vanaf het begin t/m regel 10
LIST 10-90 SAVE de regels 10 tot en met 90

Druk nu RECORD en PLAY in op de cassettereorder.

Als de recorder stopt typt u:

PRINT#1:CLOSE 1 (druk op RETURN)

U programma staat nu opband in merge.

Programma van tape halen:

(Laad nu eerst op de gewone manier het programma in waaraan het MERGE programma moet worden toegevoegd. REDAKTIE).

Spoel de tape naar het begin.

Type nu POKE 19,1:OPEN 1,1,0,"NAAM PROG" (druk op RETURN).

Druk nu op de PLAY toets van de recorder.

Als de recorder STOPT dan zet u hem NIET UIT.

Wis nu het scherm en typ het volgende in:

Druk driemaal CURSOR DOWN

POKE 153,1:POKE198,1:POKE631,13:?"CURSOR HOME" (druk op RETURN)

Als de recorder stopt staat u MERGE programma in het geheugen.

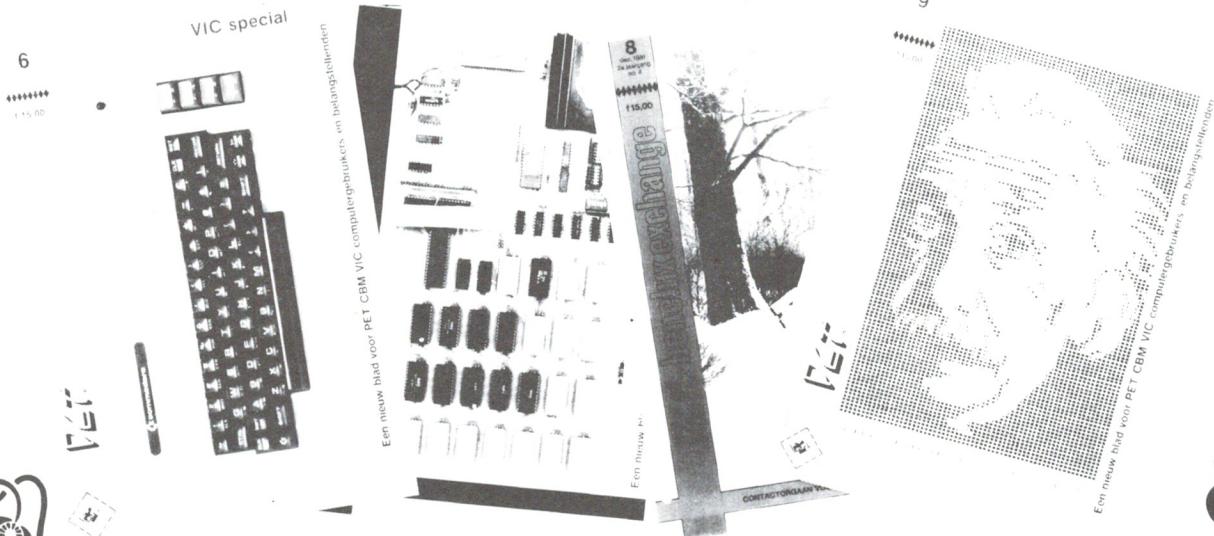
Dit geeft meestal een SYNTAX ERROR hier hoeft u niet op te letten.

Enkele goede redenen om u aan te sluiten bij de PBE

De PBE, afkorting van PET Benelux Exchange, is een grote, onafhankelijke gebruikersgroep, waarbij op dit moment ca 1500 PET/CBM en VIC gebruikers zijn aangesloten. De PBE is in het najaar van 1979 opgericht door Copytronics, waar zich de administratie en het secretariaat bevindt. De voornaamste activiteiten van de PBE zijn:

- * Regelmatische maandelijkse bijeenkomsten in het oosten en westen van ons land. Regio-Oost bijeenkomsten worden verzorgd door Jos en Hans Courbois (080)566315. Locatie: zaal Levensmorgen, Hazenkampsweg 36, Nijmegen (tweede zaterdag). Regio-West bijeenkomsten worden verzorgd door Bill Tjon en Ben de Winter, resp. bereikbaar onder (070)946156 en (015) 611648. Locatie: Lodewijk Makeblidge College, R. Holstlaan 2 te Rijswijk (vierde zaterdag). De openingstijden zijn van ca 11.00 uur - 16.30 uur. Hieraan voorafgaand worden cursussen in machinetaalprogrammeren gehouden. Er wordt gedemonstreerd op zelf meegebrachte computers en de belangstelling is erg groot. De entree bedraagt fl. 3,- per persoon, waarvoor men een nieuwsbrief krijgt toegestuurd voor de volgende bijeenkomst. Hierin staan ook tips en aanbiedingen voor/door gebruikers (niet commercieel).
- * Daarnaast is er het PBE kwartaalblad, waarvoor u fl. 45,- per jaar betaalt en dat tevens een lidmaatschap van de PBE gebruikersgroep inhoudt. Dit blad geeft belangrijke aanvullingen: vanaf PBE-6 ook voor VIC gebruikers. De uitgebrachte nummers zijn nog beschikbaar en op gebruikersbijeenkomsten ter inzage. Het lidmaatschap voor 1982 begint met PBE-9 waarin o.a. de VIC ROM-geheugenindeling, referentiekaart met PEEKS en POKEs, VIC schema's, simpele machinetaalmonitor, tips en korte programma's zijn opgenomen. Er is verder een printservice met losse of voorgemonteerde printjes, die voor VIC gebruikers binnenkort interessant wordt.
- * De PBE beschikt over duizenden programma's voor de PET/CBM die heel goedkoop ter beschikking staan voor de aangesloten gebruikers (zie PBE-9). Aan een VIC programmabibliotheek met honderden programma's wordt momenteel hard gewerkt!

Inl: Copytronics, B. v. Suchtelenstraat 46, 7413 XP Deventer
tel. 05700 - 31895 (Johan Smilde).



PRIJSLIJST

geldig vanaf 1-3-1982

VIC hardware

	Adviesprijs incl. B.T.W.
VIC-20	1381,—
VIC-1530 cassettereorder	266,—
VIC-1515 printer	1497,—
VIC-1540 single floppy disc 170K	2242,—
VIC-1210 3K RAM	168,—
VIC-1110 8K RAM	248,—
VIC-1111 16K RAM	404,—
VIC-1212 programmers-aid	157,—
VIC-1211a super expander + 3K RAM	224,—
VIC-1213 machinetaal monitor	157,—
VIC-1020 moederbord	648,—

VIC ROM-spelletjes

VIC-1901 Avengers	98,—
VIC-1904 Super slot	98,—
VIC-1906 Alien	98,—
VIC-1907 Jupiter Lander	98,—
VIC-1908 Draw poker	98,—
VIC-1909 Road race	98,—
VIC-1910 Rat race	98,—
VIC-1912 Mole attack	98,—
VIC-1919 Sargon schaak (binnenkort leverbaar)	

VIC cassette-programma's

VIC-1610 demonstratie	38,—
VIC-1620 Kleur/geluid	38,—
VIC-1630 Spelletjes 1	38,—
VIC-1640 Adressen	38,—

Accessoires

VIC lege cassettes, per paar	11,—
VIC lijstpapier, 500 vel	38,—
VIC gebruiksaanwijzing	34,—
VIC gebruikersclub, per jaar	50,—
VIC diskettes, 10 stuks	183,—
VIC joystick	41,—
VIC paddle, per paar	73,—
Learn programming on the VIC, Engels boek	21,—
VIC revealed schemaboek, Engels boek	59,—
Informatieboek voor programmeurs	59,—

Datatronic ROM-programma's

VIC-STAT	175,—
VIC-GRAPH	175,—
VIC-FORTH	249,—

De prijzen kunnen nog enigszins gewijzigd worden.

handic®
Benelux B.V.



Westerweg 198E, 1852 AP Heiloo. Postbus 213 1850 AE Heiloo,
tel. 072 - 33 76 44

DEALERLIJST:

A

Aalten	Erba b.v.	05437- 2351	B
Afferden	Macom	08853- 1647	C
Alkmaar	TOS computercentrum	072-156540	C
Alkmaar	Matsunaga	072-129337	B
Alkmaar	Fa. v. Kampen	072-116589	B
Alkmaar	Sibo	072-114929	B
Alkmaar	Fa. v. d. Gragt	072-126046	H
Alkmaar	Bakker Dijk b.v.	072-114268	C
Alkmaar	Wastora	072-127127	C
Alkmaar	Trendshop	072-155454	C
Alphen a/d Rijn	Groen Stereo	01720- 73083	B
Alphen a/d Rijn	Trendshop	01720- 72580	C
Amersfoort	Morelisse	033- 16052	C
Amersfoort	Radio Centrum	033- 15772	C
Amstelveen	Radio Valkenberg	020-432470	B
Amsterdam	Radio Vos	020-736154	B
Amsterdam	Attent Electra	020-934006	H
Amsterdam	Compu 2000	020-360903	C
Amsterdam	Kool HiFi	020-656369	C
Amsterdam	Radio Valkenberg	020-184022	B
Amsterdam	A.R.S. Elopta	020-251922	B
Amsterdam	Allwave	020-225344	C
Amsterdam	Radio Rotor	020-125759	B
Amsterdam	Fa. Elcon	020-279378	C
Amsterdam	Computercollectief	020-223573	C
Almelo	Trendshop	020-727757	C
Almelo	Electronicahuis	05490- 19191	C
Almelo	Leo de Rijter	05490- 18648	C
Almelo	Abbink	05490- 60461	B
St. Annaparochie	Radio Kampen	05180- 1906	B
Apeldoorn	v. Essen Radio	055-212485	B
Apeldoorn	Radio Putto	055-214106	B
Appelscha	Radio Oldersma	05162- 1591	H
Arnhem	Hupra	085-426716	B
Arnhem	Te Kaat	085-432445	C
Arnhem	Telemark	085-456838	B
Asten	Fa. Jeukens	04936- 3388	B
Assen	Audio Service Marree	05920- 13034	C
Assen	T.S.C.	05920- 17787	C
Arkel	M.C.P. b.v.	01831- 1566	C

B

Balk	Fa. Haantjes	05140- 2395	H
Beilen	Fa. de Groot	05930- 3839	H
Bergen op Zoom	Trendshop	01640- 56595	C
Bergen op Zoom	Fa. Rein de Jong	01640- 36028	B
Beverwijk	Radiodokter	02510- 26292	B
Beverwijk	Wijkerbaan Autobedrijf	02510- 43193	H
Beverwijk	Westerveld	02510- 24150	C
Boekel	Fa. v.d. Broek	04922- 2207	B
Born	Fa. Wibo	04498- 51248	H
Den Bosch	Desiree Camp	073-138323	B
Den Bosch	Ben v. Dijk Electro	073-216232	B
Den Bosch	Malmberg	073-215565	C
Breda	Indelec	076-142333	C
Breda	Breda Electra	076-135173	B
Breda	Fa. Cohen	076-134462	B
Den Burg	v. Wijngaarden	02220- 2695	H
Bussum	Radio Velt	02159- 17315	B
Bussum	Computerhouse Bussum	02159- 33700	C

C

Castricum	Trendshop	02518- 51441	C
Cuyk	Rutten	08850- 16344	B

D

Damwoude	Fa. v. d. Galiën	05111- 1396	B
Delft	Allwave	015-126322	C
Delft	Electr. Centrum Delft	015-134429	C
Delft	E.H.S.	015-132234	C
Delft	Radio Reno	015-132194	B
Deurne	Electro Mennen	04930- 9389	B
Doetinchem	Hobby Electr.	08340- 23329	C
Dokkum	Electr. Terpstra	05190- 4000	C
Doorn	Autoshop De Eend	03430- 4076	H

Verklaring van de letters:

H - handic produkten

C - Commodore VIC-20 produkten

B - Commodore VIC-20 en handic produkten

E

Doorsspijk	Fa. v. Zeeburg	05258- 365	B
Dordrecht	Radiobeurs Louter b.v.	078-134918	B
Drachten	Fa. v. d. Meulen	05120- 12352	B
Drachten	T.V. Technische Dienst	05120- 17541	C
Drunen	Dekkers b.v. Radio & T.V.	04163- 75833	B
Ede	Sisas Holland	08380- 38075	C
Ede	Eylander	08380- 17548	B
Eindhoven	Trendshop	040-451186	C
Eindhoven	Fa. Bombeek	040-441834	B
Eindhoven	Fa. Reijers	040-522888	C
Eindhoven	Compu 2000	040-445255	C
Eindhoven	Vogelzang	040-447955	B
Elburg	Bouwman Radio	05250- 3777	B
Emmeloord	Fa. Kollekoren	05270- 12256	C
Emmen	Crescendo	05910- 13580	C
Emmen	E.H.C. Micronics	05910- 13859	C
Enkhuizen	Electr. Orbit.	02280- 12904	C
Enschede	Radio Nijhuis	053-315169	C
Ermelo	Ves elektro	03410- 12786	C

F

Franeker	Fa. Tinga	05170- 2525	C
Franeker	Fa. Ettema	05170- 2090	B

G

Gaanderen	Centen Electra	08350- 7241	B
Geleen	Cuvos Electra	04494- 47709	B
Geldermalsen	Graham b.v.	03455- 2341	B
Genemuiden	L. Roetman	05208- 1389	C
Gennep	v. Veenendaal	08851- 3334	C
Goes	IMHA	01100- 13941	B
Goor	Fa. Vrielink	05470- 4050	C
Gorkum	Fa. Sommer	08130- 32644	H
Gouda	Willems Expert	01820- 12890	B
's Gravenzande	Trendshop	01830- 12888	C
Groningen	Fa. Koenen	01748- 3136	H
Groningen	A.V.A. Autoshop	050-128066	B
Groningen	Vorstenberg comm. center	050-121524	B
Groningen	Telec b.v.	050-129374	C
Groningen	Computerwinkel	050-131427	C

H

Haaksbergen	Fa. v. Ulsen	05427- 1276	B
Den Haag	Quality Sellers	070-888844	C
Den Haag	Stuut en Bruin	070-604993	B
Den Haag	Rueb b.v.	070-559919	B
Den Haag	Radio Gerrése	070- 45542	C
Den Haag	Allwave	070-649400	C
Haarlem	Fa. Kiekens	023-320485	H
Haarlem	Blue Cat	023-327236	H
Haarlem	Display Electronica	023-322421	C
Harderberg	Fa. Oostenbrink	05232- 1720	B
Harderwijk	Ridero	03410- 18245	C
Heemskerk	Fa. v. Campen	02510- 42919	B
Hengelo	Radio Nijhuis	074-917567	C
Hengelo	Computershop Hengelo	074-428726	C
Heerde	Veron Electr.	05782- 1540	B
Heerhugowaard	Fa. Leegwater	02207- 12309	B
Heerlen	Vogelzang	045-716055	B
Heiloo	Radio Bakker	072-330262	B
Den Helder	Elab component supply service	02230- 30375	B
Den Helder	Fa. Proton	02230- 19068	B
Helmond	Fa. Sibo	02230- 13310	B
Hillegom	Fa. Westerhof	04920- 46680	B
Hilversum	Kall-Tronic	02520- 15605	B
Hilversum	Computerworld	035- 12633	C
Hoek van Holland	Trendshop	035- 42013	C
Hoofddorp	Electro Holland	01747- 4819	H
Hoofddorp	W + L Automatisering	02503- 31890	C
Hoofddorp	Fa. Gehrels	02503- 14965	B
Hoogeveen	Doeven Electronica	05280- 69679	B
Hoorn	Sibo	02290- 17763	B

Hoorn	Video 2000	02290- 14779	C
Hulst	Elton Autoshop	01140- 12261	C
Hulst	Colijn Data Systemen	01147- 834	C

J
Joure Radio Rijpkema 05138- 2656 B

K
Kampen Elcynamic 05202- 11671 B
Krabbendijke Fa. Goud 01134- 2010 H
Krommenie Knijnenberg 075-289606 C
Kloetinge Colijn Data Systemen 01100- 14008 C
Krimpen a/d IJssel Comptour 01807- 19817 C

L
Leiden L.C.L. 071-125700 H
Leiden Fa. de Groot 071- C
Leiden Vlasveld Electronica 071-120848 C
Leiden Allwave 071-124567 C
Lekkerkerk Fa. Zwijsenburg 01805- 1327 H
Lekkerkerk Electro Slager 01805- 1869 B
Lelystad Micron Electronics 03200- 44830 C
Lichtenvoorde Fa. Krabbenborg 05443- 2713 H
Lieren Fa. v. d. Kamp 05766- 1744 H
Lisse Radio Beurs 02521- 12176 B

M
Maastricht Vogelzang 043- 14169 B
Maastricht Fa. Zeguers 043- 32072 H
Meerssen T.V. Deusing 043-642079 C

N
Nistelrode Ben. v. Dijk 04124- 1503 B
Nijkerk v. d. Klokk Geurtsen 03494- 51382 B
Nijmegen Fa. v. d. Camp 080-442747 C
Nijmegen Fa. Baas Esso Service 080-441663 H
Nijmegen Fa. v. d. Water 080-554182 B
Nijmegen v. d. Broek 080-774322 H
Nijmegen Macom Systems 080-233878 C
Nijmegen Video Nijmegen 080-224820 C
Nijverdal Doornbal b.v. 05486- 12297 B
Nijverdal Radio Vo 05486- 12728 C

O
Oosterhout Fa. Peeters 01620- 53612 B
Oostkapelle Info Boek Zeeland 01188- 2305
Oss Ben. v. Dijk Electro 04120- 33295 B

P
Purmerend Radio Valkenberg 02990- 20727 B
Pijnacker Allwave 01736- 3961 C

R
Raalte Bekman Electra 05720- 2511 B
Reusel Rivago Micromix 04976- 1979 C
Ridderkerk De Malle Molen 01804- 25782 C
Roermond Byte Electron 04750- 10250 C
Roermond Fa. Zeguers 04750- 14479 B
Roosendaal H & B Techn. Buro 01650- 34171 B
Rotterdam Dil Electronics 010-854213 C
Rotterdam Compu 2000 010-117524 C
Rotterdam Delmee Autoshop 010-844304 H
Rotterdam VIC Computercentrum 010-137823 C
Rotterdam Radio Ultra 010-191680 C
Rotterdam DCS Electronica 010-769900 B
Rotterdam F.G. Electronics 010-145553 H
Rotterdam Botlek Stores 010-161133 C
Rotterdam L. Taxt 010-297410 C

S
Sassenheim Fa. Duynstee 02522- 10398 B
Schagen Sibo 02240- 98383 B
Schiedam Trendshop 010-739601 C
Schiedam Radiohuis v. d. Bend 010-733855 B
Sevenum Emckevort + Bors 04767- 1851 H
Sittard Fa. Harmens 04490- 16658 H
Sittard Fa. Wibo 04490- 13070 H
Sneek Blom Radio 05150- 13383 B
Someren Fa. v. Ottendijk 04937- 1204 H
Spijkenisse Electorama 01880- 22022 C

Stadskanaal	Leo Electronics	05990- 19004	B
Stadskanaal	Commix	05990- 20086	C
Steenwijk	Fa. Beute	05210- 12349	C

T
Tholen Quist Duine 01660- 2505 H
Tiel Fa. Rieuwers 03440- 13919 C
Tiel T.V. Servicedienst Tiel 03440- 13907 B
Tilburg Nico v. d. Braak 013-432153 B
Tilburg Fa. Rosemeisl 013-323049 H
Tilburg Mitchel Electronics 013-360848 C
Tilburg N. v. Helfteren 013-670906 C
Tilburg N. v. Helfteren 013-355605 C

U
Uden Ben. v. Dijk Electro 04132- 64944 B
Uithuize Lambeek 05953- 1298 H
Utrecht Display Electronics 030-315655 C
Utrecht Radio Centrum 030-319636 B

V
Varsseveld Visscher 08352- 2749 B
Veenendaal Hupra 08385- 24222 B
Veghel v. Aalst Electronic 04130- 41370 B
Venlo Fa. Bauer 077- 17154 C
Venlo Enckevert en Bors 077- 27278 H
Vlaardingen Radiohuis v. d. Bend 010-342481 B
Vlaardingen Fa. Molenaar 010-341516 C
Vlissingen Dijkhuizen 01184- 12981 C
Volendam Hi-Fi Volendam 02993- 65451 C
Voorhout IJsselmuiden 02522- 11618 C
Voorthuizen Fa. v. Loon 03429- 1359 B

W
Waalwijk Fa. Dekkers 04160- 34836 B
Wageningen Fa. Mateman 08370- 12444 H
Weert Lako b.v. 04950- 34164 H
Weert Harthold Weekers 04950- 33392 B
Wierden Engberts & Olthuis 05496- 1565 H
Wieringerwerf Fa. v. Zoonen 02272- 1232 H
Winschoten Fa. Oost 05970- 14234 C
Winschoten Typeschool nieuwe stijl 05970- 14669 C
Wommels Fa. Kooistra 05159- 1341 H

IJ
IJmuiden Fa. v. Campen 02550- 14516 B

Z
Zaandam Wastora 075-127127 C
Zaandam Radio Valkenberg 075-168255 B
Zaandam De Prijzenkraker 075-162562 B
Zaandijk Fa. Zeltron 075-282974 C
Zwolle Radio Centrum 05200- 12233 H
Zwolle Electronicahuis 05200- 13804 C
Zwolle Wehkamp 05200- 77711 C



VIC-GEBRUIKERS CLUB
POSTBUS 12972
1100 AZ AMSTERDAM