

Grafisch display

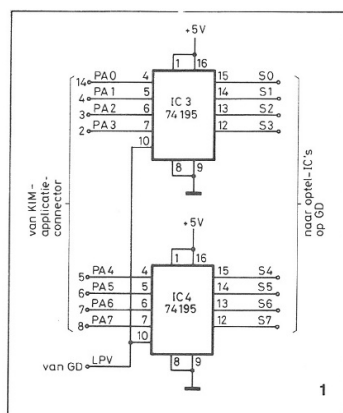
Monitor voor de KIM

M. Dohmen,

R. Koekoek

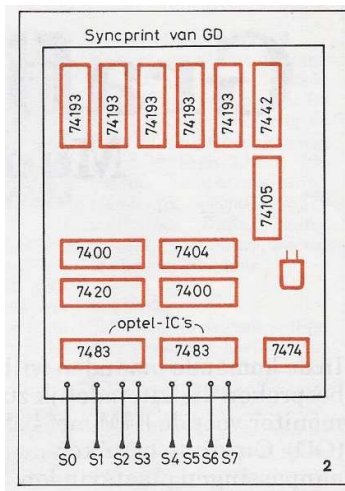
In de komende maanden willen wij een aantal routines bespreken die uiteindelijk zullen leiden tot een volledige monitor voor de KIM met behulp van het grafisch display (GD). Om dit te bereiken moeten wat hardware-aanpassingen plaatsvinden. Te verwachten zijn een aantal routines om onder meer lijnen te trekken en figuren te tekenen, variërend van eenvoudige vierkanten en kubussen tot ingewikkelde ellipsen, parabolen en hyperbolen. Dit allemaal en nog veel meer werkt met behulp van een totaal andere programmeertaal, een soort „tekentaal“. Verder een complete machinetaalmonitor met disassembler en volledige cursorbesturing en een nieuwe cassette-lees-schrijf-routine met een opslagsnelheid van 2400 baud (dit is 15,6K per minuut) met softwarebesturing van de recorders. Uiteraard bij deze cassetteroutine een „linking-loader“, die universeel toepasbaar is en uitbreidingen toelaat.

De opzet is als volgt. Het complete pakket bestaat voor een groot gedeelte uit subroutines, die door een klein hoofdprogramma worden gestuurd. Deze routines komen gewoon achter elkaar in het geheugen te staan. Degenen die geen complete monitor willen hebben, kunnen deze subroutines los gebruiken, daar ze ook op zichzelf hun nut kunnen bewijzen. Zo ontstaat dan een „subroutine-arsenaal“. Degenen die wel een complete machinetaalmonitor willen hebben, zullen echter een paar wijzigingen in, en toevoegingen aan, de hardware moeten maken.



1. Zoals u wellicht weet bestaat er de mogelijkheid een hardware scroll-up te maken op het GD. Hiervoor zijn 4 IC's nodig. Zonder deze uitbreiding werkt de monitor niet.
2. Voor de monitor is een groter toetsenbord nodig (28 toetsen). Ik mag echter aannemen dat u, als u geen ASCII-toetsenbord heeft, al lang van plan was een uitgebreider toetsenbord te kopen. Verderop in dit artikel vindt u een toetsenbord beschreven, dat zeer goedkoop is en ruimschoots voldoet (voor zo'n f 30,00 heeft u een goedwerkend toetsenbord). Een echt ASCII-toetsenbord blijft natuurlijk aan te raden.

3. Niet noodzakelijk, maar erg handig is een schakeling, die ervoor zorgt dat met software een luidspreker aan en uit kan worden gezet. Deze kan van nut zijn in combinatie met de monitor (denk bijv. aan een pieptoon nadat CRLF is gegeven).



Hardware-scroll-up

De ingangspoort van het GD kan tot nu toe vier commando's verwerken, namelijk:

000_2 = doe niets.

001_2 = klok x-adres in.

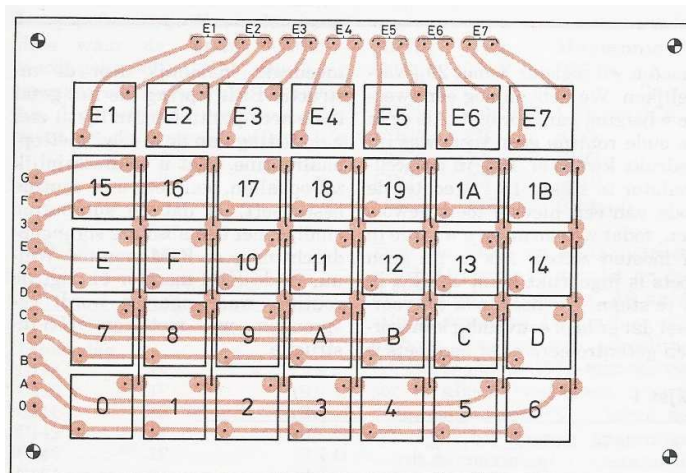
010_2 = klok y-adres in.

011_2 = schrijf een punt op het scherm.

Er kunnen echter zeven signalen worden gedetecteerd. De scroll-up maakt gebruik van de lichtpendetectie (101_2).

Zodra er 101_2 op de inputpoort verschijnt, wordt LPV (lichtpen verticaal) laag en worden de IC's 3 en 4 actief (zie afb. 1). Deze twee IC's zijn 4bits-schuifregisters. De

ingangen van deze IC's zijn verbonden met poort A, die ook naar het grafisch display toe gaat. De uitgangen zijn verbonden met de optel-IC's op de syncprint van het grafisch display (zie ook afb. 2). Als deze IC's er bij u nog niet opzitten dan moet u deze IC's nog vast solderen (tweemaal 7483). Als nu LPV laag wordt, wordt de data, die op PA stond, doorgeschoven naar de optel-IC's. Hier wordt de data opgeteld bij het verticale adres en ontstaat er aldus een scroll-up. Als u bijvoorbeeld de tekst op het GD een regel wil laten opschuiven, zet u gewoon $0A_{16}$ op poort A ($0A_{16}$ is het aantal puntjes verticaal dat een karakter nodig heeft). Er wordt natuurlijk van uitgegaan dat de inhoud van de latches van 8 bits (IC 3 en 4) nul waren.

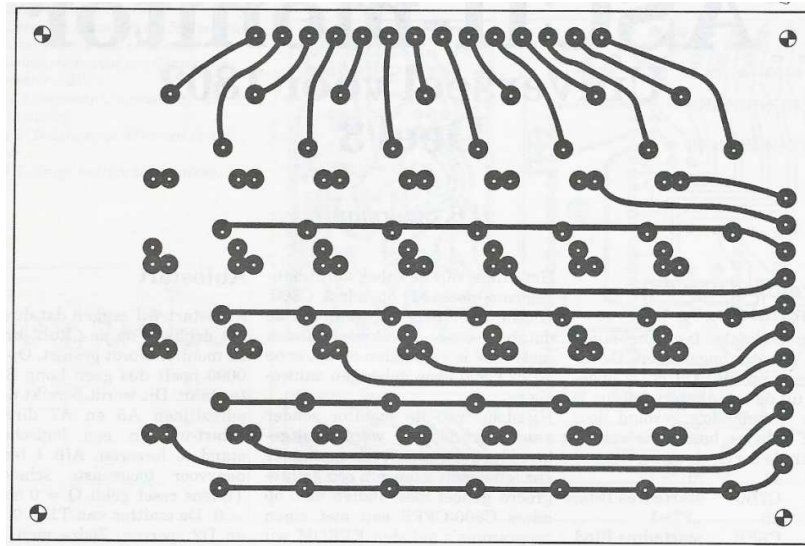


Toetsenbord

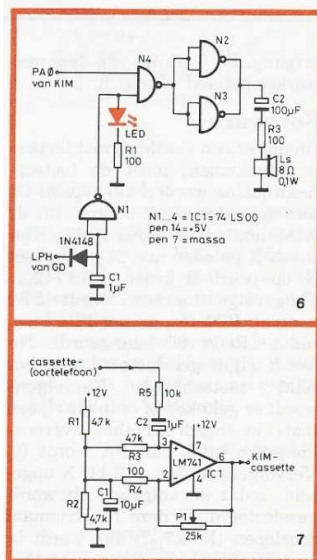
Voor de uitgebreide monitor heeft u een toetsenbord met minimaal 28 toetsen nodig. Een ASCII-toetsenbord is natuurlijk ideaal, maar omdat niet iedereen er het geld voor over heeft, wordt hier een goedkoop alternatief besproken. Op de KIM is één pennetje van het binair-naar-decimaal-omzetter-IC ongebruikt gelaten. Door hier

een extra rij toetsen aan vast te maken en een nieuwe toetsophaalroutine te schrijven, heeft u 28 toetsen tot uw beschikking gekregen. In afb. 3. is een gedeelte van de schakeling rond het KIM-toetsenbord en de toegevoegde hardware getekend. Zoals u ziet krijgen de nieuwe toetsen de toetscode 15_{16} tot en met $1B_{16}$. Het is een beetje

onhandig om naast het oorspronkelijke KIM-toetsenbord nog zeven losse toetsen te bedienen, die waarschijnlijk een eindje van het KIM-toetsenbord af zullen liggen. Door 28 nieuwe toetsen op een print te bevestigen en deze te verbinden met de KIM heeft u alle toetsen bij elkaar zitten. Als u dit doet is het aan te raden om zeven extra toetsen op de print te bevestigen en tevens genoeg ruimte te laten voor verdere uitbreidingen. Als u goede toetsen koopt hoeft u het toetsenbord nooit meer te vervangen. U kunt nu ook de toetsen van eigen opschriften voorzien. Wat deze opschriften moeten zijn, volgt later. In afb.4 ziet u het printontwerp van het toetsenbord. Als u niet over de nodige middelen beschikt voor het etsen van een print kunt u wel een voorgeboorde print van toetsen voorzien, hetgeen net zo handig is.



Luidsprekeraanpassing



In afb.6 ziet u het complete schema van de luidsprekeraanpassing. Voor deze aanpassing wordt het pennetje LPH (lichtpen horizontaal) gebruikt. Normaal is deze uitgang hoog, maar zodra er 110₁₂ op de ingangspoort van het grafisch display verschijnt, verschijnt er een blokgolf met een frequentie van 1 MHz op deze uitgang. Hierdoor kan condensator C1 zich ontladen en wordt de uitgang van poort N1 hoog. De LED gaat branden ter indicatie dat de luidspreker met PA0 is verbonden. Het signaal, afkomstig van PA0 wordt via N4 aangeboden aan de versterker bestaande uit N2 en 3 en wordt dan naar een luidspreker gevoerd. Een extra aanpassing betreft de versterker, die het zwakke signaal, dat van de cassetterecorder afkomt, moet versterken, zodat het signaal aan de KIM kan worden

aangeboden (zie afb.7). Deze aanpassing is natuurlijk niet noodzakelijk, maar aan te bevelen als u voor altijd van dat „gesodemeter" met de volumeregelaar af wilt zijn of als

u genoeg heeft van het lawaai dat uit de luidspreker van uw goedkope cassette recorder komt. De ingang van de versterker wordt nu gewoon verbonden met de line-uitgang. De werking van deze versterker spreekt voor zich.

Software

Omdat er een vierde rij met toetsen is bijgekomen, moet de toetsophaalroutine worden aangepast. De toetsophaalroutine begint in de KIM-monitor op adres 1F6A. Hier wordt X geladen met 21₁₆ en wordt X op poort B gezet (PB:1742₁₆). Tengevolge hiervan wordt PB0 hoog en PB1 tot en met PB4 laag, zodat „ROW 0" laag wordt. Nu wordt rij 0 geselecteerd

van het KIM toetsenbord.

Vervolgens wordt er gekeken of er in die rij een toets is ingedrukt (het overeenkomstige bitje van PA wordt 0). Vervolgens wordt er 2 bij X opgeteld, zodat de volgende rij wordt geselecteerd. Is deze lus driemaal doorlopen (X=27₁₆?) dan wordt òf de toetswaarde in A gezet, òf 15₁₆ in A gezet als er geen toets was ingedrukt. Een zelfde routine hebben we voor het nieuwe toetsenbord nodig met die

Lijst 1

```

RGS.6502 COMPUTER BULLETIN DE PAGE 01

0000:
0010:
0020:
0030:
0040:
0050:
0060:
0070:
0080:
0090:
0100:
0110: E000          ORG    #E000
0120:
0130:
0140:
0150:
0160: E000          ONEY *  #1F02
0170: E000          KEVIN *  #1F7H
0180:
0190: E000 A2 21      TOETS  LOXIM #21  EERSTE RIJ IN REG X
0200: E002 A0 01      MOTO   LOVIN #01
0210: E004 20 02 1F  JSR     ONEYV TOETS INGEDRUKT IN DEZE RIJ?
0220: E007 00 07      BNE     ZETOM  ZO JA, ZET WAARDE OM
0230: E009 E0 29      CPXIM #29  VIER RIJEN GEHOED OM
0240: E00B 00 F5      BNE     MOTO   ZO NEE, VOLGENDE
0250: E00D A9 FF      LOVIN #FF  ZO JA, ZET #FF IN ACCU
0260: E00F 60        RTS        EN KEER TERUG
0270: E010 4C 7H 1F  ZETOM  JMP     KEVIN  ZET TOETSWAARDE OM

```

wijziging, dat we vier rijen moeten selecteren in plaats van drie rijen. In plaats van register X met 27₁₆ te vergelijken, moeten we register X met 29₁₆ vergelijken. We hebben nog een tweede wijziging aangebracht. Als er in de oude routine geen toets was ingedrukt kwam er 15₁₆ in de accumulator te staan. 15₁₆ is echter de code van een nieuwe toets geworden, zodat we een andere waarde in A moeten zetten. Als er nu geen toets is ingedrukt komt er FF₁₆ in A te staan. Dit heeft ook tot voordeel dat er heel eenvoudig kan worden gecontroleerd of er een toets is ingedrukt, namelijk door de instructie BPL (spring als het getal in de accu positief is). In lijst 1 ziet u de listing van de nieuwe toetsophaalroutine. Wat u waarschijnlijk zal opvallen, wanneer u de routine bestudeert, is dat de subroutine eindigt met een absolute sprongopdracht naar de KIM-monitor. Welnu, in de KIM-monitor eindigt de routine, waar naartoe wordt gesprongen, wèl met een RTS-instructie.

Bijschriften deel 1:

Afb. 1 Aanpassing voor de hardware-scroll-up.

Afb. 2 Plaats op de syncprint waar de IC's en de draden van de aanpassing moeten komen te zitten (tweemaal 7483).

Afb. 3 Toetsenbord en een gedeelte van de KIM-hardware.

Afb. 4 Printontwerp van het nieuwe toetsenbord.

Afb. 5 Componentenopstelling van het toetsenbord.

Afb. 6 Luidsprekeraansluiting.

Afb. 7 Versterker voor de cassette recorder.

Lijst 1 Nieuwe toetsophaalroutine.

In het eerste deel (dec. '81) hebben wij de diverse hardware aanpassingen besproken, die nodig zijn om onze monitor te laten werken op de KIM. In dit deel bespreken we de software, zoals die in de eerste en een deel van de tweede EPROM kan worden ondergebracht. Behandeld worden de besturing van de scrolling, de luidspreker en de cursor, die met variabele frequentie knippert. Verder de commando's „wis het scherm“, „wis vanaf de cursor tot het einde van het scherm“, „wis vanaf de cursor tot het einde van de regel“ en druk een pointer, byte, nibble of ASCII-karakter af. De monitor heeft uiteindelijk ongeveer 6K aan geheugenruimte nodig, wat gezien de vele mogelijkheden erg weinig is.

Wanneer u moeilijkheden heeft met het aansluiten van EPROM's, raad ik u aan het artikel „Moederkaart voor 6502“ te lezen (RB, juli 1981). Met behulp van 4K EPROM-kaarten (RB, september 1981) kan gemakkelijk 8K aan EPROM op de KIM worden aangesloten. In tabel 1 ziet u de indeling van deze geheugenruimte. Vanaf adres E100 tot en met E382 vinden we een al bestaand programma, namelijk „Letters op het grafisch display“, geschreven door de heer D. M. de Boer. Vanaf adres E000 vindt u het in het voorgaande deel behandelde toetsophaalprogramma en vanaf adres E020 het apart beschreven programma „Lijnen op het grafisch display“ van de heer R. Koekoek. We zullen nu de diverse besturingen en commando's nader toelichten aan de hand van de bijbehorende lijsten.

Lijst 1

		1830 ;	
		1840 ;	
		1850 ;	
		1860 ;	ROUTINE VOOR LUIDSPREKERSTURING
		1870 ;	
		1880 ;	
E8FA-	20 9C E3	1890	LUIDS
E8FD-	05 3E	1900	
E8FF-	8D 47 17	1910	
E902-	06 3F	1920	LUSG
E904-	EA	1930	
E905-	EA	1940	
E906-	EA	1950	
E907-	CA	1960	LUSH
E908-	00 FD	1970	
E90A-	EE 00 17	1980	
E90D-	0D 47 17	1990	
E910-	10 F0	2000	
E912-	4C 96 E3	2010	

```

JSP LSAAN ;INITIALISEER
LDA #GELENG ;HAAL LENGTE VAN GELUID
STA TIMER2+*#07 ;
LDX #GEFRE0 ;HAAL FREQUENTIE
NOP ;
NOP ;
NOP ;
DEX ;TEL AF
BNE LUSH ;
INC PAD ;TOGGLE LUIDSPREKER
LDA TIMER2+*#07 ;TIMER KLAAR?
BPL LUSG ;
JMP L5UIT ;LUIDSPREKER UIT

```

Luidspreker

In lijst 1 en 2 zien wij een simpel stukje software. Eerst wordt gesprongen naar de subroutine LSAAN, waar de waarde \$06 in poort B wordt gezet. Hierdoor wordt bit 0 van poort A verbonden met de luidspreker, doordat LPH, een penntje op het grafisch display,

zoals eerder besproken actief wordt. Vervolgens wordt er een toon opgewekt met een aan-uitverhouding van 50 %. De duur is instelbaar van ca. 0,001 tot ca. 0,25 s en de frequentie van 500 tot ca. 20000 Hz. Alvorens het programma aan te roepen moet u de duur op adres 008E zetten en de cyclustijd (1/frequentie) op adres 008F. De duur wordt in de timer geladen, welke een deelfactor van 1024 geeft. X wordt geladen met de cyclustijd. In een lus wordt X telkens met 1 verminderd, totdat deze gelijk aan nul is geworden. Hierna wordt PA0 „getoggeld“, waardoor er een toon ontstaat. Dit gaat door tot de tijd om is. Om de luidspreker uit te zetten wordt \$00 in poort B geladen, wat tot gevolg heeft dat LPH „hoog“ wordt. Voordat dit kan werken moeten uiteraard de datarichtingsregisters van de in/uitpoorten goed worden gezet. Dit goed zetten gebeurt

in de routine INITIO, zie lijst 3. Op de plaats waar nu NOP's staan, komt later een JSR-instructie naar een toetsophaalroutine.

Lijst 2

```

0070 ;
0080 ;
0090 ; ROUTINE SCROLL UP
00A0 ;
00B0 ;
E384- 18 0920 SCRUP CLC ;
E385- 08 0930 CLD ;
E386- A5 A9 0940 LDA *SCROLL ;TEL REGELHOOGTE OP
E388- 65 A8 0950 ADC *REGELH ;BIJ DE
E38A- 85 A9 0960 SCRSCT STA *SCROLL ;REGELTELLER
E38C- A5 A9 0970 SCROUT LDA *SCROLL ;TEL DIT OP
E38E- 80 00 17 0980 STA PAD ;BIJ HET
E391- A9 05 0990 LDA #05 ;VERTICALE ADRES
E393- 80 02 17 1000 STA PAD ;VAN HET DISPLAY
E396- A9 08 1010 LSUIT LDA #00 ;ZET PAD GOED EN
E398- 80 02 17 1020 STA PAD ;LUIDSPREKER UIT
E39B- 60 1030 RTS ;
1040 ;
1050 ;
1060 ;
1070 ; ROUTINE LUIDSPREKER AAN
1080 ;
1090 ;
E39C- A9 06 1100 LSAAN LDA #06 ;MARK LPH ACTIEF
E39E- D0 F8 1110 BNE LSUIT+02 ;

```

Scrolling

In lijst 2 ziet u de routine SCRUP, waarmee alle tekst op het scherm één regel naar boven kan worden geschoven (scroll up). De data in SCROLL (adres 00A9) houdt bij hoeveel de tekst al was opgeschoven. De data in REGELH (adres 00A8) onthoudt de regelhoogte. Eerst worden het carry- en het decimalbit nul gemaakt om fouten te voorkomen.

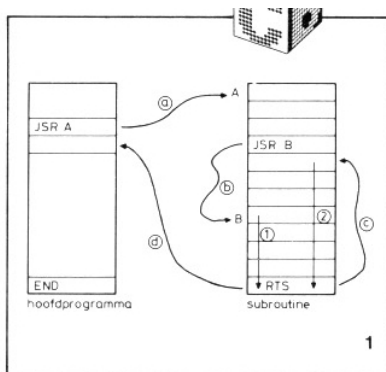
De regelhoogte wordt bij de regelteller opgeteld en het resultaat wordt in SCROLL gezet. Deze waarde wordt in PAD geladen en \$05 in PBD (LPV). Hierdoor wordt SCROLL door de optel-IC's opgeteld bij het verticale adres van het grafisch display. Vervolgens wordt PBD goed gezet. Dit komt overeen met het uitzetten van de luidspreker, zodat deze twee routines konden worden gecombineerd. Dat spaart weer geheugenruimte. Met behulp van routine SCRDOWN kan de tekst op het scherm een regel naar beneden worden geschoven (scroll down). Zij werkt volgens hetzelfde principe als SCRUP, behalve dat de regelhoogte wordt afgetrokken van de regelteller.

Cursorbesturing

De in lijst 4 weergegeven routine voor de besturing van de cursor is er niet een van het type waarbij de cursor, telkens wanneer deze routine wordt aangeroepen, op de goede plaats wordt gezet, maar een routine die enkele honderden malen per seconde wordt doorlopen. U moet zich dus voorstellen dat enkele zaken als adressen en registers een vorige keer al waren goed gezet. XCOR (adres 00A1) bevat de huidige X-coördinaat en YCOR (adres 00A2) de huidige Y-coördinaat van de cursor. Stel dat de routine al eens was aangeroepen. In de routine OUT zouden dan XCOR en YCOR gelijk zijn gemaakt aan respectievelijk LASTX en LASTY, ofte wel de laatst behandelde cursorpositie zou gelijk zijn gemaakt aan de te behandelen positie. Wanneer in een ander programma de X- en Y-coördinaat van de cursor niet zijn veranderd (de cursor hoeft niet te worden verplaatst), zijn bij het opnieuw doorlopen van de routine LASTX en XCOR en LASTY en YCOR aan elkaar gelijk. Routine OUT wordt dan niet uitgevoerd. In register Y wordt de Y-coördinaat gezet en in register X de X-coördinaat. Wel wordt gekeken of de cursor niet moet worden geïnverteerd; deze moet immers knipperen. CAAN (adres 00AB) houdt bij of de cursor aan of uit was. Op adres E3C2 wordt de cursorteller (CCOUNT) opgehoogd en vergeleken met de waarde voor de knippersnelheid (CSPEED). Zijn deze aan elkaar gelijk, dan wordt bij INVERT de cursor geïnverteerd. CCOUNT wordt geladen met \$FB

(-5), en zo wordt bijgehouden, hoeveel puntjes er moeten worden gezet; in dit geval vijf. CAAN (adres 00AB) wordt goed gezet en register Y wordt opgehoogd om de juiste plaats op het scherm te kiezen. De cursor moet immers onder het karakter komen. Vervolgens wordt LUS7 vijfmaal doorlopen, waardoor de puntjes op het scherm worden geschreven met als resultaat dat CCOUNT nul is geworden.

Stel nu eens dat de X- en Y-coördinaat in een ander programma werden veranderd. Er wordt dan naar OUT gesprongen. Hier wordt gekeken of de waarde in CAAN even dan wel oneven is. Wanneer deze oneven is, was de cursor aan en wordt naar INVERT gesprongen om de cursor op de oude positie uit te zetten. De cursor op de nieuwe positie wordt aangezet als de routine voor de cursorbesturing een tweede maal wordt doorlopen. Was de oude cursor al uit, dan wordt alleen LASTX gelijk gemaakt aan XCOR en LASTY aan YCOR. Door vervolgens CCOUNT gelijk te maken aan CSPEED wordt de cursor direct op de nieuwe plaats zichtbaar.



Afdrukken van pointers, bytes, nibbles en karakters

In de routine van lijst 5 is tot het uiterste gebruik gemaakt van zogenoemde „nested loops". Dit levert een stukje code op dat driemaal zo beknopt is als vergelijkbare routines en dat is een doel van iedere rechtgeaarde programmeur. Omdat de structuur nogal complex is, wordt de werking verklaard aan de hand van de uitgeschreven versie volgens lijst 6. Dit is dus de volgorde van de instructies zoals de computer

ze tegenkomt. Daar stukken van het hoofdprogramma meerdere malen moeten worden uitgevoerd, worden er sprongen naar subroutines gedaan, die eigenlijk geen subroutines zijn. De stack houdt bij waar de volgende instructie zich bevindt. In afb. 1 vindt u een voorbeeld van een „nested loop". Wanneer de computer naar een subroutine springt, wordt de programmateller op de stack gezet. Bij terugkeer uit de routine wordt de inhoud van de stack geladen in de programmateller en gaat de computer op de aangegeven plaats verder. Wordt er echter gesprongen naar een subroutine die zich in het programma A zelf bevindt (JSR B) en bovendien eindigt met een RTS, dan resulteert dit in het tweemaal uitvoeren van routine B. De werking van de printroutine verloopt analoog.

Nog even het volgende. Wanneer een cijfer moet worden afgedrukt, moet dit eerst worden omgezet naar ASCII. De code voor de „0" is \$30, voor de „1" is dat \$31 en zo tot en met 9. De ASCII-codes voor letters sluiten niet precies aan op die voor cijfers. De „A" heeft de code \$41. Om van 9 naar A te gaan, hetgeen het geval is bij hexadecimaal tellen, moet men \$07 optellen bij de code voor 9 ($41 - 3A = 7$). Deze methode van omzetting is in dit programma ook toegepast. Wanneer het getal \$2B moet worden afgedrukt, wordt dit eerst vier plaatsen naar rechts geschoven: \$2B wordt \$02. Dit getal is kleiner dan \$0A en er wordt dus \$30 bij opgeteld: $\$02 + \$30 = \$32$ (ASCII-code voor

„2"). De „2" wordt afgedrukt en vervolgens wordt \$2B weer in de accu gehaald vanaf de stack. Door een AND met \$0F worden de hoogste vier bits afgezonderd: \$2B wordt \$0B. Dit getal is groter dan \$0A en dus wordt er \$37 bij opgeteld: $\$0B + \$37 = \$42$ (ASCII-code voor „B"). Als laatste wordt de „B" afgedrukt.

Tabel 1

Adres	Label	Data	Beschrijving
E000	TOETS		; routine (RB, december 1981, blz. 38?
E014	ZWA	A9 18	; schrijf zwart
E016		D0 06	
E018	WIT	A9 10	; schrijf wit
E01A		D0 02	
E01C	INV	A9 08	; schrijf geïnverteerd
E01E	SCHKL	85 AF	; bewaar in LKLEUR
E020 LIJN			; routine (RB, februari 1982, blz. 92)
E056		B9 C3 E0	; deze drie instructies dienen in
E07A		A5 AF	;de routine „Lijnen op het grafisch
E0C0		4C 69 E0	;display" te worden aangepast
E0CB	CLS		; routine „wis scherm"
E0E1	PRPNTX		; print pointer
E0EA	PRBYTE		; print byte
E0F3	PRHEX		; print hex byte
E0FE	PRASC		print ASCII-karakter
E100	LETTER		; routine (RB, juni 1979, blz. 47)
E2D4		20 58 E3	; deze vijf instructies dienen in het
E2E5		20 58 E3	;programma „Letters op het grafisch
E2FE		20 58 E3	;display" van de heer De Boer te
E338		20 58 E3	worden veranderd
E34C		65 A8	
E383	SCRUP		; scroll up
E3A0	SCRDWN		; scroll down
E3A9 INITIO			; initialiseer de in- en uitvoer
E3B4 CURS			; besturing van de cursor
E400 ASCTAB			; ASCII-karakertabel (zie tabel 2)
E542			; tot en met E71F moet \$FF bevatten
E720 MANTAB			; ASCII-tabel vervolg
E780			; tot en met E800 moet \$FF bevatten

Lijst 3

```

1120 ;
1130 ;
1140 ;
1150 ; ROUTINE SCROLL DOWN
1160 ;
1170 ;
E3A0- 38 1180 SCRNIN SEC ;
E3A1- 08 1190 CLD ;
E3A2- A5 A9 1200 LDA *SCROLL ;TREK REGELHOOGTE AF
E3A4- E5 A8 1210 SEC *REGELH ;VAN DE TELLER
E3A6- 4C 8A E3 1220 JMP SCSET ;
1230 ;
1240 ;
1250 ;
1260 ; INITIALISEER DE IN- EN UITVOER
1270 ;
1280 ;
E3A9- A9 1F 1290 INITIO LDA #1F ;
E3AB- 80 03 17 1300 STA #00 ;
E3AD- A9 FF 1310 LDA #FF ;
E3AB- 80 01 17 1320 STA #00 ;
E3B3- 68 1330 RTS ;

```

Lijst 4

```

1340 ;
1350 ;
1360 ;
1370 ; ROUTINE VOOR CURSORESTURING
1380 ;
1390 ;
E3B4- A6 A0 1400 CURS LDY *LASTX ;ZIJN DE HUIDTIGE
E3B6- A4 AE 1410 LDY *LASTY ;ZIJN DE HUIDTIGE
E3B8- E4 A1 1420 CPY *XCOR ;GELIJN AAN DE
E3BA- 00 32 1430 BNE OUT ;ALTYSTE
E3BC- C4 A2 1440 CPY *YCOR ;ZIJN DE HUIDTIGE
E3BE- 00 2E 1450 BNE OUT ;
E3C0- A5 AC 1460 LDA *COUNT ;MOET DE CURSOR AL
E3C2- E6 AC 1470 INC *COUNT ;AANDEEN
E3C4- C5 AA 1480 CMP *CSPEED ;GEINVERTEERD?
E3C6- 00 25 1490 BNE OUT-#01 ;NEE, DAN TERUG
E3C8- A9 FB 1500 INVERT LDA #FB ;ZO JA, INVERTEER
E3CA- 05 AC 1510 STA *COUNT ;ZET HANTAL PUNTJES
E3CC- E6 AB 1520 INC *CARN ;ZET ULG GOED
E3CE- C8 1530 INY ;VOLGENDE PLAATS
E3CF- A0 02 17 1540 LUS7 LDA #0 ;NACHT OP SYNC
E3D2- 18 FE 1550 BFL LUS7 ;
E3D4- 8E 00 17 1560 STX #0 ;KLOK ADRESSEN
E3D7- EE 02 17 1570 INC #0 ;
E3DA- EE 02 17 1580 INC #0 ;
E3DD- 8C 00 17 1590 STX #0 ;
E3E0- A9 08 1600 LDA #08 ;
E3E2- EE 02 17 1610 INC #0 ;
E3E5- 8D 02 17 1620 STA #0 ;
E3E8- E8 1630 INC ;
E3E9- E6 AC 1640 INC *COUNT ;VOLGENDE PUNT
E3EB- D0 E2 1650 BNE LUS7 ;
E3ED- 68 1660 RTS ;
E3EE- A5 AB 1670 OUT LDA *CARN ;MOET ER WORDEN
E3F0- AA 1680 BSR A ;GEINVERTEERD?
E3F1- 80 05 1690 BCS INVERT ;ZO JA, DOE DAT
E3F3- A5 A1 1700 LDA *YCOR ;ZET DE NIEUWE
E3F5- 85 A0 1710 STA *LASTY ;CURSOR OP
E3F7- A5 A2 1720 LDA *YCOR ;ZIJN PLAATS
E3F9- 85 AA 1730 STA *LASTY ;
E3FB- 85 AA 1740 LDA *CSPEED ;ZET DIRECT AAN
E3FD- 85 AC 1750 STA *COUNT ;
E3FF- 68 1760 RTS ;

```

Lijst 6

```

B5 00 ;HAAL EERSTE BYTE
48 ;BEWAAR
B5 01 ;HAAL TWEEDE BYTE
48 ;BEWAAR
4A 4A 4A 4A ;NEEM HOOOGSTE VIER BITS
C9 0A ;GROTER DAN 9 ?
18 ;
30 02 ;
69 07 ;ZO JA TEL ER $37 BIJ OP
69 30 ;ZO NEE, TEL ER $30 BIJ OP
85 A0 ;BEWAAR IN "LETTER"
20 00 E1 ;SUBROUTINE LETTER
68 ;HAAL TWEEDE BYTE
29 0F ;NEEM LAAGSTE VIER BITS
C9 0A ;GROTER DAN 9 ?
18 ;

```

Lijst 5

```

2440 ;
2450 ;
2460 ;
2470 ; ROUTINE VOOR FRAMTEN VAN POINTERS
2480 ; NIBBLES, BYTES EN KARAKTERS
2490 ;
2500 ;
E0E1- B5 00 2510 PRFNTR LDA *PPPOINT.X ;HAAL BYTE
E0E3- 48 2520 PHA ;BEWAAR
E0E4- B5 01 2530 LDA *PPPOINT+101.X ;HAAL VOLGENDE BYTE
E0E6- 20 E0 E0 2540 JSR PRBYTE ;DOK AF
E0E9- 68 2550 PLA ;
E0EA- 48 2560 PRBYTE PHA ;BEWAAR
E0EB- 4A 2570 LSR A ;HAAL HOOOGSTE
E0EC- 4A 2580 LSR A ;VIER BITS
E0ED- 4A 2590 LSR A ;
E0EE- 4A 2600 LSR A ;
E0EF- 20 F5 E0 2610 JSR PRHEX+002 ;
E0F2- 68 2620 PLA ;HAAL BYTE OP
E0F3- 29 0F 2630 PRHEX AND #0F ;HAAL LAAGSTE VIER BITS
E0F5- C9 0A 2640 CMP #0A ;9?
E0F7- 18 2650 CLC ;
E0F8- 30 02 2660 BMI NEX1 ;
E0FA- 69 07 2670 ADC #07 ;ZO JA, TEL $37 OP
E0FC- 69 30 2680 NEX1 ADC #30 ;ZO NEE, TEL $30 OP
E0FE- 85 A0 2690 PRASC STA *LETTER ;ZET OP SCHERM

```

Lijst 6

```

B5 00 ;HAAL EERSTE BYTE
48 ;BEWAAR
B5 01 ;HAAL TWEEDE BYTE
48 ;BEWAAR
4A 4A 4A 4A ;NEEM HOOOGSTE VIER BITS
C9 0A ;GROTER DAN 9 ?
18 ;
30 02 ;
69 07 ;ZO JA TEL ER $37 BIJ OP
69 30 ;ZO NEE, TEL ER $30 BIJ OP
85 A0 ;BEWAAR IN "LETTER"
20 00 E1 ;SUBROUTINE LETTER
68 ;HAAL TWEEDE BYTE
29 0F ;NEEM LAAGSTE VIER BITS
C9 0A ;GROTER DAN 9 ?
18 ;
30 02 ;
69 07 ;ZO JA, TEL ER $37 BIJ OP
69 30 ;ZO NEE, TEL ER $30 BIJ OP
85 A0 ;BEWAAR IN "LETTER"
20 00 E1 ;SUBROUTINE LETTER
48 ;HAAL EERSTE BYTE
4A 4A 4A 4A ;NEEM HOOOGSTE VIER BITS
C9 0A ;GROTER DAN 9 ?
18 ;
30 02 ;
69 07 ;ZO JA, TEL ER $37 BIJ OP
69 30 ;ZO NEE, TEL ER $30 BIJ OP
85 A0 ;BEWAAR IN "LETTER"
20 00 E1 ;SUBROUTINE LETTER
68 ;HAAL EERSTE BYTE
29 0F ;NEEM LAAGSTE VIER BITS
C9 0A ;GROTER DAN 9 ?
18 ;
30 02 ;
69 07 ;ZO JA, TEL ER $37 BIJ OP
69 30 ;ZO NEE, TEL ER $30 BIJ OP
85 A0 ;BEWAAR IN "LETTER"
20 00 E1 ;SUBROUTINE LETTER

```

```

30 02      ;
69 07      ;ZO JA, TEL ER $37 BIJ OP
69 30      ;ZO NEE, TEL ER $30 BIJ OP
85 A0      ;BEWAAR IN "LETTER"
20 00 E1    ;SUBROUTINE LETTER
68         ;HAAL EERSTE BYTE
48         ;BEWAAR
4A 4A 4A 4A ;NEEM HOOGSTE VIER BITS
C9 0A      ;GROTER DAN 9 ?
18         ;
30 02      ;
69 07      ;ZO JA, TEL ER $37 BIJ OP
69 30      ;ZO NEE, TEL ER $30 BIJ OP
85 A0      ;BEWAAR IN "LETTER"
20 00 E1    ;SUBROUTINE LETTER
68         ;HAAL EERSTE BYTE
29 0F      ;NEEM LAAGSTE VIER BITS
C9 0A      ;GROTER DAN 9 ?
18         ;
30 02      ;
69 07      ;ZO JA, TEL ER $37 BIJ OP
69 30      ;ZO NEE, TEL ER $30 BIJ OP
85 A0      ;BEWAAR IN "LETTER"
20 00 E1    ;SUBROUTINE LETTER

```

Bijschriften deel 2:

Lijst 1 Routine voor de besturing van de luidspreker.

Lijst 2 Routine voor de bewerking „scroll up”.

Tabel 1 Indeling van de geheugenruimte van de eerste EPROM.

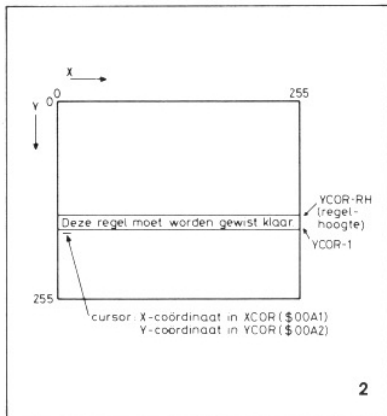
Lijst 3 Routine voor „scroll down” en de initialisatie van de in- en uitvoer.

Lijst 4 Routine voor de besturing van de cursor.

Lijst 5 Routine voor het printen van pointers, bytes, nibbles en karakters.

Lijst 6 Opengewerkte versie van de printroutine.

Afb. 1 Voorbeeld van een „nested loop” met behulp van JSR-instructies.



Wachtroutine

De in lijst 7 weergegeven wachtroutine begint met drie NOP-instructies, die niet mogen worden weggehaald, daar hier later een sprong naar een toetsophaalroutine wordt geplaatst. Na Y te hebben bewaard volgt een dubbele tellus. De waarde in X bepaalt de duur van de vertraging. De inhoud van het X-register gaat dus verloren.

Lijst 7

```

1480 ;
1490 ;
1500 ;
1510 ; ROUTINE VOOR WACHTTIJDEN
1520 JBR #E8DC
1530 ;
1540 ;
E8DC- EA 1550 WAIT NOP ;
E8DD- EA 1560 NOP ;
E8DE- EA 1570 NOP ;
E8DF- A2 04 1580 WAIT LDV #04 ;URSTE WACHTTIJD
E8E1- 96 1590 WAIT TXN ;
E8E2- 48 1600 PHA ;BEWAAR Y
E8E3- A0 FF 1610 LUS11 LDV #FF ;255
E8E5- 88 1620 LUS10 DEV ;
E8E6- D0 FD 1630 BNE LUS10 ;
E8E8- CA 1640 DEC ;
E8E9- D0 F8 1650 BNE LUS11 ;
E8EB- 68 1660 PLA ;HAAL Y
E8EC- A8 1670 TXN ;
E8ED- 60 1680 RTS ;

```

Lijst 8

```

2280 ;
2290 ;
2300 ;
2310 ; HAAK SCHERM SCHOON
2320 ;
2330 ;
E0CB- A9 1C 2340 CLS LDR #1C ;ZET MISCODE
E0CD- 80 02 17 2350 CLS1 STA PEO ;IN POORT E
E0CE- A9 00 2360 LDR #00 ;ZET CURSORLAG
E0D2- 85 A6 2370 STA XCARN ;UIT
E0D4- A0 04 2380 LDV #04 ;LUS VIERMAAL
E0D6- 20 DC E8 2390 LUS JSR WAIT ;WACHT OP TOETS
E0D9- 88 2400 DEV ;
E0DA- D0 FA 2410 BNE LUS ;
E0DC- A9 00 2420 SCRNUL LDR #00 ;HERSTEL SCROLL
E0DE- 4C 8A E3 2430 JMP SCSET ;

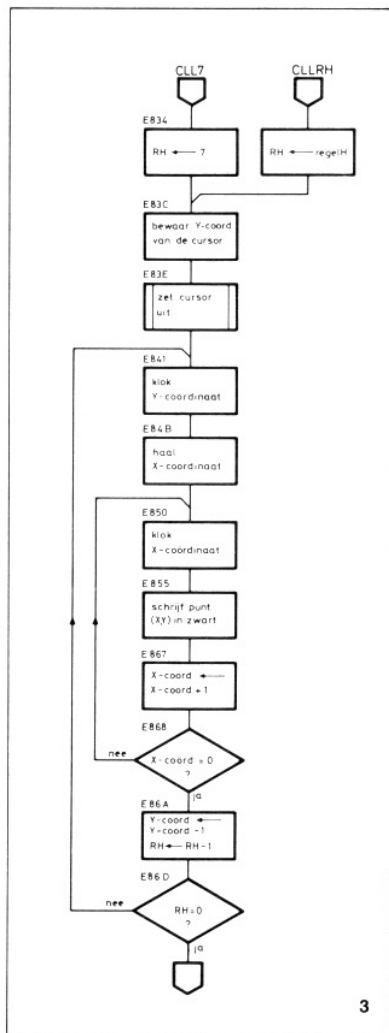
```

Lijst 9

```

0930 ;
0930 ;
1000 ;
1010 ; ROUTINE CLEAR LINE
1020 ;
1030 ;
E834- A2 07 1040 CLL7 LDV #07 ;
E836- D0 02 1050 BNE CLLRH+02 ;0000
E838- A6 A8 1060 CLLRH LDV #FEELH ;
E83A- A5 A2 1070 LDR #VCOR ;HAAL Y-COÖRDINAAT
E83C- 85 9F 1080 LUSB STA #Y ;BEWAAR
E83E- 20 EE E8 1090 JSR RESULT ;
E841- A9 02 1100 LUSC LDR #02 ;SCHRIJF OP GO
E843- 8D 02 17 1110 STA PEO ;
E846- A5 9F 1120 LDR #Y ;
E848- 8D 00 17 1130 STA PEO ;
E84B- A4 A1 1140 LDV #XCOR ;
E84D- EA 1150 NOP ;
E84E- EA 1160 NOP ;
E84F- EA 1170 NOP ;
E850- A0 02 17 1180 LUSA LDR PEO ;WACHT OP SYNC
E853- 10 FB 1190 BPL LUSA ;
E855- A9 01 1200 LDR #01 ;
E857- 8D 02 17 1210 STA PEO ;
E85A- 8C 00 17 1220 STV PEO ;
E85D- A9 03 1230 LDR #03 ;
E85F- 8D 02 17 1240 STA PEO ;
E862- A9 18 1250 LDR #18 ;
E864- 8D 02 17 1260 STA PEO ;
E867- C8 1270 INV ;VOLGENDE PUNT
E868- D0 E6 1280 BNE LUSA ;X-COÖRDINAAT = 0 ?
E86A- C6 9F 1290 DEC #Y ;VERLAAG POINTER
E86C- CA 1300 DEC ;YH = 0 ?
E86D- D0 D2 1310 BNE LUSC ;
E86F- 60 1320 RTS ;

```



Wissen van het scherm

De routine CLS in lijst 8 is ingewikkelder dan u denkt. Eerst wordt de wiscode (\$1C) in poort B geladen, waarna er moet worden gewacht. Op het eerste gezicht is dit wachten wat omslachtig geprogrammeerd. Het lijkt weinig zinvol de wachtroutine viermaal te doorlopen wanneer men net zo goed de waarde voor X viermaal zo groot had kunnen kiezen. Het antwoord is simpel. Er wordt naar WAITK gesprongen omdat er tijdens het wachten, tot het scherm is gewist, een toets kan worden ingedrukt. In deze perioden wordt dus viermaal naar de nog te bespreken toetsophaalroutine gesprongen. Bij sommige monitors kun je pas een nieuw commando via het toetsenbord geven, wanneer het vorige commando is uitgevoerd. Dit is hier niet het geval, omdat de toetsophaalroutine een ingedrukte toets zolang opslaat totdat een lopend programma is geëindigd. Wanneer het scherm is gewist, moet ook de cursor uit zijn. Dit wordt gedaan door CAAN op nul te zetten. De routine eindigt met een sprong naar SCRSET, daar na het wissen van het scherm de scrolling goed moet worden gezet. De optel-IC's in het grafisch display worden gevuld met \$00.

Wis een regel

De routine in lijst 9 is heel wat interessanter en wordt verduidelijkt aan de hand van afb. 2. De getekende cursor heeft de coördinaten (X,Y) ofte wel (XCOR, YCOR). Om de betreffende regel te wissen moeten er $[RH * 256 - XCOR]$ puntjes zwart worden gemaakt. De waarde in YCOR wordt in register YC gezet. Dit is de Y-coördinaat van het zwart te schrijven puntje (zie ook afb. 3). De cursor wordt uitgezet en de Y-coördinaat wordt naar het display gestuurd. Vervolgens wordt een aantal malen de X-coördinaat opgehoogd en de betreffende punten met zwart geschreven. Wanneer X, na het passeren van \$FF, nul is geworden, worden Y en YC met 1 verminderd. Dit herhaalt zich totdat YC gelijk aan nul is geworden.

Wis tot het einde van het scherm

Wanneer u het vorige heeft begrepen, doorziet u direct hoe de routine in lijst 10 te werk gaat. RH moet gelijk worden aan $SCROLL - YCOR - 1$. Nu zult u denken: waarom moet RH niet gelijk zijn aan $- YCOR - 1$? Stel, u bent aan het werken met de monitor en er was

gescrollt. De onderkant van het scherm heeft dan als Y-coördinaat SCROLL, want als alles bijvoorbeeld tien puntjes omhoog is geschoven komen de puntjes die boven verdwijnen er beneden bij. Omdat in routine CLEOS alles tot aan de onderkant van het scherm moet worden gewist, moet van SCROLL (aantal puntjes dat het scherm verticaal is opgeschoven) de Y-coördinaat worden afgetrokken. Daar moet dan nog eens 1 van worden afgetrokken, omdat alles vanaf de cursor moet worden gewist. Tevens wordt RH in X en SCROLL in YC gezet, wat overeenkomt met het verschuiven van de cursor naar de onderkant van het scherm. Dit is gedaan om de routine LUSB, waar naartoe wordt gesprongen, niet wezenlijk te hoeven veranderen. Hier wordt namelijk alles wat boven de cursor staat gewist tot RH regels boven de cursor. Door RH gelijk te maken aan de Y-coördinaat van de onderkant van het scherm min de Y-coördinaat van de cursor min 1 en de Y-coördinaat van de cursor gelijk te maken aan die van de onderkant van het scherm, wordt een routine „wis tot het einde van het scherm" gecreeërd, welke overeen komt met de routine „wis een regel".

Cursor uit

Het enige dat de routine in lijst 11 doet is de X- en de Y-coördinaat van de cursor in X respectievelijk Y zetten en naar routine OUT springen. Dit is een onderdeel van de cursorbesturingsroutine in lijst 4. Het X-register blijft hierbij ongeschonden.

ASCII-tabel

Geheel los van de besproken software staat de ASCII-tabel. Deze tabel met ASCII-karakters voor het grafische display is wel samen met het programma „Letters op het grafische display" uitgezonden door Hobbyscoop (een programma van de NOS), doch nooit gepubliceerd. Dat is bij deze dus gebeurd. De eerste kolom uit tabel 2 geeft de ASCII-code voor het karakter. De tweede kolom geeft de vorm van het karakter weer. De derde kolom bevat de code voor de manier waarop het karakter moet worden afgedrukt. Dit is al eerder besproken en we zullen er daarom niet verder op ingaan. Deze waarden komen in de EPROM te staan, beginnend op adres E720. De vierde kolom geeft de data die nodig is om de karaktermatrix te vullen. Eén byte stelt één kolompuntje voor. Wanneer er \$01 staat, wordt het bovenste puntje van de kolom aangezet, het onderste puntje als er \$80 staat. Het hangt dan van de waarde in de derde kolom af of het geheel moet worden gedraaid of gespiegeld. De in kolom vier afgedrukte waarden komen in de EPROM te staan, beginnend op adres E400.

Lijst 10

	1330	:	
	1340	:	
	1350	:	
	1360	:	ROUTINE CLEAR TILL END OF SCREEN
	1370	:	
	1380	:	
E870-	D8	1390	CLEOS
E871-	18	1400	CLD
E872-	A5 A9	1410	CLC
E874-	A6	1420	LDA *SCROLL
E875-	E5 A2	1430	TRV
E877-	AA	1440	SEC *YCOR
E878-	88	1450	DEV
E879-	98	1460	TVR
E87A-	4C 3C E8	1470	JMP LUSB

Lijst 11

	1690	:	
	1700	:	
	1710	:	
	1720	:	ROUTINE CURSOR UIT
	1730	:	
	1740	:	
E8EE-	8A	1750	CRSUIT
E8EF-	48	1760	TXA
E8F0-	A6 AD	1770	PHA
E8F2-	A4 AE	1780	LDX *LASTX
E8F4-	20 EE E3	1790	LDX *LASTY
E8F7-	68	1800	JSR OUT
E8F8-	AA	1810	PLA
E8F9-	60	1820	TAX

Tabel 2

ASCII-waarde		ASCII-tabel 2		ASCII tabel 1	
hex	Karakter	hex		hex	
20	spatie	82		00 00	
21	!	01		BE	
22	"	A2		0E 00	
23	#	A3		28 FE 28	
24	\$	05		48 54 FE 54 24	
25	%	05		C6 26 10 C8 C6	
26	&	05		6C 92 AC 40 A0	
27	'	01		0E	
28	(	03		38 44 82	
29	)	03		82 44 38	
2A	*	A3		44 28 FE	
2B	+	A3		10 10 7C	
2C	,	22		80 60	
2D	-	82		10 10	
2E	.	01		80	
2F	/	05		C0 20 10 08 06	
30	0	05		7C A2 92 8A 7C	
31	1	03		8C FE 80	
32	2	04		C4 A2 92 8C	
33	3	04		44 92 92 6C	
34	4	04		1E 10 10 FC	
35	5	04		9E 92 92 62	
36	6	04		7C 92 92 64	
37	7	04		C2 22 12 0E	
38	8	82		6C 92	
39	9	04		4C 92 92 7C	
3A	:	01		28	
3B	;	02		80 68	
3C	<	04		10 28 44 82	
3D	=	82		28 28	
3E	>	04		82 44 28 10	
3F	?	05		04 02 B2 12 0C	
40	@	05		7C 82 BA B2 9C	
41	A	82		FC 12	
42	B	04		FE 92 92 6C	
43	C	04		7C 82 82 44	
44	D	04		FE 82 82 7C	
45	E	04		FE 92 92 82	
46	F	04		FE 12 12 02	
47	G	04		7C 82 A2 64	
48	H	82		FE 10	

49	I	A2	82 FE
4A	J	04	60 80 80 7E
4B	K	05	FE 10 28 44 82
4C	L	04	FE 80 80 80
4D	M	A3	FE 04 08
4E	N	05	FE 04 08 10 FE
4F	O	82	7C 82
50	P	04	FE 12 12 0C
51	Q	05	7C 82 A2 42 BC
52	R	04	FE 12 32 CC
53	S	04	4C 92 92 64
54	T	A3	02 02 FE
55	U	82	7E 80
56	V	A3	3E 40 80
57	W	A3	7E 80 70
58	X	A3	C6 28 10
59	Y	A3	06 08 F0
5A	Z	05	C2 A2 92 8A 86
5B	[	04	FE 82 82 82
5C	\	05	06 08 10 20 C0
5D	]	04	82 82 82 FE
5E	↑	A3	08 04 FE
5F	—	61	F0
60	`	03	00 01 02
61	a	04	40 A8 A8 F0
62	b	04	FE 88 88 70
63	c	03	70 88 88
64	d	04	70 88 88 FE
65	e	04	70 A8 A8 10
66	f	03	08 FC 0A
67	g	24	1C A2 A2 7E
68	h	04	FE 08 08 F0
69	i	01	FA
6A	j	03	40 80 7A
6B	k	04	FE 20 50 88
6C	l	03	82 FE 80
6D	m	05	F8 08 F0 08 F0
6E	n	04	F8 08 08 F0
6F	o	82	70 88
70	p	24	FE 22 22 1C
71	q	24	1C 22 22 FE
72	r	04	F8 10 08 08
73	s	04	90 A8 A8 48
74	t	03	08 7E 88
75	u	82	78 80

76	v	A2	78 80
77	w	A3	78 80 60
78	x	A3	88 50 20
79	y	24	1E A0 A0 7E
7A	z	04	88 C8 A8 98
7B	{	03	10 6C 82
7C		01	FF
7D	}	03	82 6C 10
7E		43	80 70 08
7F		04	AA 54 AA 54

Bijschriften deel 3:

Afb. 2 Werking van de routine „wis een regel“.

Tabel 2 ASCII-tabel.

Lijst 7 Routine voor wachttijden.

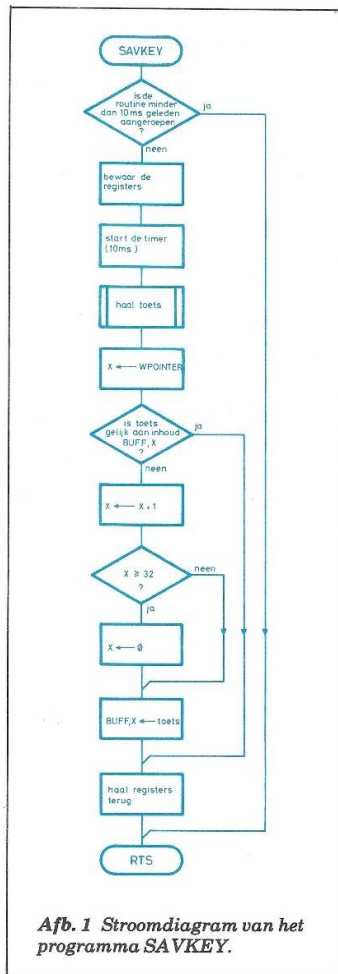
Lijst 8 Routine voor het wissen van het scherm.

Lijst 9 Routine voor het wissen van een regel.

Lijst 10 Routine voor het wissen tot het eind van het scherm.

Lijst 11 Routine om de cursor uit te zetten.

Afb. 3 Stroomdiagram van de routine „wis een regel“.



In het voorgaande hebben we een aantal routines beschreven die nodig waren voor de monitor. De behandeling van een speciale routine is echter achterwege gelaten. Deze routine, de toetsophaalroutine, wordt nu in zijn geheel besproken.

Zoals u wellicht is opgevallen, kwamen in een aantal routines (CURSOR, WAIT en WIS LIJN) een drietal NOP's voor. We zeiden dat deze later zouden worden vervangen door een JSR-instructie. Nu zouden we deze NOP's kunnen vervangen door JSR KEYIN, welke begint op adres \$E000 (deze is besproken in deel 1). We zouden ons er dan echter te gemakkelijk vanaf maken. Een sprong naar die toetsophaalroutine in programma's als CURSOR, WAIT en WIS LIJN zou erg onlogisch lijken. De toetsophaalroutine die we hebben gemaakt bestaat uit drie delen:

1. Haal toets op en bewaar hem.
2. Creëer repeat-functie.
3. Voer bewaarde toetsen uit.

Uit de praktijk is gebleken dat veel computers geen repeatfunctie kennen en dat dit een groot gemis is; denk aan de de toets „CRLF" of de „+"-toets van de KIM. Een tweede nadeel, dat gevonden is bij het uitproberen van diverse systemen, is dat, wanneer een toets wordt ingedrukt tijdens het uitvoeren van een programma

(bijvoorbeeld TREK LIJN of WIS SCHERM), deze niet wordt geïnterpreteerd door de computer. Ook wanneer meerdere toetsen na elkaar worden ingedrukt, wordt in het uiterste geval alleen de eerste uitgevoerd. De hier beschreven toetsophaalroutine bezit een automatische repeat-functie. Verder is het mogelijk om tot een totaal van zestien toetsen op te slaan, terwijl een programma loopt, die na afloop ook weer na elkaar worden uitgevoerd. Op het eerste gezicht lijkt zestien toetsen veel, maar als u bedenkt dat het tekenen van een figuur op het beeldscherm relatief lang duurt, is het toch wel gewenst om tot zestien toetsen te kunnen „onthouden".

Toetsophaalroutine

In afb. 1 ziet u het stroomdiagram van dit programma. Er wordt gebruik gemaakt van de timer. Dit is gedaan om de routine wat sneller te laten verlopen. In de routine WAIT wordt deze routine bijvoorbeeld vijfmaal aangeroepen. Dit is vrij snel achter elkaar. Om te voorkomen dat voor niets de routine opnieuw moet worden uitgevoerd is er een timer die de tijd bepaalt tussen het aanroepen van de routine en het aanroepen van de routine de keer ervoor. Werd hij minder dan 10 ms geleden aangeroepen dan heeft het geen nut om naar het toetsenbord te kijken. Is het langer dan 10 ms geleden dan worden de

registers op de stack gezet en wordt de timer opnieuw gestart. Vervolgens wordt naar het toetsenbord gekeken. Dit gebeurt twee keer om de toets te onderdrukken. Nu we de toets hebben binnengehaald, moet deze nog worden bewaard. Dit kan slechts op één manier, namelijk met behulp van een FIFO-buffer. FIFO is de afkorting voor een Engelse term: First In First Out, wat zoveel betekent als: de waarde die als eerste de buffer in ging, moet er ook weer als eerste uit. Dit in tegenstelling tot de stack van een microprocessor, wat een LIFO-buffer is (Last In First Out). Voor de FIFO is nodig:

1. Een stukje geheugenruimte.
2. Een „wijzer" die aangeeft waar de laatst uitgelezen waarde staat (RPOINT).
3. Een „wijzer" die aangeeft waar de laatst ingelezen waarde staat (WPOINT).

In de routine SAVKEY hebben we met deze twee pointers nog niets te maken. We zetten gewoon de toets op de eerst volgende lege plaats, als het tenminste een nieuwe toets is. Want stel dat de toets nog was ingedrukt. De toets zou dan opnieuw op een plaats in de buffer worden gezet. Het gevolg zou zijn dat binnen 0,1 s de hele FIFO-buffer met één toetswaarde zou zijn gevuld. Maar we zitten met nog een probleem. Stel dat de toets tweemaal achter elkaar wordt ingedrukt. De vorige toets was dan gelijk aan de nieuwe, waardoor de laatste niet in de buffer zou worden gezet. Dit euvel kan op twee manieren worden opgelost:

1. In de toetsophaalroutine wordt gewacht totdat de toets weer is losgelaten. Een eerste nadeel is echter dat door het wachten andere programma's worden opgehouden. Een tweede nadeel is dat de repeat-functie niet gemakkelijk meer mogelijk is.
2. Er wordt aangegeven dat er een nieuwe toets is ingedrukt. (In Basic komt u iets soortgelijks tegen bij de CR-LF-toets om aan te geven dat er een nieuwe regel wordt ingevoerd.) Een nadeel is hier dat er na elke toets door middel van een andere toets moet worden aangegeven dat er een nieuwe toets ingedrukt gaat worden (denk aan de „+"-toets op de KIM die het volgende adres selecteert).

Toch hebben we gekozen voor methode 2, hoewel daar de nadelen groter lijken. Dit nadeel is echter uit de weg geruimd door een nieuwe toets te creëren, namelijk de „geen"-toets. Elke keer als een toets wordt ingedrukt volgt daarna automatisch de „geen"-toets. In de toetsophaalroutine wordt namelijk \$FF in A gezet als er geen toets was ingedrukt. Door de „geen"-toets als een aparte toets te beschouwen blijft het geheel „real time" en is repeat-functie mogelijk. In plaats van 16 bytes wordt de FIFO-buffer nu wel 32 bytes lang, omdat na elke toets ook \$FF moet worden opgeslagen om aan te geven dat de toets was losgelaten. Het hele programma is nu eenvoudig te begrijpen (lijst 1).

REPEAT of SAVKEY

Het tweede programma, dat we in dit deel bespreken is heel wat lastiger te begrijpen dan het eerste. Hier zijn namelijk twee programma's in elkaar verweven. Een vlag bepaalt welk van de twee programma's wordt doorlopen:

1. Creëer de repeat-functie.

2. Voer SAVKEY uit.

Deze vlag, aangeduid met TELLER, bevindt zich op adres \$0081. Is de waarde \$00 dan wordt SAVKEY uitgevoerd, hetgeen inhoudt dat de toets wordt opgehaald die het eerst in de buffer werd gezet. Is TELLER ongelijk aan nul dan wordt REPEAT uitgevoerd. Het hierin toegepaste principe is ongeveer gelijk aan dat van de Mini-assembler voor de 6502 (RB, maart en april 1981), waarin een toets meerdere functies kreeg afhankelijk van de tijd dat deze was ingedrukt, wat door een teller werd bijgehouden. Op deze manier wordt hier bepaald hoe snel de repeat verloopt. Als TELLER gelijk is aan nul, is er geen repeat. Bij de nu volgende uitleg van het programma gaan we daar even van uit.

Lijst 1

```
1150 ; ROUTINE BEWAAR TOETS
1160 ;
1170 ;
E800- 2C 07 17 1150 SAVKEY BIT TIMER ;ZIJN DE 10 MS AL OM?
E803- 10 2E 1190 BPL RET ;
E805- 48 1200 PHA ;BEWAAR
E806- 0A 1210 TVA ;REGISTERS
E807- 48 1220 PHA ;OP DE
E808- 98 1230 TVA ;STACK
E809- 48 1240 PHA ;
E80A- A9 0A 1250 LDA #0A ;START TIMER OPNIEUW
E80C- 00 07 17 1260 STA TIMER ;(10,24 MS)
E80F- 20 20 02 1270 DEBOH JSR GETKEY ;INDIRECTE SPRONG NAAR
1280 ;TOETSOPHAALROUTINE VIA RAM
E812- 05 04 1290 STA #KEYV ;OM DENDER TE
E814- 20 20 02 1300 JSR GETKEY ;ONDERDRUKKEN
E817- 05 04 1310 CMP #KEYV ;LOSDELATEN?
E819- 00 F4 1320 BNE DEBOH ;NEEN, OPNIEUW
E81B- A6 83 1330 LDA #APOINT ;IS HET AANGEMEZEN
E81D- 00 00 02 1340 CMP BUFF,X ;KARAKTER UIT DE
E820- F0 0C 1350 BFO OUT ;BUFFER GELIJK?
E822- E8 1360 INH ;NEEN, VOLGENDE PLATS
E823- E8 20 1370 CPX #020 ;BUFFER VOL?
E825- 90 02 1380 BCC BUFNU ;NEEN, ZET TOETS OF 'GEEN TOETS'
E827- A2 00 1390 LDY #000 ;RESET BUFFER
E829- 90 00 02 1400 BUFNU STA BUFF,X ;IN DE BUFFER
E82C- 86 83 1410 STX #APOINT ;BEWAAR POINTER
E82E- 68 1420 OUT PLA ;AARL
E82F- A6 1430 TVA ;REGISTERS
E830- 68 1440 PLA ;TERUG
E831- A6 1450 TVA ;AAN DE
E832- 68 1460 PLA ;STACK
E833- 68 1470 RET RTS ;KEER TERUG
1480 ;
1490 ;
1500 ; ;LS
1504 ;
0820- 4C 00 E0 1505 GETKEY JMP TOETS ;INDIRECTE ADRESSERING VAN
1506 ; TOETSOPHAALROUTINE VIA RAM
```

Lijst 2

```
1530 ; ROUTINE VOOR REPEAT-FUNCTIE
1540 ; OF OPSLAAN VAN EEN TOETS
1550 ;
1560 ;
1570 ;
E87D- A0 00 1580 RESKEY LDY #000 ;GEEN REPEAT
E87F- 84 81 1590 RKEY STX #TELLER ;ZET TELLER
E881- A6 82 1600 SAVKEY LDY #APOINT ;ZIJN DE POINTERS
E883- E4 83 1610 CPY #APOINT ;AAN ELKAR GELIJK?
E885- F0 12 1620 BEQ REPT ;ZO JA, REPEAT
E887- E8 1630 INX ;NEEN, VOLGENDE BUFFERPLAATS
E889- E0 20 1640 CPY #020 ;BUFFER VOL?
E88C- A2 00 1650 BCC #BUF ;
E88E- 86 82 1670 NFBUF LDY #000 ;JA, RESET BUFFER
E890- B0 00 02 1680 STX #APOINT ;ZET POINTER
E893- 20 00 02 1690 LDA BUFF,X ;AARL TOETS UIT BUFFER
E896- 4C 82 E8 1700 JSR SAVKEY ;AAN DE BEWAARROUTINE
E899- A6 81 1710 REPT JMP EXIT ;TERUG
E89B- CA 1720 LDY #TELLER ;AARL TELLER
E89D- CA 1730 DEX ;AARL TELLER = 1?
E89F- F0 1C 1738 BEQ RTN ;ZO JA, TERUG
E8A1- E3 1740 INX ;AARL TELLER = 0?
E8A3- F0 02 1750 BEQ BPREV ;ZO JA, VIA RENDE
E8A5- C6 81 1760 DEC #TELLER ;NETH, REPEAT
E8A8- 20 20 02 1770 BPREV JSR GETKEY ;AARL TOETS
E8AB- 85 84 1780 STA #KEYV ;BEWAAR DE TOETS
E8AD- 20 23 02 1790 JSR SROUT ;KEER HET UIT
E8AF- 20 20 02 1800 JSR GETKEY ;AAN DE BEWAARROUTINE
E8B1- C5 84 1810 CMP #KEYV ;
E8B3- D0 F1 1820 BNE BPREV ;
E8B5- C5 80 1830 EXIT CMP #KEYV ;HET HET TOETS?
E8B7- F0 08 1840 BEQ SAVKEY ;ZO JA, OPNIEUW
E8B9- 85 80 1850 STA #KEYV ;ZET TOETS WIST
E8BB- 30 C3 1860 BMI RESKEY ;AARL HET #0?
E8BD- 4C 26 02 1870 RTN JMP TRKEY ;NEEN, LIND
```

Bijchriften deel 4:

Afb. 1 Stroomdiagram van het programma SAVKEY.

Afb. 2 Stroomdiagram van het programma RESKEY.

Lijst 1 Programma SAVKEY.

Lijst 2 Programma RESKEY.