RB
RADIO BULLETIN

maandblad voor
toegepaste elektronica
jrg 47 • nr. 6 • juni 1978
ned ƒ 3.25 – belgië F 55

2708

µP EPROM programmeer-apparaat

postfading op elke recorder

VHF-TV modulator

druppelladen van accu's

6
1978

RADIO BULLETIN

Radio Bulletin is een maandelijkse uitgave van uitgeverij De Muiderkring BV.
Nijverheidswerf 17-21, Bussum.
Postadres: postbus 10, 1400 AA Bussum (Holland).
Tel.: 02159-31851, Telex: 15171.
Postgiro 83214.
Bank: Amro-bank, Weesp, rek. nr. 48.49.54.563.

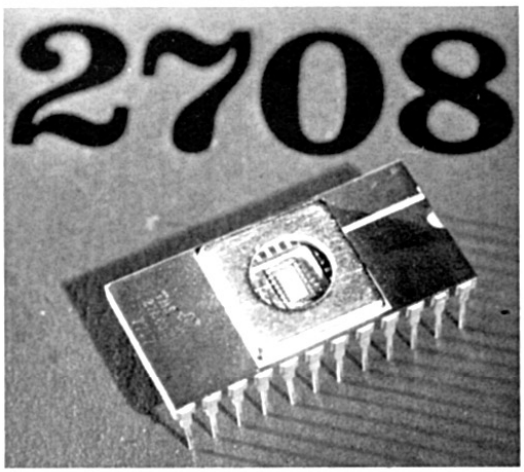
Redactie
hoofdredacteur: W. Hesselink
eindredacteur: J. G. Arends
technische redacteuren: D. M. de Boer, J. van de Pol, D. J. F. Scheper
audioredacteur: W. Jak
redactiesecr.: A. J. Vlaswinkel
techn. adv.: H. B. Stuurman

Telefonisch spreekuur, uitsluitend over in RB gepubliceerde schema's:
iedere maandag tussen 16.00 en 17.00 uur op tel. nr. 02159-31851.

Abonnementen
Abonnementsprijs *f.* 32,50 per vol kalenderjaar. Voor een abonnement, dat in de loop van het jaar wordt opgegeven, geldt een naar rato lager tarief. Abonnementen worden aan het eind van ieder kalenderjaar automatisch verlengd, tenzij uiterlijk 30 november bericht van opzegging is ontvangen. Betaling van abonnementsgeld uitsluitend d.m.v. de toegezonden *accept-girokaart*. Teneinde vertraging in de afwikkeling van correspondentie over abonnementszaken te voorkomen verzoeken wij u vriendelijk in brieven en telefoongesprekken steeds uw *abonneenummer* te vermelden. Dit nummer is afgedrukt op de adreswikkel van het blad.

Advertenties
Tarieven worden op aanvraag verstrekt. Teksten en illustratiemateriaal dienen uiterlijk op de 6de van de maand, voorafgaande aan de maand van verschijning, in het bezit te zijn van de advertentieafdeling: J. J. de Wit en mv. M. Schram-Sluyk.

RB in België
RB heeft ook een speciale Belgische editie. Voor abonnementen en advertenties wordt uitgeverij De Muiderkring in België vertegenwoordigd door: Maarten Kluwer's Internationale Uitgevers Onderneming NV, Generaal Capiaumontstraat 15, B2600 Berchem-Antwerpen, Tel. 031-36.05.24, Giro 000-0925940-75, Kredietbank 405-3035001-96.



De 2708, 8192 programmeerbare bitjes in één IC, goed om een 1k programma in op te slaan.

Inhoud

Volgende maand in RB

synthesizer voor blazers,
volledige beschrijving van een synthesizer, die kan worden 'bespeeld' als een blaasinstrument

verlichtingsautomaat voor aquarium

frequentie- en impulsteller
tevens klok en klokgelijkzetter.

frequentie- en capaciteitsmeter

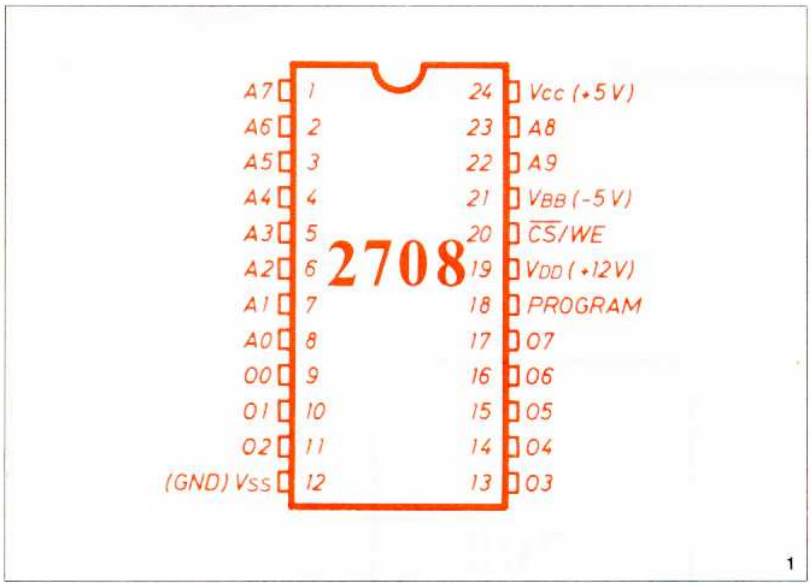
verschijnt maandelijks
juni 1978
47ste jaargang/nr. 6

EPROM PROGRAMMEER APPARAAT

MET DE KIM

J.M. v.d. PEIJL

Al eerder hebben we het in RB gehad over geheugenuitbreiding (november 1977). Toen betrof het een RAM uitbreiding, dus een extra geheugen waarin geschreven en gelezen kan worden. Waarschijnlijk heeft u ook wel eens in stilte gedacht aan extra ROM, waarin u dan de meest voorkomende programma's zet. Het voordeel is dan dat de programma's meteen na inschakelen beschikbaar zijn. Er is echter één maar: Hoe krijg je je programma in ROM? Een ROM wordt tijdens fabricage van een door de klant gewenst programma voorzien. De kosten hiervan zijn echter zó hoog, dat alleen zeer grote series op deze manier gemaakt kunnen worden. Een aantrekkelijk alternatief is de PROM, Programmable, Read Only Memory. Dit geheugen kan m.b.v. een PROM programmer door de gebruiker zelf van de gewenste inhoud worden voorzien. Eenmaal bij de PROM is het nog maar één stap verder naar de EPROM.
Ook de EPROM (Erasable PROM) kan door de gebruiker worden geprogrammeerd. Hij kan echter bovendien met ultraviolet licht weer worden gewist. Hierdoor kunnen we een EPROM met verouderde of achterhaalde programma's gewoon wissen, en opnieuw programmeren. De meest populaire EPROM is op het moment de 2708 (1k byte) die mede door de ontwikkeling van zijn opvolgers (de 2716 en 2732 met resp. 2k en 4k byte) de afgelopen tijd sterk in prijs is gedaald. Het programmeren van zo'n EPROM is een karwei dat uitstekend door de KIM kan worden opgeknapt. In dit eerste deel kunt u lezen hoe de KIM, door toevoeging van een paar eenvoudige schakelingen, de taak overneemt van een dure PROM-programmer. In een volgend artikel zullen we dan enige uitvoeringsvormen beschrijven.



1 Bovenaanzicht van de 2708 EPROM

Het programmeren

Volgens de fabrieks speciticaties moeten alle bits van de 2708 worden geprogrammeerd. Een 'lege' 2708 bevat slechts logische enen. Door het programmeren kunnen we hier nullen van maken. De 2708 heeft 10 adreslijnen, A0 tot A9 (afb. 1). Met deze 10 adreslijnen kunnen we alle 2^10 = 1024 adressen bereiken. Op ieder adres staan 8 bits. We hebben 8 datalijnen. Deze datalijnen zijn normaal op de databus van onze KIM aangesloten. Eenmaal geselecteerd (een '0' op CS/WE, pin 20) fungeren deze datalijnen als uitgang

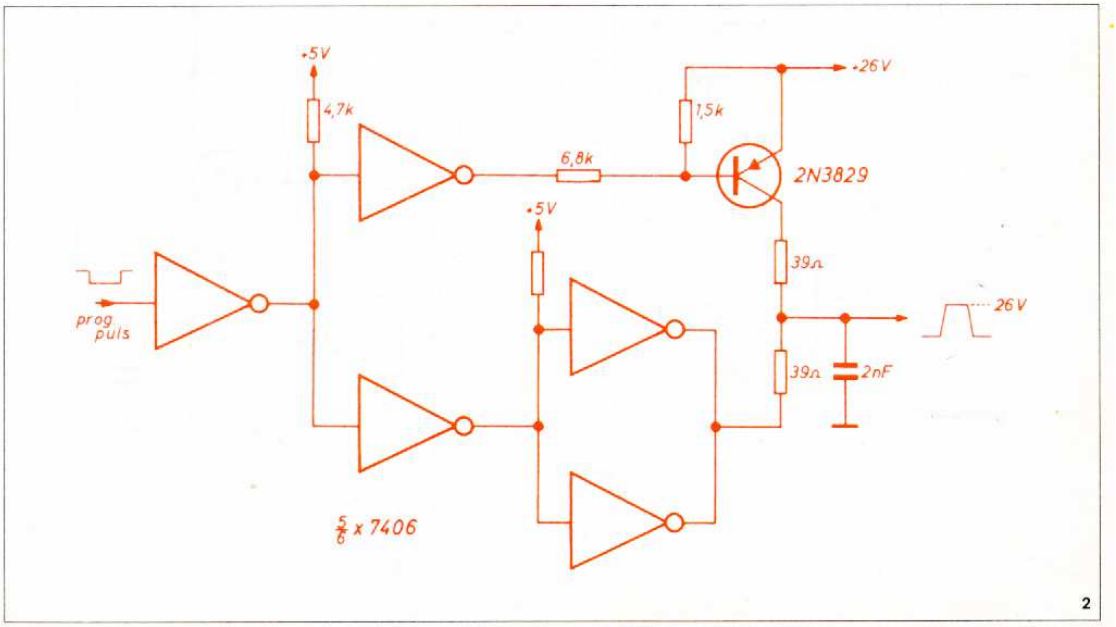
Bij het programmeren worden deze 8 datalijnen als ingang gebruikt. Zolang het programmeren duurt moet pin 20. (CS/WE) op +12 V worden gehouden.

Het programmeren geschiedt door pulsen van 26 V op de programmeer-ingang te zetten. De programmeer-ingang moet actief naar 26 V worden geschakeld en ook actief naar aarde. De flanken moeten ongeveer 1 us duren. De schakeling van afb. 2 blijkt goed te voldoen. De 7406 is een 6-voudige TTL inverter met open collectoruitgang. Op. deze uitgang mag 30 V worden gezet.

De transistor moet een snel type zijn met een geringe storage. Een 2N2905 is hier te traag. Tijdens zo'n programmeerpuls mogen data en adreslijnen niet veranderen. Alle adressen worden nu achter elkaar geprogrammeerd met elk een programmeerpuls van 1 ms.

Het programmeren van de hele ROM duurt dan ongeveer 1000 x 1 ms is 15.

Deze cyclus wordt 100 x herhaald en dan is de EPROM ingelezen. Het hele programmeerproces wordt bestuurd door onze KIM.

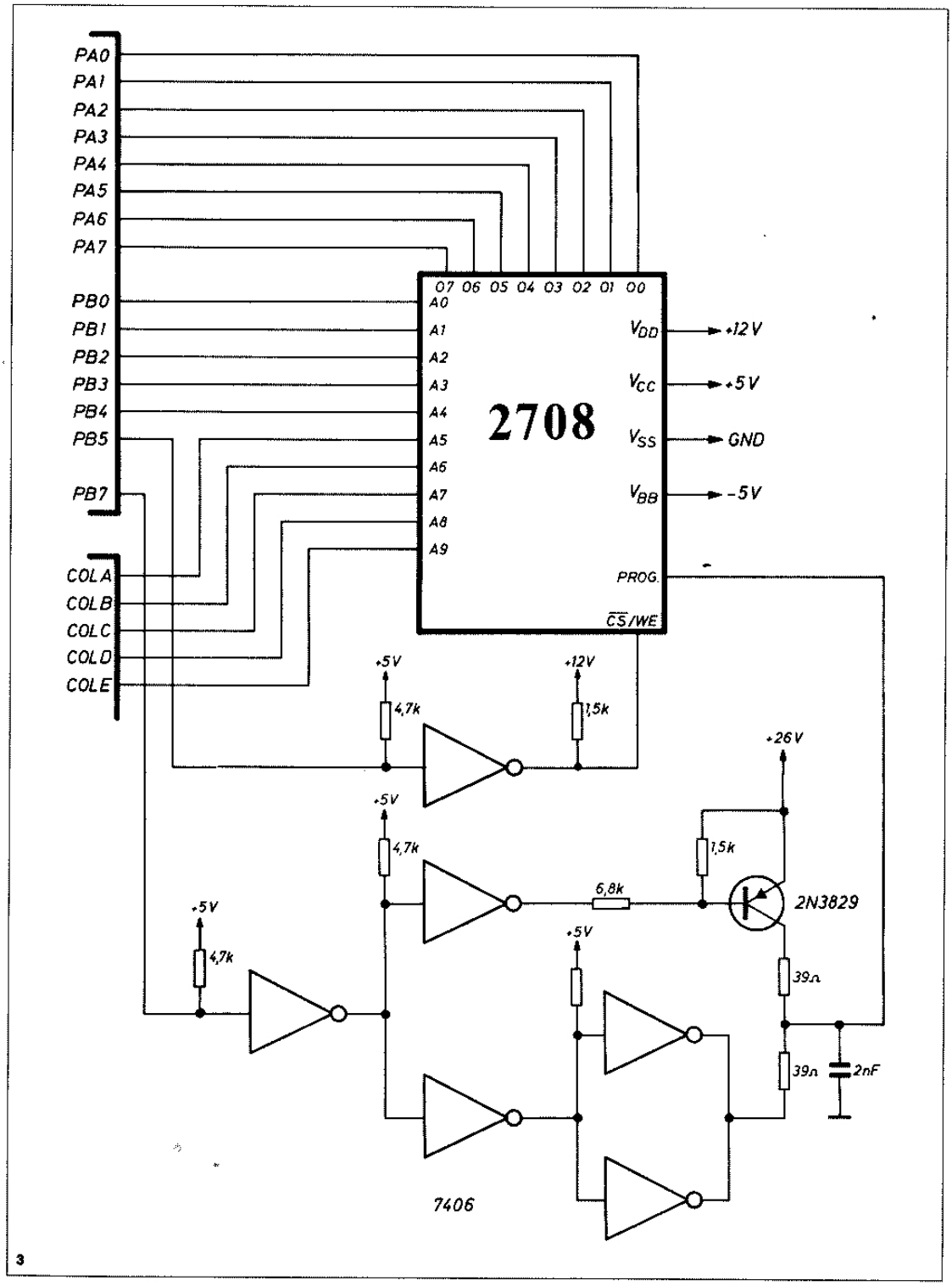


2. De interface tussen de KIM en te programmeren EPROM

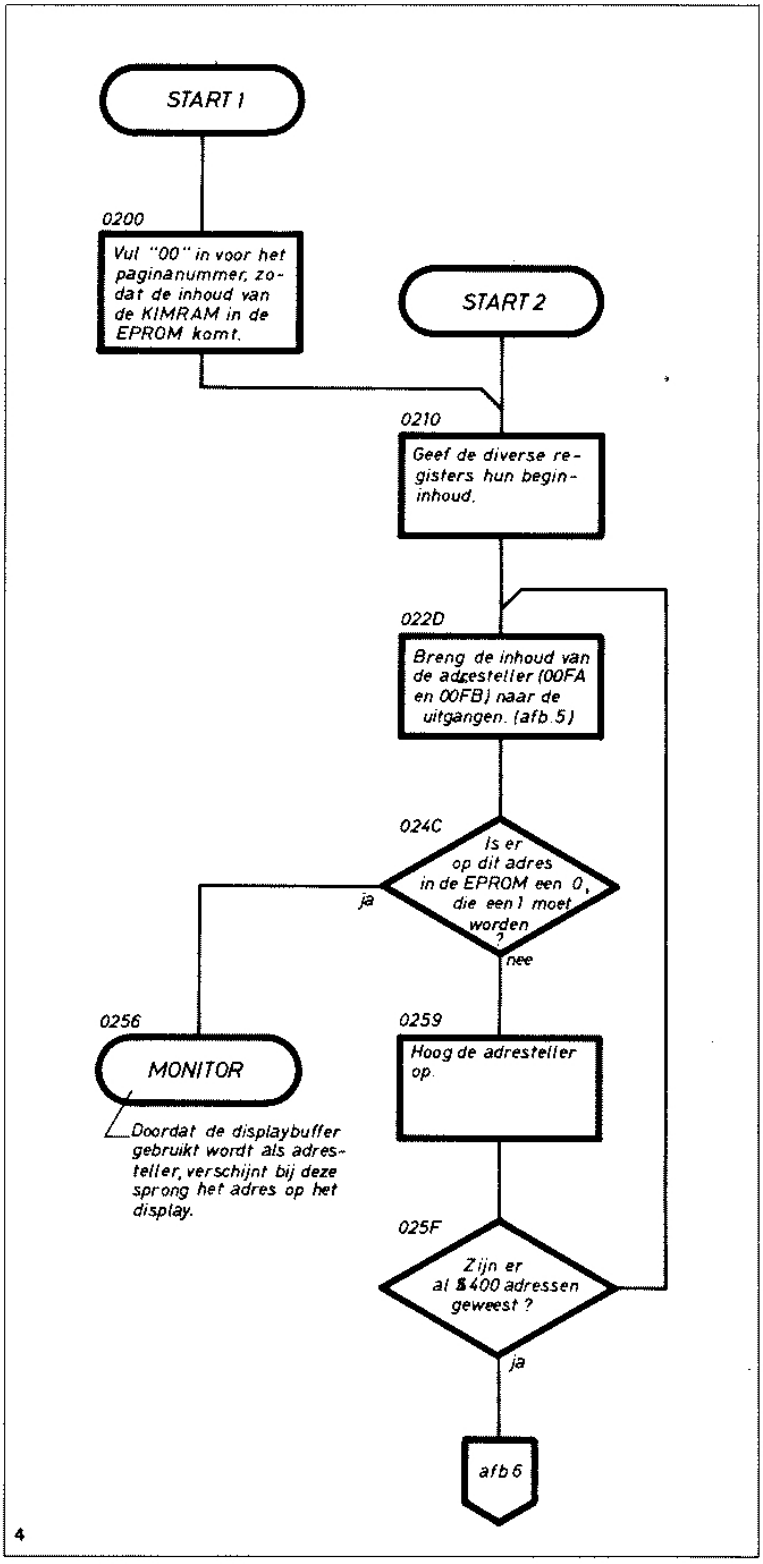
De aansluitingen bij het programmeren

We hebben gezien dat voor het programmeren 10 adreslijnen, 8 datalijnen, 1 chipselectlijn en 1 programmeerlijn, dus totaal 20 aansluitingen nodig zijn. Onze KIM heeft 15/0 lijnen. Er moeten dus nog 5 lijnen ergens anders vandaan komen. Hiervoor gebruiken we nog 5 I/O lijnen van het keyboard/display. We kunnen het display dus niet laten werken tijdens het programmeren. Maar dit was anders ook uitgesloten omdat ‘één keer door de SCAND loop van het. Monitor programma 4 milliseconden kost, terwijl de programmeerpulsen maximaal 1 ms mogen duren. De aansluitingen van de 2708 bij het programmeren staat in afb. 3. We zien dat de datalijnen zijn aangesloten op PPA (peripheral port A), de 5 laagste adreslijnen op PBO ... PB4 en de 5 hoogste adreslijnen op COLA ... COLE. De inverter in de CS/WE lijn dient om het logische 1” niveau op 12V te brengen. Tijdens het programmeren moet PB5 dus '0' blijven, waardoor CS/WE 12 V wordt. Bij het uitlezen (de inhoud van de 2708 moet worden gecontroleerd) moet PB5 '1' worden, waardoor CS/WE “0' wordt. De programmeerpulsen worden gemaakt d.m.v. PB7. De adressen worden geteld met zero-page-adressen FA en FB. Dit heeft als voordeel dat bij een eventuele fout de displaybuffer reeds gevuld is met het adres van de fout. De plaatsen 1780 t/m 1788 worden ook gebruikt door ons hulpprogramma. Dus deze adressen moeten we verder vermijden Het programma en vergelijkprogramma bestaat uit 3 delen:

A het eerste vergelijkprogramma.
B het programmeerprogramma
C het tweede vergelijkprogramma.



3 Het complete programmeer apparaat zoals het op de KIM kan worden aangesloten

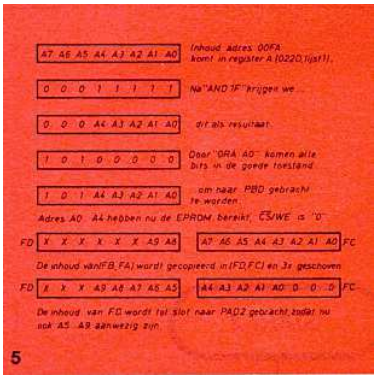


4 Flowdiagram van het eerste vergelijkprogramma. Wanneer nergens een '0' in een '1' verandert gaan we door naar het programmeerprogramma

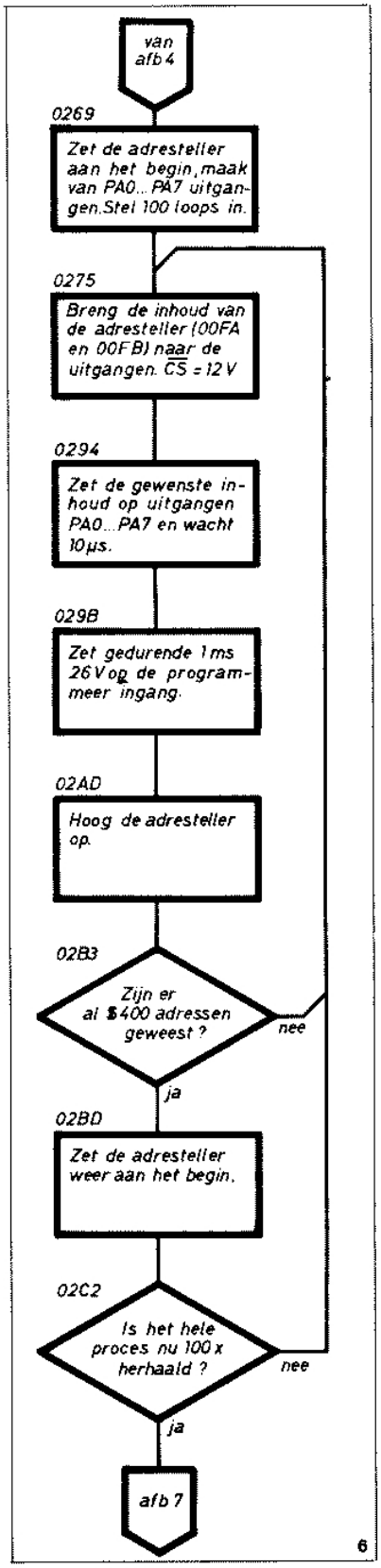
Het eerste vergelijkprogramma

Voordat het eigenlijke programmeren begint gaan we eerst kijken of het wel mogelijk is ons programma in de 2708 te schrijven. Er geldt namelijk de regel dat we nooit van een 0 een 1 mogen maken. Als de 2708 goed is gewist is het programmeren natuurlijk altijd mogelijk, maar het komt weleens voor dat er een O blijft 'hangen'. Verder hebben we de mogelijkheid ons programma te 'updaten'. Dit betekent dat we een bestaand programma in de 2708 kunnen uitbreiden, mits er op de niet gebruikte adressen 'SFF' staat. We schrijven daartoe eerst de inhoud van onze 2708, die op de programmeerplaats staat naar een stuk 1k RAM geheugen. Het nieuwe stuk programma wordt bij de RAM geschreven, en vervolgens programmeren we de EPROM opnieuw Zolang er van een '0' géén '1' gemaakt hoeft te worden, is het niet nodig de EPROM te wissen

We moeten dus voor elk adres controleren of er niet van een 0 een 1 gemaakt zou worden. Ons vergelijkprogramma start op adres 0210 + n.x 400. Deze adresnotatie lijkt misschien vreemd maar geeft aan dat het programmeerprogramma relocatable is Dat betekent dat het overal neergezet kan worden in de adresruimte van de KIM. Deze verplaatsbaarheid is bereikt door in het programma geen eigen subroutines te gebruiken en geen 'omdat we hierbij aangewezen zijn op absolute adressen. Voordat we 'ons programma starten moeten we op adres 1780 de eerste 8 bits (het bladzijdenummer) vermelden van het programma dat we in de ROM willen schrijven. Als we starten op 0200 + n.400 wordt er 00 genoteerd op 1780en daarna gaan we verder met 0210. We programmeren dan dus vanuit de KIM RAM. Het flowdiagram van het eerste vergelijkprogramma vinden we in afb.4. Het eerste blok bevat alle voorbereidingen. De interrupt wordt uitgeschakeld, we gaan naar de binaire mode om adresberekeningen te kunnen maken. FA, FB wordt onze adressenteller. Van 'onze' I/O lijnen maken we uitgangen, behalve van de PPA die op de datalijnen zit van de 2708. We willen de 2708 immers uitlezen. Daarna komen we bovenin de verify-loop (adres 0220). We beginnen om het adres dat op FB, FA 'staat naar de adreslijnen van onze 2708 te brengen.



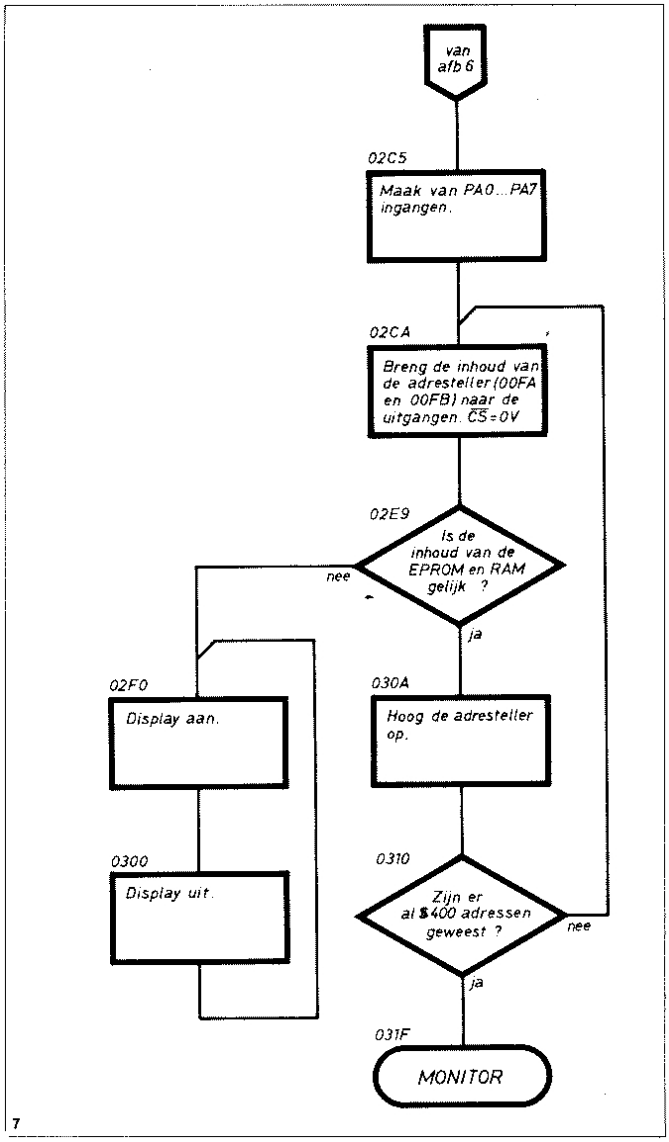
Daarvoor zijn enige manipulaties nodig (zie afb. 5). Van FA worden d.m.v. AND 1F, de 3 belangrijkste bits afgestript. Vervolgens worden 'd.m.v. ORA AO de bits 7 en 5 op een logische 1 gebracht (adres 022F en 0231). Dan wordt de accumulatorinhoud naar 1702 geschreven. Het resultaat is dat chipselect en de programpulse op 0 Volt komen en dat de 5 onbelangrijkste bits van FA naar de adreslijnen A0 - 4 van de 2708 worden gebracht Hierna worden FA en FB naar de hulpadressen FC en FD geschreven. Dat is nodig omdat we hiermee wat gaan schuiven. Na 3 x ASL en ROL FD hebben we de 5 belangrijkste adres bits op de juiste plaats staan in FD en deze worden nu doorgegeven via 1740 naar COLA - COLE. Nu kunnen we gaan vergelijken. D.m.v. LDA 1700 wordt de



inhoud van het 1e ROM adres naar de accumulator gebracht (adres 024C). Nu wordt een logische OR gedaan met de inhoud van het adres aangegeven door FB, FA. Door deze laatste bewerking mag de accumulator niet veranderen omdat dit zou inhouden dat we van een 0 een 1 willen maken. We controleren hierna dan ook met een CMP 1700. Bij een not-equal-conditie (Z=0) springen we dan naar de KIM monitor en we kunnen aflezen bij welk adres het mis gaat. Bij een equal-conditie) wordt het adres in FB, FA opgehoogd. Vervolgens kijken we of we alle $3FF adressen hebben gehad. Als dit niet het geval is gaan we verder met vergelijken en anders komen we automatisch in het eigenlijke programmeerprogramma.

Het programmeerprogramma (afb. 6)

Dit programma moet 100 x worden doorlopen. We gebruiken hiervoor het Y-register. Dit Y-register wordt dus op (hexadecimaal) 64 gezet. De inhoud van adres $1780 gaat weer naar F5. Van PPA (adres $1700) worden uitgangen gemaakt. De 2708 moet nu immers worden geprogrammeerd. De_chipselect wordt 12 V (adres 0279 en 0278) en de inhoud van het adres aangegeven door FB en FA gaat naar de datalijnen via adres $1700. Vervolgens wachten we 10 us voordat de programmeerpuls wordt gegeven. Dit is nodig volgens de specificaties van de fabrikant De programmeeruitgang wordt op 26 V gebracht (adres 0298... 02A0). We tellen 1 milliseconde af met het X-register en de programmeeruitgang gaat weer naar 0 V. We krijgen weer increment FB, FA en beginnen aan het volgende adres totdat we alle 1024 adressen hebben gehad. Dan krijgen we decrement Y-register. Dit gaat zo door totdat het Y-register nul is. We komen dan vanzelf in het tweede vergelijkprogramma.



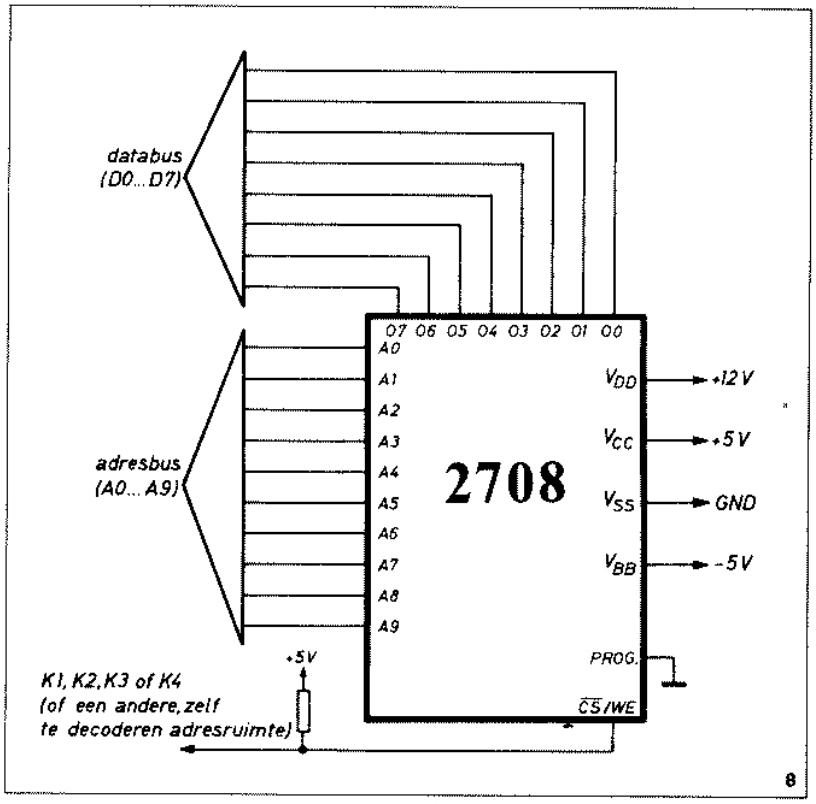
Het tweede vergelijkprogramma (afb. 7)

Dit programma is ongeveer gelijk aan het eerste programma. Maar er wordt ‘of de inhoud van de 2708 gelijk is aan die van het programma dat we hebben ingeschreven. Als dit klaar is gaat het programma terug naar het KIM monitor programma. Het gaat dan branden en het adreseen plaats hoger dan het eindadres van het programma dat is geschreven een fout wordt geconstateerd door het 2e vergelijkprogramma, gaat het display knipperen. Dit knipperen gebeurt door afwisselend 256 keer door 'SCANDS' te gaan en daarna een dubbele vertragingsloop te doorlopen. Voordat we naar SCANDS gaan moeten we eerst de I/O port 1740 weer goed zetten Dit gebeurt met de subroutine ‘INITS' van het monitorprogramma. Op F9 wordt de inhoud van 1700 geschreven. We kunnen dus ook zien wat erin de ROM gekomen is. Als we vervolgens op RESET drukken, kunnen we zien wat er op die plaats had moeten staan. Bij de oude 2708 van de firma INTEL krijgen we hier vaak foutmeldingen. Dat komt omdat dit IC bij het programmeren erg warm wordt. Als we deze 2708 later uitlezen blijkt het er altijd goed in te staan. Misschien zou een sterk ventilatortje hier helpen. Goede resultaten kreeg ik altijd met de TMS 2708 en de TMS 27L08 van Texas Instruments.

Het terugschrijfprogramma (lijst 2)

Als we bij een geprogrammeerde 2708 data willen inschrijven kunnen we dat alleen doen op een gedeelte waar jen enen staan (FF). We zetten de ROM dan op de programmeerplaats en we schrijven de ROM naar een stuk Ik RAM geheugen. Daartoe maken we gebruik van het terugschrijfprogramma. Dit programma heeft 2 startadressen nl. 100 en 110. Als we starten op 100, wordt adres $1780 op 00 gezet. We schrijven dan de inhoud van de EPROM welke op de programmeerplaats staat naar de 1K RAM van de

KIM. Het programma is vrijwel gelijk aan het te vergelijkprogramma alleen wordt hier de inhoud van de EPROM overgeschreven naar de geheugenplaats aangegeven door de inhoud van de adressen $FB en $FA. Aan het eind van dit programma springen we weer naar de KIM monitor
Door het terugschrijfprogramma te starten terwijl de programmeerplaats leeg is, komt er alleen $FF in de RAM te staan. Hierdoor zullen na het intypen van een programma alle niet gebruikte. geheugenplaatsen op $FF staan, zodat we de mogelijkheid hebben het ongebruikte deel van de EPROM later nog ‘eens te programmeren.

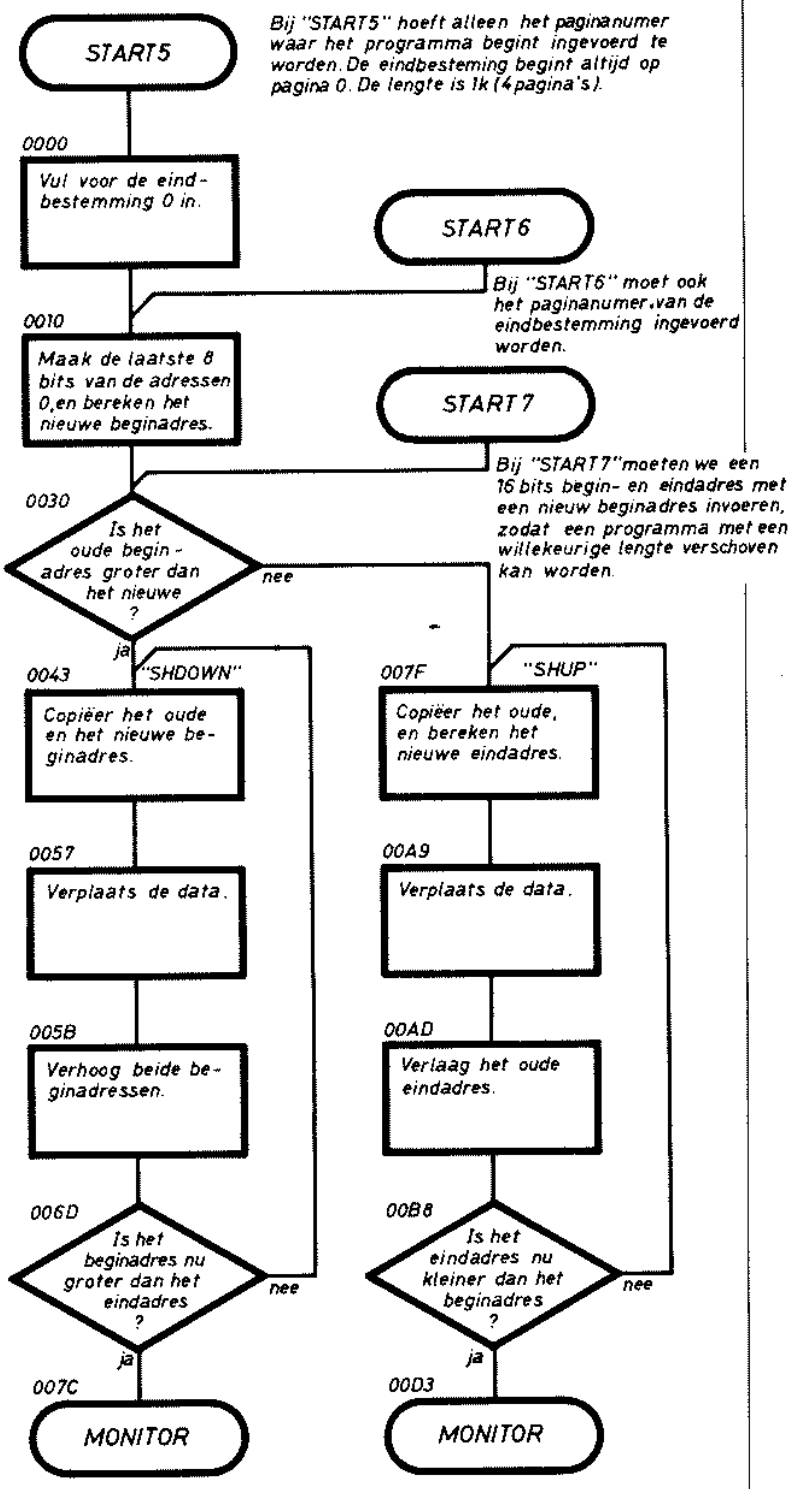


Eenmaal geprogrammeerd bevat de EPROM een programma van maximaal 1kbyte (1024 adressen). We kunnen de EPROM nu zonder verdere logica in de gedecodeerde adresruimte van 4k plaatsen (afb. 8). Door de CS/WE pin aan te ‘sluiten op K1, K2, K3 of K4 kunnen we het beginadres van de EPROM kiezen:

CS/WE aan:	adresruimte:
K1	0400...0800
K2	0800...0C00
K3	0C00 1000
K4	1000...1400

In RB november 1977 werden in deze adresruimte 2 BEM-1 kaarten geplaatst van elk 2K. In het volgende artikel (slot) zullen we een print publiceren met dezelfde aansluitingen en het zelfde formaat als de BEM-1 kaarten. Hierdoor kan op eenvoudige wijze de geheugenruimte naar behoefte gevuld worden met 4k RAM, 2k EPROM en 2k RAM of 4k EPROM, door gewoon de gewenste kaart in de socket te steken. Natuurlijk kunnen de EPROMs ook in de niet gedecodeerde adresruimte van 56k geplaatst worden, in dat geval is nog wat decodeerlogica vereist. De verschillende uitvoeringsmogelijkheden zullen in het volgende deel nog wat verder worden uitgediept.

EPROM-adres +n.400 (Hex)	gebruikte adressen	Beschrijving
0000 GO	1780	Copieer een K1 programma naar 0000-03FF (KIM-RAM) Het bladzijdenummer waar het 1k programma begint wordt op 1760 gezet
0010 GO	1780 1781	Copieer een 1k programma naar een ander stuk RAM (schrijft niet op 1780 t/m 1788). Plaats het bladzijdenummer waar het 1k programma begint op 1780 en het bladzijdenummer waar het 1k programma naartoe moet op 1781
0030 GO	1783 t/m 1788	Verschuif een stuk programma van willekeurige lengte van de ‘ene plaats naar een andere plaats. We kunnen hiermee naar ‘boven of naar beneden schuiven. 1783 en 1784: Beginadres van het te verschuiven programma. 1785 en 1786: Eindadres van het te verschuiven programma. 1787 en 1788: Beginadres waar het programma naar toe moet.
0100 GO	-	Schrijf de inhoud van een 2708 die op de programmeerplaats staat naar 0000-03FF.
0110 GO	1780	Schrijft de inhoud van een 2708 die op de programmeerplaats staat naar een stuk RAM. Plaats het bladzijdenummer waar het programma naar toe moet op 1780.
0200 GO	-	Programmeer een 2708 die op de programmeerplaats staat. Het programma dat wordt overgenomen staat op 0000-03FF
0210 GO	1780	Programmeer een 2708 die op de programmeerplaats staat. Plaats het bladzijdenummer waar het programma begint op 1780.


START5
Bij "START5" hoeft alleen het paginanumer waar het programma begint ingevoerd te worden. De eindbesteming begint altijd op pagina 0. De lengte is 1k (4 pagina's).
0000
Vul voor de eind-bestemming 0 in.
START6
Bij "START6" moet ook het paginanumer van de eindbestemming ingevoerd worden.
0010
Maak de laatste 8 bits van de adressen 0, en bereken het nieuwe beginadres.
START7
Bij "START7" moeten we een 16 bits begin- en eindadres met een nieuw beginadres invoeren, zodat een programma met een willekeurige lengte verschoven kan worden.
0030
Is het oude begin-adres groter dan het nieuwe ?
nee
ja
0043
"SHDOWN"
Copiëer het oude en het nieuwe be-ginadres.
0057
Verplaats de data.
005B
Verhoog beide be-ginadressen.
006D
Is het beginadres nu groter dan het eindadres ?
nee
ja
007C
MONITOR
007F
"SHUP"
Copiëer het oude, en bereken het nieuwe eindadres.
00A9
Verplaats de data.
00AD
Verlaag het oude eindadres.
00B8
Is het eindadres nu kleiner dan het beginadres ?
nee
ja
00D3
MONITOR

Het shiftprogramma (afb. 9)
Vooral bij het gebruik van deze EPROM programmer zal de behoefte ontstaan om programma's met een lengte van 1k te verschuiven. Ook tijdens het maken van een programma kan deze behoefte zich voordoen, zij het nu stukken met een willekeurige lengte. Voordat we het shitprogramma starten moet daarom bekend zijn:
a. Het beginadres van het te verschuiven programma op 1783 en 1784
b. Het eindadres van het te verschuiven programma op 1785 en 1786;
c Het nieuwe beginadres van het programma op 1787 en 1788

(steeds op het eerste adres de laatste cijfers, en op het tweede adres de eerste cijfers) Het eindadres van het nieuwe programma volgt natuurlijk uit deze gegevens. De adressen 1783 ..1788 gebruiken we ook als adrestellers. Omdat we indirect willen werken moeten we ook gebruik maken van zeropage adressen. Hiervoor nemen we FA, FB, FC en FD. Bij elk adres dat we verschuiven worden deze zeropage adressen weer gevuld met de waarden uit 1783 .. 1788. Het voordeel hiervan is dat we nu ook in de zeropage programma's kunnen verschuiven zonder dat onze tellers worden overschreven. Het spreekt vanzelf dat we nooit mogen shiften naar het gebied van 1783. 1788. Ons programma heeft 3 ingangen, nl 0000, 0010 en 0030. (Natuurlijk ook weer +n. 400.) Als we willen starten op 0030, moeten we vooraf de adressen 1783 ... 1788 invoeren. Als we starten ‘op 0010, verschuiven we altijd een blok van 1k. We moeten dan vooraf op 1780 aangeven waar het programma begint (paginanummer, hex) en op 1781 waar het programma naar toe moet (paginanummer, hex). Als we starten op 0000, wordt door het programma ‘00’ op adres 1781 gezet. Hierdoor kunnen we ‘een 1k blok uit het geheugen naar de KIM-RAM schrijven.Als we een programma willen verschuiven zijn er twee mogelijkheden, het programma moet naar boven (lagere adressen, of naar onderen (hogere adressen). We moeten eerst vaststellen of ons programma naar boven of naar beneden schuift. Dit kunnen we nagaan door beide beginadressen te vergelijken, dus door de inhoud van 1784, 1783 en de inhoud van 1788, 1787 van elkaar af te trekken (adressen 0030 0041). Stel dat (1784, 1783) is groter dan (1788, 1787), dan moeten we dus omhoog schuiven. In afbeelding 9 nemen we dus de linker tak van het flowdiagram. FA en FB worden gevuld vanuit 1783 en 1784, FC en FO worden gevuld vanuit 1787 en 1788. De inhoud van het adres dat op FB en FA staat gaat nu rechtstreeks naar de inhoud van het adres dat op FD en FC staat, en hiermee is de eerste regel verplaatst (adres 0057). Hierna hogen we het oude en nieuwe adres op. Vervolgens wordt gekeken of het beginadres het eindadres is gepasseerd (adres 006D), want dan moeten we natuurlijk ophouden.Als (1788, 1787) groter is dan (1784, 1783) moeten we beginnen om de inhoud van het laatste adres van ons programma te verschuiven, omdat we anders weer ons eigen programma overschrijven. In afb. 9 nemen we de rechter tak van het flowdiagram. Op FA en FB komt nu het eindadres van het te verschuiven programma (007F). Op FC en FD moet het nieuwe eindadres komen te staan. Dit adres moet eerst berekend worden door van het oude eindadres het oude beginadres af te trekken (adres

0089 ... 00A7). De aldus verkregen lengte van het programma wordt opgeteld bij het nieuwe beginadres, en het resultaat (het nieuwe eindadres) wordt op FC en FD gezet. Op adres 00A9 wordt nu de inhoud van het adres dat op FB, FA staat naar het adres dat op FD, FC staat, gebracht. Vervolgens wordt het oude eindadres verlaagd. Doordat het nieuwe eindadres steeds opnieuw wordt uitgerekend, is het niet nodig dit adres te verlagen. Zodra het eindadres het beginadres passeert, zijn we klaar en gaan terug naar het monitorprogramma.

Het programmeerprogramma in EPROM

Van de reeds besproken programma's (lijst 1, 2 en 3) is eigenlijk alleen lijst 1 strikt noodzakelijk voor het programmeren. De programma's uit lijst 2 en 3 kunnen zeer handig zijn. U kunt natuurlijk dit programmeerprogramma op cassetteband opnemen, en indien u een EPROM wit programmeren het programmeerprogramma in de KIM zetten. Dit heeft als nadeel dat minimaal ca. 290 plaatsen van de KIM-RAM nodig zijn voor dit programma. Het ligt dan ook voor de hand om als eerste programma het programmeerprogramma in EPROM te zetten. In de EPROM is dan nog ruimte genoeg voor de programma's van lijst 2 en 3. Ook kleine programma's die u zelf veel gebruikt kunt u er gelijk inzetten. U gaat dus als volgt te werk: Ten eerste typt u de programma's van list 1, 2 en 3 in de KIM. Vanaf adres $0322 kunt u zelf nog een programma dat veel voorkomt intypen. De niet gebruikte plaatsen vult u met SFF, zodat zij later nog gebruikt kunnen worden voor aanvullingen. (Bij het programmeren kunnen we van een ‘1’ een ‘0! maken, en niet omgekeerd) U zet tot slot een lege 2708 op de programmeerplaats, en start het programma vanaf adres S0200. Het programmeerprogramma programmeert nu zichzelf! Eenmaal geprogrammeerd voegen we de 2708 toe aan het geheugen van de KIM, zodat altijd direct zonder problemen geprogrammeerd kan worden.

Let op!

Wanneer u een programma in de KIM RAM heeft staan en u wit dit programma in EPROM zetten, moet u bedenken dat het programma op andere adressen terecht komt. Dit houdt in dat u voornamelijk de JMP en de JSR instructies moet corrigeren (natuurlijk niet als u naar het monitorprogramma springt).

Bij gebruik van tabellen soms ook de LDA instructies. Wanneer u over geheugenuitbreiding beschikt kunt u het programma het best ontwikkelen op de plaats waar het later in EPROM komt, dan weet u zeker dat het werkt. Ook moet u er om denken dat de EPROM ruimte niet gebruikt kan worden om tijdelijk data in op te slaan. Sommige ten hebben de aardige gewoonte om het programma zichzelf te. laten veranderen, wat bij de EPROM natuurlijk niet mogelijk is.

Vervolg

In het tweede en laatste deel zullen we nog ingaan op de extra voedingsspanning (-5 V, de 12 V en 5 V hadden we), tevens zullen enkele suggesties gegeven worden voor de praktische uitvoering van deze programmer.

Lijst 1

```
0200  A9 00     START1  LDA,imm   $00      ┐ Zet paginanummer
0202  8D 80 17          STA,abs   PAG1      │ op 0
0205  F0 09             BEQ,rel   START2    ┘ Ga naar START2
0210  78        START2  SEI,impl            ┐
0211  D8                CLD,impl            │
0212  A9 00             LDA,imm   $00       │
0214  8D 01 17          STA,abs   PADD1     │
0217  85 F9             STA,Zpage INH       │
0219  85 FA             STA,Zpage POINTL    │ Geef de diverse
021B  8D 43 17          STA,abs   PBDD2     │ registers hun
021E  AD 80 17          LDA,abs   PAG1      │ beginwaarde
0221  85 FB             STA,Zpage POINTH    │
0223  A9 1F             LDA,imm   $1F       │
0225  8D 41 17          STA,abs   PADD2     │
0228  A9 FF             LDA,imm   $FF       │
022A  8D 03 17          STA,abs   PBDD1     ┘
022D  A5 FA     VERIFY  LDA,Zpage POINTL    ┐
022F  29 1F             AND,imm   $1F       │
0231  09 A0             ORA,imm   $A0       │
0233  8D 02 17          STA,abs   PBD1      │
0236  A5 FA             LDA,Zpage POINTL    │
0238  85 FC             STA,Zpage TEMP      │
023A  A5 FB             LDA,Zpage POINTH    │ Breng het adres
023C  85 FD             STA,Zpage TMPX      │ naar PB0 . . . PB4, en
023E  A2 03             LDX,imm   $03       │ COLA . . . COLE (afb. 5)
0240  06 FC     SHIFT1  ASL,Zpage TEMP      │
0242  26 FD             ROL,Zpage TMPX      │
0244  CA                DEX,impl            │
0245  D0 F9             BNE,rel   SHIFT1    │
0247  A5 FD             LDA,Zpage TMPX      │
0249  8D 40 17          STA,abs   PAD2      ┘
024C  AD 00 17          LDA,abs   PAD1      ┐ Is er op dit adres
024F  01 FA             ORA,ind,X POINTL    │ in de EPROM een '0'
0251  CD 00 17          CMP,abs   PAD1      │ die een '1' moet worden?
0254  F0 03             BEQ,rel   GOED      ┘
0256  4C 4F 1C          JMP,abs   START     ─ Zo ja, foutmelding.
0259  E6 FA     GOED    INC,Zpage POINTL    ┐
025B  D0 02             BNE,rel   GREST1    │ Hoog adresteller op.
025D  E6 FB             INC,Zpage POINTH    ┘
025F  38        GREST1  SEC,impl            ┐
0260  A5 FB             LDA,Zpage POINTH    │
0262  ED 80 17          SBC,abs   PAG1      │ Zolang het verschil
0265  C9 04             CMP,imm   $04       │ tussen de adresteller
0267  D0 C4             BNE,rel   VERIFY    │ en het beginadres geen
0269  AD 80 17          LDA,abs   PAG1      │ $400 is, gaan we terug.
026C  85 FB             STA,Zpage POINTH    │ Zet de teller weer aan
026E  A0 64             LDY,imm   $64       │ het begin.
0270  A9 FF             LDA,imm   $FF       │ Aantal programloops
0272  8D 01 17          STA,abs   PADD1     │ Aansluitingen PA0 . . . PA7
0275  A5 FA     ST      LDA,Zpage POINTL    │ worden uitgang.
0277  29 1F             AND,imm   $1F       │
0279  09 80             ORA,imm   $80       │
027B  8D 02 17          STA,abs   PBD1      │
027E  A5 FA             LDA,Zpage POINTL    │
0280  85 FC             STA,Zpage TEMP      │ Breng adres
0282  A5 FB             LDA,Zpage POINTH    │ naar PB0 . . . PB4, en
0284  85 FD             STA,Zpage TMPX      │ COLA . . . COLE. Zet 12 V
0286  A2 03             LDX,imm   $03       │ op CS/WE.
0288  06 FC     SHIFT2  ASL,Zpage TEMP      │
028A  26 FD             ROL,Zpage TMPX      │
028C  CA                DEX,impl            │
028D  D0 F9             BNE,rel   SHIFT2    │
028F  A5 FD             LDA,Zpage TMPX      │
0291  8D 40 17          STA,abs   PAD2      ┘
0294  A1 FA             LDA,ind,X POINTL    ┐ Zet het te programmeren
0296  8D 00 17          STA,abs   PAD1      ┘ getal op de uitgang.
0299  EA                NOP,impl            ┐ Wacht 4 µs.
029A  EA                NOP,impl            ┘
029B  AD 02 17          LDA,abs   PBD1      ┐
029E  29 7F             AND,imm   $7F       │ Zet 26 volt op de
02A0  8D 02 17          STA,abs   PBD1      ┘ programmeeringang.
02A3  A2 C6             LDX,imm   $06       ┐
02A5  CA        LOOP1   DEX,impl            │ 1 ms wachten.
02A6  D0 FD             BNE,rel   LOOP1     ┘
02A8  09 80             ORA,imm   $80       ┐ Maak de programmeer-
02AA  8D 02 17          STA,abs   PBD1      ┘ ingang weer 0 volt.
02AD  E6 FA             INC,Zpage POINTL    ┐
02AF  D0 02             BNE,rel   GREST2    │ Hoog de adresteller op.
02B1  E6 FB             INC,Zpage POINTH    ┘
02B3  38        GREST2  SEC,impl            ┐
02B4  A5 FB             LDA,Zpage POINTH    │
02B6  ED 80 17          SBC,abs   PAG1      │ Zolang het verschil
02B9  C9 04             CMP,imm   $04       │ tussen de adresteller
02BB  D0 B8             BNE,rel   ST        │ en het beginadres geen
02BD  AD 80 17          LDA,abs   PAG1      │ $400 is, gaan we terug.
02C0  85 FB             STA,Zpage POINTH    │ Zet de teller weer aan
02C2  88                DEY,impl            │ het begin.
02C3  D0 80             BNE,rel   ST        │ Was dit de 100e keer?
02C5  A9 00             LDA,imm   $00       │ Zo nee, terug.
02C7  8D 01 17          STA,abs   PADD1     │ Aansluitingen PA0 . . . PA7
02CA  A5 FA     VERG2   LDA,Zpage POINTL    │ worden ingang.
02CC  29 1F             AND,imm   $1F       │
02CE  09 A0             ORA,imm   $A0       │
02D0  8D 02 17          STA,abs   PBD1      │
02D3  A5 FA             LDA,Zpage POINTL    │
02D5  85 FC             STA,Zpage TEMP      │
02D7  A5 FB             LDA,Zpage POINTH    │ Breng het adres
02D9  85 FD             STA,Zpage TMPX      │ naar PB0 . . . PB4, en
02DB  A2 03             LDX,imm   $03       │ COLA . . . COLE. Maak
02DD  06 FC     SHIFT3  ASL,Zpage TEMP      │ CS/WE weer 0 volt.
02DF  26 FD             ROL,Zpage TMPX      │
02E1  CA                DEX,impl            │
02E2  D0 F9             BNE,rel   SHIFT3    │
02E4  A5 FD             LDA,Zpage TMPX      │
02E6  8D 40 17          STA,abs   PAD2      ┘
02E9  AD 00 17          LDA,abs   PAD1      ┐ Vergelijk de inhoud
02EC  C1 FA             CMP,ind,X POINTL    │ van de EPROM met de
02EE  F0 1A             BEQ,rel   GOED2     ┘ inhoud van de RAM.
02F0  85 F9             STA,Zpage INH       ┐
02F2  20 88 1E          JSR,abs   INITS     │
02F5  8E 81 17  LOOP3   STX,abs   PAG2      │ Display aan.
02F8  20 1F 1F  AAN     JSR,abs   SCANDS    │
02FB  EE 81 17          INC,abs   PAG2      │
02FE  10 F8             BPL,rel   AAN       ┘
0300  A0 80     UIT     LDY,imm   $80       ┐
0302  88        LOOP2   DEY,impl            │
0303  D0 FD             BNE,rel   LOOP2     │ Display uit.
0305  E8                INX,impl            │
0306  D0 F8             BNE,rel   UIT       ┘
0308  F0 EB             BEQ,rel   LOOP3     ─ Terug naar display aan.
030A  E6 FA     GOED2   INC,Zpage POINTL    ┐
030C  D0 02             BNE,rel   GREST3    │ Hoog de adresteller op.
030E  E6 FB             INC,Zpage POINTH    ┘
0310  38        GREST3  SEC,impl            ┐
0311  A5 FB             LDA,Zpage POINTH    │
0313  ED 80 17          SBC,abs   PAG1      │ Zolang het verschil
0316  C9 04             CMP,imm   $04       │ tussen de adresteller
0318  D0 B0             BNE,rel   VERG2     │ en het beginadres geen
031A  A9 00             LDA,imm   $00       │ $400 is, gaan we terug.
031C  8D 03 17          STA,abs   PBDD1     │ PB0 . . . PB5 en PB7 worden
031F  4C 4F 1C          JMP,abs   START     ┘ weer ingang.
                                            ─ Naar monitorprogramma.
```

●

Lijst 2

```
0100  A9 00     START3  LDA,imm   $00      ┐ Zet het paginanummer
0102  8D 80 17          STA,abs   PAG1     │ op 0
0105  F0 09             BEQ,rel   START4   ┘ Ga naar START4.
0110  78        START4  SEI,impl           ┐
0111  D8                CLD,impl           │
0112  A9 00             LDA,imm   $00      │
0114  8D 01 17          STA,abs   PADD1    │
0117  85 F9             STA,Zpage INH      │
0119  85 FA             STA,Zpage POINTL   │ Geef de diverse
011B  8D 43 17          STA,abs   PBDD2    │ registers hun
011E  AD 80 17          LDA,abs   PAG1     │ beginwaarde.
0121  85 FB             STA,Zpage POINTH   │
0123  A9 1F             LDA,imm   $1F      │
0125  8D 41 17          STA,abs   PADD2    │
0128  A9 FF             LDA,imm   $FF      │
012A  8D 03 17          STA,abs   PBDD1    ┘
```

```
012D  A5 FA     COPY     LDA,Zpage  POINTL   ┐
012F  29 1F              AND,imm    $1F      │
0131  09 A0              ORA,imm    $A0      │
0133  8D 02 17           STA,abs    PBD1     │
0136  A5 FA              LDA,Zpage  POINTL   │
0138  85 FC              STA,Zpage  TEMP     │
013A  A5 FB              LDA,Zpage  POINTH   │  Breng het adres
013C  85 FD              STA,Zpage  TMPX     │  naar PB0 . . . PB4, en
013E  A2 03              LDX,imm    $03      │  COLA . . . COLE.
0140  06 FC     SHIFT4   ASL,Zpage  TEMP     │
0142  26 FD              ROL,Zpage  TMPX     │
0144  CA                 DEX,impl            │
0145  D0 F9              BNE,rel    SHIFT4   │
0147  A5 FD              LDA,Zpage  TMPX     │
0149  8D 40 17           STA,abs    PAD2     ═
014C  AD 00 17           LDA,abs    PAD1     ┐  Breng de inh. van de
014F  81 FA              STA,ind,X  POINTL   ┘  EPROM naar de RAM.
0151  E6 FA              INC,Zpage  POINTL   ┐
0153  D0 02              BNE,rel    GREST4   │  Hoog de adresteller op.
0155  E6 FB              INC,Zpage  POINTH   ┘
0157  38       GREST4   SEC,impl            ═
0158  A5 FB              LDA,Zpage  POINTH   ┐
015A  ED 80 17           SBC,abs    PAG1     │  Zolang het verschil
015D  C9 04              CMP,imm    $04      │  tussen de adresteller
015F  D0 CC              BNE,rel    COPY     │  en het beginadres geen
0161  A9 00              LDA,imm    $00      │  $400 is, gaan we terug.
0163  8D 03 17           STA,abs    PBDD1    │  PB0 . . . PB5 en PB7 worden
0166  4C 4F 1C           JMP,abs    START    ┘  weer ingang.  Naar monitorprogramma.
```

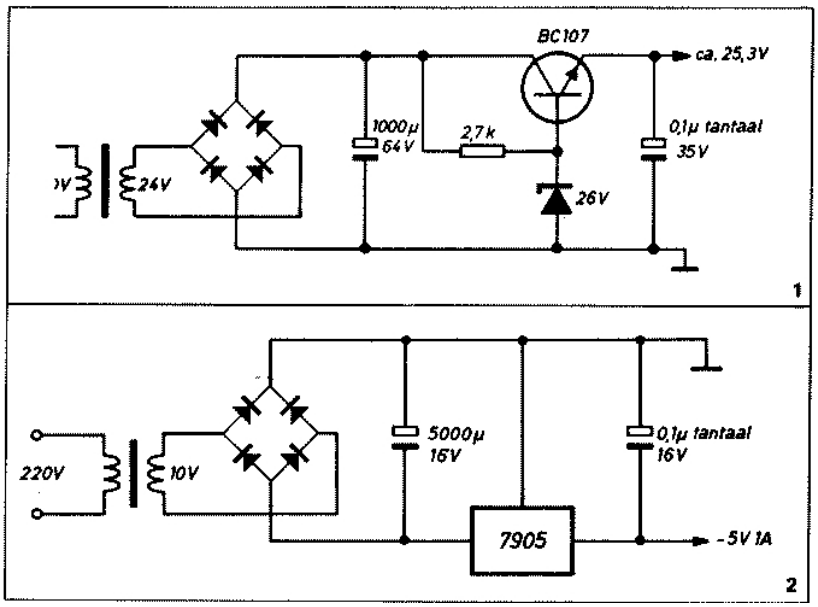
Lijst 3

```
0000  A9 00     START5   LDA,imm    $00      ┐  Zet het paginanummer
0002  8D 88 17           STA,abs    BNIEHI   │  waar het programma naar
0005  F0 0F              BEQ,rel    VERDER   ┘  toe moet op 0.
0010  AD 81 17  START6   LDA,abs    PAG2     ┐  Zet het paginanummer
0013  8D 88 17           STA,abs    BNIEHI   ┘  waar het programma naar
0016  A9 00     VERDER   LDA,imm    $00      ┐  toe moet zoals ingevoerd.
0018  8D 83 17           STA,abs    BOUDLO   │  Vul voor de laatste cijfers
001B  8D 87 17           STA,abs    BNIELO   ┘  van het adres '00' in,
001E  A9 FF              LDA,imm    $FF      ┐  en voor de laatste cijfers
0020  8D 85 17           STA,abs    EOUDLO   ┘  van het eindadres 'FF'.
0023  AD 80 17           LDA,abs    PAG1     ┐  Zet het oude beginadres
0026  8D 84 17           STA,abs    BOUDHI   ┘  zoals ingevoerd.
0029  18                 CLC,impl            ┐
002A  D8                 CLD,impl            │  Bereken het oude eind-
002B  69 03              ADC,imm    $03      │  adres, uitgaande van een
002D  8D 86 17           STA,abs    EOUDHI   ┘  lengte van 1k.
0030  78        START7   SEI,impl            ═
0031  D8                 CLD,impl            ┐
0032  A2 00              LDX,imm    $00      │  Trek het oude en nieuwe
0034  38                 SEC,impl            │  beginadres van elkaar af
0035  AD 83 17           LDA,abs    BOUDLO   │  om te bepalen of omhoog
0038  ED 87 17           SBC,abs    BNIELO   │  dan wel naar beneden moet
003B  AD 84 17           LDA,abs    BOUDHI   │  worden verplaatst.
003E  ED 88 17           SBC,abs    BNIEHI   │
0041  90 3C              BCC,rel    SHUP     ═
0043  AD 83 17  SHDOWN   LDA,abs    BOUDLO   ┐
0046  85 FA              STA,Zpage  POINTL   │
0048  AD 84 17           LDA,abs    BOUDHI   │
004B  85 FB              STA,Zpage  POINTH   │  Copieer de oude en nieuwe
004D  AD 87 17           LDA,abs    BNIELO   │  beginadressen.
0050  85 FC              STA,Zpage  TEMP     │
0052  AD 88 17           LDA,abs    BNIEHI   │
0055  85 FD              STA,Zpage  TMPX     ═
0057  A1 FA              LDA,ind,X  POINTL   ┐  Verplaats de inhoud van
0059  81 FC              STA,ind,X  TEMP     ┘  oud naar nieuw.
005B  EE 87 17           INC,abs    BNIELO   ┐
005E  D0 03              BNE,rel    GREST5   │  Hoog het nieuwe adres op.
0060  EE 88 17           INC,abs    BNIEHI   ┘
0063  EE 83 17  GREST5   INC,abs    BOUDLC   ┐
0066  D0 03              BNE,rel    GREST6   │  Hoog het oude adres op.
0068  EE 84 17           INC,abs    BOUDHI   ┘
006B  F0 0F     GREST6   BEQ,rel    MON      ═  Buiten geheugencapaciteit.
006D  38                 SEC,impl            ┐
006E  AD 85 17           LDA,abs    EOUDLO   │  Indien het beginadres
0071  ED 83 17           SBC,abs    BOUDLO   │  hoger is dan het eind-
0074  AD 86 17           LDA,abs    EOUDHI   │  adres, zijn we klaar.
0077  ED 84 17           SBC,impl   BOUDHI   ┘
007A  B0 C7              BCS,rel    SHDOWN   ═  Zo niet, terug.
```

```
007C  4C 4F 1C  MON      JMP,abs    START    ═  Klaar.
007F  AD 85 17  SHUP     LDA,abs    EOUDLO   ┐
0082  85 FA              STA,Zpage  POINTL   │  Copieer het oude
0084  AD 86 17           LDA,abs    EOUDHI   │  eindadres.
0087  85 FB              STA,Zpage  POINTH   ═
0089  38                 SEC,impl            ┐
008A  AD 85 17           LDA,abs    EOUDLO   │
008D  ED 83 17           SBC,impl   BOUDLO   │
0090  85 FC              STA,Zpage  TEMP     │
0092  AD 86 17           LDA,abs    EOUDHI   │
0095  ED 84 17           SBC,abs    BOUDHI   │
0098  85 FD              STA,Zpage  TMPX     │  Bereken het nieuwe
009A  18                 CLC,impl            │  eindadres.
009B  A5 FC              LDA,Zpage  TEMP     │
009D  6D 87 17           ADC,abs    BNIELO   │
00A0  85 FC              STA,Zpage  TEMP     │
00A2  A5 FD              LDA,Zpage  TMPX     │
00A4  6D 88 17           ADC,abs    BNIEHI   │
00A7  85 FD              STA,Zpage  TMPX     ═
00A9  A1 FA              LDA,ind,X  POINTL   ┐  Verplaats de inhoud
00AB  81 FC              STA,ind,X  TEMP     ┘  van oud naar nieuw.
00AD  AD 85 17           LDA,abs    EOUDLO   ┐
00B0  D0 03              BNE,rel    GBOR     │  Verlaag het oude adres.
00B2  CE 86 17           DEC,abs    EOUDHI   │
00B5  CE 85 17  GBOR     DEC,abs    EOUDLO   ═
00B8  38                 SEC,impl            ┐
00B9  AD 85 17           LDA,abs    EOUDLO   │  Indien het eindadres
00BC  ED 83 17           SBC,abs    BOUDLO   │  lager is dan het begin-
00BF  AD 86 17           LDA,abs    EOUDHI   │  adres, zijn we klaar.
00C2  ED 84 17           SBC,abs    BOUDHI   ┘
00C5  90 0C              BCC,rel    MONI     ═  Naar het monitorprogramma.
00C7  A9 FF              LDA,imm    $FF      ┐
00C9  CD 85 17           CMP,abs    EOUDLO   │  Zijn we niet lager dan
00CC  D0 B1     SHUP1    BNE,rel    SHUP     │  adres $0000?
00CE  CD 86 17           CMP,abs    EOUDHI   │
00D1  D0 F9              BNE,rel    SHUP1    ┘
00D3  4C 4F 1C  MONI     JMP,abs    START    ═  Zo ja, stop.
```

Deel 2

In het vorige deel werd de (zeer eenvoudige) interface besproken die nodig was om de programmeerpulsen aan de 2708 toe te voeren. Ook het programmeerprogramma, dat er voor zorgt dat de juiste pulsen op de juiste tijd komen, kwam in dat deel aan de orde. In dit laatste deel gaan we nog even in op de extra benodigde voedingsspanningen, alsmede op enige uitvoeringsmogelijkheden.

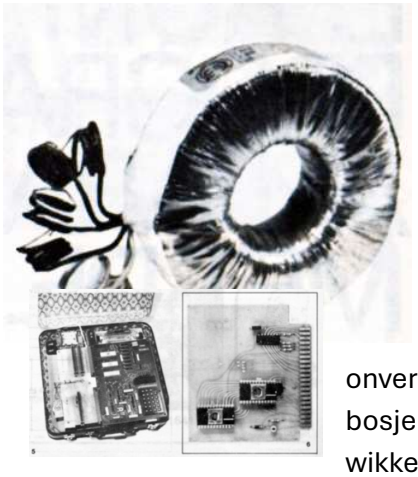


De voeding

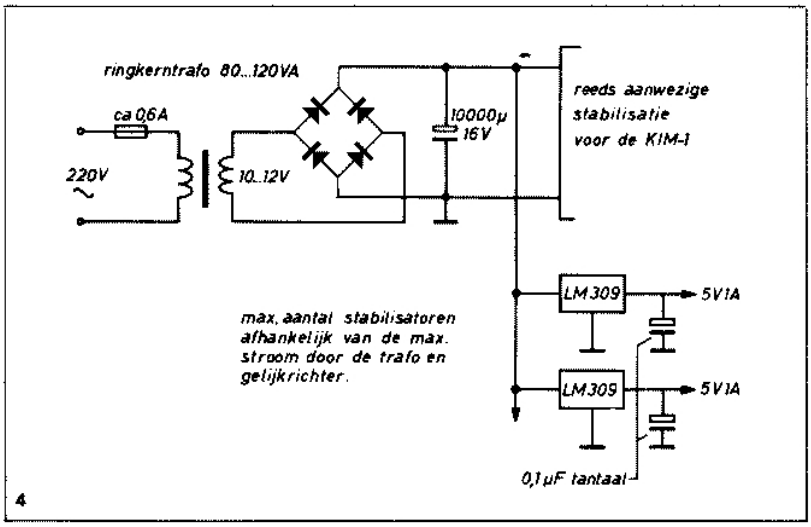
De KIM maakt zelf gebruik van +12 V 'en +5 V. Voor normaal gebruik van de 2708 is alleen nog -5 V nodig. Wanneer we gaan programmeren moet bovendien een voedingsspanning van +26 V worden gemaakt. Veel stroom trekt de 2708 niet. De 26 V voeding hoeft maximaal slechts 20 mA te leveren. Als we echter toekomstige ontwikkelingen in het oog houden (de 2716, waar ze zeker op terug komen), moeten we de voeding geschikt maken voor minimaal 30 mA. In afb. 1 is een mogelijk schema van deze voeding gegeven. De extra -5V voedingsspanning moet per 2708 ca. 45 mA kunnen leveren. In verband met toekomstige uitbreidingen is het echter verstandig de -5 V geschikt te maken voor minimaal 1 A. Dit kan het makkelijkst op de wel bekende manier volgens afb. 2. De +5 V en de +12V voeding van de KIM zullen de extra stroom voor de 2708 wel kunnen leveren (resp. 10 mA en 65 mA per 2708).

De trafo en de brugcel

Wanneer de voeding verzwaard moet. Worden doordat u uw systeem gaat uitbreiden is de trafo meestal een probleem. Door de relatief hoge prijs van een trafo wordt zijn capaciteit maar al te vaak zo gekozen dat nét de verlangde stroom kan worden geleverd. Het gevolg is dat voor iedere uitbreiding 'opnieuw een trafo moet worden gekocht om de extra stroom/spanning te leveren. Bovendien moet voor elke trafo ook weer een plaatsje



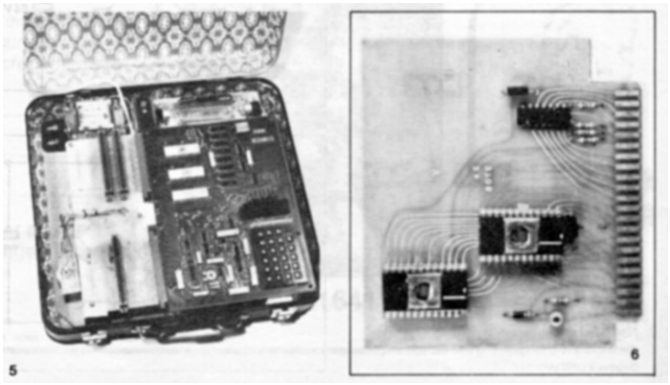
worden gevonden. In dit geval is opeens weer een spanning van -5 V en +26 V nodig, dus inderdaad wéér een trafo. Om voor eens en altijd met dit probleem af te rekenen moet u zorgen voor een trafo van zo'n 80 tot 120 VA. Het best kunt u dan een ringkern-trafo nemen (afb. 3). Deze trafo's zijn wel iets duurder, maar hebben buiten de bekende pluspunten (klein, lage verliezen, geen strooiveld) het grote voordeel dat ze makkelijk van extra windingen kunnen worden voorzien. Een onverwachte 'spanning (zoals nu +28 V) kost dan alleen een bosje wikkeldraad en wat tijd (ongeveer 240 windingen). De wikkelruimte is haast onbeperkt, en u hoeft geen blik te demonteren. Wel moet u er op letten dat u niet boven het max. schijnbare vermogen uitkomt, maar met 80 of 120 VA kunt u voorlopig voort. Ook is het verstandig de brugcel voldoende zwaar te maken (voor de +5 V voeding ca. 5.10 A). Voor elke uitbreiding kan dan gewoon een 5 V stabilisator worden toegevoegd (bv, LM309 of 7805). Een en ander is nog eens getekend in afb. 4



Uitvoeringsmogelijkheden

De auteur heeft voor deze PROM-programmer een heel eigen systeem ontwikkeld. Als basis heeft hij een moederprint gemaakt, welke rechtstreeks op de KIM past (afb. 5). Deze moederprint heeft nog 3 connectoren waar de uitbreidingskaartjes (afb. 6) inpassen. Elk kaartje heeft een geheugencapaciteit van 2k EPROM. Door deze kaartjes in de bovenste 2 connectoren te steken, komt de in totaal 4k EPROM in de gedecodeerde adresruimte van de KIM (adres 0400.....013FF). Ook is het mogelijk in deze connectoren een 2k RAM kaartje te steken.

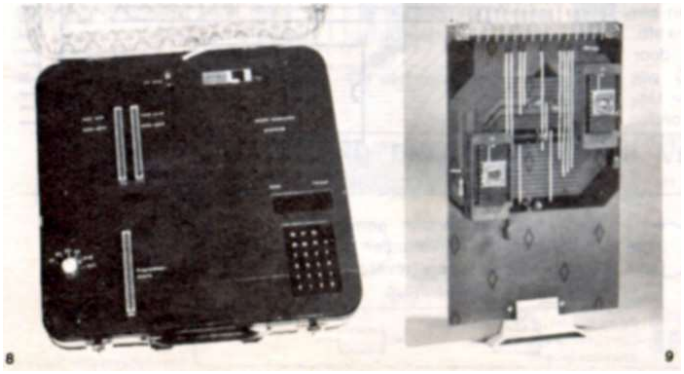
De EPROM kaartjes bevatten naast de 2 EPROMs ook nog een 7406, met een paar weerstanden. De in- en uitgangen van deze 7406 worden apart naar buiten gevoerd.



Wanneer het kaartje in de onderste connector wordt gestoken (programmeerplaats), worden de in- en uitgangen van de 7406 zó doorverbonden dat het geheel de interface vormt tussen de KIM en de te programmeren EPROM. (De EPROM moet dan in de onderste voet op het kaartje zitten.) De adres- en datalijnen zijn nu met I/O van de KIM verbonden, en het programmeren kan beginnen.
Wanneer het EPROM-kaartje normaal wordt gebruikt (in een van de twee bovenste connectoren) blijft de 7406 niet ongebruikt, in dit geval worden de inverters zo geschakeld dat er een mogelijkheid tot vector-selectie ontstaat (afb. 7). Hierdoor hebben we de mogelijkheid beide interrupt vectors en de resetvector boven in een willekeurig 1k blok te leggen. Afb. 8 geeft tenslotte het eindresultaat: een compact koffertje, met daarin een compleet ontwikkelingssysteem.

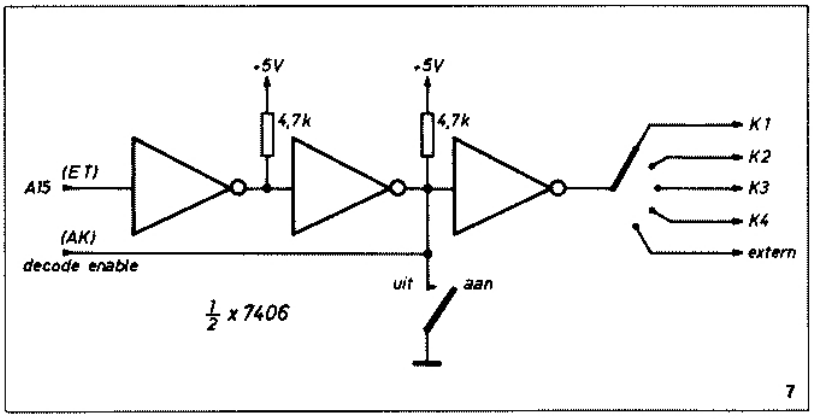
R.B. uitvoering

Hoewel bovenstaande beschrijving tot een vrij compleet geheel leidt, zijn er toch enkele

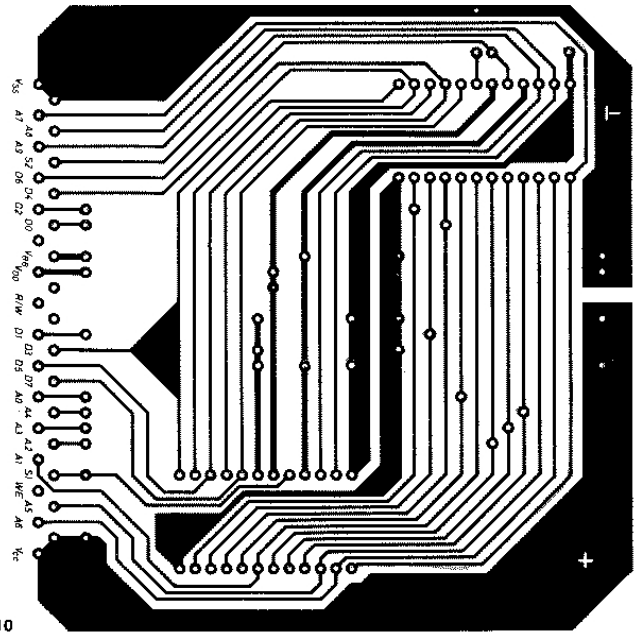


bezwaren. We denken aan de ijverige hobbyist, die een mooie behuizing voor zijn KIM heeft gemaakt, en de moederprint hier onmogelijk in kwijt kan. Of de trouwe AB-lezer, die in de gedecodeerde 4k ruimte al 2 connectoren heeft geplaatst voor de BEM-1 kaart (RB nov77).
Daarbij komt dan nog dat de auteur door gebruik van dubbelzijdige contacten op de print, verplicht was ook dubbelzijdige print te gebruiken. Nu is het zelf vervaardigen van dubbelzijdige print voor de meeste hobbyisten niet weggelegd. Ook wanneer een dergelijke print door ons verkocht zou worden, zou hij onnodig ‘duur worden. Daarom ontwierp de RB redactie een enkelzijdige eurocard, met dezelfde connector, en dezelfde aansluitingen als de BEM-1 (afb. 9) Door het gebruik van enkelzijdige print werden enkele doorverbindingen noodzakelijk, voor de prijsbewuste hobbyist zal dit nauwelijks een bezwaar zijn. Overigens kan deze enkelzijdige print makkelijk dubbelzijdig uitgevoerd worden door de doorverbindingen op film te tekenen. De print layout kunt u in afb. 10 vinden, de bijbehorende componenten opstelling in afb. 11. Doordat de BEM-kaarten gebruik maken van een 31-polige connector, terwijl de auteur een 40-polige connector gebruikte, was het niet mogelijk om ook de 7406 op deze print te zetten. Strikt

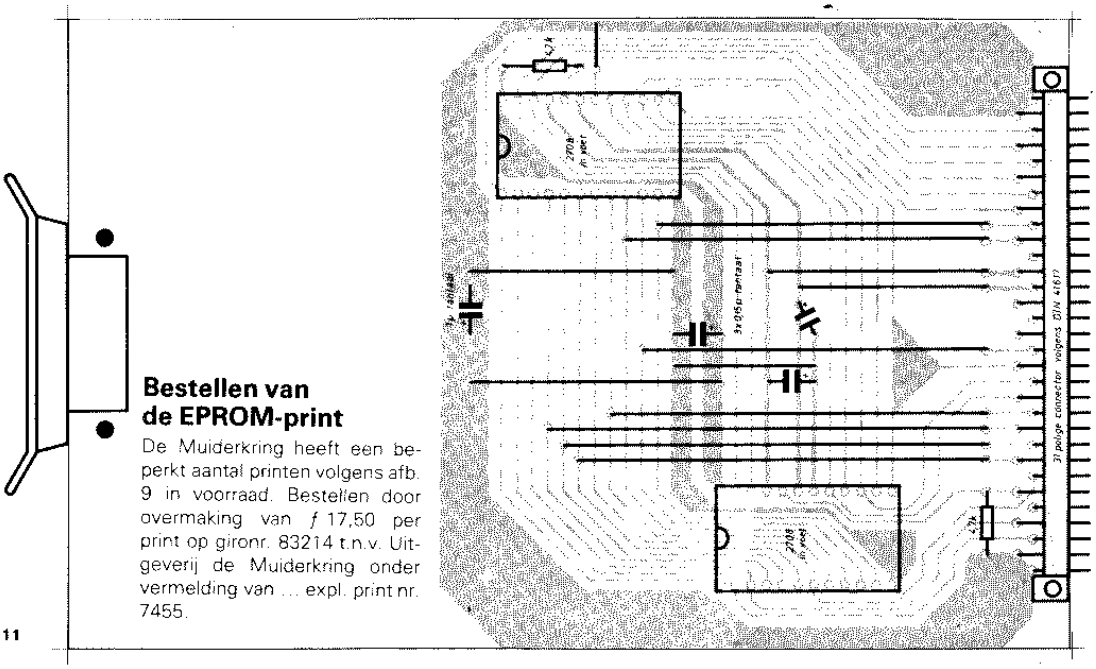
genomen is dat ook helemaal niet nodig. De 7406 wordt gebruikt in de interface tussen de KIM en de te programmeren EPROM. Deze interface kan echter ook makkelijk op een stukje Veroboard worden gemaakt. Ergens in of op de behuizing van uw KIM is dan altijd wel een plekje te vinden waar u een 24-polige L.C. voet kwijt kunt. U heeft dan een aparte 'programmeer voet', waar u de 2708 kunt programmeren. Deze programmeervoet kunt u dan in de toekomst tevens gebruiken als connector voor (zelfgebouwde) appraten welke gebruik maken van de intelligentie van de KIM. Op de programmeervoet komen immers 18 door de KIM te programmeren TTL niveaus uit (in- en uitgangen), met daarbij nog één signaal tussen +12V en 0V en een signaal tussen de +26 en 0V. Ook zijn via deze voetnog 3 voedingsspanningen te betrekken. In de luxe uitvoering kunt u voor die programmeervoet dan een voetje met hefboom nemen (rij duur), zodat het insteken en het uittrekken altijd probleemloos kan verlopen. Een vector select is met dit systeem niet gerealiseerd, maar kan door toevoeging van één 7406, een schakelaar en wat weerstanden worden gemaakt (afb. 7).



Samengevat is de RB-uitvoering van deze EPROM-programmer:
In de behuizing van de KIM bevinden zich 2 connectoren. In elke connector kan óf een BEMI (2K RAM) óf een EPROM kaart (2K EPROM) gestoken worden. Op de behuizing bevindt zich een programmeervoet, waarin een 2708 geprogrammeerd kan worden. Deze programmeervoet dient tevens als connector voor randapparatuur.



10



Bestellen van de EPROM-print

De Muiderkring heeft een beperkt aantal printen volgens afb. 9 in voorraad. Bestellen door overmaking van *f* 17,50 per print op gironr. 83214 t.n.v. Uitgeverij de Muiderkring onder vermelding van ... expl. print nr. 7455.

11