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.

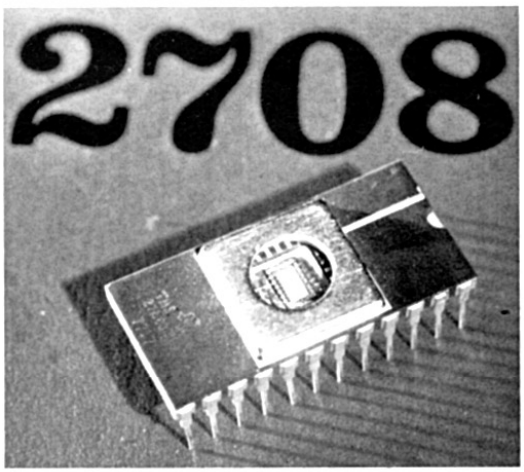
Redactie
hoofdredacteur: W. Hesselink
eindredacteur: J. G. Arends
technische redacteuren: D. M. de Boer, J. van de Pol, D. J. F. Scheper
audioredacteur: W. Jak
redactiesecr.: A. J. Vlaswinkel
techn. adv.: H. B. Stuurman

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.

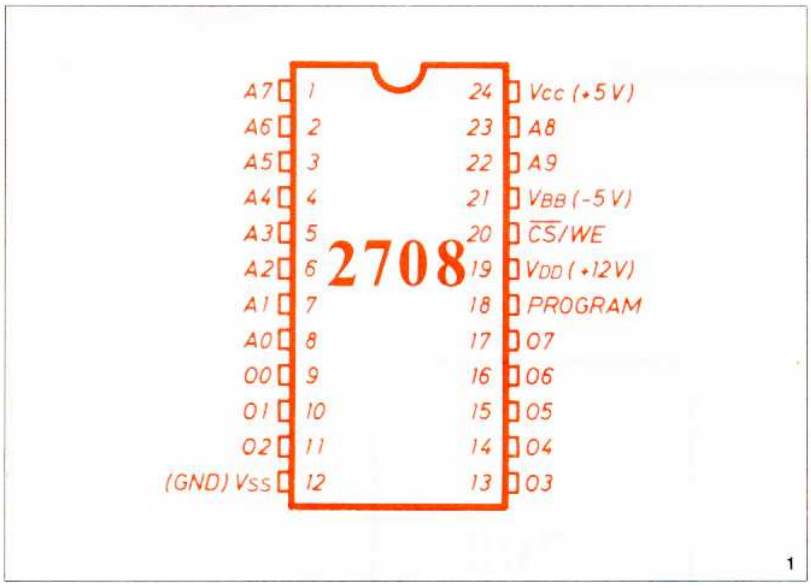
frequentie- en capaciteitsmeter

verschijnt maandelijks
juni 1978
47ste jaargang/nr. 6

EPROM PROGRAMMER WITH THE KIM

J.M. v.d. PEIJL

We have already discussed memory expansion in RB (November 1977). That concerned a RAM expansion, i.e. additional memory in which data can be written and read. You have probably also thought about additional ROM, in which you could put the programs you use most often. The advantage is that the programs are immediately available after switching on. There is, however, one problem: how do you get your program into ROM? A ROM is programmed during manufacture with a program specified by the customer. The cost is so high that this method is only practical for very large production runs. An attractive alternative is the PROM, Programmable Read Only Memory. Using a PROM programmer, the user can program this memory himself with the desired contents. Once we have reached the PROM, it is only one step further to the EPROM.
The EPROM (Erasable PROM) can also be programmed by the user. In addition, it can be erased again using ultraviolet light. Thus an EPROM containing obsolete or outdated programs can simply be erased and reprogrammed. The most popular EPROM at present is the 2708 (1K byte), which has fallen considerably in price recently, partly because of its successors, the 2716 and 2732, with 2K and 4K bytes respectively. Programming such an EPROM is a task that the KIM can handle very well. In this first part you can read how, by adding a few simple circuits, the KIM takes over the job of an expensive PROM programmer. In a subsequent article we will describe several practical versions.



1. Top view of the 2708 EPROM

Programming.

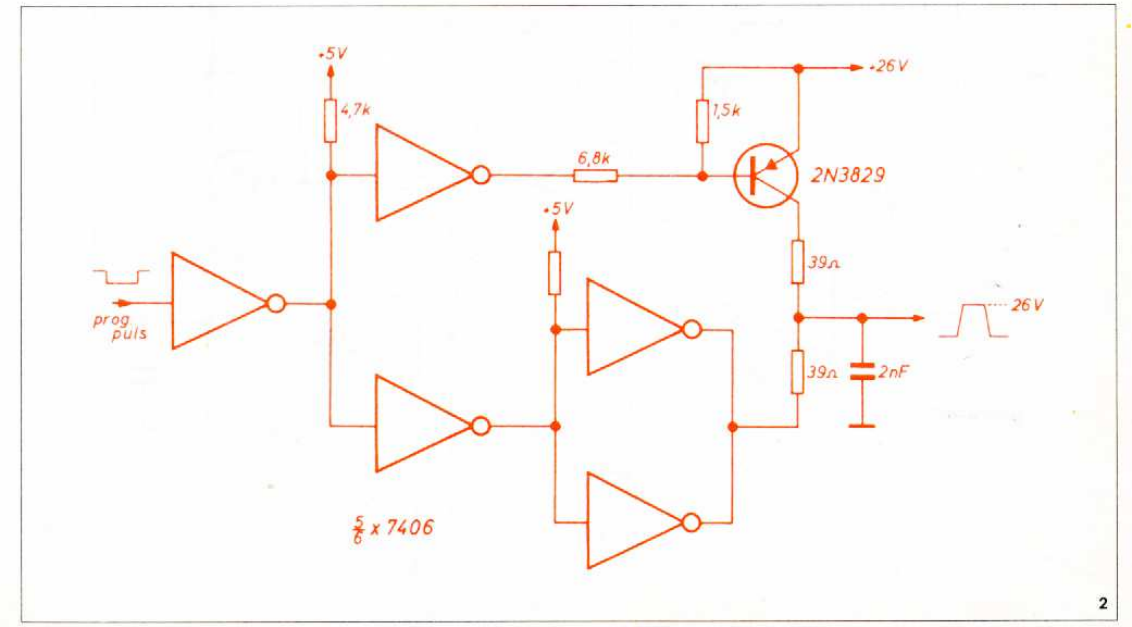
According to the manufacturer's specifications, all bits of the 2708 must be programmed. An "empty" 2708 contains only logical ones. By programming, we can turn these into zeroes. The 2708 has 10 address lines, A0 to A9 (Fig. 1). With these 10 address lines we can access all 2^10 = 1024 addresses. Each address contains 8 bits. We have 8 data lines. These data lines are normally connected to the data bus of our KIM. Once selected (a "0" on CS/WE, pin 20), these data lines act as outputs. During programming these 8 data lines are used as inputs. During programming, pin 20 (CS/WE) must be held at +12 V.

Programming is done by applying 26 V pulses to the programming input. The programming input must be actively switched to 26 V and also actively to ground. The edges should take approximately 1 µs. The circuit in Fig. 2 has proved to work well. The 7406 is a six-fold TTL inverter with open-collector outputs. Up to 30 V may be applied to this output.

The transistor must be a fast type with low storage time. A 2N2905 is too slow here. During such a programming pulse, the data and address lines must not change. All addresses are programmed consecutively, with a 1 ms programming pulse for each.

Programming the entire ROM then takes approximately 1000 × 1 ms is 1 s.

This cycle is repeated 100 times, after which the EPROM has been programmed. The entire programming process is controlled by our KIM.

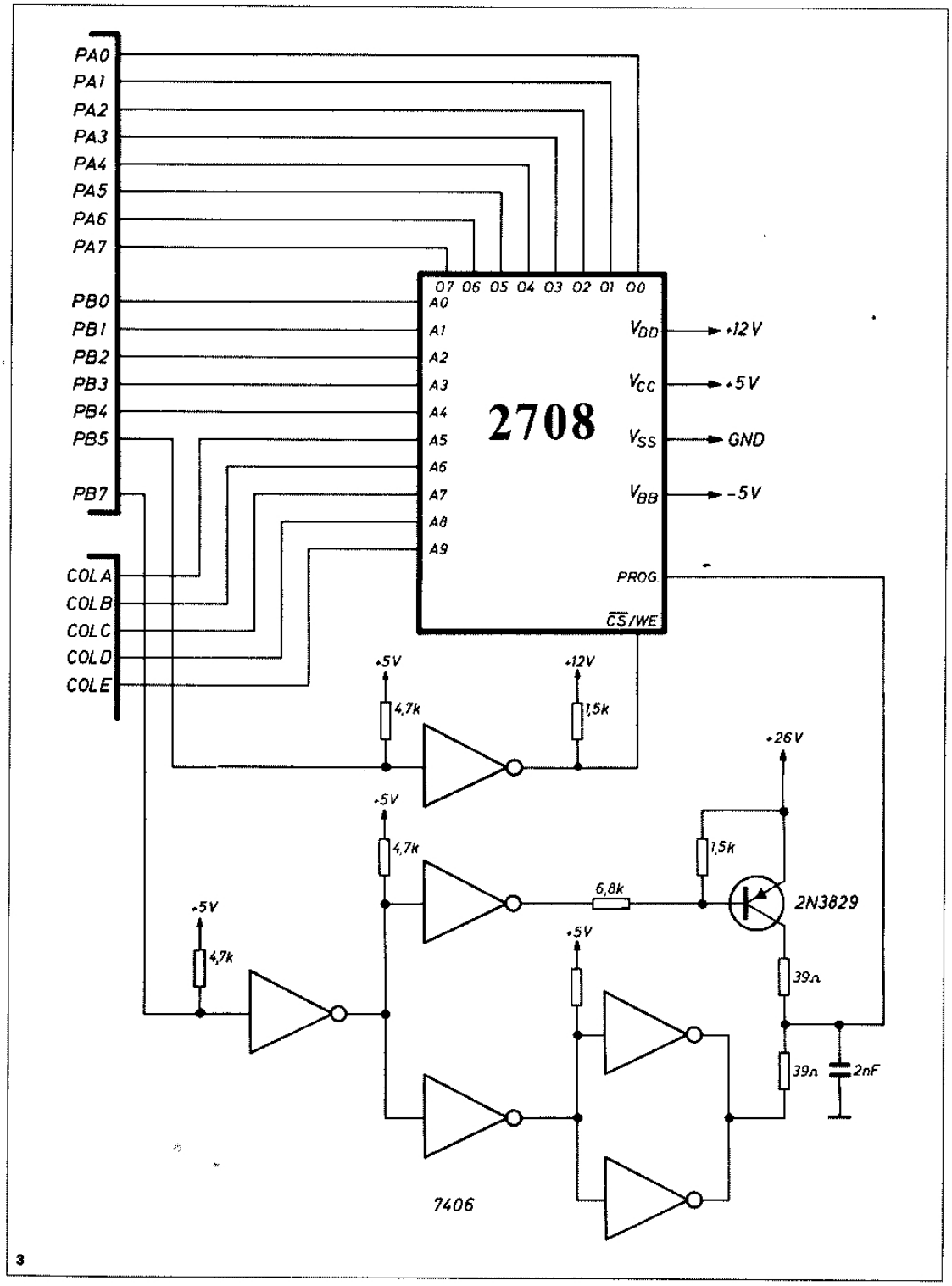


2, Interface between the KIM and the EPROM to program

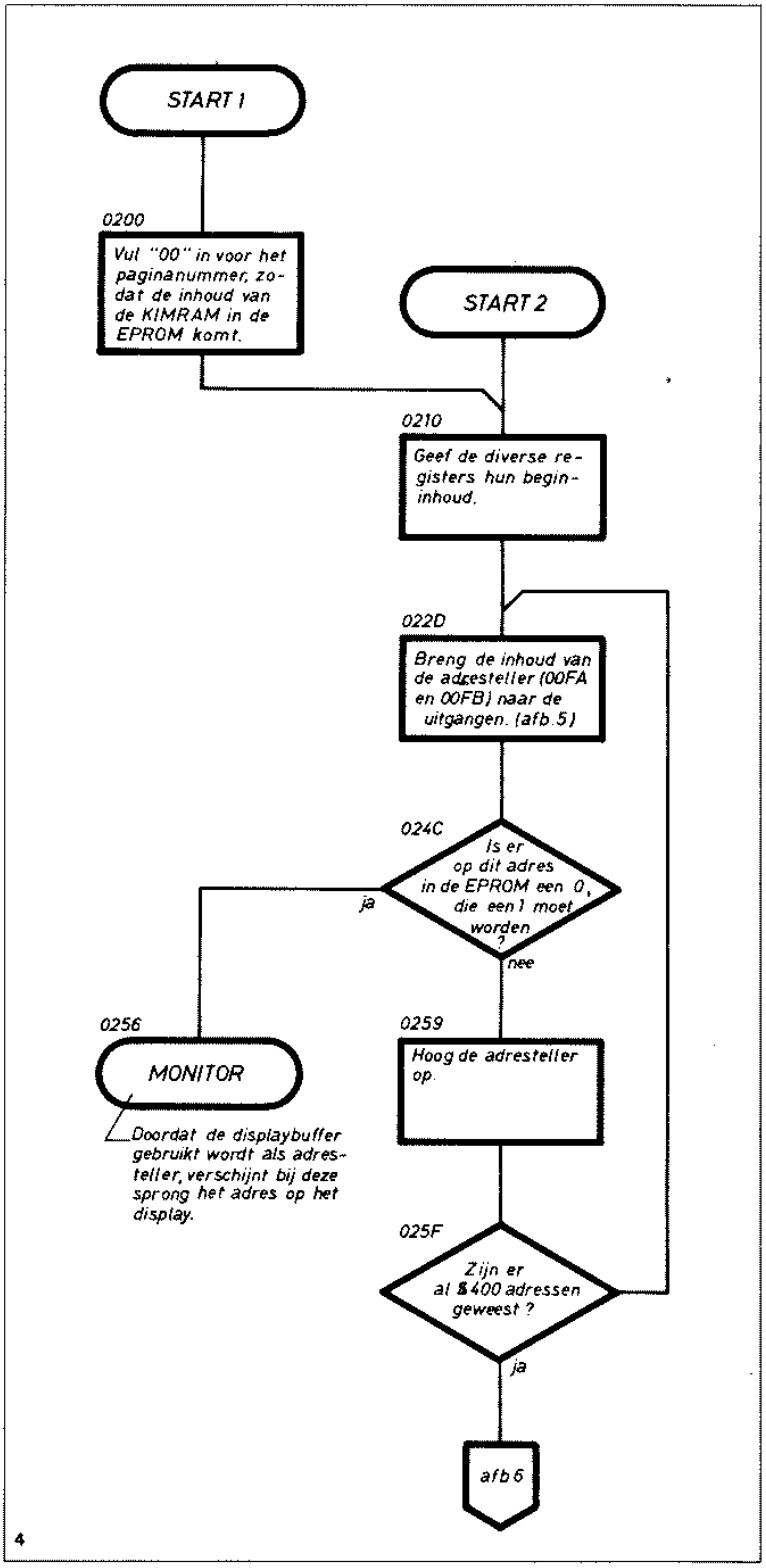
Connections during programming

We have seen that programming requires 10 address lines, 8 data lines, 1 chip-select line and 1 programming line, a total of 20 connections. Our KIM has 15 I/O lines. There are therefore still 5 lines that must come from somewhere else. For this we use 5 more I/O lines of the keyboard/display. The display therefore cannot operate during programming. But this would otherwise also be impossible because one pass through the SCAND loop of the monitor program takes 4 milliseconds, while the programming pulses may last a maximum of 1 mS. The connections of the 2708 during programming are shown in Fig. 3. We see that the data lines are connected to PPA (Peripheral Port A), the 5 lowest address lines to PB0 ... PB4 and the 5 highest address lines to COLA ... COLE. The inverter in the CS/WE line serves to bring the logical "1" level up to 12 V. During programming PB5 must therefore remain "0", causing CS/WE to be 12 V. During reading (the contents of the 2708 must be checked), PB5 must be "1", causing CS/WE to become "0". The programming pulses are generated by PB7. The addresses are counted using zero-page addresses FA and FB. This has the advantage that in the event of an error the display buffer is already filled with the address of the error. Locations 1780 through 1788 are also used by our utility program, so these addresses must be avoided from now on.

The program and comparison program consist of 3 parts:
A — the first comparison program.
B — the programming program.
C — the second comparison program.



3 The complete circuit of te the programmer to be connected to the KIM



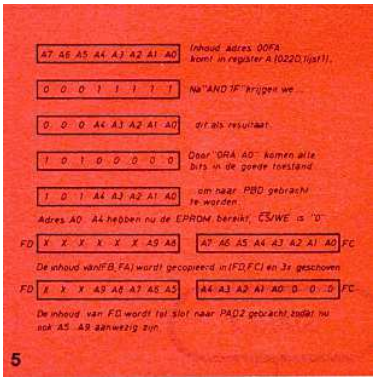
Flowchart of the first comparison program. HWN nowhere a '0' changes into a '1' we continue with the programming.

The first comparison program

Before the actual programming begins, we first check whether it is possible to write our program into the 2708. There is a rule that we must never change a 0 into a 1. If the 2708 has been properly erased, programming is of course always possible, but occasionally a 0 remains "stuck". We also have the possibility of "updating" our program. This means that we can extend an existing program in the 2708, provided that $FF is present at the unused addresses. To do this, we first copy the contents of our 2708, which is in the programming socket, to a section of 1K RAM memory. The new section of the program is written into the RAM, and then we program the EPROM again. As long as no 0 has to be changed into a 1, it is not necessary to erase the EPROM.

We therefore have to check for each address whether a 0 would have to be changed into a 1. Our comparison program starts at address 0210 + n × 400. This address notation may seem strange, but it indicates that the programming program is relocatable. This means that it can be placed anywhere in the address space of the KIM. This relocatability is achieved by not using its own subroutines in the program and not using [the OCR/source is incomplete here]. Before starting our program we must put the first 8 bits (the page number) of the program that we want to write into the ROM at address 1780. If we start at 0200 + n × 400, 00 is entered at 1780 and we then continue with 0210. We are then programming from KIM RAM. The flow diagram of the first comparison program is shown in Fig. 4.

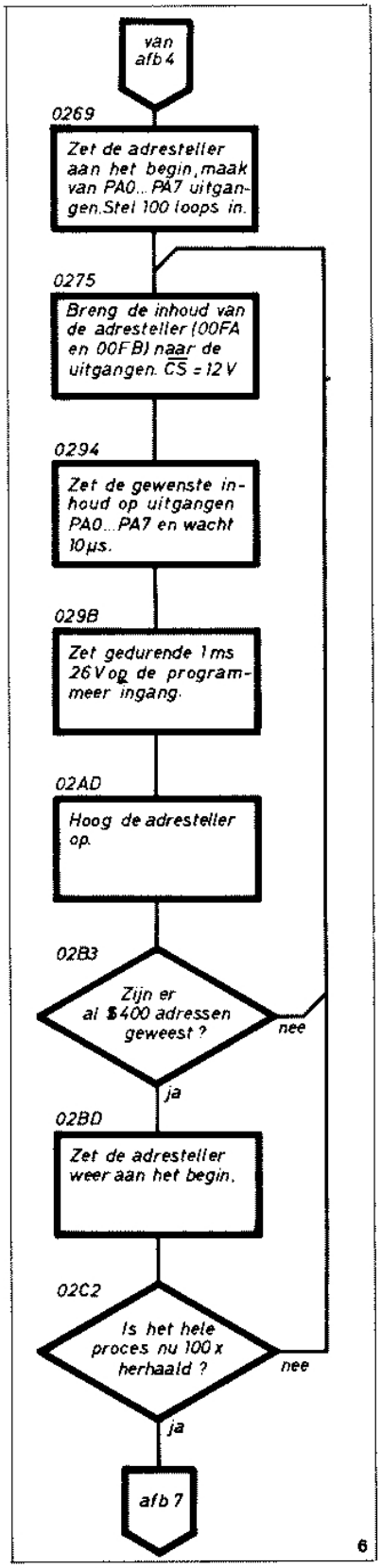
The first block contains all preparations. The interrupt is disabled, and we go to binary mode so that address calculations can be made. FA, FB become our address counter. We make our I/O lines outputs, except for PPA, which is connected to the data lines of the 2708. We want to read the 2708, after all. We then arrive at the top of the verify loop (address 0220).We begin by transferring the address contained in FB, FA to the address lines of our 2708.



Some manipulation is necessary for this (see Fig. 5). From FA, using AND 1F, the 3 most significant bits are stripped off. Then, using ORA A0, bits 7 and 5 are set to logical 1 (addresses 022F and 0231). The accumulator contents are then written to 1702. The result is that chip-select and the programming pulse become 0 volts and the 5 least significant bits of FA are placed on address lines A0–4 of the 2708.

FA and FB are then written to auxiliary addresses FC and FD. This is necessary because we are going to shift the values. After 3 × ASL and ROL FD, the 5 most significant address bits are in the correct positions in FD, and these are now passed via 1740 to COLA–COLE.

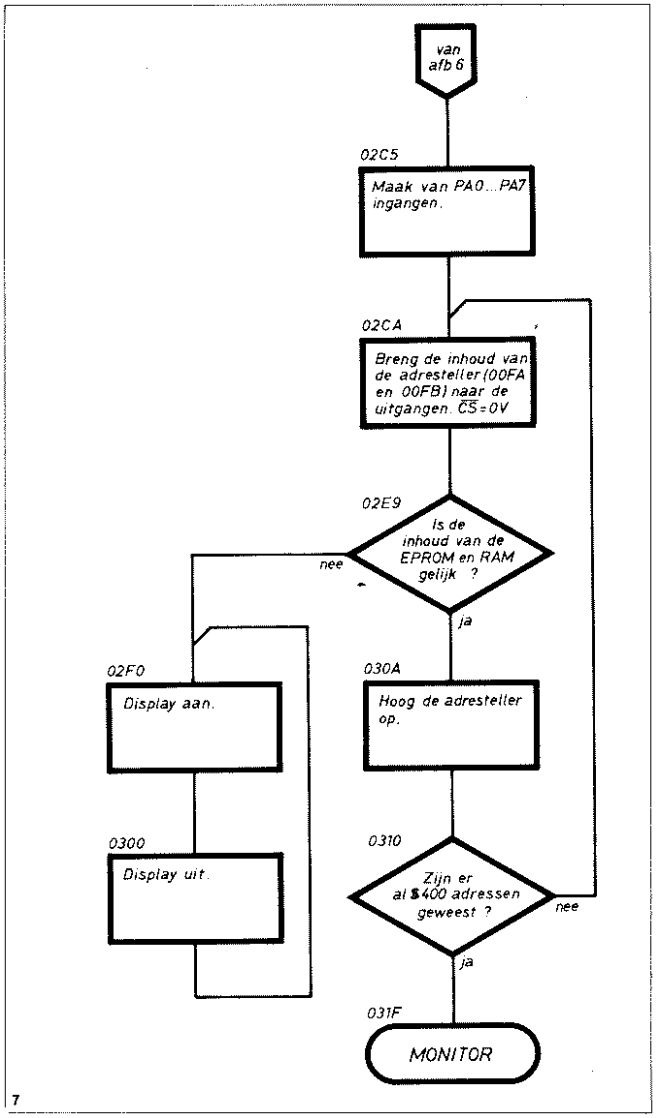
Now we can compare. Using LDA 1700, the contents of the first ROM address are brought into the accumulator (address 024C). A logical OR is now performed with the contents of the address indicated by FB, FA. After this last operation the accumulator



must not change, because that would mean that we wanted to change a 0 into a 1. We therefore check with CMP 1700. Under a not-equal condition (Z=0) we jump to the KIM monitor and can read at which address the error occurs. Under an equal condition the address in FB, FA is incremented. We then check whether all $3FF addresses have been processed. If not, we continue comparing; otherwise we automatically enter the actual programming program.

The programming program (Fig. 6)

This program must be run 100 times. We use the Y register for this purpose. The Y register is therefore set to (hexadecimal) 64. The contents of address $1780 again go to F5. PPA (address $1700) is made into outputs. The 2708 must now be programmed, after all. The chip-select is made 12 V (addresses 0279 and 0278), and the contents of the address indicated by FB and FA go to the data lines via address $1700. We then wait 10 µs before the programming pulse is given. This is required according to the manufacturer's specifications. The programming output is brought to 26 V (addresses 0298 ... 02A0). We count down 1 millisecond with the X register and the programming output goes back to 0 V. We again increment FB, FA and start with the next address until all 1024 addresses have been processed. Then the Y register is decremented. This continues until the Y register is zero. We then automatically enter the second comparison program.



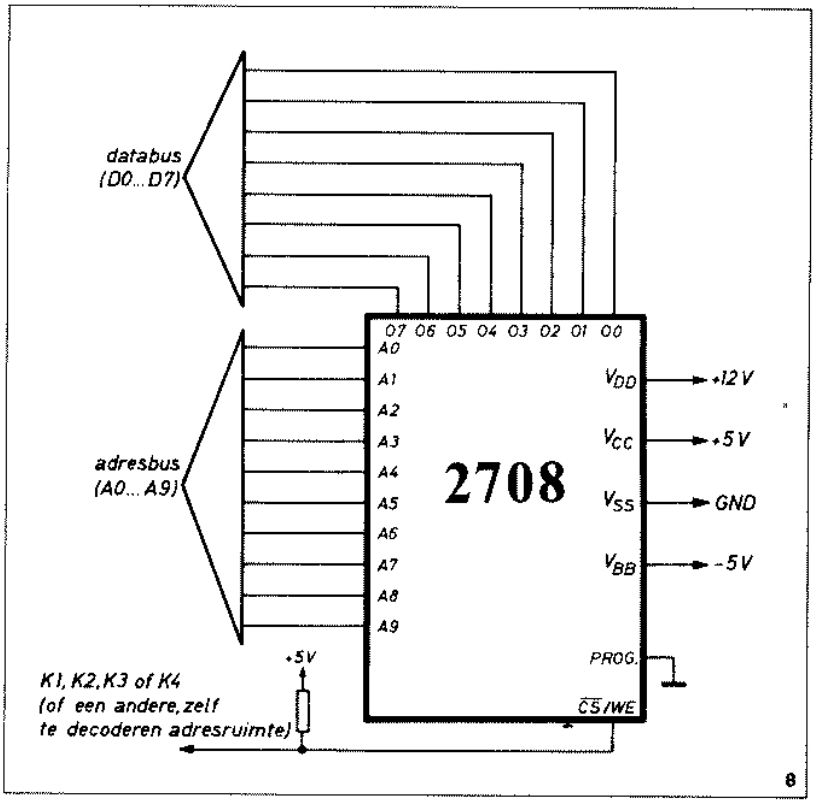
The second comparison program (Fig. 7)

This program is approximately the same as the first program. However, the contents of the 2708 are checked to see whether they are equal to those of the program we have written. When this is finished the program returns to the KIM monitor program. The OCR text is damaged at this point and does not clearly preserve the sentence describing the display indication. If the second comparison program detects an error, the display flashes.
This flashing is produced by going through SCANDS 256 times alternately and then executing a double delay loop. Before going to SCANDS we must first set I/O port 1740 correctly again. This is done with the monitor-program subroutine "INITS". The contents of 1700 are written to F9. We can therefore also see what has actually entered the ROM. If we then press RESET, we can see what should have been there at that location.
With the old 2708 from INTEL we often get error messages here. This is because this IC becomes very hot during programming. When we read this 2708 later, it always turns out to contain the data correctly. Perhaps a small fan would help here. I always obtained good results with the TMS 2708 and TMS 27L08 from Texas Instruments.

The write-back program (listing 2)

If we want to write data into a programmed 2708, we can only do so in an area where ones ($FF) are present. We therefore put the ROM in the programming socket and write the ROM to a section of 1K RAM memory. For this we use the write-back program.
This program has 2 starting addresses, namely 100 and 110. If we start at 100, address $1780 is set to 00. We then write the contents of the EPROM in the programming socket to the 1K RAM of the KIM. The program is almost the same as the comparison program,

except that here the contents of the EPROM are copied to the memory location indicated by the contents of addresses $FB and $FA. At the end of this program we jump back to the KIM monitor.
By starting the write-back program while the programming socket is empty, only $FF is written into RAM. As a result, after entering a program all unused memory locations will contain $FF, so that we have the possibility of programming the unused part of the EPROM later.



Connections of the EPROM during normal use

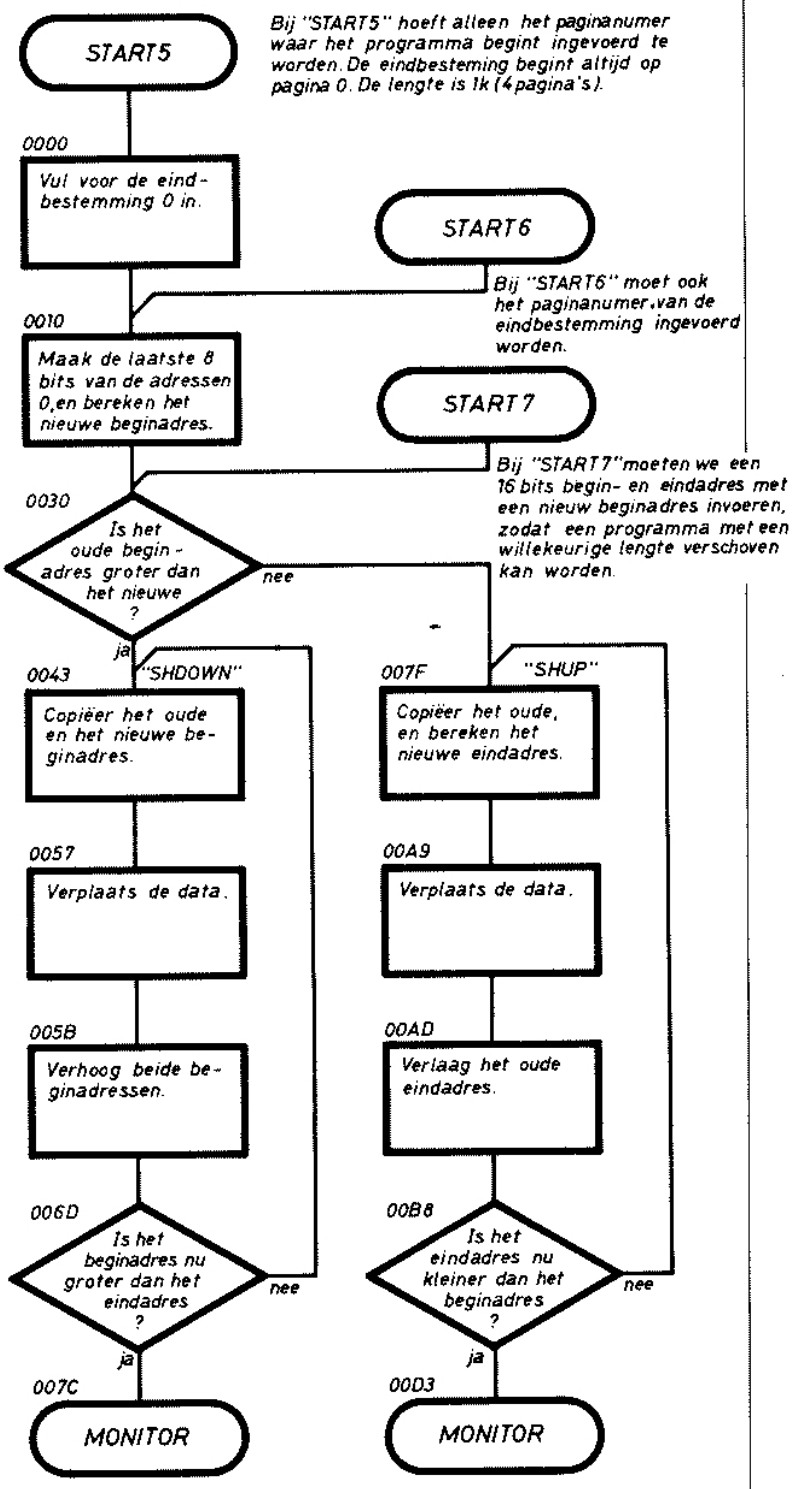
Once programmed, the EPROM contains a program of at most 1 Kbyte (1024 addresses). We can now use the EPROM without further logic in the decoded 4K address space (Fig. 8). By connecting the CS/WE pin to K1, K2, K3 or K4 we can choose the starting address of the EPROM:

CS/WE connected to:	address space:
K1	0400 ... 0800
K2	0800 ... 0C00
K3	0C00 ... 1000
K4	1000 ... 1400

In RB November 1977, two BEM-1 cards of 2K each were placed in this address space. In the next article (the final part) we will publish a PCB with the same connections and the same format as the BEM-1 cards. This makes it possible to fill the memory space as required with 4K RAM, 2K EPROM and 2K RAM, or 4K EPROM, simply by plugging the

desired card into the socket. The EPROMs can of course also be placed in the non-decoded 56K address space; in that case some additional decoding logic is required. The various construction possibilities will be discussed in somewhat more detail in the next part.

EPROM address +n.400 (Hex)	used addresses	Description
0000 GO	1780	Copy a KIM program to 0000–03FF (KIM-RAM). The page number where the 1K program begins is put at 1760.
0010 GO	1780 1781	Copy a 1K program to another section of RAM (does not write to 1780 through 1788). Put the page number where the 1K program begins at 1780 and the page number where the 1K program is to go at 1781.
0030 GO	1783 through 1788	Move a section of program of arbitrary length from one place to another. We can move it upwards or downwards. 1783 and 1784: Starting address of the program to be moved. 1785 and 1786: Ending address of the program to be moved. 1787 and 1788: Starting address where the program is to go.
0100 GO	-	Write the contents of a 2708 in the programming socket to 0000–03FF.
0110 GO	1780	Write the contents of a 2708 in the programming socket to a section of RAM. Put the page number where the program is to go at 1780.
0200 GO	-	Program a 2708 in the programming socket. The program to be transferred is at 0000–03FF.
0210 GO	1780	Program a 2708 in the programming socket. Put the page number where the program begins at 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
"SHDOWN"
0043
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
"SHUP"
007F
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

The shift program (Fig. 9)

Especially when using this EPROM programmer, the need will arise to shift programs with a length of 1K. This need can also arise while developing a program, although in that case the sections may have arbitrary lengths. Before starting the shift program, the following must therefore be known:

a. The starting address of the program to be shifted at 1783 and 1784;
b. The ending address of the program to be shifted at 1785 and 1786;
c. The new starting address of the program at 1787 and 1788.

(Always put the low-order figures at the first address, and the high-order figures at the second address.) The ending address of the new program follows naturally from these data.

We also use addresses 1783 ... 1788 as address counters. Because we want to work indirectly, we must also use zero-page addresses. For this we take FA, FB, FC and FD. For each address that we shift, these zero-page addresses are filled again with the values from 1783 ... 1788. The advantage is that we can now also shift programs in zero page without overwriting our counters. It goes without saying that we must never shift into the area 1783 ... 1788.

Our program has 3 entry points, namely 0000, 0010 and 0030. (Of course, again + n × 400.) If we want to start at 0030, we must first enter the addresses 1783 ... 1788. If we start at 0010, we always shift a 1K block. We must then first enter at 1780 where the program begins (page number, hex) and at 1781 where the program is to go (page number, hex). If we start at 0000, the program puts "00" at address 1781. This allows us to write a 1K block from memory to KIM RAM.

If we want to move a program there are two possibilities: the program has to move upwards (to lower addresses), or downwards (to higher addresses). We must first determine whether our program moves upwards or downwards. We can determine this by comparing the two starting addresses, i.e. by subtracting the contents of 1784, 1783 from the contents of 1788, 1787 (addresses 0030–0041).

Suppose (1784, 1783) is greater than (1788, 1787). Then we must shift upwards. In Fig. 9 we therefore take the left branch of the flow diagram. FA and FB are filled from 1783 and 1784; FC and FD are filled from 1787 and 1788. The contents of the address specified by FB and FA now go directly to the address specified by FD and FC, and the first line has thereby been moved (address 0057). We then increment the old and new addresses. Next we check whether the starting address has passed the ending address (address 006D), because then we must of course stop.

If (1788, 1787) is greater than (1784, 1783), we must start by shifting the contents of the last address of our program, because otherwise we would overwrite our own program again. In Fig. 9 we take the right branch of the flow diagram. FA and FB now contain the ending address of the program to be shifted (007F). The new ending address must be placed in FC and FD. This address must first be calculated by subtracting the old starting address from the old ending address (addresses 0089 ... 00A7). The resulting

program length is added to the new starting address, and the result (the new ending address) is placed in FC and FD.
At address 00A9 the contents of the address specified by FB, FA are now transferred to the address specified by FD, FC. The old ending address is then decremented. Because the new ending address is recalculated each time, it is not necessary to decrement this address. As soon as the ending address passes the starting address, we are finished and return to the monitor program.

The programming program in EPROM

Of the programs already discussed (listings 1, 2 and 3), only listing 1 is actually strictly necessary for programming. The programs in listings 2 and 3 can be very useful. Of course you can record this programming program on cassette tape, and whenever you want to program an EPROM, load the programming program into the KIM. The disadvantage is that this program requires at least about 290 locations of KIM RAM. It therefore makes sense to put the programming program into EPROM as the first program. There is then still enough room in the EPROM for the programs in listings 2 and 3. You can also put small programs that you use frequently into it at the same time.
So proceed as follows: First type the programs from listings 1, 2 and 3 into the KIM. Starting at address $0322 you can enter another program that you use frequently. Fill the unused locations with $FF, so that they can later be used for additions. (When programming we can change a "1" into a "0", and not the other way around.) Finally put an empty 2708 in the programming socket and start the program at address $0200. The programming program now programs itself! Once programmed, add the 2708 to the KIM's memory, so that programming can always be done directly without problems.

Attention!

When you have a program in KIM RAM and want to put this program into EPROM, you must remember that the program will end up at different addresses. This means that you will mainly have to correct the JMP and JSR instructions (of course not when jumping to the monitor program). When tables are used, sometimes the LDA instructions also need to be corrected.
If you have a memory expansion, it is best to develop the program at the location where it will later be placed in EPROM; then you know for certain that it works. You must also remember that the EPROM space cannot be used to store temporary data. Some programs have the nice habit of modifying themselves, which of course is not possible with EPROM.

Continuation

In the second and final part we will discuss the additional supply voltages (-5 V; we already had the 12 V and 5 V), and some suggestions will also be given for the practical construction of this 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.
```

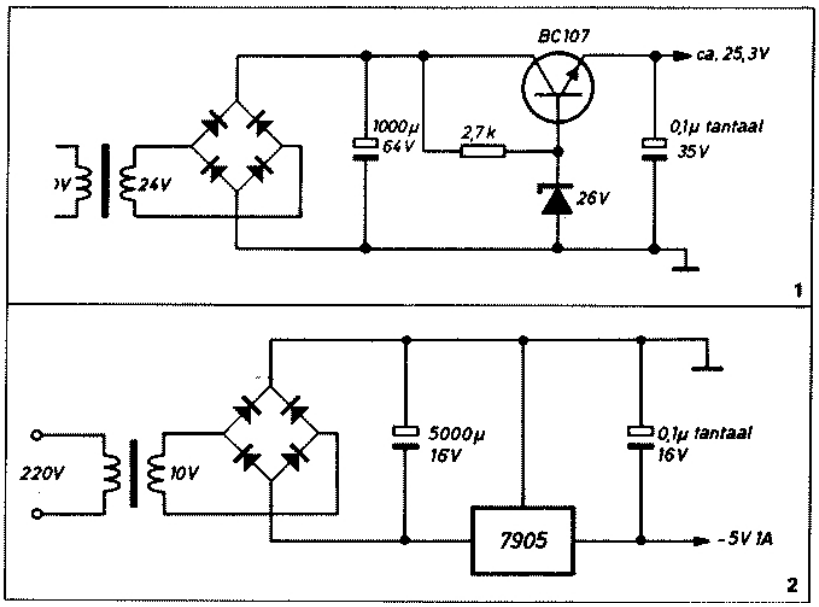
Lijst 3

```
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.
```

Part 2

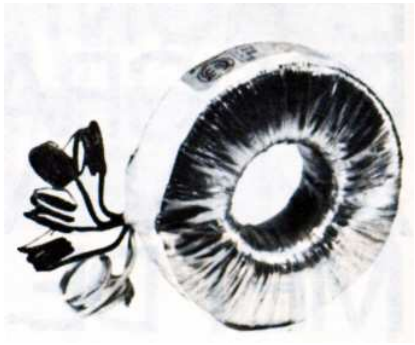
In the previous part the (very simple) interface required to supply the programming pulses to the 2708 was discussed. The programming program, which ensures that the correct pulses occur at the correct time, was also covered in that part. In this final part we will briefly discuss the additional supply voltages required, as well as some construction possibilities.



The power supply

The KIM itself uses +12 V and +5 V. For normal use of the 2708, only -5 V is additionally required. When we program, a +26 V supply voltage must also be generated. The 2708 does not draw much current. The 26 V supply only needs to deliver a maximum of 20 mA. If, however, we keep future developments in mind (the 2716, which we will certainly return to), the supply should be suitable for at least 30 mA. Fig. 1 shows a possible circuit for this power supply.

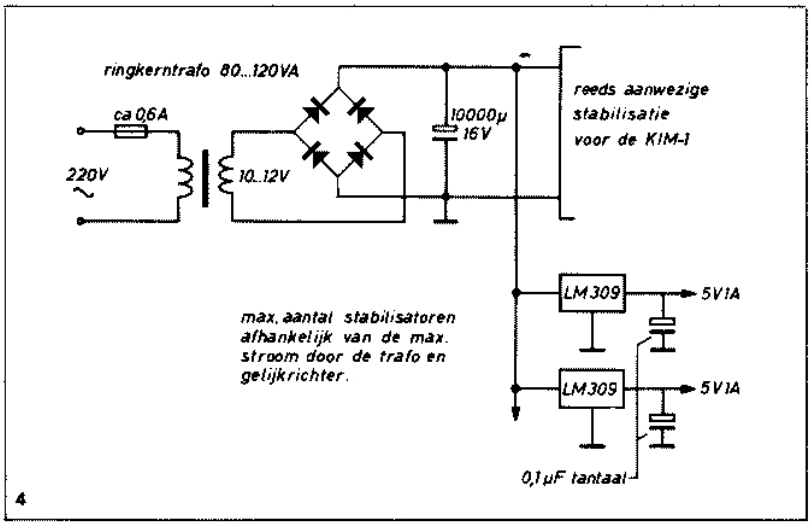
The additional -5 V supply must be capable of delivering about 45 mA per 2708. In view of future expansion, however, it is sensible to make the -5 V supply suitable for at least 1 A. This can most easily be done in the familiar manner shown in Fig. 2. The +5 V and +12 V supplies of the KIM should be able to provide the additional current for the 2708 (respectively 10 mA and 65 mA per 2708).



The transformer and bridge rectifier

When the power supply has to be increased because you are expanding your system, the transformer is usually a problem. Because of the relatively high price of a transformer, its capacity is very often chosen so that it can supply just the required current. The result is that for every expansion a new transformer has to be purchased to provide the additional current/voltage. Moreover, a place has to be found for each transformer.

In this case we suddenly need -5 V and +26 V again, so indeed another transformer. To get rid of this problem once and for all, you should provide a transformer of about 80 to 120 VA. A toroidal transformer is the best choice (Fig. 3). These transformers are somewhat more expensive, but in addition to their familiar advantages (small, low losses, no stray field), they have the major advantage that extra windings can easily be added.An unexpected voltage (such as +28 V in this case) then costs only a bundle of winding wire and some time (about 240 turns). The winding space is almost unlimited, and you do not have to dismantle the laminations. You must of course make sure that you do not exceed the maximum apparent power, but with 80 or 120 VA you can go on for the time being.It is also sensible to make the bridge rectifier sufficiently heavy-duty (for the +5 V supply about 5–10 A). For every expansion a 5 V regulator can then simply be added (e.g. LM309 or 7805). The whole arrangement is shown once again in Fig. 4.



Construction possibilities

The author has developed a very specific system for this PROM programmer. As a basis he made a motherboard which fits directly onto the KIM (Fig. 5). This motherboard has 3 more connectors into which the expansion cards (Fig. 6) fit. Each card has a memory capacity of 2K EPROM. By plugging these cards into the upper two connectors, a total of 4K EPROM is placed in the decoded address space of the KIM (address 0400 ... 13FF). It

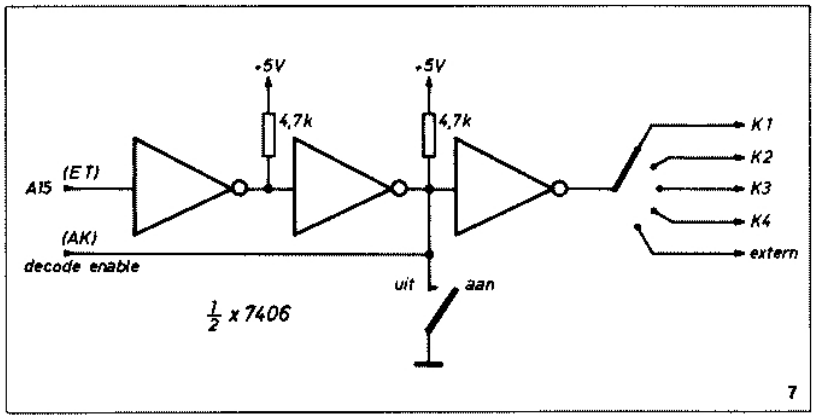
is also possible to plug a 2K RAM card into these connectors.
In addition to the two EPROMs, the EPROM cards contain a 7406 and a few resistors. The inputs and outputs of this 7406 are brought out separately. When the card is inserted into the lower connector (programming socket), the inputs and outputs of the 7406 are interconnected in such a way that the whole thing forms the interface between the KIM and the EPROM to be programmed. (The EPROM must then be inserted in the lower socket on the card.) The address and data lines are now connected to the KIM's I/O, and programming can begin.
When the EPROM card is used normally (in one of the two upper connectors), the 7406 does not remain unused. In this case the inverters are connected so that vector selection becomes possible (Fig. 7). This gives us the possibility of placing both interrupt vectors and the reset vector at the top of an arbitrary 1K block.
Fig. 8 finally shows the result: a compact carrying case containing a complete development system.

R.B. version

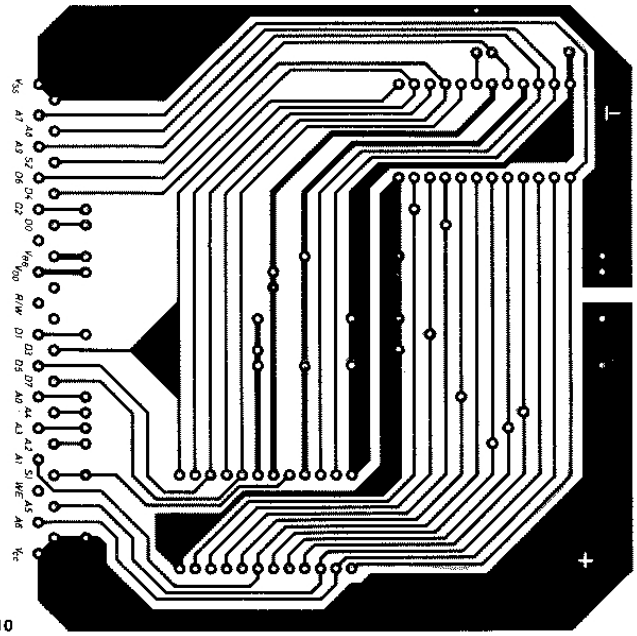
Although the above description leads to a fairly complete system, there are nevertheless some objections. We are thinking of the enthusiastic hobbyist who has made a nice enclosure for his KIM and cannot possibly fit the motherboard into it. Or the faithful RB reader who has already installed two connectors in the decoded 4K space for the BEM-1 card (RB Nov. 77).
In addition, because the author used double-sided contacts on the PCB, he was obliged to use a double-sided PCB as well. Manufacturing double-sided PCBs one self is beyond the reach of most hobbyists. Even if such a PCB were sold by us, it would unnecessarily become expensive. Therefore the RB editorial staff designed a single-sided Eurocard, with the same connector and the same connections as the BEM-1 (Fig. 9).
The use of a single-sided PCB made some wire links necessary; for the cost-conscious hobbyist this will hardly be an objection. Incidentally, this single-sided PCB can easily be made double-sided by drawing the wire links on film.

The PCB layout can be found in Fig. 10, and the corresponding component placement in Fig. 11. Because the BEM cards use a 31-pin connector while the author used a 40-pin connector, it was not possible to put the 7406 on this PCB as well. Strictly speaking, this is not necessary either.
The 7406 is used in the interface between the KIM and the EPROM to be programmed. However, this interface can also easily be made on a small piece of Veroboard. Somewhere in or on the enclosure of your KIM there will always be a place where you can fit a 24-pin IC socket.
You then have a separate "programming socket", in which you can program the 2708. In the future this programming socket can also be used as a connector for (home-built) devices that make use of the intelligence of the KIM. After all, the programming socket

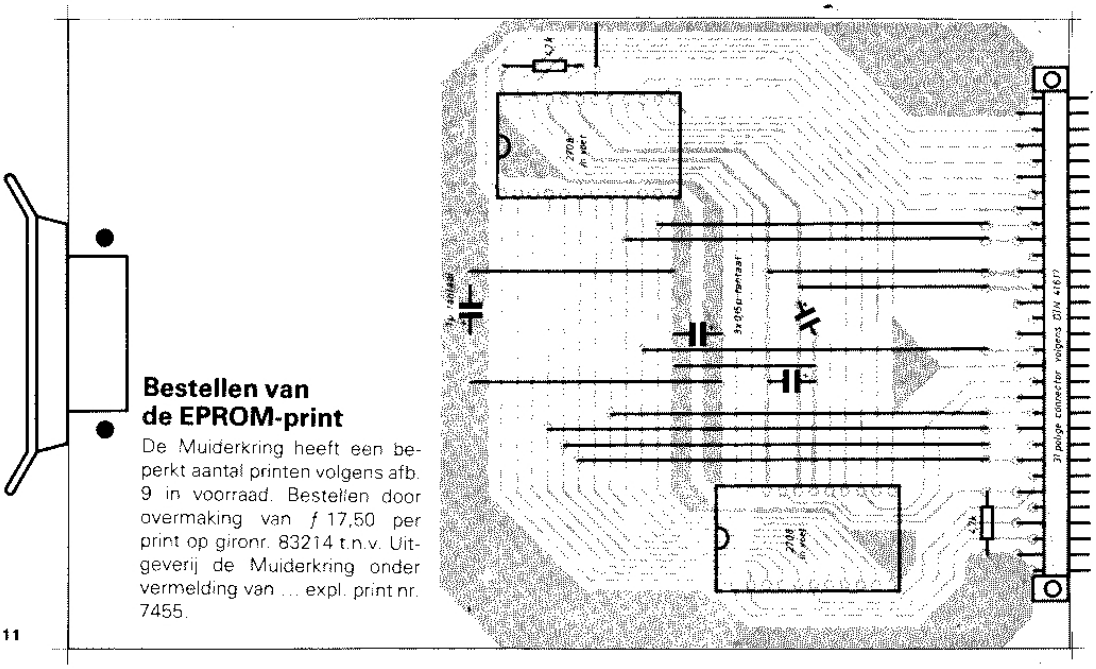
provides 18 TTL-level signals that can be programmed by the KIM (inputs and outputs), plus one signal between +12 V and 0 V and a signal between +26 V and 0 V.
Three supply voltages can also be obtained via this socket. In the deluxe version you can use a lever-operated socket for the programming socket (rather expensive), so that inserting and removing the IC can always be done without problems.
Vector select is not implemented with this system, but it can be made by adding one 7406, a switch and some resistors (Fig. 7).



In summary, the RB version of this EPROM programmer is:
Inside the KIM enclosure there are 2 connectors. Each connector can accept either a BEM-1 (2K RAM) or an EPROM card (2K EPROM). On the enclosure there is a programming socket in which a 2708 can be programmed. This programming socket also serves as a connector for peripherals.



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