J. M. v. d. PEIJL

EPROM PROGRAMMEER-APPARAAT MET DE KIM

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.

Het programmeren

Volgens de fabrieksspecificaties 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 t/m 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.

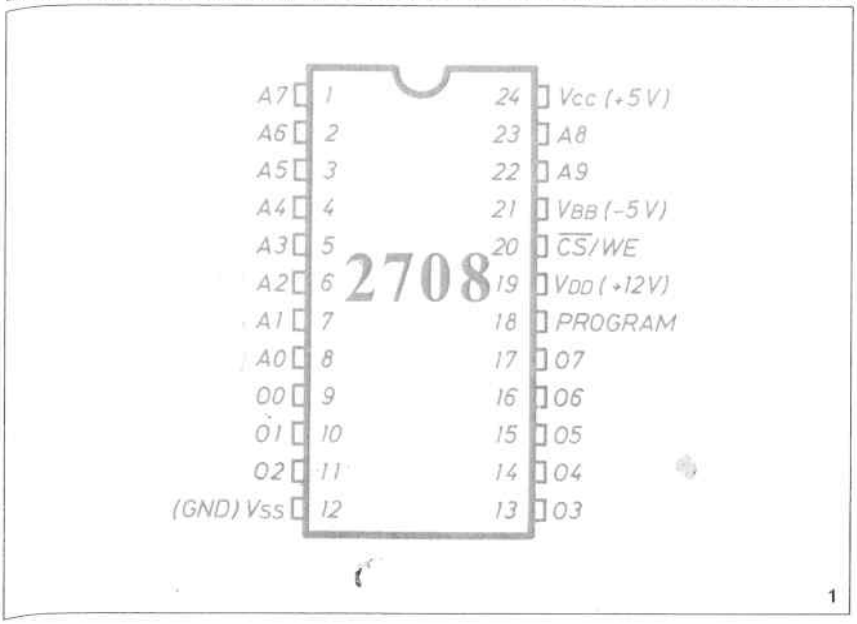
Bij het programmeren worden deze 8 datalijnen als ingang gebruikt. Zolang het programmeren duurt moet pin 20 (CS/WE) op +12 V worden gehouden.

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 µs 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 1 s.

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



1

1 *Bovenaanzicht van de 2708 EPROM*

2 *De interface tussen de KIM en de te programmeren EPROM*

De aansluitingen bij het programmeren

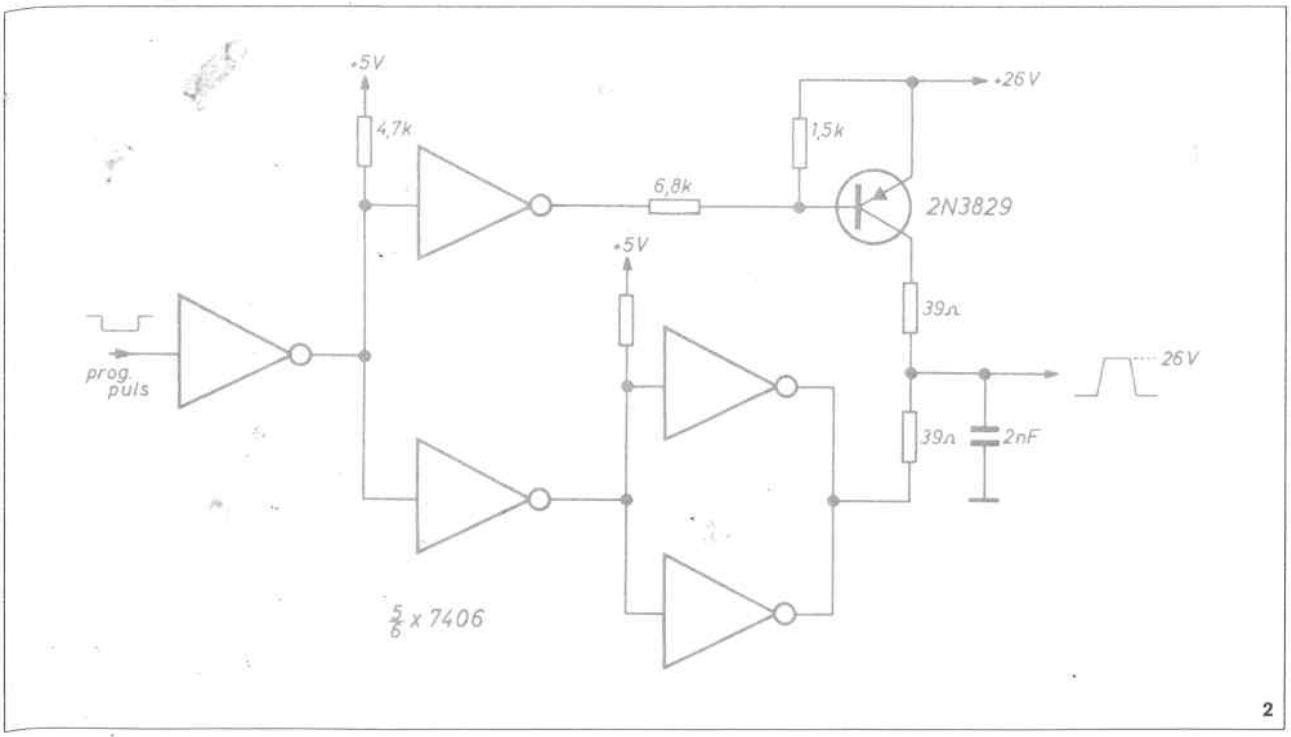
We hebben gezien dat voor het programmeren 10 adreslijnen, 8 datalijnen, 1 chipselectlijn en 1 programmeerlijn, dus totaal 20 aansluitingen nodig zijn. Onze KIM heeft 15 I/O lijnen. Er moeten dus nog 5 lijnen ergens anders vandaan komen. Hiervoor gebruiken we nog 5 I/O lijnen van het keyboard/display. 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 monitorprogramma 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 PB0 . . . PB4 en de 5 hoogste adreslijnen op COLA . . . COLE. De inverter in de $\overline{CS}$/WE lijn dient om het logische '1' niveau op 12 V te brengen. Tijdens het programmeren moet PB5 dus '0' blijven, waardoor $\overline{CS}$/WE 12 V wordt. Bij het uitlezen (de inhoud van de 2708 moet worden gecontroleerd) moet PB5 '1' worden, waardoor $\overline{CS}$/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:

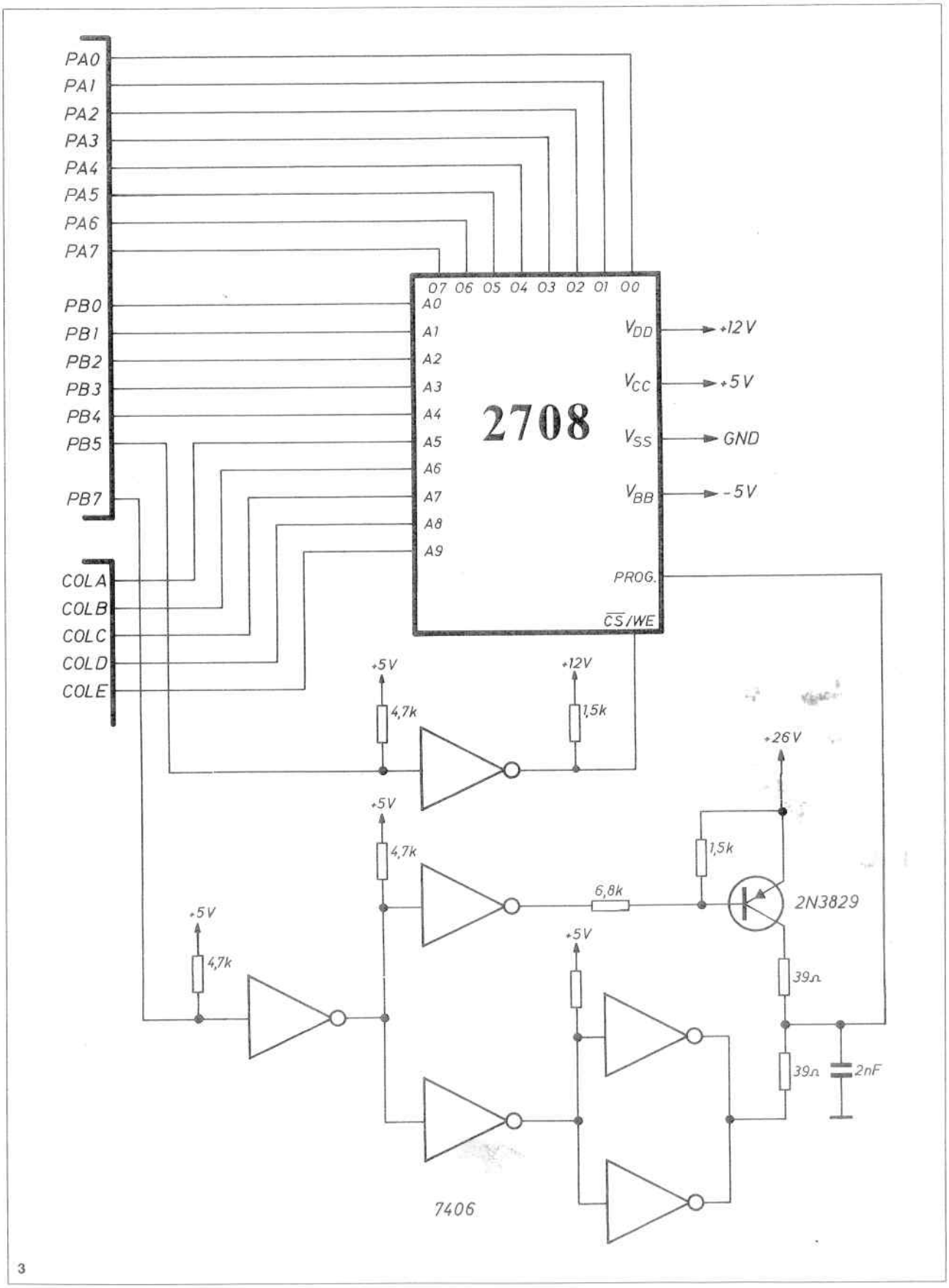
A het eerste vergelijkprogramma

B het programmeerprogramma

C het tweede vergelijkprogramma.



2


PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB7
COL A
COL B
COL C
COL D
COL E
O7 O6 O5 O4 O3 O2 O1 O0
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
2708
VDD
VCC
VSS
VBB
+12 V
+5 V
GND
-5 V
PROG.
CS/WE
+5V
4,7k
+12V
1,5k
+26 V
+5V
4,7k
1,5k
6,8k
2N3829
+5V
4,7k
+5V
39Ω
39Ω
2nF
7406

3

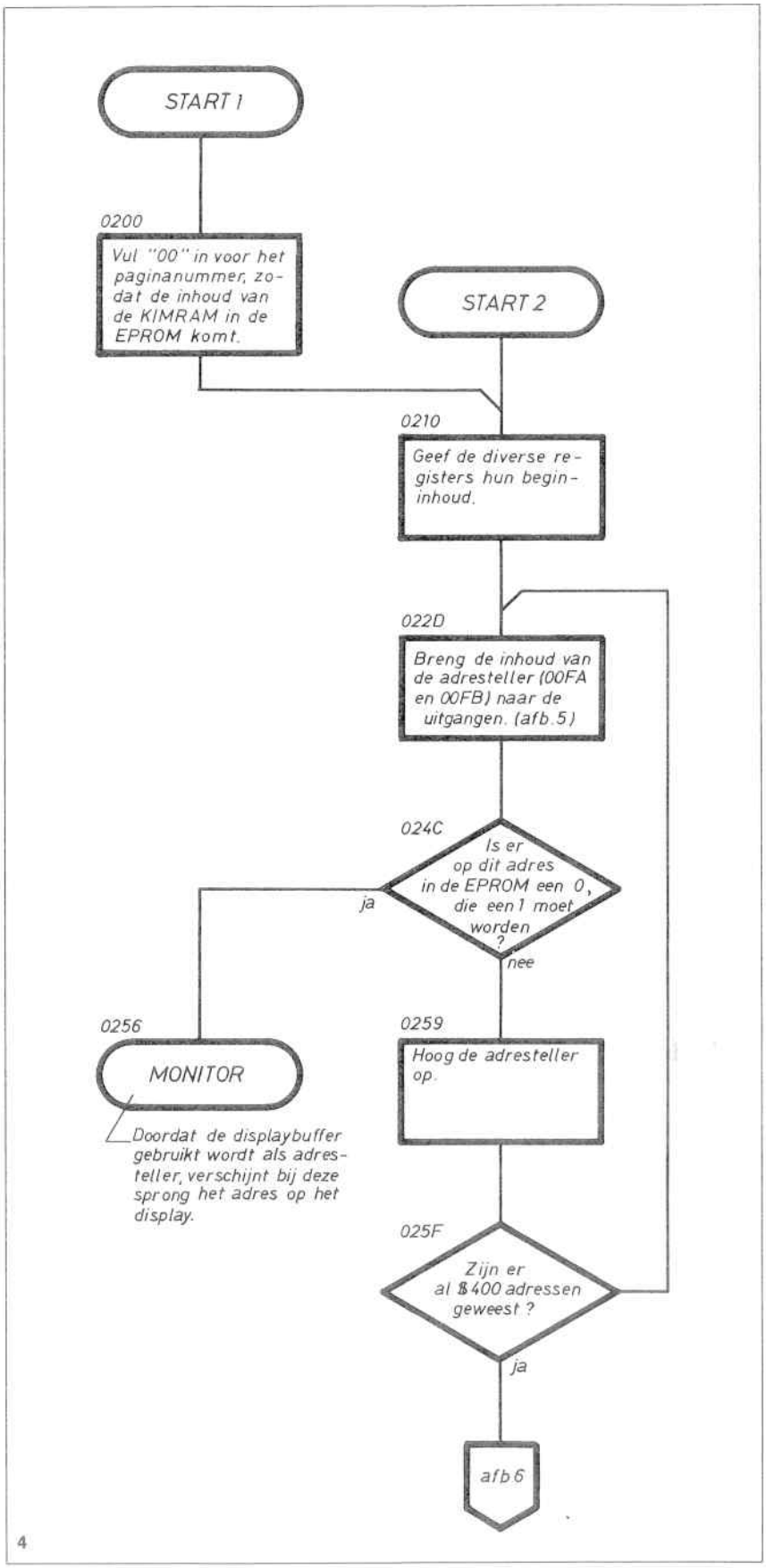
3 *Het complete programmeerapparaat, zoals het op de KIM kan worden aangesloten*

4 *Flowdiagram van het eerste vergelijkprogramma. Wanneer nergens een '0' in een '1' verandert, gaan we door naar het programmeerprogramma. (afb. 6)*

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 0 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 '$FF' 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 JMP's, 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 1780 en daarna gaan we verder met 0210. We programmeren dan dus vanuit de KIM-RAM. Het flowdiagram van het eerste



4