```
EPEPEPEPEPEP    EPEPEPEPEP
EP              EP        EP
EP              EP        EP
EP              EP        EP
EPEPEPEP        EPEPEPEPEP
EP              EP
EP              EP
EP              EP
EPEPEPEPEPEP    EP
```

De DOS65 EPROMprogrammer

Software handleiding V1.03 dd. 23-01-1989

Deze handleiding werd geschreven door:

N. de Vries
Mari Andriessenrade 49
2907 MA Capelle aan den IJssel
Nederland

Zowel voor deze handleiding, het programma EP, de hardware schakeling als de print geldt:

Inhoud:

N.B. Voor de bouw en installatie van de EP hard- software wordt men naar de aparte hardwaremanual verwezen.

Hoofdstuk 1: Inleiding.

Het is alweer geruime tijd geleden, dat het tijdschrift Elektuur een EPROMprogrammer publiceerde, geschikt voor de standaard Elektuurbus. De naam van dit apparaat is EPROMmer. Hij is in staat (op primitieve wijze) EPROMs van het type 2716 of 2732 te programmeren.

De tijd heeft echter intussen niet stilgestaan. De EPROMs groeiden gestaag in capaciteit, terwijl het aantal algoritmen om EPROMs te programmeren ook uitbreiding onderging. De EPROMmer is derhalve zo langzamerhand als verouderd te beschouwen.

De kleinste gangbare EPROM is nog steeds de 2716, die als 2k woorden van 8 bit is georganiseerd. De grootste EPROM die thans gekocht kan worden is de 27512, die 64k woorden van 8 bit, of populairder 64k byte groot is. Behalve een verschil in capaciteit, en dus ook in behuizing, is er ook een belangrijk verschil in het programmeeralgoritme. De 2716 wordt geprogrammeerd door bij ieder byte een programmeerpuls te geven van 50 milliseconde. De totale programmeertijd is dus minimaal 2048 x 50 ms = 102.4 seconden ofwel ca. 1 minuut en 42 seconden. Zouden we een 27512 op deze manier gaan programmeren, dan zouden we 54 minuten en 37 seconden bezig zijn! Dit is op zijn zachtst gezegd onpraktisch. Intussen heeft de techniek ervoor gezorgd, dat ook het algoritme sneller werd: de normale programmeertijd voor een 27512 ligt bij ca. 9 minuten.

Het ontbreken van een programmer voor de Elektuurbus die ook de moderne EPROMs met de moderne algoritmen kan programmeren, was aanleiding tot het ontwerpen van de EPROMprogrammer EP, een afkorting van EPROM-programmer. EP bestaat uit twee delen: een print met hardware en een stuk software dat het algoritme opwekt. EP programmeert alle gangbare single 5 Volt EPROMs in 24 en 28 pins behuizing. De beschikbare programmeerspanningen zijn 12.5, 21 en 25 Volt. De software ondersteunt drie algoritmen: standaard 50 ms, het Intel Inteligent algoritme en het Intel Quick Pulse algoritme. Dit laatste algoritme vertegenwoordigt de laatste stand van de techniek.

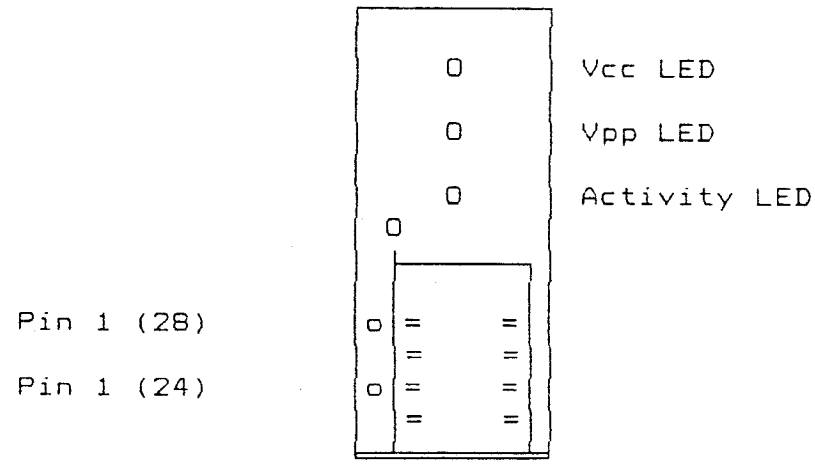
Hoofdstuk 2: Gebruik van EP.

2.1 Hardware.

Om EP te kunnen gebruiken, moet het systeem worden voorzien van de EP hardware. Deze bestaat uit een Eurokaart met en PIA en een VIA die op de bus gestoken wordt en een kleiner printje dat de ZIF-voet en wat LEDs herbergt. De bouw en installatie van de kaart, alsmede de installatie van de EP software worden in de aparte EP hardware manual beschreven.
Na installatie kan de software worden opgestart.

2.2 De indicator LEDs.

De ZIF-print is voorzien van een vijftal LEDs, die als volgt zijn opgesteld:



De Vcc LED brandt als de voedingsspanning voor de EPROM is ingeschakeld. Dit is normaliter alleen tijdens de uitvoering van commando's het geval. Als de EP software op een commando wacht is de LED gedoofd, en zijn alle pinnen van de ZIF-voet spanningsloos.
De Vpp LED brandt als de programmeerspanning is ingeschakeld. Afhankelijk van het EPROMtype kan deze spanning voorkomen op pin 1, 22 of 23 van de ZIF socket. De programmeerspanning wordt alleen tijdens de programmeerfase van het P-commando ingeschakeld.
De activity-LED is verbonden met de adreslijn A7 van de EPROM. Tijdens de uitvoering van commando's zal deze LED in principe knipperen. Hiermee wordt vooral een indruk verkregen van de snelheid waarmee het programmeerproces vordert. Tijdens het uitvoeren van lees-, verify, test en empty commando's verandert de adreslijn A7 meestal zo snel dat de LED flauw zal branden.
Van de LEDs bij de pinnen 1 en 3 van de ZIF-voet brandt er altijd maar 1. Deze LEDs geven aan in welke ZIF-voetpin pin 1 van de te bewerken EPROM moet komen.

BELANGRIJK: INDIEN 1 OF MEER VAN DE DRIE BOVENSTE LEDs BRANDEN, MAG ER GEEN EPROM IN OF UIT DE ZIF-VOET GEPLAATST OF VERWIJDERD WORDEN!!

Alleen als alle drie LEDs gedoofd zijn, zijn alle ZIF-voet pennen spanningsloos.

2.3 Opstarten.

Indien de installatie volgens de EP hardware handleiding correct is uitgevoerd, staat op de systeemschijf de juiste versie van de driversoftware onder de naam EP. Door EP (CR) in te typen, wordt de software opgestart. Met de software wordt ook de monitor MON65 meegeladen. Op het scherm verschijnt na het opstarten:

```
EP Vx.yy
 by NdV

The following commands are available:

B - Set RAM & EPROM address boundaries. C - Calculate RAM checksum.
D - Set EPROM type.                     E - Check if EPROM is empty.
I - Set address increment.              M - Call MON65 monitor.
P - T + program EPROM + V.              Q - Quit program.
R - Read EPROM contents into RAM.       S - Show EPROM contents directly.
T - Test if EPROM is programmable.      V - Verify EPROM with RAM.
? - Show this list.

Address increment = 1
EPROM start: $0000                RAM start: $1000
EPROM  end : $1FFF                RAM  end : $2FFF

2764 21V Intel > _ <---- prompt
```

Op het kleine printje moeten nu alle LEDs gedoofd zijn, behalve de LED bij pin 1 van de ZIF-voet. De gebruiker kan nu de te bewerken EPROM in de ZIF-voet zetten. Pin 1 van de EPROM moet naast de brandende pin 1 LED komen.

2.4 De prompt.

De prompt, aangegeven met een pijl, bestaat uit drie delen. Het eerste is het gekozen EPROMtype. De EPROM in de socket moet uiteraard van hetzelfde type zijn. Is dit niet het geval, dan dient men met het commando D (zie hoofdstuk 3) het juiste EPROMtype te selecteren. De EPROM mag hierbij in de ZIF-voet blijven zitten. De volgende typen zijn mogelijk:

2716	2732	2732A	2764	2764A
27128	27128A	27256	27256A	27512
2532	2564			

Het tweede deel van de prompt is de gebruikte programmeerspanning. Deze kan 12.5, 21 of 25 volt zijn. De programmeerspanning staat soms expliciet op de EPROM vermeld. Een te hoge spanning vernielt de EPROM; een te lage zal programmeerfouten geven. Zolang er niet geprogrammeerd wordt, is deze waarde niet belangrijk: alleen tijdens de programmeerfase van het commando P wordt de programmeerspanning op de EPROM gezet. Alle andere commando's veroorzaken alleen spanningen op TTL-niveaus op de EPROM-voet. Het is dus niet bezwaarlijk om een 12.5 Volt 2764A uit te lezen met de programmer ingesteld op een 21 Volt 2764.

Het derde deel van de prompt is het thans gekozen algoritme. Er zijn er drie, in volgorde van snelheid zijn dit:

50 ms	Het klassieke algoritme
Intel	Het Intel Inteligent Algoritme
Quick	Het Intel Quick Pulse Algoritme

Achter de prompt staat de cursor, hier voorgesteld door een underscore. De EP werkt met 1-letter commando's, die zowel in lower als in uppercase kunnen worden ingetypt. De commando's hebben geen van alle argumenten. EP laat de invoer altijd in uppercase zien. Niet geldige commando's worden niet geaccepteerd en ook niet ge-echo-ed. Wordt een geldig commando ingetypt, maar bedenkt men zich, dan kan dit weer verwijderd worden door op de DEL of RUB-toets te drukken. Het gekozen commando wordt uitgevoerd door op de return-toets te drukken.

Een beschrijving van alle commando's staat in hoofdstuk 3. In EP worden alle getallen (adressen, data) in hexadecimaal ingevoerd en afgedrukt. Overal waar letters moeten worden ingetypt, mag dit in zowel lower, als uppercase gebeuren. Lower case letters worden in getallen ook als lower case getoond, maar wel als uppercase behandeld.

Hoofdstuk 3. Commandobeschrijvingen.

In dit hoofdstuk worden alle mogelijke commando's uitgelegd. De commando's staan in alfabetische volgorde. De uitleg van een commando begint steeds aan het begin van een pagina.

3.1 Commando B: kies de adresgrenzen voor EPROM en RAM.

Na het invoeren van B en return, verschijnt de huidige instelling op het scherm. Om dit commando zinnig te kunnen gebruiken, eerst iets over de beperkingen.
Ten eerste kent dit commando geen defaults, anders dan de huidige instelling. De default instelling kan worden verkregen door met D commando een EPROM met een andere grootte te kiezen. Dit zet de adresstap op 1, het EPROM startadres op 0000, het EPROM eindadres op de grootte van de EPROM (behalve 27512) en het RAM startadres op 1000. De tweede beperking wordt gevormd door het feit dat er slechts 32k byte geheugen voor opslag van de EPROM-inhoud beschikbaar is. Dit is voldoende om de complete inhoud van alle ondersteunde typen behalve de 27512 bij een adresstap van 1 te kunnen opslaan. De RAM-geheugenindeling is, als EP geladen is, als volgt:

Adres	Gebruik
0000-00FF	Zero page
0100-01FF	Stack
0200-0FFF	EP software
1000-8FFF	Vrij voor opslag EPROM inhoud
9000-9FFF	MON65 (meegeladen met EP)
A000-A9FF	Vrij voor DOS65 utility
AA00-AFFF	DOS buffers
C000-DFFF	DOS65

Er is in totaal dus 32k byte vrij om de EPROM inhoud op te slaan en te bewerken. Het eerste bruikbare RAM adres is 1000. Dit is de default waarde voor het RAM startadres. De defaultwaarden voor de begin- en eindadressen van de EPROM zijn respectievelijk 0000 en het maximum EPROM adres, behalve voor de 27512. Voor deze laatste EPROM is het eindadres default 7FFF.
Na het tonen van de huidige instelling wordt om een nieuw EPROM startadres gevraagd. Invoeren van alleen (CR) laat de huidige waarde staan en springt naar de invoer van het EPROM-eindadres. (ESC) gevolgd door (CR) verlaat het B-commando, waarbij alle instellingen ongewijzigd blijven. Behalve deze twee bijzondere mogelijkheden, kan men natuurlijk ook een hexadecimaal adres invoeren. Men kan net zoveel cijfers invoeren als men wil. Zijn het minder dan vier, dan wordt het ingevoerde getal aan de voorzijde denkbeeldig aangevuld met nullen totdat het vier cijfers heeft. Zijn het er meer dan vier, dan worden de laatste vier cijfers genomen. Het intoetsen van DEL of RUB wist het laatst ingevoerde karakter. Control-Y laat de huidige bufferinhoud zien. De invoer van een adres dient te wordt afgesloten met een (CR).
De volgende vraag betreft het EPROM eindadres. Hier geldt hetzelfde als bij het startadres: alleen (CR) laat het huidige adres staan, (ESC) gevolgd door (CR) verlaat het commando met de huidige instelling, waarbij een eventueel veranderd EPROM

startadres de nieuwe waarde krijgt. Verder kan men een adres invoeren, waarbij weer dezelfde regels gelden.
Na de (CR) achter het EPROM eindadres controleert EP de beide EPROM adressen. Indien er iets fout is, wordt er 'EPROM size exceded' gemeld, en vraagt EP om een nieuw EPROM beginadres. Hier kan men weer met (CR), (ESC), enz. verder. De volgende oorzaken kunnen aanleiding voor de melding zijn:

- Het startadres ligt buiten de geldige adresgrenzen van de gekozen EPROM.
- Het eindadres ligt buiten de geldige adresgrenzen van de gekozen EPROM.
- Het beginadres is hoger dan het eindadres.
- Bij de 27512 is een groter deel dan 32k byte gekozen.

Indien de EPROM adressen geaccepteerd worden, vraagt EP tenslotte om het RAM beginadres. Ook hier geldt weer: alleen (CR) laat de huidige waarde staan, (ESC) gevolgd door (CR) verlaat het commando, waarbij eventuele veranderingen in de EPROMadressen wel aangepast worden. Ook kan weer op dezelfde manier een adres worden ingevoerd.
Na de (CR) rekent EP het nieuwe RAM-eindadres uit. Indien dit beneden 9000 ligt worden de huidige adressen getoond en verschijnt de prompt weer. Indien adres 8FFF wordt overschreden, verschijnt er 'RAM size exceded' en wordt om een nieuw EPROM start adres gevraagd, enzovoort. De melding 'RAM size exceded' verschijnt ook, wanneer een RAM-beginadres beneden 1000 wordt ingevoerd.

Opmerking: de adresstap.
Men dient reeds bij het invoeren van de EPROMadressen rekening te houden met de gekozen adresstap: Het gebruikte RAMbereik wordt immers ook door de waarde van de adresstap bepaald. Een voorbeeld: bij een adresstap van 4 is de maximale grootte van de te gebruiken EPROM 32k (de beschikbare RAM-ruimte) gedeeld door 4 is 8k byte. Bij deze instelling kan dus maximaal een gehele 2764/2564 geprogrammeerd worden, bij grotere EPROMs zal men dan een deel van de EPROM moeten kiezen. Eventueel moet men dus eerst met het commando I de juiste adresstap instellen. Voor een nadere uitleg van de adresstap, zie hoofdstuk 6.

3.2 Commando C: reken de RAM-checksum uit.

De checksum is het 16-bit resultaat van de optelling van alle bytes in een blok data. Bij deze optelling wordt de carry van de vorige optelling steeds verwaarloosd. Deze manier van checksum bepalen komt overeen met de methode die gebruikt wordt op de meeste professionele EPROMprogrammers.

De checksum die EP laat zien bij het lezen, programmeren en verifieren van EPROMs heeft steeds betrekking op de data in de EPROM. Hierbij wordt steeds uitgegaan van de gekozen adresgrenzen binnen de EPROM. In sommige gevallen echter dient de checksum een bepaalde waarde te hebben. Een voorbeeld hiervan zijn ROMs voor IBM-PC's: deze dienen een checksum te hebben waarvan het low byte nul is. Dergelijke EPROM-inhouden hebben in zo'n geval ook een byte dat dient om de checksum van de gehele EPROMdata te kunnen corrigeren. Met het C-commando kan de checksum van gekozen RAMbereik bepaald worden nadat deze data gewijzigd is, maar voordat er een EPROM geprogrammeerd wordt.

Met C (CR) wordt de checksum van de RAM-data uitgerekend en getoond. Men kan een bepaald deel van het RAM kiezen door middel van het B-commando. Het C-commando gebruikt de ingestelde waarden voor het RAMbegin- en -eindadres, alsmede de huidige instelling voor de adresstap.

3.3 Commando D: kies een EPROMtype.

Na het invoeren van het commando D verschijnt op het rechter deel van de statusregel het huidige type, in normaal video. Door nu op de plus- en/of mintoetsen te drukken kan men door het scala aan typen stappen en steeds resp. het volgende of het vorige type kiezen. De lijst is circulair, dat wil zeggen dat na selectie van het laatste type de eerste weer verschijnt bij een plus, of na het kiezen van het eerste type verschijnt het laatste bij een min.

Op de statusregel verschijnt niet alleen het typenummer, maar ook de programmeerspanning en het algoritme. Combinaties van programmeerspanningen en algoritmen die volgens de datasheets niet zijn toegestaan of niet voorkomen (bijvoorbeeld 2764, 21V, Quick of 27256, 21V, 50 ms) zijn niet opgenomen.

Door op (CR) te drukken wordt het type op de statusregel gekozen. Als het basistype niet veranderd is (dus bijvoorbeeld van 2764 21V 50ms naar 2764A 12.5V Quick) blijven de adresgrenzen op de huidige instelling staan. Wordt echter een ander basistype gekozen (bijvoorbeeld 27128 naar 27512, maar ook 2764 naar 2564), dan worden de adresgrenzen op de default waarden gezet. Bovendien meldt EP dan: 'WARNING: boundaries reset to defaults'. Met het tonen van de thans ingestelde waarden voor de adresgrenzen gevolgd door de gewijzigde prompt wordt de devicekeuze afgerond.

De defaultwaarden zijn als volgt:

EPROM startadres: 0000
EPROM eindadres : grootte van de EPROM, max. 7FFF
RAM startadres : 1000
RAM eindadres : 1000 + grootte van de EPROM, max. 8FFF

3.4 Commando E: kijk of een EPROM leeg is.

Met het commando E kan men zien, of een EPROM leeg is, bijvoorbeeld om te controleren of een EPROM goed werd gewist. Dit commando vergelijkt de EPROM-data met de waarde FF (de ongeprogrammeerde toestand). Indien het gekozen EPROM adresbereik uitsluitend FF als data bevat, volgt vervolgens de melding:

EPROM is empty.

Wordt een andere waarde dan FF gevonden, dan volgt de melding:

EPROM is NOT empty.

Dit commando doet alleen een uitspraak over het al of niet leeg zijn, er wordt geen adres-, data- of checksuminformatie gegeven.

3.5 Commando I: verander de adresstap.

Behalve het rechttoe-rechtaan werken met adressen, heeft EP ook mogelijkheden om met 16- en 32-bit data te werken. Dit wordt mogelijk gemaakt door de variabele adresstap. Met dit commando wordt de adresstap gekozen. Na het geven van het I-commando wordt de huidige adresstap getoond, en wordt om een nieuwe waarde gevraagd. (Voor een gedetailleerde uitleg van de adresstap, zie hoofdstuk 6). De gebruiker kan nu 4 dingen invoeren:

1 (CR)	Dit zet de adresstap op 1.
2 (CR)	Dit zet de adresstap op 2.
4 (CR)	Dit zet de adresstap op 4.
(CR)	Dit laat de huidige adresstap staan.

Andere waarden dan 1, 2 of 4 worden niet geaccepteerd.

3.6 Commando M: roep MON65 aan.

Het commando M roept de met EP meegeladen monitor MON65 aan als subroutine. Dit houdt in, dat na het invoeren van het commando Q in de monitor men weer in EP terugkeert.
Dit commando is aanwezig, omdat met EP geen RAM-data verplaatst of veranderd kan worden. Toevoegen van deze functies zou het dupliceren van de mogelijkheden van MON65 betekenen. De meegeladen versie van MON65 is de normale uitvoering: derhalve is de gehele MON65 manual van toepassing.
Door Q in MON65 in te voeren keert men weer terug in EP, dat reageert door de prompt te laten zien en om een commando te vragen.

3.7 Commando P: programmeer een EPROM.

Commando P bestaat eigenlijk uit drie commando's: eerst wordt gekeken of de EPROM inderdaad geprogrammeerd mag worden. EP controleert of een nul-bit in de EPROM door het programmeren in een een-bit zou veranderen. Is dit het geval, dan verschijnt er: 'Fail' en wordt het commando beëindigd.

Als de EPROM programmeerbaar blijkt, verschijnt er OK en wordt met programmeren begonnen. Hiertoe wordt de programmeerspanning ingeschakeld en de Vpp-LED zal gaan branden. Vervolgens wordt de EPROM in de voet geprogrammeerd volgens het thans gekozen algoritme. Op de volgende regel verschijnt er: 'Programming...', gevolgd door het EPROMadres dat thans geprogrammeerd wordt. Hiermee heeft men een indruk hoe het programmeerproces vordert.

Bij het 50 ms algoritme wordt het gehele adres getoond; bij de andere algoritmen verschijnen alleen de eerste twee cijfers van het adres.

Bij het Inteligent en het Quick Pulse algoritme is het mogelijk, dat een EPROMadres niet binnen het maximum aantal programmeerpulsen geprogrammeerd kan worden. In zo'n geval verschijnt er 'Program fail.' en wordt het P-commando beeindigd. Programmeerfouten bij het 50 ms algoritme komen pas in de verify-fase van het P-commando aan het licht.

Om een zo kort mogelijke programmeertijd te bewerkstelligen, worden adressen waarin FF als data geprogrammeerd moet worden, overgeslagen. De bit-test garandeert namelijk dat er ook FF in de EPROM staat op die adressen.

Als het laatste EPROMadres geprogrammeerd is, volgt een verify; de EPROM-inhoud wordt met de te programmeren RAM-data vergeleken. Zie hiervoor het V-commando, paragraaf 3.12.

3.8 Commando R: kopieer een EPROM-inhoud naar RAM.

Het commando R leest de inhoud van de EPROM in de ZIF-voet in het RAM-geheugen. Hierbij worden de ingestelde adresgrenzen en de huidige adresstap in acht genomen. Na het lezen van de EPROM wordt de checksum van de gelezen data getoond. Indien de checksum van de data van te voren bekend is (bijvoorbeeld omdat deze op het etiket van de EPROM staat) kan de checksum een indicatie zijn of de data in orde is en correct werd gelezen.

3.9 Commando Q: verlaat EP.

Door het invoeren van het commando Q wordt de EP-software verlaten en keert men terug naar het DOS. Hierbij wordt het scherm gewist. Omdat EP een subroutine is (het eindigt op een RTS) kan het eventueel ook door andere programma's als subroutine worden aangeroepen. In zo'n geval wordt met Q teruggekeerd naar het aanroepende programma.

3.10 Commando S: laat de EPROM-inhoud zien.

Met het commando S kan rechtstreeks in de EPROM in de ZIF-voet gekeken worden. De EPROM-inhoud wordt dus NIET in RAM geladen. Dit kan zinvol zijn als de RAM-inhoud niet verloren mag gaan. De uitvoer van het S-commando heeft de vorm van een normale hexdump. De data wordt zowel in hex, als in ASCII getoond. Het S-commando begint met de uitvoer met het huidig ingestelde EPROM-startadres, en dumpt vervolgens 16 regels van 16 bytes (1 pagina, ofwel 256 bytes). Daarna wordt de voedingsspanning van de EPROM uitgeschakeld en wacht EP op een toets. Er kunnen nu drie toetsen worden gebruikt:

1. (CR). Met deze toets verlaat men het S-commando. De prompt verschijnt weer.
2. +. Na het invoeren van een plus worden de volgende 16 regels gedumpt. Hierna kan men weer een plus, een min of een (CR) invoeren. Bij het herhaald invoeren van plus-toetsen, zal op een gegeven moment het EPROM-eindadres gepasseerd worden. De volgende plus geeft dan een dump beginnend op EPROMadres nul.
3. -. Na het invoeren van een min wordt het vorige blok van 16 regels gedumpt. Hierna kan men weer een plus, een min of een (CR) invoeren. Indien het huidig gedumpte blok begint met EPROMadres nul, dan zal bij het invoeren van een min een blok gedumpt worden dat begint met het maximum EPROMadres, verminderd met FF.

3.11 Commando T: test of de EPROM programmeerbaar is.

Met het commando T kan gekeken worden of een EPROM geprogrammeerd kan worden met de data die thans in RAM staat. Dit commando wordt ook door het P-commando uitgevoerd voordat het werkelijke programmeren begint.
Het T-commando zelf kent ook twee fasen, die voor de gebruiker niet zichtbaar zijn. Eerst wordt gekeken of de EPROM leeg is. Is dit het geval, dan verschijnt er 'EPROM is empty' op het scherm, gevolgd door de prompt.
Is de EPROM niet leeg, dan wordt er vervolgd met een test die controleert of een programmeeractie een nul-bit in de EPROM zou veranderen in een een-bit. (Dit is alleen mogelijk door een EPROM te wissen). Indien dit niet gebeurt, verschijnt er 'EPROM is programmable' en verschijnt de prompt. Indien de EPROM niet geprogrammeerd kan wordt met de huidige RAMdata verschijnt er 'EPROM is NOT programmable'. In dit geval zal men de EPROM eerst moeten wissen.

3.12 Commando V: vergelijk EPROM- en RAMdata.

Het commando V vergelijkt de inhoud van het gekozen EPROMbereik met het gekozen RAM-bereik. Indien er geen verschillen zijn wordt er 'EPROM equals RAM' afgedrukt en verschijnt de prompt. Zijn er verschillen, dan wordt het eerste verschil gemeld door het tonen van EPROMadres, EPROMdata en RAMdata in hex, zodat er eventueel conclusies uit de verschillen getrokken kunnen worden. Door het invoeren van een spatie wordt het volgende verschil getoond. Indien er geen verschillen meer zijn verschijnt er 'No more differences' gevolgd door de prompt. Wil men de verdere verschillen niet weten, dan kan door het invoeren van (CR) het commando beëindigd worden.
Het commando P roept na het programmeren dit commando aan.

3.13 Commando ?: laat alle mogelijke commando's zien.

Het commando ? laat een lijstje zien met alle commando' met een korte beschrijving. Deze tekst verschijnt ook bij het starten van EP. Na het afdrukken ervan verschijnt de prompt weer.

Hoofdstuk 4. De keuze van het EPROM-type en het programmeeralgoritme.

4.1 EPROMtypen.

EP ondersteunt een groot aantal gangbare EPROMtypen. Hieronder een lijstje van de typen met hun organisaties:

2716	2048 x 8 bit = 16384 bit of:	2k byte
2732	4096 x 8 bit = 32768 bit of:	4k byte
2764	8192 x 8 bit = 65536 bit of:	8k byte
27128	16384 x 8 bit =131072 bit of:	16k byte
27256	32768 x 8 bit =262144 bit of:	32k byte
27512	65536 x 8 bit =524288 bit of:	64k byte
2532	4096 x 8 bit = 32768 bit of:	4k byte
2564	8192 x 8 bit = 65536 bit of:	8k byte

De typen 2716, 2732 en 2532 zijn ondergebracht in een 24-pins behuizing, de overige typen zitten in een 28-pins behuizing.

4.2 Keuze van het EPROM-type.

De hierboven genoemde typenummers zijn te beschouwen als basistypenummers. Deze zitten verstopt in het complete typenummer dat op de behuizing vermeld staat.
In het algemeen bestaat een typenummer op een EPROM uit de volgende delen:

1. Het fabrikanten-voorvoegsel.
2. Het basistypenummer.
3. Een code voor de behuizing.
4. Een code voor het temperatuurbereik.
5. Een code voor de toegangstijd.

(De delen 3 en 4 zijn soms gecombineerd).

4.2.1. Het fabrikanten-voorvoegsel.

Iedere fabrikant heeft zijn eigen, unieke voorvoegsel dat in de meeste gevallen uit een of meer letters bestaat. Soms bevat het voorvoegsel informatie over het soort IC of de technologie waaruit het IC is opgebouwd. Een aantal gangbare voorvoegsels zijn:

AM (AMD)
DQ (Seeq)
ET (Eurotechnique)
HN (Hitachi)
M (SGS)
M5L (Mitsubishi)
MBM (Fujitsu)
MCM (Motorola)
TC (Toshiba CMOS)
TMM (Toshiba NMOS)
TMS (Texas Instruments)
i (Intel)
(uP)D (NEC)

4.2.2 Het basistypenummer.

Dit nummer komt na het fabrikanten-voorvoegsel, en begint in het geval van EPROMs meestal met een 2. Sommige fabrikanten, zoals Seeq hanteren daarnaast afwijkende nummers. Het basistypenummer vertelt iets over de grootte (in kilo-bits) en de pinning (25- of 27-serie). Als het basistypenummer met 24 begint dat betreft het een 27-serie EPROM in een kunststofbehuizing, die eenmalig geprogrammeerd kan worden. Een voorbeeld hiervan is een 24256. Dit is een 27256 in plasticbehuizing. EP kan de 24-reeks ook programmeren: kies het overeenkomstige 27-type.
Het basistypenummer kan nog gevolgd worden door een A. In dat geval betreft het een EPROM van hetzelfde type als het type zonder A, maar met een lagere programmeerspanning. Ook zijn de A-typen meestal wat sneller dan hun standaard-broertjes.
Tenslotte kan er na de 27 of 24 nog een C in het basistypenummer voorkomen. Het betreft dan een EPROM gemaakt in CMOS-technologie; dus met een laag stroomverbruik. Deze EPROMs hebben zonder uitzondering hetzelfde programmeeralgoritme als hun NMOS-broertjes.

4.2.3. Behuizing/temperatuurbereik.

Deze codes staan achter het basistypenummer en bevatten nooit een A, die immers een afwijkende programmeerspanning aangeeft. Deze codes zijn voor iedere fabrikant uniek. Voor het gebruik met EP zijn ze niet belangrijk: als het huisje in de ZIF-voet past is alles OK.

4.2.4. Snelheidscodes.

Deze code staat achter de code voor de behuizing. De meeste fabrikanten gebruiken een code die met een streepje begint en waaruit direct de toegangstijd in honderden of tientallen nanoseconden kan worden afgeleid. Als de code ontbreekt dan betreft het een EPROM met standaard toegangstijd. De standaard toegangstijd is niet voor ieder merk EPROM dezelfde, maar ligt meestal rond de volgende waarde:

Basis	Standaard	A-type
2716	450 ns	-
2732	450 ns	300 ns
2764	300 ns	250 ns
27128	300 ns	250 ns
27256	250 ns	200 ns
27512	250 ns	-
2532	450 ns	-
2564	450 ns	-

De toegangstijd is wel belangrijk voor de uiteindelijke toepassing van de EPROM maar niet voor EP: de 3 MHz versie van EP leest EPROMs zodanig langzaam uit, dat typen met een toegangstijd van 4 us (4000 ns) nog betrouwbaar kunnen worden gelezen.

4.3. Keuze van de programmeerspanning.

Tijdens het programmeren van een EPROM wordt op een pen van de EPROM een extra hoge spanning gezet, de programmeerspanning. Deze spanning heeft afhankelijk van het type EPROM een verschillende waarde. De hardware van EP kan spanningen van 5, 12.5, 21 en 25 Volt genereren.
LET OP: A-versies hebben altijd een lagere programmeerspanning dan standaardtypen.
Ofschoon de programmeerspanning alleen bij de grotere EPROMs op het huisje staat (in de vorm: 'PGM @ 21 V' o.i.d.) zit er wel enige regelmaat in. Dit zijn de meest voorkomende waarden:

Basis	Standaard	A-type
2716	25 Volt	-
2732	25 Volt	21 Volt
2764	21 Volt	12.5 Volt
27128	21 Volt	12.5 Volt
27256	21 Volt	12.5 Volt
27512	12.5 Volt	-
2532	25 Volt	-
2564	25 Volt	-

WAARSCHUWING: KEUZE VAN EEN TE HOGE PROGRAMMEERSPANNING HEEFT ONMIDDELLIJKE VERNIELING VAN EEN EPROM TOT GEVOLG ZODRA DE PROGRAMMEERSPANNING INGESCHAKELD WORDT.
Raadpleeg daarom bij twijfel altijd eerst het datasheet!

4.4. Keuze van het programmeeralgoritme.

In de loop der tijden zijn door het voortschrijden der techniek alsmede uit praktische overwegingen als totale programmeertijd, een aantal programmeeralgoritmes ontstaan. EP ondersteunt de drie meest gangbare, het 50 ms algoritme, het Intel Inteligent algoritme en het Intel Quick Pulse algoritme.
Helaas is het niet zo, dat men een willekeurige EPROM met een willekeurig (bij voorkeur het snelste) algoritme kan of mag programmeren. Allereerst is er het datasheet van de fabrikant van de EPROM. Dit datasheet schrijft over het algemeen een algoritme voor. Soms mag een EPROM naar keuze met twee verschillende algoritmen geprogrammeerd worden; een 2764 mag bijvoorbeeld vrijwel altijd met 50 ms of met het Inteligent algoritme geprogrammeerd worden. Net als bij de programmeerspanning is er geen vaste regel, maar wel een algemene tendens aan te geven. Op de volgende pagina staat een lijstje:

Basis	Standaard	Alternatief
2716	50 ms	-
2732	50 ms	-
2732A	50 ms	-
2764	Inteligent	50 ms
2764A	Inteligent	Quick Pulse
27128	Inteligent	50 ms
27128A	Inteligent	Quick Pulse
27256	Inteligent	-
27256A	Inteligent	Quick Pulse
27512	Inteligent	Quick Pulse
2532	50 ms	-
2564	50 ms	-

Ook hier geldt weer: bij twijfel het datasheet raadplegen! Een gedetailleerde beschrijving van de door EP gegenereerde algoritmen vindt men in hoofdstuk 5.

Hoofdstuk 5. Gedetailleerde algoritme beschrijving.

5.1 Het klassieke 50 milliseconde algoritme.

Dit algoritme is het oudst van alle algoritmen; het is ook het traagst. Het wordt voornamelijk gebruikt bij de typen met de kleinste organisaties. Het algoritme is betrekkelijk eenvoudig.
Een stroomdiagram van het algoritme vindt men in bijlage A. Het programmeren van een EPROM-locatie vindt plaats door de EPROM-databus in three-state te zetten, vervolgens het te programmeren adres aan te bieden en stabiel te houden, de data op de EPROM-databus te zetten en tenslotte een puls met een lengte van 50 milliseconde op de programmeerpin aan te bieden. Een controle of de data ook werkelijk in de EPROM is geprogrammeerd vindt tijdens het programmeerproces niet plaats; het algoritme schrijft dan ook voor dat er na het programmeren van alle te programmeren adressen een vergelijking tussen de EPROM inhoud en te programmeren data dient plaats te vinden met afgeschakelde programmeerspanning en nominale voedingsspanning. Programmeerfouten komen dus pas bij het verifieren aan het licht.
Het is bij dit (en ook bij de andere) algoritme toegestaan EPROM locaties in willekeurige volgorde en willekeurig aantal te programmeren. Ook mag een locatie worden overgeprogrammeerd, mits er geen nul-bits in een-bits veranderen. De EPROM hoeft dus niet noodzakelijk leeg te zijn.

5.2. Het Intel Inteligent Algoritme.

Dit algoritme werd zoals de naam al aangeeft, door Intel ontwikkeld en werd voor het eerst gebruikt voor de 2764. In tegenstelling tot het 50 ms algoritme is dit algoritme intelligent, hetgeen inhoudt dat de totale lengte van de programmeerpuls niet langer is dan noodzakelijk om de EPROM correct te programmeren. Omdat tijdens het programmeerproces ook in de EPROM wordt gelezen, wordt dit algoritme ook wel als interactief algoritme aangeduid.
Het stroomdiagram van dit algoritme is in bijlage B gegeven. Het programmeerproces voor een locatie verloopt als volgt. Eerst wordt de voedingsspanning van EPROM verhoogd van 5 naar 6 Volt. Dit heeft een verschuiving van alle logische niveaus tot gevolg, en wel zodanig dat het niveau waarop een logische nul wordt herkend a.h.w. naar beneden wordt verlegd. (Bij 5 Volt is een nul gedefinieerd als een spanning van 0.8 Volt of lager. Omdat bij 6 Volt wordt uitgelezen zal het nul-niveau bij 5 Volt dus tenminste 5/6 maal 0.8 = .667 Volt zijn). Vervolgens wordt de programmeerspanning ingeschakeld en de EPROM-databus in three-state gezet. Daarna worden het te programmeren adres en de te programmeren data op de EPROM gezet. Nu begint het programmeerproces door het aanbieden van een puls van 1 milliseconde op de programmeerpin. Na het verstrijken van de puls wordt de EPROM uitgelezen door OE en CS beide laag te maken. (Bij sommige typen heeft dit het afschakelen van de programmeerspanning tot gevolg). Wordt de data niet juist teruggelezen, dan volgt een volgende programmeerpuls van 1 milliseconde, enzovoort.

Dit pulsen wordt volgehouden totdat 1 van de volgende twee condities optreedt:

- Het totaal aantal pulsen bedraagt 25
- De data wordt correct teruggelezen.

In het eerste geval wordt het programmeerproces gestopt en dient de EPROM afgekeurd te worden. EP meldt in dat geval 'Program Fail'.

In het tweede geval wordt dezelfde locatie overgeprogrammeerd, en wel met een pulslengte van driemaal het aantal pulsen die tot dusver nodig waren om de locatie te programmeren, of met driemaal het aantal pulsen dat tot dusver nodig was. EP gebruikt de laatste mogelijkheid.

Ook bij dit algoritme is een verificatie van de EPROMdata bij nominale voedingsspanning voorgeschreven, en wel vanwege het feit dat de EPROM tijdens het programmeren niet op de normale voedingsspanning wordt bedreven, en omdat bij het lezen tijdens het programmeren de programmeerspanning ingeschakeld blijft.

De programmeertijd bij het Inteligent algoritme geeft een indruk van de kwaliteit van de EPROM: de kortste programmeertijd (4 ms per locatie) treedt op indien de data reeds na het geven van 1 enkele puls wordt teruggelezen. De langste tijd (100 ms per locatie, langzamer dan het 50 ms algoritme) treedt op indien iedere locatie precies 25 pulsen nodig heeft voordat de data teruggelezen wordt. In de praktijk schommelt het aantal initiële pulsen tussen de 1 en 4 stuks.

5.2.1. Variaties op het Intel Inteligent Algoritme.

Een aantal fabrikanten schrijft een algoritme voor dat in details afwijkt van het hierboven genoemde algoritme.

De eerste variatie is, dat bij het bereiken van de 25e puls, er eerst een overprogrammering met een factor drie dient plaats te vinden, gevolgd door een verify, alvorens de EPROM wordt afgekeurd. Deze variatie is niet in EP geïmplementeerd.

De tweede variatie betreft het maximum aantal initiële pulsen, en de overprogrammeerfactor. Bij het Intel Inteligent algoritme bedragen deze waarden 25 en 3. Hieronder staat een lijstje met een aantal variaties:

Naam	A (Initieel)	B (overprogrammeerfactor)
AMD Interactive	25	2ms vast
Fujitsu Quick Pro	20	1
Intel Inteligent	25	3
Intel Inteligent B	15	4 (Alleen bij 27128)
Toshiba Fast I	15	4
Toshiba Fast II	20	4

Diegenen die het zo precies mogelijk willen doen kunnen EP eventueel aanpassen. Op het startadres+3 staat namelijk het maximale aantal initiële pulsen, op startadres+4 de overprogrammeerfactor. Bij de normale implementatie van EP zijn dit de adressen $203 en $204.

5.3. Het Intel Quick Pulse algoritme.

Dit algoritme is supersnel. Het zelfs zo snel dat men duidelijk begint te merken dat de programmer nog meer moet doen dan alleen programmeerpulsen genereren.
Het algoritme komt grotendeels overeen met het Inteligent algoritme, zij het dat er twee verschillen zijn. Het eerste verschil is, dat de pulslengte niet 1 milliseconde maar 100 microseconde bedraagt. Het tweede verschil is, dat er niet overgeprogrammeerd wordt. Voor de rest is alles hetzelfde, inclusief het maximum aantal programmeerpulsen per locatie: 25. Bij overschrijding daarvan meldt EP ook hier 'Program Fail'.

5.4. Theoretische programmeertijden.

Hieronder staat een lijst met de theoretische minimale programmeertijden. In de praktijk zal het programmeren wat langer duren, omdat EP eerst een bit-test doet en na het programmeren verifieert. Ook kost het genereren van het EPROM-adres en het ophalen/wegschrijven van de RAM-data enige tijd.

Type	50 ms	Intel min.	Intel max.	Quick min.	Quick max.
2716	1'42''				
2732	3'24''				
2764	6'49''	0'33''	13'39''	0.82''	20.5''
27128	13'39''	1'05''	27'18''	1.64''	41''
27256		2'11''	54'37''	3.27''	1'22''
27512		4'22''	1h48'55''	6.55''	2'44''
2532	3'24''				
2564	6'49''				

Hoofdstuk 6. De adresstap.

Met het commando I kan de adresstap worden ingesteld op 1, 2 en 4. De adresstap is de waarde die EP bij het RAM-adres telt, alvorens de volgende EPROM locatie te programmeren met de inhoud van het dan verkregen RAM-adres. De adresstap wordt bij alle commando's die het RAM van de computer gebruiken, gebruikt. De adresstap komt van pas bij het bewerken van data bedoeld voor 16- en 32-bit microprocessors.

Deze processors benutten 16- of 32-bits data tegelijkertijd. Daar EPROMs meestal 8-bit breed zijn, worden bij deze processors steeds paren respectievelijk stellen van 4 EPROMs gebruikt. Bij een 16-bit systeem bevat de ene EPROM steeds de even bytes, en de andere de oneven bytes. Bij een 32-bit processor gaat dit nog een stap verder; EPROM 1 van de set van vier bevat de bytes waarvan het adres op binair 00 eindigt, EPROM 1 de bytes met 01, EPROM 2 met 10 en tenslotte bevat EPROM 3 alle bytes met adressen die binair op 11 eindigen.

Kopieren van dergelijke EPROMs kan zonder meer met EP en iedere andere programmer gedaan worden. Wil men de data echter veranderen, dan moet deze in leesbare volgorde in het geheugen komen. Hiervoor is de adresstap ingevoerd. Bij 16-bit data zet men de adresstap op 2, bij 32-bit data op 4. Verder werkt men als volgt:

Kies met het commando B de gewenste EPROM adresgrenzen, en het gewenste RAM-start adres, dat EVEN moet zijn. Plaats vervolgens de even-EPROM in de ZIF-voet, en lees hem uit met het R-commando. De EPROM-data wordt nu op even RAM-adressen geladen. Verwijder de EPROM uit de ZIF-socket en vervang hem door de oneven EPROM die erbij hoort. Kies weer commando B. Laat de EPROM adressen staan door tweemaal (CR) in te voeren. Kies voor het RAM begin adres dezelfde waarde, doch ONEVEN, en wel 1 hoger. Lees nu de oneven EPROM in (met het R-commando). Deze data komt op de oneven RAM-adressen, precies tussen de data van de even EPROM in. De data kan nu normaal bestudeerd, veranderd en gesaved worden.

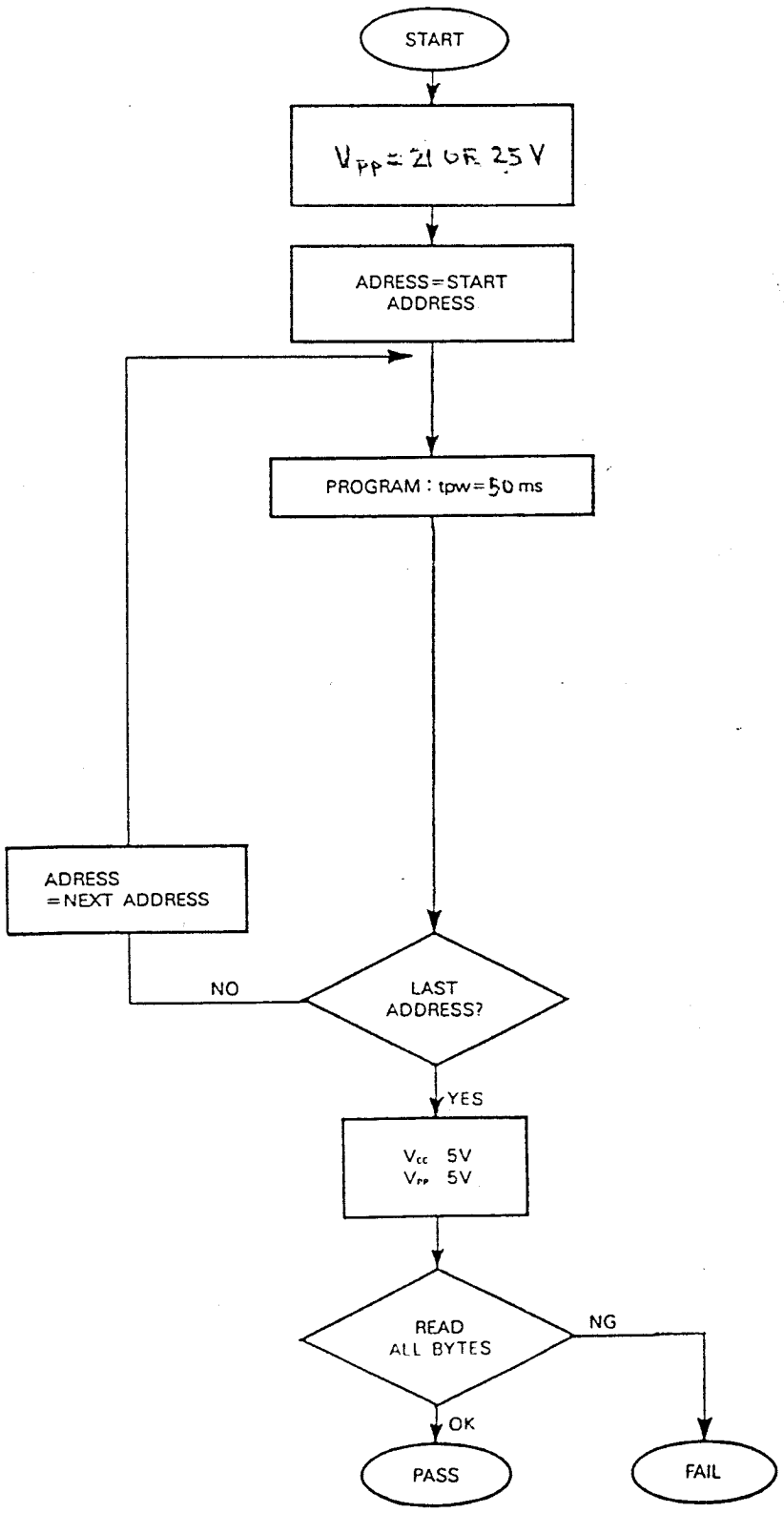
Na eventuele veranderingen laat men de adresstap en de adresgrenzen staan, en programmeert dus eerst de oneven EPROM. Daarna kiest men met B weer dezelfde EPROM-adressen, maar een even RAM beginadres, nu 1 lager (dus hetzelfde adres als bij het inlezen van de even EPROM). Daarna programmeert men de nieuwe even EPROM.

De gang van zaken bij 32-bit is dezelfde; zij het dat hier alles viermaal gedaan moet worden; men kieze in dit geval bij voorkeur een RAM-start adres dat resp. op 0, 1, 2 en 3 eindigt.

Het commando C vormt geen uitzondering op de andere commando's: ook bij de bepaling van de checksum wordt de ingestelde adresstap gebruikt. Dit houdt in, dat het C-commando de checksum uitrekent van de data die in 1 enkele EPROM van een paar of stel zou terechtkomen. Dit is niet altijd gewenst: soms moet de data in het stel of paar EPROMs een bepaalde checksum hebben. In zo'n geval kieze men voor het geven het C-commando de 27512 EPROM, een adresstap van 1 en het begin- en eindadres van de data waarover men de checksum wil weten. Voor het programmeren zet men dan het type, het adresbereik en de adresstap weer op de oude waarde.

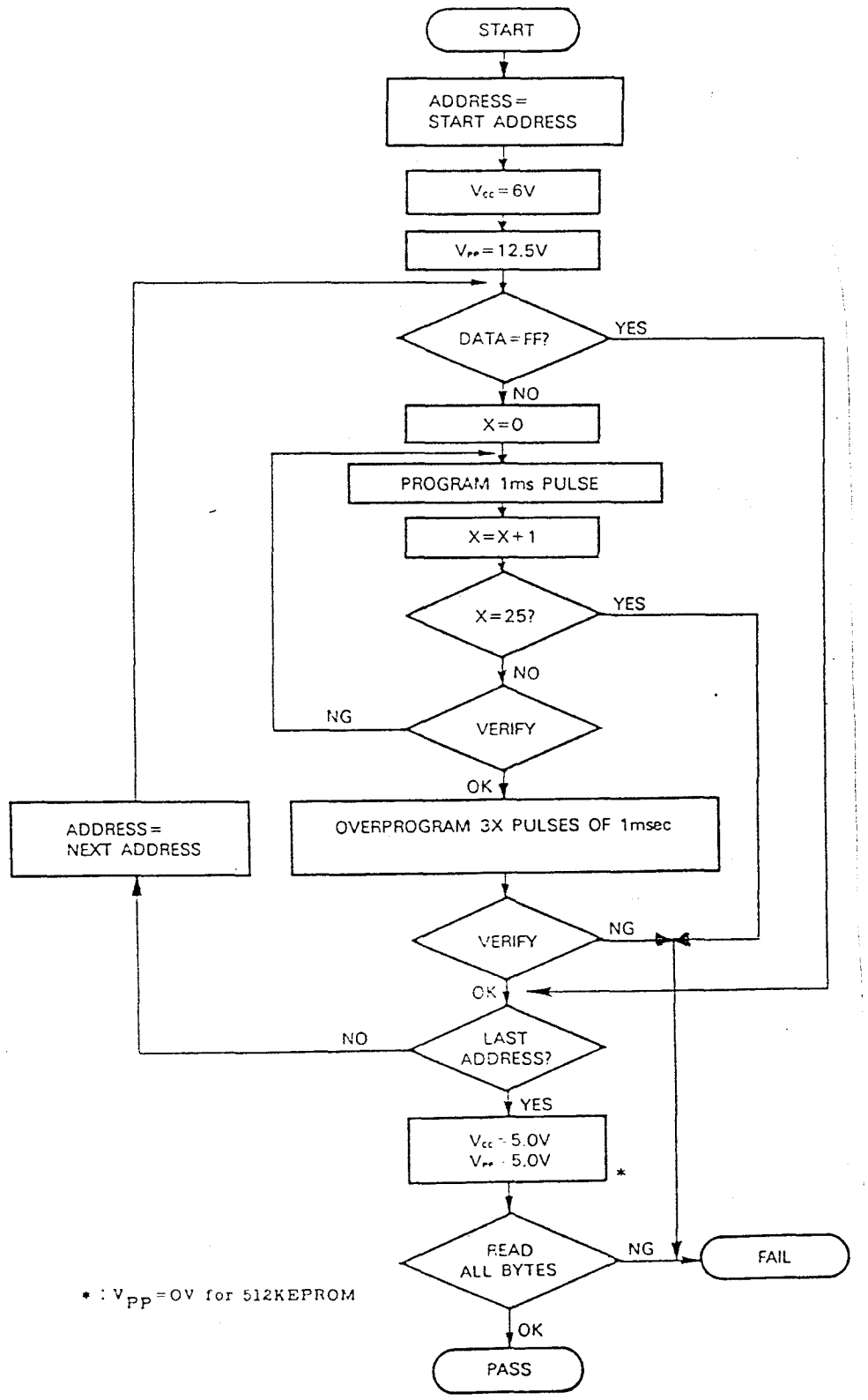
Bijlage A.

Flow chart 50 ms algoritme.



Bijlage B.

Flow chart Intel Inteligent programmeeralgoritme.



Bijlage C.

Flow chart Intel Quick Pulse programmeeralgoritme.