Automatisering deeltjesversneller

Achterop, S.

Award date:
1978

AUTOMATISERING DEELTJES VERSNELLER

S. ACHTEROP

Verslag van het afstudeerwerk, uitgevoerd bij de vakgroep Digitale Systemen.

Coach: Prof. ir. A. Heetman

INHOUDSOPGAVE
==========

-The 200kV industrial ion implantation system
-The 350kV heavy ion accelerator

Inleiding

Het afstuderen is op de volgende manier opgezet. Het project is opgedeeld in een aantal deelprojecten. Deze deelprojecten zijn al tijdens die projecten afzonderlijk gedocumenteerd, en deze documenten vormen het grootste deel van dit verslag. (zie de inhoudsopgave)

De rest van het verslag bestaat uit een samenvatting en afronding van het totole afstudeerwerk. Dit stuk plaatst tevens de afzonderlijke documenten in het totale project. Voor een motivatie van de gekozen opzet verwijs ik naar het hoofdstukje: doelstellingen.

het onderwerp

Het onderwerp van mijn afstuderen luidt:
automatisering deeltjesversneller.
Iets uitgebreider komt dat op het volgende neer:
Onderzoek in hoeverre het mogelijk is om een deeltjesversneller, die gebruikt wordt bij de productie van halfgeleiderkomponenten, te automatiseren, en probeer dit, voor zover als mogelijk, te realiseren.
Het gebruik van een kleine computer (bijv. een 8 bits machine, zoals de 8080) is hierbij een randvoorwaarde.
Dit onderzoek is gedaan ten behoeve van het bedrijf High Voltage Engineering (Europe) B.V. gevestigd te Amersfoort. (verder af te korten als HV)
Een nevendoel is, dat HV door dit onderzoek bekend wordt met de mogelijkheden en beperkingen van microcomputers e.d.
Dit onderwerp is ontstaan door mijn al bestaande kontakten met dit bedrijf. Ik heb in de loop van de afgelopen jaren al verscheidene keren bij dit bedrijf werkzaamheden verricht, waardoor ik vrij goed op de hoogte ben met de situatie in Amersfoort. Dit is voor mij vooral in het begin van het project van veel nut gebleken.
Tijdens mijn I-1 fase is het idee voor dit onderzoek als afstudeerproject ontstaan.
Prof. ir. A. Heetman van de vakgroep EB heeft zich bereidt verklaart om van de zijde van de TH als coach op te treden, terwijl van de zijde van HV ik vnl. geholpen ben door de heren F. van de Velde en J. Stronkhorst.
De afstudeerperiode is gestart op 1 september 1977 en wordt hopelijk 26 oktober 1978 afgesloten door de diploma uitreiking.

de deeltjesversneller en het ontstaan van het project.

De firma HV heeft bekendheid gekregen door de fabrikage van de zg. "van de Graaf versneller". Dit is een lineaire deeltjesversneller waarbij de benodigde hoogspanning verkregen wordt door de bekende van de Graaf generator.
Deze deeltjesversnellers worden van oudsher gebruikt voor kernfysische experimenten in laboratoria van universiteit en industrie.
De laatste jaren worden deeltjesversnellers steeds meer gebruikt bij de productie van halfgeleiderkomponenten. Tot voor enkele jaren werd bij het maken van halfgeleiders vnl. gebruik gemaakt van diffusietechnieken om de verontreinigingen in het halfgeleiderkristal te brengen.
Een techniek die de laatste jaren erg in opkomst is gekomen, is het implanteren van de verontreinigingen. Hiertoe worden de verontreiningen als ionen met een bepaalde snelheid (=kinetische energie) in het kristal geschoten. Na een kleine verhitting van het kristal blijken de ionen zich netjes in het rooster te hebben gepast.
Het implanteren van de verontreiningen heeft t.a.v. diffusietechnieken verschillende voordelen, waarvan ik er een aantal wil noemen.

- plaatsbepaling
  Bij implantatie is het mogelijk om d.m.v. de deeltjesenergie te bepalen hoe diep de deeltjes in het kristal dringen. De concentratie van de verontreinigingen uitgezet als functie van de diepte in het kristal bij diffusie is een dalende e-macht, terwijl er nu veel meer mogelijkheden bijgekomen zijn. Zo is het nu mogelijk om direct een begraven laag te implanteren.
- directe meting van de hoeveelheid geimplanteerde deeltjes
  Als we de te implanteren wafer via een coulombmeter met aarde verbinden, dan kunnen we direct de hoeveelheid (geladen) deeltjes meten die geimplanteerd worden.
- een grotere verscheidenheid in de te implanteren deeljes is nu mogelijk in vergelijking met de mogelijkheden van diffusietechnieken.
- de homogeniteit van de aangebrachte verontreinigingen over het kristaloppervlak is nu beter te krijgen.
- reproduceerbaarheid
  Was het met diffusietechnieken alleen mogelijk om voor halfgeleiders afkomstig van dezelfde wafer een goede reproduceerbaarheid te verkrijgen, nu is datzelfde mogelijk voor halfgeleiders van verschillende wafers. Dit wordt als een van de grootste voordelen gezien van de ionenimplantatie.
- schone productie
  Het implanteren gebeurd onder vacuumcondities, dit betekent dat het bijna niet voorkomt dat er ongewenste verontreinigingen in het kristal geraken.

Naast de voordelen is het belangrijkste nadeel dat de machine gecompliceerder is dan de diffusiemachine, met alle gevolgen van dien.
Met een lineaire deeltjesversneller is het mogelijk de gewenste ionen te produceren.
De firma HV heeft hiertoe een aantal deeltjesversnellers ontwikkeld, zie hiervoor de bijlagen.
Deze machines zijn echter niet optimaal geschikt om in een fabriek als productie eenheid te werken, daar het aanzetten en in bedrijf houden van de machine met de hand gebeurd en vrij arbeidsintensief en tijdrovend is. Met het goedkoper worden van computers werpt zich de vraag op, of het niet rendabel is, om dit (althans gedeeltelijk) te automatiseren.
Deze vraag is een andere formulering van mijn afstudeeronderwerp. Aan het begin van mijn afstuderen heb ik die vraag al met ja beantwoord, en mijn afstuderen heeft dan ook grotendeels bestaan uit het zover mogelijk realiseren van een ontwerp daartoe.

De werkelijke vraag waar het, althans voor HV om ging, was in hoeverre is er voor HV de mogelijkheid een dergelijke machine te ontwikkelen.
Hoewel er nog wel een aantal onzekerheden zij die de vraag oproepen of het mogelijk is om een volledig geautomatiseerde implanter te maken, is het ook los van die onzekerheden wel vast komen te staan dat een implanter in verregaande mate te automatiseren is. Het is wel zo dat er nog heeft wat moet gebeuren voor dat zo`n ma-

chine realiteit is, mede omdat in een groot aantal komponenten in de bestaande machine zitten die grondig gewijzigd zullen moeten worden als we volledig gebruik willen maken van de microprocessor en e.v.t. andere LSI komponenten.
Ik denk dat tot dit laatste besloten moet worden wil een dergelijk project een goede kans van slagen hebben.

Een van de onzekerheden in het project is het automatiseren van de ionenbron. Die heeft een zodanig moeilijk beschrijfbaar gedrag, dat het nog de vraag is of het mogelijk is om hem goed te automatiseren. Het lijkt mij een zeer interresant project, omdat dat voor zover mij bekent iets echt nieuws is.
Hoewel het aan het begin van het afstuderen in de bedoeling lag om al tijdens het afstudeerproject een gedeelte van de gemaakte software werkelijk op een implanter (of gedeelte daarvan) uit te testen, bleek dat het binnen het afstuderen niet haalbaar was.

Doelstellingen

Het doel van het afstuderen is, behalve te laten zien dat je zelfstandig wetenschappellijk kunt werken, er iets van te leren. Dit laatste is voor mezelf het belangrijkste, en daarom wil ik proberen dit een beetje uit te werken, nl te beschrijven wat ik heb willen leren, en wat er van terecht is gekomen. Hierover denkende ben ik tot een vrij verrassende conclusie gekomen, nl. dat als je probeert in alle gebieden waar je werkzaam bent goed te werken, dat je dan eigenlijk steeds op hetzelfde terecht komt.
Dit wordt in verschillende vakgebieden steeds anders benoemd, maar het duidt m.i. toch steeds op hetzelfde, en dat is: Wil je een probleem aanpakken, maak dan eerst een duidelijke probleemstelling en construeer daarna pas de oplossing, je daarbij beperkend tot dingen die je kunt overzien.(of: probeer steeds vanuit expliciete doelstelingen te vertrekken) Ik zeg bewust construeer omdat mijn ervaring is, dat als de probleemstelling echt goed uitgewerkt is dat dan het maken van de oplossing vaak niet meer dan construeren is. Ik denk dat de meeste mensen dit nogal triviaal vinden, maar als je kijkt hoe weinigen zich er werkelijk aan houden (in mijn ogen althans) is het nog niet zo triviaal. Een van de belangrijkste dingen die ik, naast het concrete technische werk gedaan heb is me met bovenstaande vragen bezig gehouden en het als operationele vaardigheid zoveel mogelijk proberen te ontwikkelen.
In de afstudeerverslagen die ik in de laatste jaren bekeken heb, vind ik van een bewust bezig zijn met dingen die ik zojuist beschreven heb erg weinig terug.
Ik denk dat het nuttig is om ook hieraan tijdens het afstuderen (bijv. via het coachen) bewust (meer) aandacht te geven.

Het gebied waar ik het bij mezelf het meest heb ontwikkeld is het programmeren. De genoemde werkwijze wordt meestal "structured programming" genoemd. Hierbij staat voorop het jezelf beperken tot dingen die je kunt overzien.
Bij het schrijven van een programma of procedure komt die werkwijze op het volgende neer. Beschrijf eerst wat het programma moet

doen, d.i. beschrijf de data waarop de procedure werkt, en beschrijf (liefst wiskundig) welke "transformatie" het programma uitvoert op de data. Vervolgens kan het algorithme geconstrueerd worden uitgaande van de "transformatie" die het programma uit moet voeren. Hoewel ik niet wil beweren dat ik deze werkwijze consequent heb toegepast, ben ik nu vrij goed in staat dit voor toekomstige programma's wel te doen. Dit is iets wat ik tijdens mijn afstuderen heb geleerd.

Een tweede gebied is de aanpak van het gehele project. Om het geheel een beetje overzichtelijk te maken heb ik het project verdeelt in deelprocecten. Deze deelprojecten zijn ook steeds afzonderlijk gedocumenteerd. Deze documenten wormen het tweede gedeelte van mijn verslag. Anders gezegd heb ik eerst een kapstok gebouwd om alles aan op te hangen.
Deze aanpak heeft me, ook bij het maken van het verslag, veel diensten bewezen, daar het grootste gedeelte van mijn verslag uit in de loop van het jaar gemaakte documenten bestaat, zodat het maken van het verslag vnl. bestaat uit het schrijven van deze inleiding.

O.a. om de bovengenoemde ideeen zoveel mogelijk te operationaliseren heb ik tijdens mijn werk geprobeerd zo veel mogelijk oog te hebben voor de fouten die ik gemaakt heb, steeds met het idee: wat kan ik hieruit leren.

Hoewel ik me realiseer dat wat ik hierboven beschreven heb (nog) niet goed gefundeerd is, heb ik er tkoch enige dingen over op willen schrijven in de hoop dat dat voor mij een hulpmiddel zal zijn om mijn gedachten hierover beter te ordenen.

Er zijn nog een aantal andere (concretere) dingen die ik me ten doel heb gesteld, die ik hier kort wil noemen.

- practisch leren werken met microprocessoren, en oog krijgen voor practische toepassingen.
- het schrijven van een groot assembler programma.
  dit is eigenlijk een middel om wat aan de vorige doelstelling te doen. Het schrijven van een groot assembler programma reken ik eigenlijk tot de dingen die je niet goed kunt overzien, maar om practische redenen (het afwezig zijn van een PASCAL vertaler voor de 8080) en om goed de mogelijkheden van een assembler en de processor te leren kennen heb ik een multiprogrammeringssysteem (M.S.) geschreven in assembler.
  Ik ben van mening dat het bij het gebruik van microprocessoren goed mogelijk is om met een hogere programmeertaal (van het type PASCAL) efficiente programma's te schrijven. De taal zou dan wel de mogelijkheid moeten hebben om als body van een procedure machine code van de betreffende machine te bevatten.
- zelf een multiprogrammeringssysteem schrijven.
  Hoewel ik het niet helemaal zelf ontwikkeld heb, (ik heb gebruik gemaakt van de software die voorhanden was van het APL terminal systeem) is het wel zo dat ik nu vrij goed de mogelijkheden en beperkingen van parrallel processen heb leren kennen door het in assembler schrijven van een M.S.

De opzet van het project.

Zoals gezegd is het project in het begin meteen opgedeeld in deelprojecten. Deze deelprojecten zijn steeds afzonderlijk gedocumenteerd.
In het eerste stuk "algemene uitgangspunten" staan de opzet en de deelprojecten vermeldt zoals ze in het begin gekozen zijn. Wellicht ten overvloedde zij gezegd dat de verschillende stukken documentatie (nu hoofdstukken in dit verslag) in de loop van het jaar gemaakt zijn. Om het gehele project te kunnen bekijken zijn ze bewust onveranderd overgenomen. Hier wil ik volstaan met een kort overzicht te geven van de gedane werkzaamheden.

In het eerste gedeelte van het afstuderen heb ik me vooral bezig gehouden met literatuurstudie, en wel op twee gebieden.

Het eerste is onderzoeken in hoeverre er al iets op het gebied van automatisering van deeltjesversnellers bekend is.([11]) Het bleek dat er al vrij veel gedaan is aan de bundelgeleiding. De ionenbundel in een versneller moet vanaf de bron gebracht worden naar het doel (in ons geval de halfgeleiderwafer). Tijdens die weg komt de bundel door komponenten als: afbuigmagneten,electrostatische lenzen, de versnellerbuis e.d. Bij het doel worden aan de deeljesbundel bepaalde eisen gesteld. Veranderd (al dan niet expres) een van de komponenten, dan zullen de meeste andere ook bijgeregeld moeten worden om de bundel weer aan de gestelde eisen te laten voldoen.
Deze laatste regeling is al voor verschillende versnellers gebouwd, (bij het cyclotron van de THE afd. Natuurkunde is dit ook gebeurd. [10]) Het blijkt dat dit vrij goed te automatiseren is. Van het automatiseren van een totale versneller heb ik in de literatuur eigenlijk niets kunnen vinden.

Een tweede gedeelte van de literatuurstudie bestond uit het vertrouwd raken met "real time programming" ([1],[2],[4],[5],[8] en [9]) en "discrete time control systems" ([3],[6] en [7]).

Na deze aanloopperiode is de te besturen machine onder de loep genomen en is er een beschrijving gemaakt van de implanter. (documentatie eerste stap) Met deze beschrijving is in overleg met HV de machine beschreven d.m.v. een toestandsdiagram (doc.2), waardoor het mogelijk werd om de besturing van de machine op enigszins gestructureerde wijze te beschrijven. (doc.3)
Toen eenmaal bekend was hoe de machine te bedienen, kon begonnen worden met het programmeerwerk. Dit is in twee gedeelte`s gebeurd. Eerst is een stuk basissoftware (operating system) gebouwd, waardoor het mogelijk werd om parallel meerdere processen op de 8080-processor te laten
draaien. Dit is gebeurd in de vorm van een programma waarin de werking van een horloge gesimuleerd wordt.

Dit programma is eerst in PASCAL geschreven (doc.5, doc.6) en daarna met de hand vertaald naar de assemblertaal. (doc.7) Het belangrijkste gedeelte van dit assemblerprogramma bestaat uit de zg. "nucleus" van het multiprogrammeringssysteem. Het moet vrij eenvoudig mogelijk zijn om deze nucleus als aparte module op te zetten, waardoor hij universeel toepasbaar wordt.
Met deze nucleus (die ong. 600 bytes lang is) kunnen verschillende

processen parrallel op de 8080 draaien. De belangrijkste primitieven zijn de semaphoren en de operaties P en V daarop. Rekenen we het beschreven timerproces ook bij de nucleus, dan ontstaat er een pakket wat vergelijkbaar is met het RMX pakket van de firma Intel, alleen is het kleiner en heeft het minder mogelijkheden.

Om een indruk te geven van de snelheid van de ontwikkelde nucleus volgen hier nog enkele tijden uitgaande van een gemiddelde instructietijd van 7 microseconde, komen we tot de volgende executietijden.(in microseconde)

P-operatie
114 of 784 bij het afbreken van het draaiende proces
V-operatie
114 of 294 bij het wekken van een geblokkeerd proces
schakelover (het eigenlijke overschakelen naar een volgend proces)
343
wait for interrupt
532
interrupthandler
616
T.b.v. de communicatie tussen de verschillende processen zijn variabelen van het type fifolist geintroduceerd met daarop de operaties: consumeer en produceer. Hiervoor gelden de volgende tijden.
produceer
452 als de buffer noch vol, noch leeg is
1106 als de buffer vol is
616 als de buffer leeg is
voor consumeer gelden dezelfde tijden

Naast het timerproces zijn ook een aantal processen beschreven t.b.v. de bediening van de terminal (nl. een keyboardhandler, printproces en outputhandler (voor de opmaak van uitgebreidere boodschappen voordat ze naar het printproces gestuurd worden)) Dit stuk software is de basissoftware waar ik bij het programmeren van de uiteindelijke besturingsprocessen van uit ben gegaan.

Uitgaande van het toestandsdiagram is in het laatste gedeelte van het afstuderen een algorithme geschreven om de implantatiemachine te besturen. (doc.8 en doc.9) Dit programma is afgezien van het uitwerken van enkele procedures klaar.
Hoewel hiermede het afstuderen is afgesloten, is het automatiseringsproject nog niet afgerond. De volgende stap in dit project zou kunnen zijn het definitief bepalen van een te bouwen proefmodel van een bestuurde implanter.
Eerst dan heeft het zin om de software verder uit te werken en op een 8080 systeem draaiend te maken.

Na dit overzicht van het afstudeerwerk, bestaat de rest van het verslag uit de rapporten die in de loop van het afstuderen gemaakt zijn.

algemene uitgangspunten

Het gehele project duidelijk projectmatig opzetten.
Om tot het eindresultaat te komen moeten er vele stappen genomen worden. Iedere stap moet bewust gebeuren en er moet nauwkeurig beschreven zijn wat er in de stap gebeurd is.
Iedere stap mag heel klein zijn, maar moet altijd een stap vooruit zijn. De nadruk moet niet komen te liggen op het eindresultaat, maar op de aanpak van de weg ernaar toe.
Dit met het doel voor ogen dat wat er af is ook goed en betrouwbaar werkt! De betrouwbaarheid van het resultaat staat voorop en niet het resultaat.
Ik moet beperkingen die de overzichtelijkheid bevorderen en die voor de rest niet werkelijk beperkend werken zo snel mogelijk invoeren.
Ik moet proberen een struktuur te scheppen waaraan ik me kan vasthouden, maar die me aan de andere kant genoeg bewegingsruimte laat. Ik moet zo laat mogelijk beginnen met details, voorlopig wil ik alleen nog maar een machine regelen die in een aantal toestanden kan verkeren. Bij iedere toestand behoort een aantal processen die in een bepaalde toestand verkeren.
Het motto is: doe maar niet te moeilijk of teveel, het is zo al ingewikkeld genoeg!
Fouten moeten bewust vermeden worden en de fouten die geheid gemaakt worden moeten eenvoudig te corrigeren zijn.
Het "proces afstuderen" moet ook gestruktureerd gebeuren.
Een manier van struktureren is de documentatie.

Documentatie in stappen.

De documentatie die bij een stap hoort staat als file op floppy disk
De filenaam is als volgt:
filenaam::= DOC.i
i geeft het rangnummer van de stap aan en een letter er achter geeft een aanvulling of herziening van de DOC.i file aan.
Iedere stap dient vooraf gegaan te worden door een lijstje van bijlagen (meestal schema's e.d.) en een beschrijving van de betekenis van de stap.

Te nemen of genomen stappen:

0 Voorstudie
  literatuur doorgeworsteld op gebied van:
        real time programmeren
        discrete time control systems
        bestaande geautomatiseerde versnellers

1 Maken blokschema met alle voor de besturing belangrijke komponenten erop, waardoor het geheel misschien iets overzichtelijker wordt.

2 beschrijving van in welke toestanden de machine mag verkeren en de toegestane overgangen.

3 Beschrijving van de logische processen in de procesregelaar.

4 Terugkoppeling van 1,2 en 3 naar HVE om daaruit de definitieve versies te maken.

5 De opzet van het programma.

6 Voorbeeld van een programma.

7 De assemblerversie van doc.6.

8 De opzet van het uiteindelijke mainproces.

9 Het uiteindelijke programma in PASCAL.

Documentatie voorstudie

literatuurlijst:

[1] Programmieren von Prozessrechnern
Volkmar Kussl BP 7520bse

[2] PEARL, Process and Experiment Automation Realtime Language
Werum/Windauer DEW 78 WERbsr

[3] Discrete-Time en Computer Control Systems
Cadzow/Martens EC 7259

[4] Prozessautomatiserung Band 1
R.Lauber BP 7640

[5] Messen, Steurern, Regeln mit Prozessrechern
Max Syrbe EC 7259

[6] Modern Control Theory
William L.Brogan 7707684

[7] Time Domain Analysis and Design of Control Systems
Richard C. Dorf EC6532bse

[8] Real-Time concepts and concurrent Pascal
C.H. Smedema
1975 IFAC/IFIP Real-Time Programming Workshop,Boston

[9] CAMAC Proceedings april 1974 Luxembourg
BP 73372bsr

[10] On the computercontrol of the Eindhoven cyclotron
G.L.C. van Heusden
Afd. Nat. THE

[11] tijdschriften: Nuclear Science
Nuclear Instruments and Methods
Review of Scientific Instruments

[12] PASCAL user manual and report
K.Jensen N.Wirth

[13] Critical comments on the programming language PASCAL
A.N.Habermann
Acta Informatica 3,47-57(1973)

[14] PASCAL on the B6700/7700

[15] Graph theory in modern engineering
Henley,Williams CHK73HENbse

[16] Structured programming
Dahl,Dijkstra,Hoare

[17] Network: A multiprocessor program
Per Brinch Hansen
IEEE transactions on software engineering
Vol SE-4 nr.3, may 1978

[18] The logic of computer programming
Manna,Waldinger
idem, Vol SE-4 nr.3, may 1978

[19] de software gebouwd bij het APL terminal systeem
vakgroep EB.

Documentatie eerste stap.

bijlagen:
- tekeningen HV DX-SK-806
  EX-7
  EX-11
- The 200kV industrial ion implantation system.
- The 350kV heavy ion accelerator

beschrijving:

De eerste stap bestaat uit het maken van een blokschema van de implanter. De hierin benoemde komponenten zijn de enige die voor mij van belang zijn.
Deze worden hieronder nog apart beschreven.

BESCHRIJVING KOMPONENTEN:

Achter de meeste namen van de komponenten staan tussen haakjes de toestanden van dat onderdeel vermeldt. Aan de hand van deze toestanden wordt de machine later beschreven. Een bepaalde kombinatie van al deze toestanden geeft een toestand van het toestandsdiagram van de implanter aan.

< : te klein
> : te groot

MOTORG (aan,uit)
Motorgenerator verzorgt energievoorziening binnen de miniseparator. Moet aan staan voor juiste werking, terwijl de andere toestand uit is.

MAGNEET (goede waarde,<,>)
Separatiemagneet, waar de magneetstroom via 3 organen bediend wordt.
MAGNET UP
MAGNET DOWN
dit zijn de aan-uit regelingen die een motor-potmeter systeem bedienen.
MAGNET FINE
dit is een analoge fijnregeling.

LENS (goed,fout)
elektrostatische tripletlens waarvan de instelling geregeld kan worden d.m.v. een hoogspanningsvoeding van 0 - 30 kV. via:
LENS PS CONTROL

BRON (rust,instabiel,standby,ok)
ionenbron die bediend wordt d.m.v. een servosysteem dat 5 variac's in de separator kan bedienen, nl.

ANODE
FILAMENT
MAGNET
OVEN
GAS
de werking van de bron kan bekeken worden via het telemetersysteem.Uitgelezen kunnen worden de
waarden van:
ANODE CURRENT
ANODE VOLTAGE
FILAMENT CURRENT
FILAMENT VOLTAGE
MAGNET CURRENT
OVEN CURRENT

FAN
luchtkoeling voor de ionenbron.
wordt bediend via het CONTROL PANEL (zie hieronder)

GAS
gastoevoer voor de ionenbron
wordt geregeld van een servo, zie onder BRON

VOORPOMP
TMPOMP (turbo moleculear pomp)
vacuumpomp voor het vacuum in de separator,
die bediend worden via het CONTROL PANEL
de waarde van het vacuum kan gemeten worden via
VACUUMMETER (aan,uit)

VACUUM (goed,fout)

VARIAC'S,SERVO (uit,aan)
het eerder vermelde servosysteem, dat 5 organen bedient zoals beschreven onder BRON

MASSMETER (goede massa,<,>)
instrument dat de massa van de deeltjes meet dat uit de miniseparator komen. (indien aanwezig)
het magneetveld wordt met een hallprobe gemeten, terwijl ook de versnelspanning nog gemeten wordt die de deeltjes bij de bron moeten doorlopen.
Uit deze gegevens berekent de massmeter de massa van eventuele aanwezige deeltjes.

KLEP (open,dicht)
pneumatische klep, die open is bij normaal bedrijf.

BPM
beam profile monitor is een mechanische scanner waarmee de vorm en intensiteit van de deeltjesbundel gemeten kan worden.

Dit kan op een oscilloscoop weergegeven worden.

HOOGSPANNING (aan,uit)
aan: (0,goede waarde,<,>,instabiel)
de uiteindelijke versnelspanning die maximaal 320kV kan bedragen, wat een totale maximale eindenergie oplevert van 30 + 320 = 350 keV.

KOELVLOEISTOF (aan,uit)
in de miniseparator moeten een aantal komponenten vloeistof gekoeld zijn (nl. vacuumpomp, magnet, source).
deze koeling moet natuurlijk in bedrijf zijn.

CONTROLPANEL
bedieningspaneel in de miniseparator waarmede (met de hand) aan of uitgeschakeld kunnen worden:
IONPS (aan,uit)
MAGNET POWER SUPPLY (aan,uit)
VACPUMP (aan,uit)
ELECTRONICS (aan,uit)
MASSMETER/SCANNER (BPM) (aan,uit)
TELEMETER (aan,uit)
H.V.SUPPLY (voorversnelspanning van 10.,20 of 30 kV)
(aan,uit)
FAN (aan,uit)
LENS P.S. (aan,uit)
KLEP

TELEMETER
communicatiesysteem dat gegevens via glasfibers naar aardpotentiaal brengt.

TUBE PS (aan,uit)

TRIPLET LENS (aan,uit)
aan: (goed,fout)
elektrostatische triplet lens die de bundel in principe op het target moet focusseren.

BEAM SWEEP (aan,uit)
aan: (ok,<,>)
elektrostatische afbuiging waardoor de spot over de wafer gescand kan worden ten tijde van de expliciete implantatie.

CURRENT INTEGRATOR (aan,uit) {bedient neutral trap}
aan: (rust,instellen,klaar)
meter voor het meten van de totale hoeveelheid geimplanteerde deeltjes.

NEUTRAL TRAP (aan,uit)
aan: (rust,<,>,ok,onder beheer van CURR INT)

elektrostatische afbuiging om aanwezige ongeladen deeltjes van de geladen te onderscheiden.

TARGET
doel waarop de deeltjes worden geschoten.

FARADAY CUP
meetpunten voor het meten van de breedte van de gescande beam en zijn positie.

VACUUM POMP 2	(aan,uit)
VACUUM BENEDEN	(goed,fout)
VACUUM METER 2	(aan,uit)
BPM 2	(aan,uit,uitgetrokken)

Documentatie.2

beschrijving: hier wordt de machine beschreven d.m.v. toestanden. De beschrijving van de machine zal vanaf dit moment alleen maar via die toestanden gebeuren.

Hoe kijk ik tegen de machine aan?

De implanter is een machine met een aantal ingangsgrootheden (nu schakelaars en knoppen) en uitgangsgrootheden (nu lampjes en meters).
Een bepaalde instelling van deze grootheden noemen we een toestand.
De implanter kan in zeer veel toestanden verkeren. Voor een juiste werking is alleen een beperkt aantal interresant. We bedrijven de machine zo dat hij zich alleen in een van de "interresante" toestanden kan bevinden.
Blijkt dat de toestand van de machine anders is dan logisch te verwachten, dan is er iets mis en moet er een speciaal programma aangeroepen worden.
Er zijn 2 soorten toestanden, nl. degenen die voor ons van belang zijn en de rest. Degenen die voor ons van belang zijn, nummeren we van 1 t/m N en de rest van de toestanden geven we de kollektieve naam >N.
Verder is het zo dat we niet iedere willekeurige toestandsovergang toelaten. Ook hier zijn er een aantal die we toelaten en de rest verklaren we niet interresant.
De verschillende toestanden en de gewenste overgangen zouden we nu nog in een mooi plaatje kunnen weergeven, we noemen dit dan het toestandsdiagram van de machine.
We kijken vanaf nu niet meer naar de implanter maar eigenlijk alleen naar het toestandsdiagram.

Ik stel me het normale bedrijf van de machine als volgt voor:
via een inputorgaan wordt een opdracht gegeven om naar toestand j te gaan (met meestal een aantal parameters toegevoegd)
De computer moet dan de machine via toegestane overgangen en gewenste toestanden naar toestand j brengen.
Staat de machine op een bepaald moment in een toestand >N dan moet de computer een voorgeschreven handeling verrichten. (bijv. machine naar een van de gewenste toestanden brengen, dit melden en op verdere opdrachten wachten.)

Toestanden van de implanter:

toestand 1: uit (machine volledig uit)
        motorgenerator uit
        magneet uit

```
        lens uit
        bron uit
        fan uit
        pompen uit
        variac's op nul
        massmeter uit
        klep dicht
        BPM uit
        hoogspanning uit
        koelvloeistof uit
        H.V. power supply uit
        electronics uit


toestand 2: rust (machine op inactief, maar wel gereed voor
opstarten)
        motorgenerator aan
        vacuumpompen aan
        vacuum o.k.
        electronics aan,gekoppeld aan motorgen. (voor boven);
                        beneden altijd aan
        koelvloeistof aan
        vacuummeter aan

"comment"

Dit is de toestand waarin de machine verkeert als hij niet
gebruikt wordt of binnenkort gebruikt gaat worden.
Voor de 2-de t/m de 6-de toestand geldt dat ook aan alle eisen
van vorige toestanden wordt voldaan.;

toestand 3: aan 1 (bron gestart)
        variac's ingeregeld
        fan aan
        ion source p.s. aan

"comment"
Dit is de eerste toestand waar de machine door moet als hij
aangezet wordt.
De bron wordt op een laag,pitje gezet, waarin hij lekker sta-
biel is, en wat een goed startpunt is voor verdere regeling.;

toestand 4: aan 2 (net plaatje op eerste BPM, miniseparator
                                                    klaar)
        klep open
        magneet aan
        magneetveld op goede waarde
        BPM aan
        lens p.s. aan
        HV PS aan (10,20,30kV)
        nette gefocusseerde bundel op BPM
```

```
"comment"
Tweede stap in het opstarten van de implanter, nl. een nette
bundel uit de miniseparator (en aan het begin van de versnel-
lerbuis);

toestand 5: in bedrijf (alle functies o.k. voor productie)
        nette gefocuseerde bundel op 2-de BPM
        eindenergie o.k.
        aantal deeltjes/sec o.k.

"comment"
Dit is de voor-productie fase, de implanter is nu direct
voor productie gereed.;

toestand 6: productie (gevraagde productie draaien)
        beam scanning o.k.
        BPM's uit

"comment"
productie fase waarin af en toe, bijv. tussen het implanteren
van twee wafers in de bundeleigenschappen gekontroleerd en
e.v.t. gekorrigeerd dienen te worden.;

toestand 7: test (t.b.v. testen van de implanter)

"comment"
toestand waarin vanaf de console de machine met de hand bediend
kan worden, voor servicen e.d.
De regelingen die de waarden van allerlei
variabelen bewaakten in de toestand voor deze zijn uit.;


Toegestane overgangen:

van i naar i + 1 ; i<6
van i naar 7 ; i<7
van 7 naar hoogste toestand die door alleen "verlagen"
        van de toestand verkregen wordt.
van >N naar geldige toestand
```

Documentatie.3

beschrijving: hierin worden de bij iedere toestand en toestandsovergang behorende verzameling van logische processen geinventariseerd.

Deze logische processen draaien onder beheer van een programma nl. de procesregelaar. De procesregelaar wordt op zijn beurt weer bestuurd door het besturingsprogramma. De procesregelaar en het besturingsprogramma vormen samen het totale programma.

toestand 1 (uit)
processen:
- toestandsbewaking
  bewaking van vaste logische waarden die bij een toestand horen, bijv.: vacuum o.k.,klep dicht. Hiermee kan zeer snel het belangrijkste deel van de status van de machine gecontroleerd worden. Zijn er 8 waarden die gecontroleerd moeten worden dan bestaat het programma (op assemblerniveau) uit maar drie instructies, nl inlezen, vergelijken en conditionele sprong.
- alarm
  procedure die de aard van de storing opspoort, evt. al ingrijpt en dit doorgeeft aan het besturingsprogramma.

toestand 2 (rust)
processen:
- toestandsbewaking
- vacuumbewaking
  evt. in toestandsbewaking op te nemen door de grenswaardenbewaking hardware uit te voeren.

toestand 3 (aan1)
processen:
- toestandsbewaking
- vacuumbewaking
- bronbewaking

toestand 4 (aan2)
processen:
- toestandsbewaking
- vacuumbewaking
- massabewaking
- focussering op BPM
- bronbewaking

toestand 5 (in bedrijf)
processen:

toestandsbewaking
vacuumbewaking
bronbewaking
massabewaking
focussering op BPM
focussering op 2-de BPM
hoogspanningsbewaking
implantatiesnelheid

toestand 6 (productie)
processen:
toestandsbewaking
vacuumbewaking
massabewaking
bronbewaking
focussering op BPM
focussering op 2-de BPM
hoogspanningsbewaking
implantatiesnelheid
beam scanner bewaking
implantatiebesturing
wafer transport

toestand 7 (test)
processen:
"console:=scheduler"

toestandsovergangen

van 1 naar 2
motorgenerator aanzetten
vacuumpompen aanzetten
electronics aanzetten
koelvloeistof aanzetten
wachten tot vacuum o.k.

van 2 naar 1
motorgenerator afzetten

vacuumpompen afzetten
electronics afzetten
koeling uitzetten

van 2 naar 3
fan aanzetten
ion power supply aan
opstartprocedure bron

van 3 naar 2
fan uit
ion p.s. uit

variacs naar 0

van 3 naar 4
klep open
magneet aan
magneet naar juiste waarde
BPM aan
H.V. p.s. aan
lens p.s. aan
procedure voor net plaatje op BPM

van 4 naar 3
klep dicht
magneet uit
BPM uit
lens p.s. uit
H.V. p.s. uit

van 4 naar 5
2-de BPM aan
neutral trap aan
tripplet p.s.
tube lens ps aan (preset value)
hoogspanning naar juiste waarde
procedure voor net plaatje op 2-de BPM
regelen naar juiste stroomsterkte

van 5 naar 4
2-de BPM uit
neutral trap uit
hoogspanning uit
tripplet uit
tube lens uit

van 5 naar 6
sweep system aan en inregelen
wafersysteem aan
beamscanning instellen
BPM`s uit

van 6 naar 5
beamscanner uit
BPM`s aan (?)

van 6 naar 7
alle regelingen uitzetten en de controle aan de console geven.

van 7 naar i
voorlopig moet hij met de hand naar een bekende toestand gebracht worden.

van >N naar i
voorlopig met de hand naar een bekende toestand.

Documentatie.5

De opzet van het programma.

beschrijving:
Hierin wordt de globale opzet van het programma beschreven. Een eerste poging om het programma op te zetten is mislukt. Deze poging staat beschreven in de vroegere doc.5 die nu doc.5A heet. Na een eerste bespreking van die eerste poging bleek al vlug dat de aanpak niet zorgvuldig genoeg was, en ook niet erg flexibel t.a.v. veranderingen.
Toen is dan ook besloten tot de huidige opzet.
Het aardige is dat ik in het allereerste begin ook een dergelijke opzet wou gebruiken, maar er toen maar van heb afgezien omdat ik vermoedde dat een dergelijke aanpak tot teveel overhead zou leiden en voor een microprocessor te hoog gegrepen was. De ervaring in de vakgroep EB opgedaan met het APL-terminal systeem vertelde ons dat het heel goed haalbaar was.
Wat is de gekozen aanpak: Het algorithme wordt in PASCAL geschreven en d.m.v. een aantal gebouwde standaardprocedures is het mogelijk om in PASCAL te praten over processen die parrallel en onafhankelijk draaien (onafhankelijk afgezien van de aangebrachte synchronisatie t.b.v. kommunikatie tussen de processen)

Het Multiprogrammeringssysteem.

Het programma bestaat uit standaardroutines (waardoor het mogelijk wordt om stukken programma`s als processen te gaan behandelen) en een aantal procedure`s waarin die processen beschreven worden. De body van het programma bestaat uit niet meer dan de start van de processen en fungeert verder als het idle-proces wat gezien de gekozen opzet altijd aanwezig dient te zijn.
De naam idle-proces is in dit geval niet helemaal terecht, want dit proces is in ons geval het debug programma. Dit heeft als voordeel dat het mogelijk blijft om als de processen draaien gewoon te blijven debuggen!
We noemen dit gehele programma het multiprogrammeringssysteem (M.S.) en de standaardprocedures heten: de nucleus (kern)
In het nu volgende gedeelte wordt de opzet van het programma toegelicht. Dit moet dan het lezen van het programma vergemakkelijken.

De opzet van het programma.

Uitgaande van een PASCAL machine wil ik het volgende,
en wel de mogelijkheid hebben meerdere processen parallel uit te voeren.
Wat zijn de problemen:
- -wat is een proces.
- -hoe maak ik het mogelijk dat meerdere processen parral-

lel op deze machine kunnen draaien.

Wat versta ik onder een proces.

1)een blok data waarin de toestand van het eigenlijke proces beschreven staat, dit noemen we de monitordata van het proces.

2)het algorithme dat het werkelijke proces beschrijft en een stack die bij dat proces hoort.

ad1) Wat moet ik van het proces weten:

- naam(=adres)
- logische toestand, de mogelijheden zijn:
  - dood [niet in rrlst of sedlist]
  - gereed om te draaien, maar wachtende op een processor [in rrlst]
  - draaiend op een processor [=runningproces]
  - geblokkeerd [door de uitvoering van een P of een wait for interrupt operatie] doordat een eenheid (bijv. printer of buffer) niet beschikbaar is (in gebruik, vol, leeg)
- is het proces geblokkeerd, de blokkeringsoorzaak.
- als het proces niet draaiende is, dan moet bekend zijn waar straks (door)gestart moet worden.
- de semaphoor die bij het proces hoort.
  (met deze semaphoor kan ik d.m.v. de P operatie een proces zichzelf laten stoppen, terwijl doorstarten met een V operatie mogelijk is)

ad2)

- het algorithme beschrijf ik in de vorm van een procedure en wel op de volgende manier:

```
"procedure" proces;
    {definities en declaraties}
    "begin" initialisatie;
            "while" true "do" werk
    "end"; {proces}
```

- bekend moet zijn waar de bij het proces horende stack staat.

Opmerkingen betreffende de nucleus van het M.S.

- de taak van de nucleus is niet veel meer dan het op aanvraag bijwerken van de monitordata.
- de operaties op de monitordata gebeuren door procedures van de nucleus. Vanuit een proces kan een procedure van de nucleus aangeroepen worden. Deze beslist steeds wat er moet gebeuren en kan een verandering aanbrengen in de toestand van de processen.
- de datablokken kunnen gelinkt worden voor het vormen van wachtlijsten, bijv. de wachtlijst van processen die gereed zijn om te draaien, maar waarvoor op dat moment geen processor beschikbaar is.
  (de ready to run list (rrlst))
- om het gebruik van gemeenschappelijke resources netjes te

regelen, wordt gebruik gemaakt van semaphoren.
Een semaphoor bestaat (naast zijn naam(=adres)) uit een niet negatieve integer en een wachtlijst van processen die voor die semaphoor (=seinpaal) staan te wachten. (de semaphoor-datalist (sedlist))

- een van de gemeenschappelijke resources is de processor zelf. Gezien de speciale aard en taak van de processen wordt hij anders behandelt.
  Zo is de ready to run list een wachtlijst van processen die net als bij een semaphoor staan te wachten tot ze toegang krijgen tot de resource (de processor)
  Kan een proces niet verder met de uitvoering van zijn taak, dan wordt dat proces gestopt en de volgende van de rrlst gehaald.
- de wachtlijsten bij de semaphoren werken op first in - first out basis (dit om deadlock`s te voorkomen) terwijl de rrlst als een stack (=last in first out) werkt.
- alle procedures van de nucleus werken onder bescherming van "disable interrupt".

Een gebruikelijke gang van zaken is, dat de nucleus gestart wordt vanuit proces A, bijv. door de P statement, dat de nucleus beslist dat proces B nu verder mag gaan en daarom de processor aan proces B geeft.
Op PASCAL niveau gaan we nu in P het proces A uit en komen er weer ergens in proces B weer in.
We zijn nu heel duidelijk aan het springen in de proggramma text. De processen worden beschreven als een procedure en wat gebeurd er: ergens in procedure A wordt procedure P aan geroepen en het resultaat is o.a. dat naar een bepaalde plaats in procedure B wordt gesprongen.
Dit is iets wat in PASCAL gelukkig verboden is.
Een oplossing is het schrijven van een paar "smerige" procedures die achter de schermen foeselen, waardoor de sprong van proces A naar B verkregen wordt.
Deze procedures kunnen niet in PASCAL geschreven worden, maar wel in de machinetaal, waar we wel over een dergelijk sprongmechanisme beschikken.
De verantwoording is dat het doel de middelen heiligt en dat na het gefoesel het resultaat toch weer netjes achtergelaten wordt, zodat je je dit in een enkel uitzonderlijk geval kunt permitteren.
Deze procedures hebben een body in machinecode die de vertaler letterlijk overneemt als de vertaalde kode van de procedure (zoals dat in ALGOL 60 toegestaan is)
Die vertaler ben ik voorlopig zelf.

Ik wil de mogelijkheid hebben om de uitvoering van een proces voor een bepaalde tijd te stoppen, of vanuit een proces een ander proces na een gegeven moment door te starten.
Hiertoe dienen de procedure`s suspend(tijd:integer) en

set klok(n:proces;t:integer).
Om deze procedure`s hun werk te laten doen maken we gebruik van een in software gesimuleerde wekker. (de variabelen zijn van het type: wekker, en het algoritme (dat het aflopen van de wekker verzorgt) is een gedeelte van de timer. Ieder proces heeft zo`n wekker)
Ook het uitschakelen van het door de bovengenoemde procedure`s gestarte proces is mogelijk door de procedure:
reset klok(n:proces). (het uitzetten van de wekker)

In doc.6 staat een voorbeeld van een programma wat volgens deze opzet werkt. Het mainproces bestaat uit het simuleren van een horloge. Het proces dat de werking van een horloge beschrijft is een proces waarin de tijd een belangrijke rol speelt, en waarvan de werking goed te overzien is. Mede daarom is het geschikt om de huidige opzet van het systeem te testen.

De eigelijke simulatie van het horloge gebeurd in het mainproces. Dit proces doet niets meer dan steeds de buffer "mainmelding" leeg halen en op grond van de inhoud aktie ondernemen.
Het proces kan in twee toestanden verkeren. Is de toestand "gestart" dan staat het proces te wachten op een boodschap van de keyboardhandler. (met daarin de opdracht het horloge aan te zetten op de aangegeven tijd)
Na het ontvangen van die boodschap gaat het mainproces naar de toestand "klok aan". Vanaf nu wordt bij ieder "timerbericht" (die iedere seconde komen) die tijd opgehoogd en evt. uitgeprint. Het nu aanstaande horloge kan via "tijd" opnieuw gelijk gezet worden en uitgelezen worden via "hoe laat".

Het keyboardproces heeft als taak de van het keyboard komende karakters te verwerken.
Na het intikken van het eerste karakter van een boodschap probeert de keyboardhandler het display te claimen. Na het intikken en verwerkt zijn van de boodschap wordt het display weer vrijgegeven. Het verwerken van de karakters houdt in het echoen van de karakters (via het printproces) naar het display. Komt er een "carriage return" van het keyboard, dan is een boodschap afgesloten en wordt de in het inputbuffer staande boodschap op sytax gecontroleerd (verwerk zin) en evt. naar het mainproces gestuurd.
Het proces "escape" zorgt ervoor dat het display niet onnodig lang geclaimd wordt, dit in verband met andere naar het display te sturen boodschappen.

Het outputproces dient ervoor om bepaalde boodschappen als nette strings naar het display te sturen.
Door de beperkte mogelijkheden van PASCAL op dit gebied is de PASCAL tekst meer een vertaling van de assemblerversie, vandaar de onduidelijke structuur. Om dit proces op een nette manier te beschrijven zullen we naar een andere taal toe moeten. (of PASCAL op dit punt uitbreiden)

Het printproces tenslotte stuurt de ASCCI-karakters op interrupt basis naar het display.

De assemblerversie

Het PASCAL programma is om het op de 8080 te laten draaien met de hand vertaalt naar de assemblertaal.
Voor een aantal procedures was dit echt nodig (omdat we uitgaan van een aantal niet in PASCAL te beschrijven zaken), maar omdat er geen PASCAL vertaler voorhanden was is het gehele programma met de hand vertaald. Hoewel het niet aan te raden is om een dergelijk groot programma met de hand uit te coderen, is het toch gebeurd, mede om de 8080 processor er goed door te leren kennen. Verder is het zo dat de assemblerversie iets sneller is dan de PASCAL versie, wat voor een dergelijk stuk basissoftware een voordeel te noemen is.
Vanuit de assemblerversie gezien heeft de PASCAL versie de plaats ingenomen van de flowchart. Mijn ervaring is dat deze manier van werken te verkiezen is voor het werken met een flowchart. Het grote voordeel is dat in de PASCAL tekst veel meer dingen zijn vast gelegd. Het vertalen vanuit de PASCAL tekst is nu bijzonder eenvoudig geworden. (en eigenlijk ook meer de taak van een compiler, maar dit terzijde). Tot slot is het zo dat de PASCAL tekst minstens zo gemakkelijk te produceren is dan een gelijkwaardige flowchart.

Bij het lezen van de assembler versie vinden we de PASCAL tekst als kommentaar terug.
De assembler tekst is door het programma: ASM80 vertaald naar objectcode. Deze code is door het programma: LOCATE omgezet in direct uitvoerbare code. Deze code staat onder de naar HORLOG op de floppy disc.
Daar de gebruikte timer- en modemkaart (momenteel) alleen in systeem 1 van de SYS8000 serie passen, kan het programma alleen daar op gedraaid worden.
Op de modemkaart kan niet direct een terminal aangesloten worden, daar de modemkaart zelf in feite een terminal is. Om toch deze modemkaart te kunnen gebruiken is al bij het APL terminal systeem een kastje gemaakt dat tussen de kaart en de terminal geschakeld kan worden.

```
"program" horloge;
{label declaration part}
{constant declaration part}
"const" maxpnr=6; {maximum aantal processen}
        maxspl=20; {maximum aantal semaph.}
        maxadres=16383; {16k geheugen}
        stackl=64; {stacklengte in bytes}
        max=20; {max+1=lengte van de fifobuffers}

        {procesnamen}
        idle=0;
        mainproces=1;
        timer=2;
        keyboardhandler=3;
        outputhandler=4;
        escape=5;
        PRINT=6;

        {boodschappen in mainmelding}
        timerbericht=1;
        tijd=2;
        hoe laat=3;

        {boodschappen in output}
        {de vaste}
        ping=0;
        vraag tijd=1;    {tik de tijd in s.v.p.}
        syntaxfout=2;    {syntaxfout in kommando}
        {de variabele}
        tijd=128;        {tijd (uur,min,sec:....}

{type definition part}
"type"  proces=0..maxpnr ; {procesnr}
        ss=0..maxspl; {0=scheduled; 1..maxspl=rangnr`s semaph.}
        byte=0..255;
        woord=0..FFFFH; {16 bit woord}
        adres=0..maxadres;
        procesdata="record"next:@procesdata;
                           name:proces;
                           processem:ss;
                           bloorz:ss;
                           stpoint:adres
                    "end";
        semphdata="record"count:integer;
                          inspointer:@procesdata;
                          rempointer:@procesdata
                  "end";
        wekker="record"tijd:integer;
```

```
                        naam:proces;
                        aktief:boolean
                "end";
        stack="array"[0..stackl]"of" byte;
        interrpt=(recready,trready,timerint); {alle interupt-
                                                oorzaken}
        fifolist="record"buffer:"array"[0..max]"of"byte;
                        inspointer:0..max;
                        rempointer:0..max;
                        AVB,ALB,mutex:ss
        {aantal volle/lege bufferplaatsen en mutual exclution}
                 "end";

{variable declaration part}
"var"monitordata:"record"runningproces:proces;
                        procesdescr:"array"[proces]"of"
                                                @procesdata;
                        semphdescr:"array"[ss]"of"
                                                @semphdata;
                        rrlst:@procesdata
                 "end";
     timerdata:"record"nr:"array"[proces]"of" wekker;
                        timermutex:ss
                "end";
     mainmelding,output,printer:fifolist; {kommunikatiebuffers}
        inputb,buffer:ss; {t.b.v. de terminalhandler}
     een,twee,drie:byte; {data welke door de buffer
                                beschermt wordt}

{procedure declararion part}
"procedure" create system;
        { de initialisatie routine}
        "const"np=....;{beginadres van proces data records e.d.}
        "var"i:integer;
             newpointer:adres;{variabele die de procedure
                new gebruikt voor het reserveren van geheugen}
        "procedure"initprocessorstatus(naam:proces);
        "var"i:proces;j:integer;
             PC:"array"[proces]"of"adres;
        {ik veronderstel dat dit array al geinitialiseerd is,
         het komt in ROM te staan.}
        "procedure"push(stpoint,data:woord);
                "begin"
                * zet programcounter op de bedoelde stack *
                "end";
        "begin"
         "with"monitordata.procesdescr[naam]@"do"
                "begin"
                push(stpoint,PC[i]);
                "for"j:=1"to"4"do"
                push(stpoint,0)
                {(H,L),(D,E),(B,C) en (PSW) op de stack}
```

```
        "end";
"end";
"procedure"createproces(naam:proces);
        "var"stp:@stack;
             prdscr:@procesdata;
        "begin" new(prdscr);
                monitordata.procesdescr[naam]:=prdscr;
                "with"monitordata.procesdescr[naam]@"do"
                "begin"name:=naam;
                       processem:=naam; {naam en proces
                                 hebben zelfde volgnr.}
                       "if"naam"neq"idle"then"
                       "begin"
                { idle heeft al een stack, en draait al}
                       bloorz:=0;
                       new(stp);
                       stpoint:=stp
{is geen pascal!
 is verder correct(?) als stp naar de bodem van de
 stack wijst!}
                       "end"
                "end";
                initprocessorstatus(naam)
        "end";
"procedure"createsemph(sem:ss);
        "var"sdscr:@semphdata;
        "begin" new(sdscr);
                monitordata.semphdescr[sem]:=sdscr;
                "with"monitordata.semphdescr[sem]@"do"
                "begin"count:=0;
                       inspoint:=nil;
                       rempoint:=nil
                "end"
        "end";
"procedure" init("var"j:fifolist);
        "begin""with"j"do"
                "begin"inspointer:=0;rempointer:=0;
                {AVB is in createsemph al geinitiali-
                 seerd}
                monitordata.semphdescr[ALB]@.count:=
                                          max+1;
                "end"
        "end";
"begin" {van create system}
        newpointer:=np;
        "for"i:=0"to"maxpnr"do" create proces(i);
        "for"i:=0"to"maxspl"do" create semph(i);
        {volgnummers andere semaphoren}
        i:=maxpnr;
        timerdata.timermutex:=succ(i);
        mainmelding.AVB:=succ(i);
        mainmelding.ALB:=succ(i);
```

```
                mainmelding.mutex:=succ(i);
                output.AVB:=succ(i);
                output.ALB:=succ(i);
                output.mutex:=succ(i);
                printer.AVB:=succ(i);
                printer.ALB:=succ(i);
                printer.mutex:=succ(i);
                inputb:=succ(i);
                buffer:=succ(i);
                "with"monitordata"do"
                "begin"runningproces:=0;
                       rrlst:=procesdescr[0]
                       procesdescr[0].next:=procesdescr[0]
                "end";
                "with"timerdata"do"
                        "for"i:=0"to"maxpnr"do"nr[i].aktief:=
                                                      false;
                init(mainmelding);init(output);init(printer)
        "end"; {van create system}


"procedure" redprocessorstatus; "begin""end";
"procedure" herstelprocessorstatus; "begin""end";
"procedure" interruptenable; "begin""end";
"procedure" interruptdisable; "begin""end";
"procedure" enable hardware int; "begin""end";
"procedure" disable hardware int; "begin""end";

"function" sema of(p:proces): ss;
        "begin"
        sema of:=monitordata.procesdescr[p]@.processem
        "end";


"procedure"schakelover(p:proces); {hier wordt naar een andere stack
                            overgeschakeld}
        "begin" redprocessorstatus;{onzinnige data, wel plaats
                                    reserveren op de stack}
                runningproces:=p;
                herstelprocessorstatus;
                interruptenable
        "end";{schakelover}

"procedure"scheduler;
        { haalt het volgende proces van de rrlst}
        "var"p:proces;
        "begin""with"monitordata"do"
                "begin"p:=rrlst@.name;
                       rrlst:=rrlst@.next
                "end";schakelover(p)
        "end";
```

```
"procedure" readytorun(p:proces);
                {zet p op de readytorun lijst}
        "begin""with"monitordata"do"
                "begin"
                procesdescr[p]@.next:=rrlst;
                rrlst:=procesdescr[p]
                "end"
        "end";


"procedure"P(semph:ss);
        "begin" interruptdisable;
                "with"monitordata"do"
                "with"semphdescr[semph]@"do"
                  "if"count>0
                    "then""begin"count:=count-1;interruptenable
                          "end"
                    "else"
                      "begin"
                        "with"procesdescr[runningproces]@"do"
                        "begin"next:=nil;bloorz:=semph"end";
                        "if"rempoint=nil
                          "then"rempoint:=procesdescr[running
                                                      proces]
                          "else"inspoint@.next:=
                                procesdescr[runningproces];
                        inspoint:=procesdescr[runningproces];
                        scheduler
                      "end"
        "end"; {P}


"procedure"V(semph:ss);
        "var"hulp:@procesdata;
        "begin"interruptdisable;
          "with"monitordata"do"
            "with"semphdescr[semph]@"do"
              "if"rempoint=nil
                "then"count:=count+1
                "else"
                  "begin"{haal van wachtlijst semaphoren en
                          plaats in readytorun lijst}
                    hulp:=rempoint;
                    "with"hulp@"do"
                      "begin"
                        rempoint:=next;
                        next:=rrlst;
                        bloorz:=0
                      "end"
                    rrlst:=hulp
                  "end";
                interrupt enable
```

```
        "end"; {V}

"procedure" enable(naam:interrpt);
        "begin"
        * enable de genoemde interrupt *
        "end";

"procedure" disable(naam:interrpt);
        "begin"
        * disable en reset de genoemde interrupt *
        "end";

"procedure" wait for interrupt(naam:interrpt);
        "begin" interrupt disable;
                {evt. procesdexcr[runningproces]@.next:=nil
                 en interruptoorzaak opslaan in procesdes-
                 criptor i.v.m. analogie met de P-operatie}
                enable(naam);
                scheduler
        "end";

"procedure" interrupthandler(naam:interrpt);
        "function"interruptproces(naam:interrpt):proces;
                "begin"
                "case"naam"of"
                        recready:interruptproces:=
                                        keyboard;
                        trready:interruptproces:=PRINT;
                        timerint:interruptproces:=timer
                        "end"
                "end";
        "begin"interrupt disable; {gebeurt al automatisch}
               redprocessorstatus;
               readytorun(runningproces);
               runningproces:=interruptproces(naam);
               disable(naam);
               herstelprocessorstatus
        "end";

"procedure" produceer("var"j:fifolist;item:byte);
        "begin""with"j"do"
                "begin"P(ALB);
                       buffer[inspointer]:=item;
                       inspointer:=(inspointer+1)"mod"(max+1);
                       V(AVB)
                "end"
        "end";

"procedure"consumeer(j:fifolist;"var"item:byte);
        "begin""with"j"do"
                "begin"P(AVB);
                       item:=buffer[rempointer];
```

```
                        rempointer:=(rempointer+1)"mod"(max+1);
                        V(ALB)
                "end"
        "end";


"procedure"set klok(n:proces;T:integer);
        "begin""with"timerdata"do"
                "begin"P(timermutex);
                        "with"nr[n]"do"
                        "begin"aktief:=true;
                                tijd:=T;
                                naam:=n
                        "end";
                        V(timermutex)
                "end"
        "end"; {set klok}

"procedure"suspend(T:integer);
        "begin" set klok(runningproces,T);
                P(sema of(runningproces))
        "end"; {suspend}

"procedure"reset klok(n:proces);
        "begin" "with"timerdata"do"
                "begin" P(timermutex);
                        nr[n].aktief:=false;
                        V(timermutex)
                "end"
        "end"; {reset klok}


"procedure"timer; {het timerproces}
        "var"i:integer;
        "procedure"inittimer IC;
                "begin"
                * initialiseer het timer IC op 1 keer
                  per seconde interrupperen *
                "end";
        "begin" inittimer IC;V(timermutex);
                enable hardware int;
                "while"true"do"
                  "begin" wait for interrupt(timerint);
                    "with"timerdata"do"
                      "begin"P(timermutex);
                        "for"i:=0"to"maxpnr"do"
                        "with"nr[i]"do
                          "if"aktief"then"
                            "begin"tijd:=tijd-1;
                              "if"tijd=0"then"
                                "begin"aktief:=false;
                                        V(sema of(naam))
```

```
                                "end"
                            "end";
                        V(timermutex)
                        P(mainmelding.mutex);
                        produceer(mainmelding,timerbericht)
                        V(mainmelding.mutex);
                        "end"
                "end"
        "end"; {timer}

"procedure"PRINT; {het eigenlijke printproces}
        "var"kar:byte;
        "procedure"initialiseer het display;
                "begin"
                * initialiseer het display *
                "end";
        "procedure"output(c:byte);
                "begin"
                * stuurt een byte naar het display *
                "end";
        "begin" initialiseer het display;
                wait for interrupt(trready);
                "while"true"do"
                        "begin" consumeer(printer,kar);
                                output(kar);
                                wait for interrupt(trready)
                        "end"
        "end"; {PRINT}

"procedure"outputhandler;
        "const"aantal boodsch0=3;
               aantal boodsch1=1;
               langste=100;
        "type" text="record"l:integer;
                            textbuffer:"array"[1..langste]"of"char
                    "end";
               text1="record"a:text;
                             n:integer;
                             adr:"array"[0..n-1]"of"woord
                             {n-1 kan niet in PASCAL}
                     "end";
        "var" textdescr0:"array"[0..aantal boodsch0-1]"of"@text;
              textdescr1:"array"[0..aantal boodsch1-1]"of"@text1;
        {ik veronderstel dat de inhoud van de records korrekt
         is, staat in een code segment en komt uiteindelijk in
         PROM te staan}
              lengte,i:integer;
              item:byte;
              tempbuffer:"array"[0..4]"of"byte;
        "procedure"convert(waarde:woord);
                {converteert waarde naar decimale representatie
                 1 digit per byte}
```

```
                "var"i:integer;
                "begin"
                "for"i:=4"downto"0"do"
                        "if"waarde=0"then"tempbuffer[i]:=0
                            "else""begin"
                                tempbuffer[i]:=waarde"mod"10;
                                waarde:=waarde"div"10
                                  "end"
                "end";
        "procedure"displ;
                {zet tempbuffer als ASCII karakters in printer
                 buffer}
                "var"i,j:integer;
                "begin" produceer(printer,ord(SP)); {20H}
                        i:=0;
                        "while"(i<4"and"tempbuffer[i]=0)"do"
                                                  i:=i+1;
                        "for"j:=i"to"4"do""begin"
                          tempbuffer[j]:=ASCII(tempbuffer[j]);
                          produceer(printer,tempbuffer[j])
                                          "end"
                "end";
        "begin" initialiseer textdescr;
                "while"true"do"
                "begin"consumeer(output,item);
                        P(printer.mutex);
                        "if"MSB(item)=0"then"
                           "with"textdescr0[item]@"do"
                             "for"i:=1"to"1"do"
                                produceer(printer,
                                        ord(textbuff[i]))
                           "else""begin"
                                item:=item-128;
                                "with"textdescr1[item]@"do"
                                  "for"i:=1"to"a.l"do"
                                    produceer(printer,
                                        ord(a.textbuff[i]));
                                  "for"i:=1"to"n"do"
                                    "begin"convert(adr[i]);
                                           displ;
                                    "end"
                                "end";
                        V(printer.mutex)
                "end"
        "end"; {van de outputhandler}
"procedure" keyboardhandler; {het terminal proces}
        "var"teken:char;
             i:integer;
             toestand:(rust,intikken);
             inputbuffer:"array"[0..30]"of"char;
        "procedure"termrust;
                "begin" toestand:=rust;i:=0;
```

```
                reset klok(escape);
                V(printer.mutex)
        "end";
"procedure"escape; {hiermee wordt de terminalhandler in de
                        rusttoestand gebracht}
        "begin""while"true"do"
                "begin"P(sema of(runningproces));
                       P(inputb);
                       "if"toestand=intikken"then"
                        "begin"
                        produceer(printer,ord(`$`));
                        produceer(printer,ord(`CR`));
                        produceer(printer,ord(`LF`));
                        produceer(printer,ord(`BEL`));
                        termrust
                        "end";
                        V(inputb)
                "end"
        "end"; {escape}
"procedure"verwerk(T:char);
        "begin""case"T"of"
                del:"if"i=0"then"
                        V(sema of(escape))
                        "else"
                        "begin"i:=i-1;
                        produceer(printer,ord(
                                    backspace));
                        "if"i=0"then"
                                V(sema of(escape))
                        "end";
                esc: V(sema of(escape));
                "else":"if"i=31"then"
                        V(sema of(escape))
                        "else""begin"
                                inputbuffer[i]:=T;
                                i:=i+1;
                                produceer(printer,ord(T))
                              "end"
                "end"
        "end"; {verwerk teken}
"procedure"verwerk zin;
        "begin"
        "if"i=1"then"
          "if"inputbuffer[0]=`T`"then"
                "begin" P(buffer);
                        een:=23;twee:=50;drie:=30;
                        V(buffer)
                        P(mainmelding.mutex);
                        produceer(mainmelding,tijd);
                        V(mainmelding.mutex)
                "end"
                "else""if"inputbuffer[0]=`H`"then"
```

```
                    "begin"
                    P(mainmelding.mutex);
                    produceer(mainmelding,hoe laat);
                    V(mainmelding.mutex)
                    "end"
                        "else"
                        produceer(output,syntaxfout)
              "else"produceer(output,syntaxfout)
            "end";
      "procedure" init bedieningsterminal;
            "begin" write mode instr;
                    write command instr;
                    disable(recready);
                    disable(trready)
            "end";
      "procedure" input("var"T:char);
            "begin"
            * haalt het karakter binnen en laat het in T
              achter. *
            "end";
      "begin" init bedieningsterminal;
            V(inputb);
            termrust;
            "while"true"do"
                    "begin"wait for interrupt(recready);
                      input(teken);
                      P(inputb);
                      "if"toestand=rust"then"
                            "begin"toestand:=intikken;
                                  P(printer.mutex)
                            "end";
                      "if"toestand=intikken"then"
                            "if"teken=`CR`"then"
                              "begin"
                              produceer(printer,ord(`LF`));
                              produceer(printer,ord(`CR`));
                              verwerk zin;
                              termrust
                              "end""else"
                              "begin"
                              set klok(escape,10);
                    {10 sec tijd tussen het intikken}
                              verwerk(teken)
                              "end";
                      V(inputb)
                    "end"
      "end";

"procedure" mainproces; {het mainproces}
      "var"time:"record"sec,min,hr:byte"end";
           toestand:(gestart,klok aan);
           item:byte;
```

```
"procedure"klok ophogen;
        "begin"
        sec:=(sec+1)"mod"60;
        "if"sec=0"then""begin"
                min:=(min+1)"mod"60;
                "if"min=0"then"hr:=(hr+1)"mod"24
                "end"
        "end";
"begin" toestand:=gestart; V(buffer);V(mainmelding.mutex);
  produceer(output,vraag tijd);
  "while"true"do"
    "begin" consumeer(mainmelding,item);
      "case"toestand"of"
        gestart:"case"item"of"
                  timerbericht:;
                  hoe laat:;
                  tijd:"with"time"do"
                         "begin" P(buffer);
                                 sec:=een;
                                 min:=twee;
                                 hr:=drie;
                                 V(buffer);
                                 toestand:=klok aan
                          "end"
                "end";
        klok aan:"case"item"of"
                   timerbericht:"begin" klok ophogen;
                        "if"sec=0"then"
                        produceer(output,de tijd)
                        "else""if"sec"mod"10=0"then"
                          produceer(output,ping)
                                "end";
                  tijd:"begin"toestand:=gestart;
                              P(mainmelding.mutex);
                              produceer(mainmelding,
                                        tijd);
                              V(mainmelding.mutex)
                        "end";
                  hoe laat:produceer(output,de tijd)
                "end"
          "end"
    "end"
"end";
```

```
"begin" {hiervoor moet de initialisatie van de hardware al
        gebeurd zijn}
        disable hardware int;
        create system; {nu draait het idle proces}
        readytorun(timer); {in timer: enable hardware int}
```

```
        readytorun(escape);
        readytorun(PRINT);
        readytorun(keyboardhandler);
        readytorun(outputhandler);
        readytorun(mainproces);
        scheduler; {hier wordt het eerste echte proces gestart}
        *debugprogramma*
"end". {van proces 0}
```

```
$ MACROFILE
        STKLN   STACKL
;*************************************
; "program" horloge;
;*************************************
; handvertaalde versie van PASCAL programma (DOC.6)
;*************************************
;
;*************************************
; {constant definition part}
;*************************************
MPROC   EQU     6       ; {aantal processen, afgezien van idle}
MSPL    EQU     MPROC+12        ; {aantal semaphoren, afgezien
                                ;  van de processem. van idle}
STACKL  EQU     64      ; stackl in byte
MAX     EQU     20      ; max + 1 = lengte van de fifobuffers
; {procesnamen}
IDLE    EQU     0
MAINPR  EQU     1
TIMER   EQU     2
KEYHDL  EQU     3
OUTPH   EQU     4
ESCAPH  EQU     5
PRINT   EQU     6
; {boodschappen in mainmelding}
TIMBR   EQU     1
TYD     EQU     2
HOELA   EQU     3
; {boodschappen in output}
; {de vaste}
PING    EQU     0
VRTYD   EQU     1
SYNTXF  EQU     2
; {de variabele}
DETYD   EQU     128
;*************************************
; {type definition part}
;*************************************
PNXT    EQU     0       ; proces data record
PNAME   EQU     2
PSEMA   EQU     3
BLOORZ  EQU     4
PSA     EQU     5
PDSCR   EQU     7       ; lengte van proces data record
COUNT   EQU     0       ; semaphoor data
SINSP   EQU     1
SREMP   EQU     3
SEMA    EQU     5       ; lengte van semaphoor data
WINT    EQU     1       ; wekker record
WNAME   EQU     3
WAKT    EQU     0
```

```
WEKKER  EQU     4       ; lengte van wekker
; interrupt oorzaken
RECRDY  EQU     0
TRRDY   EQU     1
TIMINT  EQU     2
FINS    EQU     MAX+1   ; fifolist
FREM    EQU     MAX+2
FAVB    EQU     MAX+3
FALB    EQU     MAX+4
FMUTEX  EQU     MAX+5
FIFOLI  EQU     MAX+6   ; lengte van fifolist
;
DSEG
;************************************
; variable declaration part
;************************************
RPROC:  DS      2       ; monitordata.runningproces
P000:   DS      2*(MPROC+1)     ; procesdescriptor
S00:    DS      2*(MSPL+1)      ; semaphoordescriptor
RRLST:  DS      2       ; ready to run list
TIMNR:  DS      WEKKER*(MPROC+1); timerdata
TMUTEX: DS      1
MAINM:  DS      FIFOLI  ; mainmelding
OUTPU:  DS      FIFOLI  ; output
PRINTE: DS      FIFOLI  ; printer
INPBUF: DS      1
BUFFER: DS      1
EEN:    DS      1
TWEE:   DS      1
DRIE:   DS      1
NEWPOI: DS      2       ; newpointer t.b.v. new
;************************************
CSEG
; {procedure declaration part}
;************************************
;************************************
; MACRO`S
;************************************
ARRI    MACRO   BASE    ; (D,E)<----(A)*2+BASE
        ADD     A
        ADI     LOW BASE
        MOV     E,A
        MVI     A,HIGH P000
        ACI     0
        MOV     D,A
        ENDM
FPTR    MACRO           ;(H,L)<---M(H,L)
        MOV     A,M
        INX     H
        MOV     H,M
        MOV     L,A
        ENDM
```

```
STPTR   MACRO           ;M(H,L)<---(D,E)
        MOV     M,E
        INX     H
        MOV     M,D
        ENDM

ARRI@   MACRO   BASE    ;(B,C)<---(BASE+2*(H,L))
        DAD     H
        LXI     B,BASE
        DAD     B
        MOV     C,M
        INX     H
        MOV     B,M
        ENDM

INDEX   MACRO   INDX    ;(H,L)<---(B,C)+INDX
        LXI     H,INDX
        DAD     B
        ENDM

PNIL    MACRO           ;M(H,L)? NIL
        MOV     A,M
        INX     H
        ORA     M
        DCX     H
        ENDM

MVPHD   MACRO           ;M(H,L):=M(D,E)
        LDAX    D
        MOV     M,A
        INX     D
        INX     H
        LDAX    D
        MOV     M,A
        ENDM

LEAVEH  MACRO   BYTE
        LXI     H,BYTE
        MOV     L,M
        MVI     H,0
        ENDM

;****************************************
; new, de PASCAL standaard procedure
; in H,L wordt de pointer achtergelaten naar de eerste
; gereserveerde byte.
; in E het aantal byte
;****************************************
NEW:    DI
        LHLD    NEWPOI
        PUSH    H
```

```
        MVI     D,0
        DAD     D
        SHLD    NEWPOI
        EI
        POP     H
        INX     H
        RET

;*************************************
; de routine DIV uit het assembler manual
;*************************************
DIV:    MOV     A,D     ;NEGATE THE DIVISOR
        CMA
        MOV     D,A
        MOV     A,E
        CMA
        MOV     E,A
        INX     D       ;FOR TWO'S COMPLEMENT
        LXI     H,0     ;INITIAL VALUE FOR REMAINDER
        MVI     A,17    ;INITIALIZE LOOP COUNTER
DV0:    PUSH    H       ;SAVE REMAINDER
        DAD     D       ;SUBSTRACT DIVISOR (ADD NEGATIVE)
        JNC     DV1     ;UNDERFLOW,RESTORE HL
        XTHL
DV1:    POP     H
        PUSH    PSW     ;SAVE LOOP COUNTER(A)
        MOV     A,C     ;4 REGISTER LEFT SHIFT
        RAL             ;WITH CARRY
        MOV     C,A     ;CY->C->B->L->H
        MOV     A,B
        RAL
        MOV     B,A
        MOV     A,L
        RAL
        MOV     L,A
        MOV     A,H
        RAL
        MOV     H,A
        POP     PSW     ;RESTORE LOOP COUNTER(A)
        DCR     A       ;DECREMENT IT
        JNZ     DV0     ;KEEP LOOPING
;
;POST-DIVIDE CLEAN UP
;SHIFT REMAINDER RIGHT AND RETURN IN DE
;
        ORA     A
        MOV     A,H
        RAR
        MOV     D,A
        MOV     A,L
        RAR
        MOV     E,A
```

```
        RET

;*************************************
; routine MOVE blok geheugen; oorsprong in H,L
                            ; doel in D,E
                            ; lengte van het blok in A
;*************************************
MOVE:   MOV     B,M
        XCHG
        MOV     M,B
        INX     H
        XCHG
        INX     H
        DCR     A
        CPI     0
        JNZ     MOVE
        RET

;*************************************
; enable 8214
;*************************************
E8214:  MVI     C,8     ; alle 8 interrupts zijn enabled
        CALL    0F821H  ; CALL SINT (thesis routine)
        RET
;*************************************
; disable 8214
;*************************************
D8214:  MVI     C,0
        CALL    0F821H  ; CALL SINT (thesis routine)
        RET

;*************************************
; routine FILL0;        zet nullen in het geheugen
;                       in A het aantal byte en
;                       in H,L het eerste adres
;*************************************
FILL0:  MVI     D,0
LO1:    MOV     M,D
        INX     H
        DCR     A
        JNZ     LO1
        RET

;*************************************
; procedure initprocessorstatus; naam in accumulator
;*************************************
INPS:   JMP     VERD1
PRCOUN: DW      IDLE    ; PC[idle], wordt niet gebruikt,
                        ; idle draait al.
        DW      MAIN    ; PC[mainm]
        DW      TIMPR   ; PC[timer]
        DW      KEYPR   ; PC[keyhdl]
```

```
        DW      OUTPPR  ; PC[outph]
        DW      ESCPRO  ; PC[escaph]
        DW      PRINPR  ; PC[print]
VERD1:  MOV     L,A
        MVI     H,0     ; inhoud A in H,L
        ARRI@   P000    ; monitordata[naam]@ in B,C
        PUSH    B       ; adres van het record bewaren
        LXI     H,0
        DAD     SP
        XCHG            ; sp van routine in D,E
        INDEX   PSA     ; stpoint in H,L
        MOV     B,M     ; inhoud van stpoint in H,L
        INX     H
        MOV     H,M
        MOV     L,B
        SPHL            ; nieuwe stpointer
        MOV     L,A
        MVI     H,0
        ARRI@   PRCOUN  ; PC[i] in B,C
        PUSH    B       ; push(stpoint,PC[i])
        LXI     H,0
        PUSH    H       ; H,L
        PUSH    H       ; D,E
        PUSH    H       ; B,C
        PUSH    H       ; PSW
        DAD     SP
        XCHG            ; processtack in D,E, routinestack in
        SPHL            ; H,L. routinestack hersteld
        POP     B       ; record adres
        INDEX   PSA
        STPTR           ; stackpointer opgeborgen
        RET

;*******************************************
; procedure create proces ; naam in accumulator
;*******************************************
CRPRO:  MVI     E,PDSCR
        CALL    NEW     ; new(pdscr)
        PUSH    PSW
        ARRI    P000    ; monitordata.procesdscr[naam]
        XCHG            ; {adres nieuw record in D,E}
        STPTR           ; :=pdscr
        XCHG            ; with monitordata.procedescr[naam]@ do
        POP     PSW
        LXI     D,PNAME
        DAD     D
        MOV     M,A     ; begin name:=naam;
        INX     H
        MOV     M,A     ; processem:=naam;
        INX     H
        MVI     E,0
        MOV     M,E     ; bloorz:=0
```

```
        CPI     0       ; if naam = 0 then
        JZ      END01
        PUSH    H
        MVI     E,STACKL
        CALL    NEW     ; new(stp)
        LXI     D,STACKL - 1
        DAD     D       ; H,L wijst naar stackbottom
        POP     D
        XCHG
        INX     H
        MOV     M,E
        INX     H
        MOV     M,D     ; stpoint:=stp
        CALL    INPS    ; end; initprocessorstatus(naam)
END01:  RET             ; end; {create proces}

;****************************************
; procedure create semaphoor    ;sem in accumulator
;****************************************
CRSEMA: MVI     E,SEMA
        CALL    NEW     ; new(sdscr)
        ARRI    S00     ; in D,E het adres van de pointer
        XCHG
        STPTR           ; :=sdscr
        XCHG            ; with monitordata.semdescr[sem]@ do
        MVI     A,SEMA
        CALL    FILL0   ; count:=0; inspoint:=nil;
                                    ; rempoint:=nil
        RET

;************************************
; procedure init("var"j:fifolist)       ;j in B,C
;************************************
INIT:   INDEX   FINS    ; with j do
        MVI     A,2
        CALL    FILL0   ; insp:=0; remp:=0;
        INX     H       ; H,L wijst naar j.ALB
        MOV     L,M
        MVI     H,0     ; j.ALB in H,L
        ARRI@   S00     ; monitordata.semphdescr[ALB]@
        INDEX   COUNT   ; .count
        MVI     M,MAX+1 ; :=max+1;
        RET

;************************************
; routine init semaphoor nummers
;************************************
INSENR: INDEX   FAVB    ; AVB
        INR     A
        MOV     M,A     ; :=succ(i)
        INX     H
        INR     A
```

```
        MOV     M,A     ; .ALB:=succ(i)
        INX     H
        INR     A
        MOV     M,A     ; .mutex:=succ(i)
        RET

;*************************************
; procedure create system
;*************************************
CREATE: LXI     H,RESTRT
        LXI     D,16
        MVI     A,24
        CALL    MOVE    ; laadt restart vanaf adres 16 {RST 2}
        MVI     A,(JMP  RESTRT)
        STA     40      ; rst5 ---> rst5
        LXI     H,40
        SHLD    41
        STA     48      ; rst6 ---> rst6
        LXI     H,48
        SHLD    49
        LXI     H,MEMORY
        SHLD    NEWPOI  ; newpointer:= np
        MVI     A,0     ; for i:=0 to maxpr do
FOR1:   PUSH    PSW
        CALL    CRPRO   ; create proces(i)
        POP     PSW
        INR     A
        CPI     MPROC+1
        JNZ     FOR1
        MVI     A,0
FOR2:   PUSH    PSW
        CALL    CRSEMA  ; create semph(i)
        POP     PSW
        INR     A
        CPI     MSPL+1
        JNZ     FOR2
        MVI     A,MPROC+1
        STA     TMUTEX  ; timerdata.timermutex:=succ(i)
        LXI     B,MAINM ; mainmelding,output,printer
        CALL    INSENR  ; .AVB,.ALB,.mutex
        LXI     B,OUTPU
        CALL    INSENR
        LXI     B,PRINTE
        CALL    INSENR
        INR     A
        LXI     H,INPBUF
        MOV     M,A     ;inputb:=succ(i)
        LXI     H,BUFFER
        INR     A
        MOV     M,A     ; buffer:=succ(i)
        LXI     H,0
        SHLD    RPROC   ; runningproces:=0
```

```
        LHLD    P000    ; procesdescr[0]
        SHLD    RRLST   ; ---> rrlst
        MOV     M,L     ; procesdescr[0].next:=procesdescr[0]
        MOV     A,H
        INX     H
        MOV     M,A
        LXI     H,TIMNR+WAKT    ; adres van timerdata.nr[0]
                        ; .aktief in H,L
        LXI     D,4     ; afstand tussen nr[i] en nr[i+1]
        MVI     A,0     ; for i:=0 to maxpnr do
FOR3:   MVI     M,0     ; nr[i].aktief:=false {=0}
        DAD     D
        INR     A
        CPI     MPROC+1
        JNZ     FOR3
        LXI     B,MAINM
        CALL    INIT    ; init(mainmelding)
        LXI     B,OUTPU
        CALL    INIT    ; init(output)
        LXI     B,PRINTE
        CALL    INIT    ; init(printer)
        RET             ; end; { van create system }

;************************************
; procedure redprocessorstatus
;************************************
REDPS:  XTHL            ; het terugkeeradres in H,L, H,L op de
                        ; stack
        PUSH    D       ; het doorstartadres staat al op de stack
        PUSH    B
        PUSH    PSW     ; nu zijn alle registers gered
        XCHG            ; het terugkeeradres in D,E
        LXI     H,0
        DAD     SP      ; stackpointer in H,L
        PUSH    D       ; het terugkeeradres weer op de stack
        XCHG            ; stackpointer in D,E
        LHLD    RPROC
        ARRI@   P000
        INDEX   PSA
        STPTR           ; stackpointer in record
        RET

;************************************
; procedure herstelprocessorstatus
;************************************
HERPS:  LHLD    RPROC   ; haal stackpointer op
        ARRI@   P000
        INDEX   PSA
        FPTR            ; sp in H,L
        POP     D       ; het terugkeeradres in D,E
        SPHL            ; verander stackpointer
        POP     PSW     ; herstel PSW
```

```
        POP     B       ; herstel B,C
        XCHG            ; terugkeeradres in H,L
        POP     D       ; herstel D,E
        XTHL            ; herstel H,L, terugkeeradres op de stack
        RET
; het doorstart adres staat boven op de stack

;*************************************
; procedure schakelover
;*************************************
SOVER:  CALL    REDPS   ; redprocessorstatus
        LHLD    P       ; runningproces:=p
        SHLD    RPROC
        CALL    HERPS   ; herstelprocessorstatus
        EI              ; enable interrupt
        RET             ; op deze return wordt overgeschakeld!

;*************************************
; procedure scheduler
;*************************************
DSEG
P:      DS      2
CSEG
SCHED:  LHLD    RRLST   ; p:=rrlst@.name
        LXI     D,PNAME
        DAD     D
        MOV     L,M
        MVI     H,0
        SHLD    P
        LHLD    RRLST   ; rrlst
        FPTR
        SHLD    RRLST   ;       :=rrlst@.next
        CALL    SOVER   ; schakelover(p)
        RET             ; end; {scheduler}

;*************************************
; procedure readytorun(p:proces)        ; p in H,L
;*************************************
RRUN:   ARRI@   P000    ; adres van procesdescr[p]@.next in B,C
        LHLD    RRLST
        XCHG            ; rrlst in D,E
        MOV     H,B     ; (B,C)-->(H,L)
        MOV     L,C
        STPTR
        DCX     H
        XCHG            ; adres van procesdescr[p]@.next in D,E
        LXI     H,RRLST
        STPTR           ; rrlst:=procesdescr[p]
        RET             ; { van readytorun }

;*************************************
; procedure P ; semph in H,L
```

```
;****************************************
PSEM:   DI              ; interruptdisable
        PUSH    H       ; with monitordata do
        ARRI@   S00     ;   with semphdescr[semph]@ do
        INDEX   COUNT   ; if count>0
        XRA     A
        CMP     M
        JZ      PSEM1
        DCR     M       ;count:=count-1
        POP     H
        EI              ; interrupt enable
        RET             ; end; {P}
PSEM1:  LHLD    RPROC
        DAD     H
        LXI     D,P000
        DAD     D
        FPTR
        LXI     D,0     ; next:=nil
        STPTR
        DCX     H
        POP     D       ; bloorz:=semph
        MOV     A,E
        XCHG
        LXI     H,BLOORZ
        DAD     D
        MOV     M,A
        INDEX   SREMP   ; if rempoint=nil
        PNIL
        JNZ     PSEM2
        STPTR           ; rempoint:=procesdescr[runningproces]
        JMP     PSEM3
PSEM2:  INDEX   SINSP   ; inspoint@.next:=procesdescr[running-
                        ;                                proces]
        FPTR
        STPTR
PSEM3:  INDEX   SINSP   ; inspoint:=procesdescr[runningproces]
        STPTR
        CALL    SCHED   ; scheduler
        RET             ; end; {P}

;****************************************
; procedure V   ; semph in H,L
;****************************************
VSEM:   DI              ; interrupt disable
        ARRI@   S00     ; with monitordata do
        INDEX   SREMP   ;   with semdescr[semph]@ do
        PNIL            ; if rempoint=nil
        JNZ     VSEM1
        INDEX   COUNT
        INR     M       ; count:=count+1
        JMP     VSEM2
VSEM1:  FPTR
```

```
        PUSH    H           ; with hulp@ do
        XCHG                ; hulp@ in D,E
        INDEX   SREMP       ; semdescr[semph] in H,L
        MVPHD               ; rempoint:=next
        DCX     D
        LHLD    RRLST       ; next:=rrlst
        XCHG
        STPTR
        DCX     H           ; bloorz:=0
        LXI     B,BLOORZ
        DAD     B
        MVI     M,0         ; end
        POP     H           ; rrlst:=hulp
        SHLD    RRLST
VSEM2:  EI                  ; interrupt enable
        RET                 ; end; {V}

;************************************
; procedure enable(naam:interrpt)       ;naam in A
;************************************
;CODE VOOR MODEM AANSTUURKAART
;     EN TIMERKAART SYS8000
TMR     EQU     28H         ; BASE ADRES TIMERKAART
BEDTER  EQU     30H         ; BASE ADRES MODEMKAART

ENABLE: CPI     RECRDY
        JZ      ENA1
        CPI     TRRDY
        JZ      ENA2
        MVI     A,70H       ; is kennelijk timerinterrupt
        OUT     TMR+2       ; enable(timerint)
        RET
ENA1:   MVI     A,5H
        OUT     BEDTER+4            ; enable(recready)
        RET
ENA2:   MVI     A,5H
        OUT     BEDTER+5            ; enable(trready)
        RET

;************************************
; procedure disable     ;naam in acc
;************************************
DISABL: CPI     RECRDY
        JZ      DIS1
        CPI     TRRDY
        JZ      DIS2
; eerst IN dan OUT!
        IN      TMR+2       ; reset de interrupt op de bus
        MVI     A,60
        OUT     TMR+2       ; disable(timerint)
        CALL    E8214
        RET
```

```
DIS1:   XRA     A
        OUT     BEDTER+4        ; disable(recready)
        CALL    E8214           ; reset de interrupt
        RET
DIS2:   XRA     A
        OUT     BEDTER+5        ; disable(trready)
        CALL    E8214           ; reset de interrupt
        RET

;*************************************
; procedure wait for interrupt  ; naam in acc
;*************************************
WFI:    DI
        CALL    ENABLE
        CALL    SCHED
        RET

;*************************************
; procedure interrupthandler    ;naam in A
;*************************************
;*************************************
; restart code, deze code wordt dynamisch geladen vanaf adr 16
;*************************************
RESTRT: CALL    REDPS   ; RST 2, dus timer ---> int 5
        MVI     A,TIMINT
        JMP     INTHAN
        CALL    REDPS   ; RST 3, ---> int 4
        MVI     A,TRRDY
        JMP     INTHAN
        CALL    REDPS   ; RST 4, ---> int 3
        MVI     A,RECRDY
        JMP     INTHAN

;*************************************
; de rest van de interrupthandler
;*************************************
INTHAN: LHLD    RPROC
        CALL    RRUN    ; readytorun(runningproces)
        MVI     H,0     ; runningproces:=
        CPI     TIMINT          ; interruptproces(naam)
        JZ      INTHA1
        CPI     RECRDY
        JZ      INTHA2
        MVI     L,PRINT
        JMP     INTHA3
INTHA1: MVI     L,TIMER
        JMP     INTHA3
INTHA2: MVI     L,KEYHDL
INTHA3: SHLD    RPROC
        CALL    DISABL
        CALL    HERPS
        EI
```

```
        RET     ; op deze return wordt overgeschakeld!

;*********************************
; procedure produceer   ; j in B,C en item in A
;*********************************
PRODU:  PUSH    PSW
        PUSH    B
        INDEX   FALB    ; adres(j.ALB)
        MOV     L,M
        MVI     H,0     ; j.ALB in H,L
        CALL    PSEM    ; P(ALB)
        POP     B
        INDEX   FINS
        MOV     E,M
        MVI     D,0     ; j.inspointer in D,E
        INDEX   0       ; (B,C) --> (H,L)
        DAD     D       ; adres(buffer[inspointer])
        POP     PSW
        MOV     M,A
        INDEX   FINS
        MOV     A,M
        CPI     MAX
        JZ      PRMOD
        INR     A
        MOV     M,A     ; inspointer:=*+1
        JMP     PROD1
PRMOD:  XRA     A
        MOV     M,A     ; inspointer:=0
PROD1:  INDEX   FAVB
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(AVB)
        RET

;*****************************************
; procedure consumeer   ; j in B,C en item in A
;*****************************************
CONSU:  PUSH    B
        INDEX   FAVB    ; adres(j.AVB)
        MOV     L,M
        MVI     H,0
        CALL    PSEM    ; P(AVB)
        POP     B
        INDEX   FREM
        MOV     E,M
        MVI     D,0
        INDEX   0       ; (B,C) --> (H,L)
        DAD     D       ; adres(buffer[rempointer])
        MOV     A,M
        PUSH    PSW     ; red item in A
        INDEX   FREM
        MOV     A,M
```

```
        CPI     MAX
        JZ      COMOD
        INR     A
        MOV     M,A     ; inspointer:=*+1
        JMP     CONSU1
COMOD:  XRA     A
        MOV     M,A     ; rempointer:=0
CONSU1: INDEX   FALB
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(ALB)
        POP     PSW
        RET

;************************************
; procedure set klok    ; n in A en T in D,E
;************************************
SETKLK: PUSH    D
        PUSH    PSW
        LEAVEH  TMUTEX
        CALL    PSEM    ; P(timermutex)
        LXI     B,TIMNR
        POP     PSW
        MVI     H,0
        MOV     L,A
        DAD     H
        DAD     H       ; (H,L):=4*A  {WEKKER=4}
        DAD     B       ; adres(nr[n])
        POP     D
        MVI     M,1     ; aktief:=true
        INX     H
        STPTR           ; tijd:=T
        INX     H
        MOV     M,A     ; naam=n {naam is overbodig in dit record}
        LEAVEH  TMUTEX
        CALL    VSEM    ; V(timermutex)
        RET

;************************************
; procedure suspend     ; T in D,E
;************************************
SUSP:   LHLD    RPROC
        MOV     A,L
        CALL    SETKLK
        LHLD    RPROC   ; proces en processem. hebben hetzelfde
                        ; volgnummer!
        CALL    PSEM
        RET

;************************************
; procedure reset klok  ; n in H,L
;************************************
```

```
RESKLK: PUSH    H
        LEAVEH  TMUTEX
        CALL    PSEM     ; P(timermutex)
        LXI     B,TIMNR
        POP     H
        DAD     H
        DAD     H
        DAD     H        ; (H,L):=4*(H,L)
        DAD     B        ; adres(timerdata.nr[n])
        MVI     M,0      ; nr[n].aktief:=false
        LEAVEH  TMUTEX
        CALL    VSEM     ; V(timermutex)
        RET

;*************************************************************
;*************************************************************

;  de parrallelle processen

;*************************************************************
;*************************************************************

; procedure timer;       { het timerproces}
;****************************************
; procedure inittimer    ; code voor gebruikte timerkaart
                         ; iedere sec een interrupt.
;****************************************
INTIMR: MVI     A,0E7H   ; init timer
        OUT     TMR+3
        MVI     A,0C0H
        OUT     TMR+2    ; reset functions
        IN      TMR+2    ; reset int. lijn
        MVI     A,63H    ; load count register (99+1 load)
        OUT     TMR
        MVI     A,30H    ; load clockrate select (10 msec)
        OUT     TMR+1
        MVI     A,60H
        OUT     TMR+2    ; start timer repetent, int disabled
        RET

;**********************************
; de timer
;**********************************
TIMPR:  CALL    INTIMR   ; inittimer
        LEAVEH  TMUTEX
        CALL    VSEM     ; V(timermutex)
        CALL    E8214    ; enable hardware interrupts
        MVI     B,0      ; j:=0  {WFI laat B,C onaangetast}
TITRUE: MVI     A,TIMINT         ; while true do begin
        CALL    WFI      ; wait for interrupt(timerint)
        INR     B        ; j:=j+1
        MOV     A,B
        CPI     1
```

```
        JNZ     TITRUE  ; j<1
        MVI     B,0     ; j mod 1 = 0
        PUSH    B
        LEAVEH  TMUTEX
        CALL    PSEM    ; P(timermutex)
        LXI     H,TIMNR
        MVI     A,0     ; i:=0
FOR4:   PUSH    PSW
        MOV     A,M
        CPI     0
        JZ      TIM2    ; if aktief then
        INX     H
        MOV     E,M
        INX     H
        MOV     D,M     ; tijd in D,E
        DCX     D       ; D,E := D,E - 1
        MOV     A,D
        CPI     0
        JNZ     TIM3
        MOV     A,E
        CPI     0
        JNZ     TIM3    ; if tijd=0 then
        DCX     H
        DCX     H
        MVI     M,0     ; aktief:=false
        POP     PSW
        PUSH    PSW
        PUSH    H
        MOV     L,A
        MVI     H,0     ; i in H,L
        CALL    VSEM    ; V(sema of(naam)); {i=naam}
        POP     H
TIM2:   POP     PSW
        LXI     B,WEKKER
        DAD     B       ; volgende nr[i].aktief in H,L
        INR     A
        CPI     MPROC+1 ; to maxpnr do
        JNZ     FOR4
        LEAVEH  TMUTEX
        CALL    VSEM    ; V(timermutex)
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    PSEM    ; P(mainmelding.mutex)
        LXI     B,MAINM
        MVI     A,TIMBR
        CALL    PRODU   ; produceer(mainmelding,timerbericht)
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
```

```
        CALL    VSEM    ; V(mainmelding.mutex)
        POP     B
TIM1:   JMP     TITRUE  ; einde hoofdlus
TIM3:   MOV     M,D
        DCX     H
        MOV     M,E     ; tijd weer opgeborgen in record
        DCX     H       ; record adres weer in H,L
        JMP     TIM2

; EINDE TIMER PROCES
;*************************************************************

;****************************************
; procedure PRINT       ; het eigenlijke printproces
                        ; het display wordt in de keyboardhand-
                        ; ler geinitialiseerd.
;****************************************
PRINPR: MVI     A,TRRDY
        CALL    WFI     ; wait for interrupt(trready)
PRTRUE: LXI     B,PRINTE        ;while true do
        CALL    CONSU   ; consumeer(printer,kar)
        OUT     BEDTER  ; output(kar)
        MVI     A,TRRDY
        CALL    WFI     ; wait for interrupt
        JMP     PRTRUE

; EINDE PRINTPROCES
;****************************************

;****************************************
; procedure outputhandler;
;****************************************
ABSCH0  EQU     3       ; const aantal boodsch0=3
ABSCH1  EQU     1       ;       aantal boodsch1=1
DSEG
;********************************
TEXTD0: DS      2*ABSCH0        ; var textdescr0
TEXTD1: DS      2*ABSCH1        ;     textdescr1
CSEG
;********************************
;vaste boodschappen voor de outputhandler
IT0:    DB      1,07H   ; ping
IT1:    DB      23,'tik de tijd in s.v.p.',0AH,0DH
IT2:    DB      26,'syntax fout in kommando!',0AH,0DH
; variabele boodschappen voor de outputhandler
IT128:  DB      19,'tijd (uur,min,sec):',3
        DW      HR,MIN,SEC
;********************************
DSEG
TEMBUF: DS      5       ; tempbuffer: array[0..4] of byte
CSEG
;********************************
```

```
; procedure convert(waarde:woord);
; waarde in B,C
;*********************************
CONV:   MVI     A,5     ; for i:=4 downto 0 do
CONV4:  PUSH    PSW
        MVI     A,0
        CMP     B
        JZ      CONV1
CONV2:  LXI     D,10    ; else
        CALL    DIV     ; bereken B,C div 10 en B,C mod 10
        LXI     H,TEMBUF
        POP     PSW
        PUSH    D
        DCR     A
        MOV     E,A
        MVI     D,0
        DAD     D       ; adres tempbuffer[i] in H,L
        POP     D
        MOV     M,E     ; tempbuffer[i]:=waarde mod 10
CONV3:  CPI     0
        JNZ     CONV4
        RET             ; end;
CONV1:  CMP     C
        JNZ     CONV2
        LXI     H,TEMBUF        ;if waarde = 0
        POP     PSW
        DCR     A
        MOV     E,A
        MVI     D,0
        DAD     D
        MVI     M,0     ; tempbuffer[i]:=0
        JMP     CONV3

;*********************************
; procedure displ;
;*********************************
DISPL:  MVI     A,20H
        LXI     B,PRINTE
        CALL    PRODU   ; produceer(printer,ord(SP))
        LXI     H,TEMBUF
        MVI     A,0
DISPL2: CPI     4       ; while (i<4
        JZ      DISPL1
        PUSH    PSW
        XRA     A       ; and
        CMP     M       ; tempbuffer[i]=0) do
        JNZ     DISPL3
        INX     H       ; H,L wijst naar volgende bufferplaats
        POP     PSW
        INR     A       ; i:=i+1
        JMP     DISPL2
DISPL3: POP     PSW
```

```
DISPL1: PUSH    PSW     ; in H,L adres uit te voeren karakter
        MOV     A,M
        ORI     30H     ; -->ASCII karakter
        PUSH    H
        LXI     B,PRINTE
        CALL    PRODU   ; produceer(printer,tempbuffer[i])
        POP     H
        INX     H
        POP     PSW
        INR     A
        CPI     5
        JNZ     DISPL1
        RET

;****************************************
; de body van de outputhandler;
;****************************************
OUTPPR: LXI     H,IT0   ; initialiseer textdescr
        SHLD    TEXTD0
        LXI     H,IT1
        SHLD    TEXTD0+2
        LXI     H,IT2
        SHLD    TEXTD0+4
        LXI     H,IT128
        SHLD    TEXTD1
OUTTRU: LXI     B,OUTPU ; while true do
        CALL    CONSU   ; consumeer(output,item)
        PUSH    PSW
        LXI     B,PRINTE
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    PSEM    ; P(printer.mutex)
        POP     PSW     ; item weer in acc.
        RLC             ; MSB in carry
        JC      OUT1    ; if MSB(item)=0 then
        RRC             ; herstel item
        MOV     L,A
        MVI     H,0
        ARRI@   TEXTD0  ; with textdescr[item]@ do
        MOV     H,B
        MOV     L,C
        MOV     A,M     ; 1 in de acc.
        INX     H       ; H,L wijst naar eerste karakter
OUT2:   PUSH    PSW     ; for i:=1 to 1 do
        PUSH    H
        MOV     A,M
        LXI     B,PRINTE
        CALL    PRODU   ; produceer(printer,ord(textbuff[i]))
        POP     H
        POP     PSW
        INX     H       ; H,L wijst naar volgende karakter
```

```
        DCR     A
        CPI     0
        JNZ     OUT2
        JMP     OUT4
OUT1:   RRC
        ANI     127         ; item:=item-128
        MOV     L,A         ; zelfde als hierboven met textd0
        MVI     H,0
        ARRI@   TEXTD1
        MOV     H,B
        MOV     L,C
        MOV     A,M
        INX     H
OUT3:   PUSH    PSW
        PUSH    H
        MOV     A,M
        LXI     B,PRINTE
        CALL    PRODU
        POP     H
        POP     PSW
        INX     H
        DCR     A
        CPI     0
        JNZ     OUT3
        MOV     A,M         ; n in A
        INX     H           ; adres van adr[0] in H,L
        MOV     C,L
        MOV     B,H
OUT5:   PUSH    PSW         ; de for statement
        PUSH    B
        MOV     H,B
        MOV     L,C         ; adres in H,L
        FPTR
        MOV     C,M
        INX     H
        MOV     B,M         ; 16 bit waarde in B,C
        CALL    CONV        ; convert(adr[i])
        CALL    DISPL       ; displ
        POP     B
        POP     PSW
        INX     B
        INX     B           ; in B,C het volgende adres
        DCR     A
        CPI     0
        JNZ     OUT5
        MVI     A,0DH       ; ord(CR)
        LXI     B,PRINTE
        CALL    PRODU       ; produceer(printer,ord(CR))
        MVI     A,0AH
        LXI     B,PRINTE
        CALL    PRODU       ; produceer(printer,ord(LF))
OUT4:   LXI     B,PRINTE
```

```
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(printer.mutex)
        JMP     OUTTRU  ; end;

;EINDE OUTPUTHANDLER
;*************************************************

;***************************************
; procedure keyboardhandler;
;***************************************
DSEG
INBUFF: DS      31      ; inputbuffer: array[0..30] of char
KEYI:   DS      1       ; i:integer
TOESTK: DS      1       ; toestand: (rust,intikken) {0 en 1}
CSEG
;***************************************
; procedure termrust;
;***************************************
TERMR:  LXI     H,0
        SHLD    KEYI    ; toestand:=rust; i:=0;
        MVI     H,0
        MVI     L,ESCAPH        ; parameter in H,L
        CALL    RESKLK  ; reset klok(escape)
        LXI     B,PRINTE
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(printer.mutex)
        RET

;***************************************
; procedure escape;     het afbreekproces
;***************************************
ESCPRO: LHLD    RPROC   ; proces=processemaphoor!
        CALL    PSEM    ; P(sema of(runningproces)
        LEAVEH  INPBUF
        CALL    PSEM    ; P(inputb)
        LXI     H,TOESTK
        XRA     A
        CMP     M       ; if toestand=intikken
        JZ      ESC1
        LXI     B,PRINTE        ; then
        MVI     A,24H   ; =ord($)
        CALL    PRODU   ; produceer(printer,ord($))
        LXI     B,PRINTE
        MVI     A,0DH   ; =ord(CR)
        CALL    PRODU
        LXI     B,PRINTE
        MVI     A,0AH   ; =ord(LF)
        CALL    PRODU
```

```
        LXI     B,PRINTE
        MVI     A,07H    ; =ord(BEL)
        CALL    PRODU
        CALL    TERMR
ESC1:   LEAVEH  INPBUF
        CALL    VSEM     ; V(inputb)
        JMP     ESCPRO

; EINDE ESCAPE PROCES
;*****************************************

;*****************************************
; procedure verwerk teken;      teken in acc.
;*****************************************
VERWRK: CPI     7FH      ; case del (=chr(7FH))
        JNZ     VERW1
        LXI     H,KEYI
        MOV     A,M
        CPI     0        ; if i=0 then
        JNZ     VERW2
VERW3:  MVI     L,ESCAPH
        MVI     H,0
        CALL    VSEM     ; V(sema of(escape))
        RET
VERW2:  DCR     A        ; i:=i-1
        PUSH    PSW
        MOV     M,A
        MVI     A,08H    ; =ord(backspace)
        LXI     B,PRINTE
        CALL    PRODU    ; produceer(printer,ord(backspace))
        MVI     A,20H    ; =ord(blank)
        LXI     B,PRINTE
        CALL    PRODU    ; produceer(printer,ord(blank))
        MVI     A,08H
        LXI     B,PRINTE
        CALL    PRODU    ; produceer(printer,ord(backspace))
        POP     PSW
        CPI     0
        JZ      VERW3
        RET
VERW1:  CPI     1BH      ; case (ESC=chr(1BH))
        JNZ     VERW4
        MVI     L,ESCAPH
        MVI     H,0
        CALL    VSEM     ; V(sema of(escape);
        RET
VERW4:  LXI     H,KEYI   ; else
        MOV     B,A
        MOV     A,M
        CPI     31       ; if i=31 then
        JNZ     VERW5
        MVI     L,ESCAPH
```

```
        MVI     H,0
        CALL    VSEM    ; V(sema of(escape))
        RET
VERW5:  MOV     E,A     ; else
        MVI     D,0
        LXI     H,INBUFF
        DAD     D
        MOV     M,B     ; inputbuffer[i]:=T
        LXI     H,KEYI
        INR     A       ; i:=i+1
        MOV     M,A
        MOV     A,B
        LXI     B,PRINTE
        CALL    PRODU   ; produceer(printer,ord(T))
        RET

;************************************
; procedure verwerk zin;
;************************************
VERWZN: LXI     H,KEYI
        MOV     A,M
        CPI     1
        JNZ     VERWZ1
        LDA     INBUFF
        CPI     54H     ; ord(T)
        JNZ     VERWZ2  ; if inputbuffer[0]=T then
        LEAVEH  BUFFER
        CALL    PSEM    ; P(buffer)
        MVI     A,23
        STA     EEN
        MVI     A,50
        STA     TWEE
        MVI     A,30
        STA     DRIE
        LEAVEH  BUFFER
        CALL    VSEM    ; V(buffer)
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    PSEM    ; P(mainmelding.mutex)
        LXI     B,MAINM
        MVI     A,TYD
        CALL    PRODU   ; produceer(mainmelding,tijd)
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(mainmelding.mutex)
        RET
VERWZ2: CPI     48H
```

```
        JNZ     VERWZ1  ; if inputbuffer[0]=H then
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    PSEM    ; P(mainmelding.mutex)
        LXI     B,MAINM
        MVI     A,HOELA
        CALL    PRODU   ; produceer(mainmelding,hoe laat)
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(mainmelding.mutex)
        RET
VERWZ1: LXI     B,OUTPU
        MVI     A,2H    ; =syntaxfout
        CALL    PRODU   ; produceer(output,syntaxfout)
        RET

;*****************************************
; procedure init bedienings terminal;
; code afhankelijk van modemkaart en de terminal
;*****************************************
IBEDT:  DI
        MVI     A,89H
        OUT     BEDTER+7        ; controlword in 8255
        MVI     A,12H
        OUT     BEDTER+1        ; hierna is de primaire 8251 klaar
                                ; voor een command instruction
        MVI     A,50H
        OUT     BEDTER+1        ; internal reset
        MVI     A,0CEH  ; 2 stopbits, parity enable, asynchr.
        OUT     BEDTER+1        ; mode instruction
        MVI     A,37H
        OUT     BEDTER+1        ; command instruction
        XRA     A
        OUT     BEDTER+4
        OUT     BEDTER+5        ; schakelaars op nul
        EI
        RET

;*****************************************
; de body van de keyboardhandler
;*****************************************
KEYPR:  CALL    IBEDT   ; init bedienings terminal
        LEAVEH  INPBUF
        CALL    VSEM    ; V(inputb)
        CALL    TERMR   ; termrust
KEYTR:  MVI     A,RECRDY        ; while true do
        CALL    WFI     ; wait for interrupt(recready)
        IN      BEDTER  ; input(teken)
```

```
        PUSH    PSW
        LEAVEH  INPBUF
        CALL    PSEM    ; P(inputb)
        LDA     TOESTK
        CPI     0       ; if toestand=rust then
        JNZ     KEY01
        INR     A       ; toestand:=intikken
        STA     TOESTK
        LXI     B,PRINTE
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    PSEM    ; P(printer.mutex)
KEY01:  POP     PSW
        CPI     0DH     ; if teken=CR then
        JNZ     KEY02
        LXI     B,PRINTE
        CALL    PRODU   ; produceer(printer,ord(CR))
        LXI     B,PRINTE
        MVI     A,0AH
        CALL    PRODU   ; produceer(printer,ord(LF))
        CALL    VERWZN  ; verwerk zin
        CALL    TERMR   ; termrust
        JMP     KEY03
KEY02:  PUSH    PSW
        MVI     A,ESCAPH        ; else
        LXI     D,10
        CALL    SETKLK  ; set klok(escape,10)
        POP     PSW
        CALL    VERWRK  ; verwerk(teken)
KEY03:  LEAVEH  INPBUF
        CALL    VSEM    ; V(inputb)
        JMP     KEYTR

; EINDE KEYBOARDHANDLER
;*****************************************


;*****************************************
; procedure mainproces;
;*****************************************
DSEG
TOESTM: DS      1       ; toestand:(gestart,klok aan) {0,1}
SEC:    DS      2       ; sec
MIN:    DS      2       ; min
HR:     DS      2       ; hr
CSEG
;***********************************
; procedure klok ophogen;
;***********************************
KLOPH:  LDA     SEC
        INR     A
```

```
        CPI     60
        JNZ     KL01
        XRA     A
KL01:   STA     SEC     ; sec:=(sec+1) mod 60
        CPI     0
        JNZ     KL02    ; if sec=0 then
        LDA     MIN
        INR     A
        CPI     60
        JNZ     KL03
        XRA     A
KL03:   STA     MIN     ; min:=(min+1) mod 60
        CPI     0
        JNZ     KL02    ; if min=0 then
        LDA     HR
        INR     A
        CPI     24
        JNZ     KL04
        XRA     A       ; hr:=(hr+1) mod 24
KL04:   STA     HR
KL02:   RET

;****************************************
; de body
;****************************************
MAIN:   MVI     A,0
        STA     TOESTM  ; toestand:=gestart
        LEAVEH  BUFFER
        CALL    VSEM    ; V(buffer)
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(mainmelding.mutex)
        LXI     H,SEC   ; initialisatie (sec,min,hr)
        MVI     A,6
        CALL    FILL0
        LXI     B,OUTPU
        MVI     A,VRTYD
        CALL    PRODU   ; produceer(output,vraag)
MTRUE:  LXI     B,MAINM ; while true do
        CALL    CONSU   ; consumeer(mainmelding,item)
        MOV     B,A
        LDA     TOESTM  ; case toestand of
        CPI     0
        JNZ     MAIN1   ; gestart
        MOV     A,B     ; case item of
        CPI     TIMBR   ; timerbericht:;
        JZ      MTRUE
        CPI     HOELA
        JZ      MTRUE   ; hoe laat:;
        LEAVEH  BUFFER
```

```
        CALL    PSEM    ; P(buffer)
        LDA     EEN
        STA     HR
        LDA     TWEE
        STA     MIN
        LDA     DRIE
        STA     SEC
        LEAVEH  BUFFER
        CALL    VSEM    ; V(buffer)
        MVI     A,1
        STA     TOESTM  ; toestand:=klok aan
        JMP     MTRUE
MAIN1:  MOV     A,B     ; klok aan: case item of
        CPI     TIMBR
        JNZ     MAIN2   ; timerbericht:
        CALL    KLOPH   ; klok ophogen
        LDA     SEC
        CPI     0       ; if sec=0 then
        JNZ     MAIN3
        MVI     A,DETYD
        LXI     B,OUTPU
        CALL    PRODU   ; produceer(output,de tijd)
        JMP     MTRUE
MAIN3:  CPI     10      ; if sec mod 10 =0 then
        JC      MAIN4
        SUI     10
        JMP     MAIN3
MAIN4:  CPI     0
        MVI     A,PING
        LXI     B,OUTPU
        CZ      PRODU   ; produceer(output,ping)
        JMP     MTRUE
MAIN2:  CPI     TYD
        JNZ     MAIN5   ; tijd:
        MVI     A,0
        STA     TOESTM  ; toestand:=gestart
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    PSEM    ; P(mainmelding.mutex)
        LXI     B,MAINM
        MVI     A,TYD
        CALL    PRODU   ; produceer(mainmelding,tijd)
        LXI     B,MAINM
        INDEX   FMUTEX
        MOV     L,M
        MVI     H,0
        CALL    VSEM    ; V(mainmelding.mutex)
        JMP     MTRUE
MAIN5:  CPI     HOELA   ; hoe laat:
        JNZ     MAIN6
```

```
        LXI     B,OUTPU
        MVI     A,DETYD
        CALL    PRODU   ;produceer(output,de tijd)
        JMP     MTRUE
MAIN6:  DI
        JMP     MAIN6

;EINDE MAINPROCES

;**********************************
;**********************************
BEGIN:  CALL    D8214
        LXI     SP,STACK
        CALL    CREATE
        MVI     H,0
        MVI     L,TIMER
        CALL    RRUN    ; ready to run(timer)
        MVI     H,0
        MVI     L,PRINT
        CALL    RRUN    ; ready to run(PRINT)
        MVI     H,0
        MVI     L,ESCAPH
        CALL    RRUN    ; ready to run(escape)
        MVI     H,0
        MVI     L,KEYHDL
        CALL    RRUN    ; ready to run(keyboardhandler)
        MVI     H,0
        MVI     L,OUTPH
        CALL    RRUN    ; ready to run(outputhandler)
        MVI     H,0
        MVI     L,MAINPR
        CALL    RRUN    ; ready to run(mainproces)
        CALL    SCHED   ; scheduler
        RST     0       ; het debug programma als idle proces
        END     BEGIN   ; {van proces 0}
```

Doc.8

Opzet van het mainproces:

We gaan er van uit dat de besturen machine beschreven kan worden door een toestandsdiagram. In ons geval is dat in doc.2 beschreven.
We willen het mainproces laten werken met dit toestandsdiagram, en wel zo dat het mainproces zoveel mogelijk onafhankelijk wordt van de eigelijke opbouw van de te besturen machine.
Het mainproces wordt hierdoor vrij universeel van opzet, terwijl we ons niet bezig hoeven te houden met speciale eigenschappen van de te besturen machine.

Het toestandsdiagram.

Voor de beschrijving van het toestandsdiagram in de gekozen programmeertaal moeten we de structuur er van exact beschrijven. In principe is het toestandsdiagram te zien als een "graph", met knooppunten en takken. De knopen vertegenwoordigen de toestanden, en de takken de mogelijke overgangen (pas op, de overgangstoestanden zoals beschreven in doc.3 zijn gewone toestanden in het toestandsdiagram!)
De machine bevindt zich op een bepaalt moment in een van zijn toestanden. Willen we de machine bedrijven, dan willen we de machine in een bepaalde toestand brengen.(We noemen dit het doel, een variabele van het type TOESTAND)
Hebben we de besturing laten weten wat we willen, dus een nieuw doel opgegeven, dan is de taak van het mainproces de machine er naar toe te geleiden.
Is het nieuwe doel bekent,dan moet de weg door het toestandsdiagram bepaalt worden om tot dat doel te geraken. Nu is het zo dat bij de te besturen machine (en bij veel andere ook) er van alle mogelijke paden door het diagram er 2 zijn die voor ons interessant zijn, en dat zijn 2 paden tussen de toestand: "machine uit" en "machine in vol bedrijf".
Er zijn vele manieren om een dergelijke machine aan of uit te zetten, maar voor de automatische besturing kiezen we er een uit om de machine aan te zetten, en een om de machine weer uit te zetten. De gekozen paden zijn die, waar we van verwachten (op grond van opgedane ervaringen met de oude handbediening) dat ze onder computerbesturing het beste voldoen.
Om met een automatische besturing tot het beste resultaat te komen, zal het zeker zo zijn dat de kale machine (zonder de besturing) op een aantal punten gewijzigd zal moeten worden om een toestandsdiagram te verkrijgen dat optimaal beantwoord aan het gestelde doel: een geautomatiseerde besturing.
In mijn afstuderen ga ik, om het project niet te groot te laten worden, uit van de bestaande machine, (die oorspronkelijk ontworpen is zonder te kijken naar een eventuele automatisering) daar het doel van het onderzoek alleen is de mogelijkheid van een

dergelijke besturing aan te tonen.
In een aantal gevallen veronderstel ik dat er hardware voorzieningen zijn getroffen, wordt bijvoorbeeld een component aangezet, dan veronderstel ik dat er een "handshake" aanwezig is, ter vervanging van het vroegere controle lampje (byv. bij de voeding van de ionenbron: IONPS is de aan/uit schakelaar, en de terugmelding is IONPWR (ionpower). (zie procesT23))

Door ons te beperken tot 2 paden in het diagram, is de taak van mainproces als volgt te beschrijven:
Komt er een opdracht om naar een nieuw doel te gaan, breng dan (indien nodig) de machine naar een van de bedoelde paden, en wandel vervolgens over dit pad naar het aangegeven doel.
De toestand waar de machine zich op een gegeven moment in bevindt wordt opgeslagen in de variabele: gemeten toestand:TOESTAND.
In "var" start:TOESTAND staat de eerste toestand van het te bewandelen pad. Als de gemeten toestand gelijk wordt aan het doel, hoeft er niet mer over een van de beide paden gelopen te worden.
Of er over het pad gelopen moet worden (in welke richting) of niet staat in de variabele: updown:(up,down,0)
Wordt er op een bepaald moment een nieuwe opdracht ingevoerd, dan moet een eventueel op dat moment draaiende taak tot stoppen gedwongen worden voordat de taak die bij het nieuwe doel hoort gestart mag worden.
Het stoppen van deze taak wordt wordt in gang gezet door de variabele continue:boolean.
De tot stoppen gedwongen taak meldt zich expliciet af door de melding: "handshake" in de buffer van het mainproces achter te laten.
Hierna kan de nieuwe taak gestart worden.

De 2 paden door het toestandsdiagram staan beschreven in de variabele diagram: "array"[TOESTAND]"of"
"record"volgende,vorige"end".
In volgende staat de volgende toestand als we het pad omhoog gaan (d.i. naar de toestand "machine in vol bedrijf") terwijl in vorige de volgende toestand staat als we naar de toestand "machine uit" gaan.
Een belangrijke, maar vrij eenvoudige taak is het regelmatig controleren van de status van de machine. Een groot deel van die status is (meestal al in hardware) te vertalen in logische waarden.
Bij een bepaalde toestand van de machine geldt voor zo`n logische variabele dat hij een van de volgende waarden moet hebben: (0,1,dontcare).
Het optreden van een fout wordt aangegeven door error=true.
Al deze logische waarden voor de op dat moment geldende toestand zijn opgeslagen in een array, het bewakingsarray.

Samenvattend wordt de machine beschreven door de volgende variabele: "var"toestandsdiagram:
"record"doel,start,toestand,gemeten toestand:TOESTAND;
updown:(up,down,0);
continue,error:boolean;

```
                diagram:"array"[TOESTAND]"of"
                        "record"volgende,vorige:TOESTAND
                        "end";
                bewakingsarray:"array"[TOESTAND,0..23]"of"
                                        (0,1,dontcare)
        "end";
```

Het type TOESTAND is een "simple variable" die als waarden heeft de namen van de toestanden uit het toestandsdiagram.

De implanter is hier beschreven als de variabele: implanter. dit is een record, waar alle grootheden beschreven staan. Hoewel het in PASCAL mogelijk is om dit op een vrij elegante wijze te doen, zou een beschrijving zoals dat mogelijk is in PEARL (lit.[2]) toch de mooiste oplossing zijn.

Toestandsprocessen.

De eigelijke besturing van de machine gebeurt door een aantal processen (taken) waarvan de werking staat onder supervisie van het mainproces, We noemen deze processen: toestandsprocessen.
Voor iedere toestand in het toestandsdiagram is er in principe een toestandsproces.
Het starten van een toestandsproces gaat d.m.v. een V-operatie op de toestandssemaphoor van dat proces, terwijl een proces gestopt wordt door het een P-operatie op zichzelf uit te laten voeren. Dit laatste wordt geinitieerd door continue=false te maken. De taak die zo`n toestandsproces heeft uit te voeren staat beschreven in doc.3, en bestaat vnl. uit het handhaven van een van de toestanden van de implanter, of het bewerkstelligen van een overgang, in welk geval de taak logisch gezien eindig is.
Een proces dat eindig is meldt zich af als het zijn taak goed volbracht heeft door de boodschap "conditie" naar het mainproces te sturen.
Ontdekt een proces dat er een fout is opgetreden dan stuurt het de boodschap "nonconditie" naar het mainproces en komt in een speciale toestand in afwachting van een reactie an het mainproces.
De structuur van een toestandsproces wordt nu als volgt:

```
"procedure"toestandsproces;
        "begin"initialisatie; P(sema of (runningproces));
        "while"true"do"
                "begin""while"continue"do"
                        "case"toestand"of"
                           T1:S1;
                           T2:S2;
                              .
                              .
                           Tj:Sj
                        "end";
                        toestand:=T1; {T1 is initiele toestand}
                        prod(mainmelding,handshake);
                        P(sema of (runningproces))
```

```
            "end"
        "end"; {van het toestandsproces}
```

De verschillende toestandsprocessen staan beschreven in doc.9

De body van het mainproces

Het mainproces is te zien als de supervisor over het gehele proces. De structuur van dit proces moet zodanig zijn dat het optimaal met deze taak bezig kan zijn.
In ons geval komt dat er op neer dat het mainproces "open staat" voor boodschappen van buiten af, vnl. via de keyboardhandler en de verschillende toestandsprocessen.Het mainproces moet dan ook zo weinig mogelijk opgehouden worden door bijv. een P-operatie die het mainproces blokkeren kan.
Een aanroep van bijv. de procedure suspend(T:integer) e.d. moet daarom vermeden worden. Door verder de buffer die bij het mainproces hoort maar groot genoeg te kiezen, zal de procedure produceer bijna nooit het stoppen van het mainproces tot gevolg hebben, en draait het mainproces eigenlijk alleen niet, als er een interrupt van buiten komt, of er geen boodschap in de buffer staat. In dit laatste geval is er voor het mainproces ook niets te doen. Hij wordt echter direct doorgestart wordt, zodra er iets in de buffer achtergelaten wordt.
Interrupts worden uiteraard zo snel mogelijk afgehandeld. De structuur van het (main)proces is (weer) eenvoudig te noemen, door veelvuldig gebruik te maken van de nucleus van het multiprogrammeringssysteem, en wel in het byzonder van de semaphoor variabelen en de P- en V-operatie daarop.

```
"procedure"mainproces;
        "begin"initialisatie;
        "with"toestandsdiagram"do"
        "while"true"do"
                "begin"consumeer(mainmelding,item);
                        "case"item"of"
                           T1:S1;
                              .
                           Tj:Sj
                        "end"
                "end"
        "end"; {mainproces}
```

Ook nu is de uiteindelijke versie te vinden in doc.9

Het programma in doc.9 is op een aantal punten (nog) niet uitgewerkt. Het voornaamste punt is wel het proces dat bij de toestand "buiten pad" behoort. Dit proces moet e voor zorgen dat als de implanter in een niet getolereerde toestand terecht is gekomen, dat de implanter naar een getolereerde toestand wordt gebracht. Dit is vnl. gebeurd door tijdgebrek. Een andere reden is, dat het niet waarschijnlijk is dat het eerste proefmadel zo uitgebreid zal

zijn als de hier beschreven implanter. Het is verstandig om de gekozen opzet op een gedeelte van de implanter uit te proberen, om het konsept op zijn geschiktheid te testen.

```
"procedure"mainproces;
        "type" TOESTAND=(opstarten,wachtopcom,T1,T12,T21,T2,T23,
                         T32,T3,T34,T43,T4,T45,T54,T5,T56,T65,T6,
                         buiten pad,leeg);
        "var"toestandsdiagram"record"
                doel,start,toestand,gemeten toestand:TOESTAND;
                updown:(up,down,0);
                continue,error:boolean;
                diagram:"array"[TOESTAND]"of"
                        "record"volgende,vorige:TOESTAND
                        "end";
                bewakingsarray:"array"[TOESTAND,0..23]"of"
                                                (0,1,dontcare)
                "end";
             implanter:
             "record"MOTORGEN,KOELVLOEISTOF,ELECTRONICS,VACPOMP,
                FAN,IONPS,VARIACS,MAGNET POWER SUPPLY,MASSMETER
                SCANNER,MAGNET UP,MAGNET DOWN,TUBE PS,HV SUPPLY,
                LENSPS,NEUTRAL TRAP,TRIPLET LENS,TELEMETER:
                                        (aan,uit);
                MAGNEET,MASSMETER:(goede waarde,te kl,te grt);
                MAGNET FINE,LENS PS CONTROL,VANODE,VFILAMENT,
                VMAGNET,VOVEN,VGAS,ANODE CURRENT,ANODE VOLTAGE,
                FILAMENT CURRENT,FILAMENT VOLTAGE,MAGNET CURRENT,
                OVEN CURRENT,GAS: real;
                BRON: (rust,instabiel,ok);
                LENS: (goed,fout);
                KLEP: (open,dicht);
                HOOGSPANNING: (uit,0,goede waarde,te kl,te grt,
                                instabiel);
                BEAMSWEEP: (uit,ok,te kl,te grt);
                CURRENT INTEGRATOR: (uit,rust,instellen,klaar);
                BPM2: (aan,uit,uitgetrokken);
                poweroff,koeling,TELPWR,DCpower,pompok,VACUUM,
                airflow,IONPWR,KLOPEN,MGNPWROK,massok,HVok,
                lenspwr,BPM2on,VARok,NEUTRLPWR,TRIPLPWR,TUBPWR,
                HOOGSPWR: boolean
             "end";
             item:byte;
        "function"next:TOESTAND;
                "var"hulp:TOESTAND;
                "begin"prod(buffer(toestand),reset);
                "if"updown=up"then"hulp:=diagram[toestand].volgende
                        "else"hulp:=diagram[toestand].vorige;
                "if"hulp=doel"then""begin"updown:=0;
                                prod(output,doel bereikt);
                                toestand:=doel
                                "end""else""begin"
                        "if"updown=up"then"
                                toestand:=diagram[hulp].volgende
```

```
                        "else"
                        toestand:=diagram[hulp].vorige
                        "end";
        V(sema of(proc(toestand)))
        "end";
"procedure"neem data over;
        "begin"
        * neemt data uit de keyboardhandler over onder
          semaphoor bescherming *
        "end";
"procedure"update toestandsdiagram;
        "var"eerste:TOESTAND;
        "begin"
        "if"error"then"
                "begin"doel:=wachtopcom;
                error:=true
                "end""else"
                "begin"
                doel:=nieuwe doel;
                meet toestand;
                "if"gemeten toestand=toestand"then"
                        "begin"eerste:=toestand;
                        start:=toestand
                        "end"
                        "else""begin"bepaal toestand;
                                     bepaal eerste;
                                     bepaal start
                                "end";
                "if"start<doel"then"updown:=up
                                "else"updown:=down;
                V(sema of(proc(eerste)))
                "end";
        continue:=true
        "end";
"procedure"verzorg output;
        "begin"
        * stuurt de benodigde data naar de outputhandler *
        "end";
"procedure"bewaak(t);
        "begin"
        bepaal(error); {kijk naar het bewakingsarray}
        "if"error"then"
                "begin" {evt direct ingrijpen}
                melden aan operateur;
                prod(mainmelding,commandl)
                "end"
        "end";
"procedure"reageer;
        "begin"
        {voorlopig alleen het toestandsproces uitzetten}
        prod(buffer(toestand),Tres);
        toestand:=wachtopcom
```

```
                "end";
        "procedure"initialisatie;
                "begin"
                * initialiseert het mainproces en de toestands-
                  processen *
                "end";


"procedure"proces T12;
        "var"toestand:(start,MGOK,pompen,klaar,err1,err2,err3,
                        err4,err5,err6);
        "begin" toestand:=start;
        P(sema of(runningproces));
        "with"implanter"do"
        "while"true"do"
        "begin"
        "while"continue"do"
          "case"toestand"of"
            start:"begin"MOTORGEN:=aan;
                  suspend(2);
                  "if"poweroff"then"
                        "begin"prod(output,MG def);
                        prod(mainmelding,nonconditie);
                        MOTORGEN:=uit;
                        toestand:=err1
                        "end""else"toestand:=MGOK
                  "end";
            MGOK:"begin"KOELVLOEISTOF:=aan;
                 suspend(5);
                 "if""not"koeling"then"
                        "begin"prod(output,koeling def);
                        prod(mainmelding,nonconditie);
                        KOELVLOEISTOF:=uit;
                        toestand:=err2
                        "end""else"toestand:=TEL
                 "end";
            TEL:"begin"TELEMETER:=aan;
                suspend(2);
                "if"TELPWR"then"toestand:=KOK
                        "else""begin"prod(output,telemeter def);
                        prod(mainmelding,nonconditie);
                        TELEMETER:=uit;toestand:=err3
                        "end"
                "end";
            KOK:"begin"ELECTRONICS:=aan;
                suspend(2);
                "if""not"DCpower"then"
                        "begin"prod(output,electr def);
                        prod(mainmelding,nonconditie);
                        ELECTRONICS:=uit;
                        toestand:=err4
                        "end""else"toestand:=ELOK
```

```
    "end";
ELOK:"begin"VACPOMP:=aan;
     suspend(2);
     "if""not"pompok"then"
            "begin"prod(output,pomp def);
            prod(mainmelding,nonconditie);
            VACPOMP:=uit;
            toestand:=err5
            "end""else""begin"toestand:=pompen
                       i:=0
                       "end"
     "end";
pompen:"begin"i:=i+1;
         "if"i<100"and""not"vacuum"then"
            "begin"suspend(180);meet(VACUUM)"end"
            "else"
       "if"VACUUM"then""begin"prod(output,vac ok);
       prod(mainmelding,conditie);
       toestand:=klaar
       "end""else""begin"toestand:=err6;
            prod(output,vac komt niet op);
            prod(mainmelding,nonconditie)
            "end"
       "end";
klaar:"begin"consumeer(buffer(toestand),item);
            {item moet komen}
             "case"item"of"
                    reset:"begin"toestand:=start;
                          P(sema of(runningproces))
                          "end"
      "end";
err1,err2,err3,err4,err5,err6:
    "begin"consumeer(buffer(toestand),item);
           "case"item"of"
             a:toestand:=start;
             b:toestand:=MGOK;
             c:toestand:=TEL;
             d:toestand:=KOK;
             e:toestand:=ELOK;
             Tres:"begin"
                    VACPOMP:=uit;
                    ELECTRONICS:=uit;
                    TELEMETER:=uit;
                    KOELVLOEISTOF:=uit;
                    MOTORGEN:=uit;
                    toestand:=start;
                    P(sema of(runningproces))
                    "end"
    "end";
    toestand:=start;
    prod(mainmelding,handshake);
    P(sema of(runningproces))
```

```
          "end"
        "end"; {of proces T12}

"procedure"procesT21;
        "begin"
        "with"implanter"do"
        "while"true"do"
                "begin"
                P(sema of(runningproces));
                VACPOMP:=uit;
                ELECTRONICS:=uit;
                KOELVLOEISTOF:=uit;
                MOTORGEN:=uit;
                prod(mainmelding,conditie)
                "end"
        "end"; {procesT21}

"procedure"procesT23;
        "var"toestand:(start,fanon,IONPOWER,bronok,err1,err2,err3);
             item:byte;
        "begin"
        initialisatie;
        P(sema of(runningproces));
        "with"implanter"do"
        "while"true"do"
        "begin"
        "while"continue"do"
        "case"toestand"of"
          start:"begin"FAN:=aan;
                suspend(4);
                "if"airflow"then"toestand:=fanon
                        "else""begin"
                        prod(output,FAN def);
                        prod(mainmelding,nonconditie);
                        FAN:=uit;
                        toestand:=err1
                        "end"
                "end";
          fanon:"begin"IONPS:=aan;
                suspend(2);
                "if"IONPWR"then"toestand:=IONPOWER
                        "else""begin"
                        prod(output,ION PS def);
                        prod(mainmelding,nonconditie);
                        IONPS:=uit;
                        toestand:=err2
                        "end"
                "end";
          IONPOWER:start de bron op; {meld zich via conditie af}
          bronok:"begin"consumeer(buffer(toestand),item)
                 "case"item"of"
                        reset:"begin"toestand:=start;
```

```
                                P(sema of(runningproces))
                                "end"
                    "end";
          err1,err2,err3:
                    "begin"consumeer(buffer(toestand),item);
                    "case"item"of"
                              a:toestand:=start;
                              b:toestand:=fanon;
                              c:toestand:=IONPOWER;
                              Tres:"begin"
                                   VARIACS:=uit;
                                   IONPS:=uit;
                                   FAN:=uit;
                                   toestand:=start;
                                   P(sema of(runningproces))
                                   "end"
                    "end";
                    toestand:=start;
                    prod(mainmelding,handshake);
                    P(sema of(runningproces))
          "end"
          "end"; {procesT23}

"procedure"procesT3;
          "begin"P(sema of(runningproces));
          "with"implanter"do"
                    "while"true"do"
                    "begin"
                    "while"continue"do"
                    "begin"
                    test de bron;
                    suspend(10)
                    "end";
                    prod(mainmelding,handshake);
                    P(sema of (runningproces))
                    "end"
          "end"; {procesT3}

"procedure"procesT32;
          "begin"
          "with"implanter"do"
          "while"true"do"
          "begin" P(sema of(runningproces));
                    VARIACS:=uit;
                    IONPS:=uit;
                    FAN:=uit;
                    prod(mainmelding,conditie)
          "end"
          "end"; {procesT32}

"procedure"procesT34;
          "var"toestand:(start,OPEN,massm,magneet,HVPS,lensps,
```

```
      plaatje,klaar,err1,err2,err3,err4,err5,err6,err7);
      item:byte;
"begin"
initialisatie;
P(sema of(runningproces));
"with"implanter"do"
"while"true"do"
"begin"
"while"continue"do"
  "case"toestand"of"
    start:"begin"KLEP:=open;
          suspend(2);
          "if"KLOPEN"then"toestand:=OPEN
                "else""begin"prod(output,klep def);
                prod(mainmelding,nonconditie);
                KLEP:=dicht;toestand:=err1
                "end"
          "end";
    OPEN:"begin"MAGNET POWER SUPPLY:=aan;
         suspend(2);
         "if"MGNPWROK"then"toestand:=massm
                "else"prod(output,magneet ps def);
                prod(mainmelding,nonconditie);
                MAGNET POWER SUPPLY:=uit;
                toestand:=err2
                "end"
         "end";
    massm:"begin"MASSMETERSCANNER:=aan;
          suspend(2);
          "if"massok"then"toestand:=magneet
                "else""begin"prod(output,massmeter/scanner
                                        def);
                prod(mainmelding,nonconditie);
                MASSMETERSCANNER:=uit;
                toestand:=err3
                "end"
          "end";
    magneet:"begin"magneet naar goede waarde;
        "if"massmeter=goede waarde"then"toestand:=HVPS
                "else""begin"prod(output,magneetregeling of
                                  massmeter def);
                prod(mainmelding,nonconditie);
                toestand:=err4
                "end"
        "end";
    HVPS:"begin"HVSUPPLY:=aan;
         suspend(2);
         "if"HVok"then"toestand:=lensps
                "else""begin"prod(output,HV ps def);
                prod(mainmelding,nonconditie);
                HVSUPPLY:=uit;toestand:=err5
                "end"
```

```
                "end";
          lensps:"begin"LENSPS:=aan;
                  suspend(2);
                  "if"lenspwr"then"toestand:=plaatje
                       "else""begin"prod(output,lens ps def);
                       prod(mainmelding,nonconditie);
                       LENSPS:=uit;toestand:=err6
                       "end"
                  "end";
          plaatje:maak mooi plaatje op BPM;
          klaar:"begin"consumeer(buffer(toestand),item);
                "case"item"of"
                       reset:"begin"toestand:=start;
                             P(sema of(runningproces))
                             "end"
                "end";
          err1,err2,err3,err4,err5,err6,err7:
               "begin"consumeer(buffer(toestand),item);
                 "case"item"of"
                       a:toestand:=start;
                       b:toestand:=massm;
                       c:toestand:=magneet;
                       d:toestand:=HVPS;
                       e:toestand:=lensps;
                       f:toestand:=plaatje;
                       Tres:"begin"LENSPS:=uit;
                                   HVSUPPLY:=uit;
                                   MASSMETERSCANNER:=uit;
                                   magneet naar nul;
                                   MAGNET POWER SUPPLY:=uit;
                                   KLEP:=dicht;
                                   toestand:=start;
                                   P(sema of(runningproces))
                             "end"
                "end"
            "end";
        prod(mainmelding,handschake);
        P(sema of(runningproces))
        "end"
        "end"; {procesT34}

"procedure"procesT4;
        "begin"P(sema of(runningproces));
                "with"implanter"do"
                "while"true"do"
                "begin"
                "while"continue"do"
                "begin"
                test de bron;
                bekijk plaatje;
                suspend(15)
                "end";
```

```
                prod(mainmelding,handshake);
                P(sema of(runningproces))
                "end"
        "end"; {procesT4}

"procedure"procesT43;
        "begin"
              "with"implanter"do"
              "while"true"do"
               "begin"P(sema of(runningproces));
               LENS:=uit;
               HVSUPPLY:=uit;
               MASSMETERSCANNER:=uit;
               magneet naar nul;
               MAGNET POWER SUPPLY:=uit;
               KLEP:=dicht;
               prod(mainmelding,conditie)
               "end"
        "end"; {proces43}

"procedure"procesT45;
        "var"toestand:(start,neutral,TRIPL,tubelens,HOOGSP,HSPreg,
                       plaatje,klaar,err1,err2,err3,err4,err5,
                       err6);
        "begin"
        initialisatie;P(sema of(runningproces));
        "with"implanter"do"
        "while"true"do"
        "begin"
        "while"continue"do"
          "case"toestand"of"
            start:"begin"BPM2:=aan;
                  suspend(2);
                  "if"BPM2on"then"toestand:=neutral
                        "else""begin"prod(output,BPM2 def);
                        prod(mainmelding,nonconditie);
                        BPM2:=uit;toestand:=err1
                        "end"
                  "end";
            neutral:"begin"NEUTRAL TRAP:=aan;
                    suspend(2);
                    "if"NEUTRLPWR"then"toestand:=TRIPL
                        "else""begin"prod(output,neutral trap def);
                        prod(mainmelding,nonconditie);
                        NEUTRAL TRAP:=uit;toestand:=err2
                        "end"
                    "end";
            TRIPL:"begin"TRIPLETLENS:=aan;
                  suspend(2);
                  "if"TRIPLPWR"then"toestand:=tubelens
                        "else""begin"prod(output,triplet def);
                        prod(mainmelding,nonconditie);
```

```
                TRIPLETLENS:=uit;toestand:=err3
                "end"
        "end;
tubelens:"begin"TUBEPS:=aan;
           suspend(2);
           "if"TUBEPWR"then"toestand:=HOOGSP
              "else""begin"prod(output,tube lens def);
              prod(mainmelding,nonconditie);
              TUBEPS:=uit;toestand:=err3
              "end"
           "end";
HOOGSP:"begin"HOOGSPANNING:=aan;
          suspend(3);
          "if"HOOGSPWR"then"toestand:=HSPreg
              "else""begin"prod(output,hoogsp def);
              prod(mainmelding,nonconditie);
              HOOGSPANNING:=uit;toestand:=err4
              "end"
          "end";
HSPreg:"begin"hoogspanning naar juiste waarde;
          "if"HOOGSPANNING=goede waarde"then"
              toestand:=plaatje
              "else""begin"prod(output,HSPregeling def);
              prod(mainmelding,nonconditie);
              hoogspanning naar nul;
              toestand:=err5
              "end"
          "end";
plaatje:maak mooi plaatje op 2de BPM:
klaar:"begin"consumeer(buffer(toestand),item);
        "case"item"of"
              reset:"begin"toestand:=start;
                    P(sema of(runningproces))
                    "end"
        "end";
err1,err2,err3,err4,err5:
     "begin"consumeer(buffer(toestand),item);
        "case"item"of"
              a:toestand:=start;
              b:toestand:=neutral;
              c:toestand:=TRIPL;
              d:toestand:=tubelens;
              e:toestand:=HSPreg;
              f:toestand:=plaatje;
              Tres:"begin"hoogspanning naar nul;
                          HOOGSPANNING:=uit;
                          TUBEPS:=uit;
                          TRIPLETLENS:=uit;
                          NEUTRAL TRAP:=uit;
                          BPM2:=uit;
                          toestand:=start;
                          P(sema of(runningproces)
```

```
                                "end"
                        "end"
          "end";
        prod(mainmelding,handshake);
        P(sema of(runningproces))
        "end"
        "end"; {procesT45}

"procedure"procesT5;
        "begin"P(sema of(runningproces));
        "with"implanter"do"
                "while"true"do"
                "begin"
                "while"continue"do"
                "begin"
                test de bron;
                bekijk plaatje;
                bekijk plaatje 2;
                suspend(30)
                "end";
                prod(mainmelding,handshake);
                P(sema of(runningproces))
                "end"
        "end"; {procesT5}

"procedure"procesT54;
        "begin""with"implanter"do"
               "while"true"do"
                "begin"P(sema of(runningproces));
                hoogspanning naar nul;
                HOOGSPANNING:=uit;
                TUBEPS:=uit;TRIPLET LENS:=uit;
                NEUTRAL TRAP:=uit;BPM2:=uitgetrokken;
                prod(mainmelding,conditie)
                "end"
        "end"; {procesT54}

        "begin"
        initialisatie;
        "with"toestandsdiagram"do"
        "while"true"do"
        "begin" consumeer(mainmelding,item);
                "case"item"of"
                  command1:"begin"neem data over;
                           "if"error"or"(continue"and"(nieuw doel
                                        "neq"toestand))
                                "then""begin"
                                   continue:=false;
                                   "if"toestand=wachtopcom
                                   "or"toestand=T1
                                   "or"toestand=T2
                                   "or"toestand=T21
```

```
                                "or"toestand=T32
                                "or"toestand=T43
                                "or"toestand=T54
                                "then"prod(mainmelding,
                                           handschake)
                                "end"
                              "else"
                              "begin""if""not"continue"then"
                                prod(output,andere verandering)
                                "else"
                                prod(output,doel al bereikt)
                              "end"
                    "end";
          handshake:"begin"update toestandsdiagram;
                     prod(output,nieuwe taak in uitvoering)
                     "end";
          command2:"begin"neem data over;
                    "if"verandering toegestaan"then"
                         verander parameter
                    "end";
          command3:"begin"neem data over;
                    verzorg output
                    "end";
          bewaking:bewaak(toestand);
          conditie:toestand:=next;
          nonconditie:reageer
        "end"
"end"
"end"; {of mainproces}
```

Uitleiding

Ik wil dit verslag eindigen met iedereen die het mij mogelijk heeft gemaakt om op deze wijze af te studeren van harte te bedan ken. Hoewel het noemen van namen met zich meebrengt dat ik waarschijnlijk mensen vergeet te noemen, wil ik toch een paar uitzonderingen maken.
Prof. Heetman wil ik bedanken omdat hij het mogelijk heeft gemaakt om bij de vakgroep EB dit onderzoek te verrichten. Van de vakgroep wil ik verder Kees Stork en Gerard Havermans bedanken voor de hulp bij de opzet en uitvoering van de gemaakte software.
Fokko van der Velde en Joop Stronkhorst van High Voltage Engeneering wil ik bedanken, vooral omdat zij het zijn die dit onderzoek in eerste instantie mogelijk hebben gemaakt.