Software voor een hybride computer

Mulleneers, J.J.M.

Award date:
1975

AFDELING DER ELEKTROTECHNIEK
TECHNISCHE HOGESCHOOL
EINDHOVEN
Groep Meten en Regelen

SOFTWARE VOOR EEN HYBRIDE COMPUTER

door J.J.M. Mulleneers

Rapport van het afstudeerwerk
uitgevoerd van juli 1974 tot mei 1975
in opdracht van prof. ir. F.J. Kylstra
onder leiding van ir. J.J.M. van Munen

SAMENVATTING

In dit rapport worden software-pakketten beschreven voor een hybride rekenmachine. Deze pakketten hebben betrekking op:
a) de communicatie tussen analoge en digitale computer en
b) het testen van de componenten van analoge machine en interface.

SUMMARY

In this report software-packages are described for a hybrid computer. These packages support:
a) the communication between analogue and digital computer and
b) the testing of the components of the analogue machine and interface.

INHOUDSOPGAVE

1 INLEIDING

In de groep FR is sinds september 1973 de koppeling tussen de digitale minicomputer PDP11 en de analoge computer HITACHI 505 gerealiseerd. Om de aldus opgebouwde hybride rekeneenheid programmatisch te kunnen sturen, is het nodig om de onder het RT11-operating system werkende BASIC en FORTRAN software pakketten uit te breiden en te modificeren. Voor een betrouwbare werking van de configuratie is regelmatige controle noodzakelijk van analoge machine en interface. Dit wordt gerealiseerd in een pakket software voor test doeleinden. De communicatie tussen analoge en digitale partner tijdens een hybride rekenproces is in een pakket hybride communicatie-routines ondergebracht. Bij het ontwerpen van software voor een hybride computer zal men rekening moeten houden met:

a) het specifieke karakter van de hybride rekenvorm.
b) de beschikbare hardware.
c) de beschikbare systeem-software.

Een nadere beschouwing van de hybride rekenvorm lijkt dan ook een gewenst startpunt voor de realisatie van software voor een hybride computer.

2 HYBRIDE REKENEN

De hybride rekenvorm is sinds de jaren vijftig op gang gekomen. De computer gebruikers die met complexe rekenproblemen werden geconfronteerd, onderkenden dat analoog of digitaal rekenen zowel specifieke voordelen als nadelen met zich meebracht. Het mengen van beide rekenvormen tot de zogenaamde hybride rekentechniek leek voor menig rekenprobleem een elegantere en betere oplossingsmethodiek te bieden.

De koppeling van digitale en analoge computer tot een hybride computer bleek specifieke eisen te stellen aan opbouw en bediening van de analoge machine, alsmede aan de interface tussen beide machines. De hardware, nodig voor deze koppeling moest een flexibele en doelmatige besturing van de analoge machine, via de voor de digitale partner gebruikte software, mogelijk maken. Dan pas kon men van de gunstige eigenschappen van analoog resp. digitaal rekenen optimaal gebruikmaken. De analoge machine heeft als primair voordeel de grote rekensnelheid, doordat bij het analoge proces de rekenbewerkingen parallel verlopen; de rekenelementen zijn gelijktijdig actief. De toestand (mode), waarin de rekenelementen van de analoge machine zich bevinden, kan door de gebruiker via bedieningsknoppen zelf bepaald worden. De mens - machine interactie is dus vrij gemakkelijk uit te voeren, waardoor de invloed van parameterwijzigingen in het rekenproces direct is na te gaan.

Bij digitaal rekenen moeten alle rekenbewerkingen achter elkaar geschakeld worden via programmering. Men heeft slechts een rekenorgaan. Dit kan vooral bij complexe problemen zoals het oplossen van differentiaalvergelijkingen tot vrij lange rekentijden voeren. De voordelen aan de digitale zijde liggen meer in de geheugenfaciliteiten die de gebruiker tot zijn beschikking staan en de flexibele programmering van de rekenbewerkingen. Bij de aanpak van een rekenprobleem dat men op een hybride computer wil oplossen zal men de taken over digitale en analoge computer zó moeten verdelen, dat van de reeds genoemde voordelen zoveel mogelijk profijt wordt getrokken. De analoge rekenmachine verzorgt de uitvoering van de zeer snelle rekendelen als b.v. het oplossen van differentiaalvergelijkingen. De digitale machine fungeert als data buffer; de data uit de analoge machine kan in haar geheugen worden

opgeslagen en bij verdere berekeningen benut worden. Bij vele rekenproblemen zoals optimaliseringsproblemen, partiele differentiaalvergelijkingen, moeten bovengenoemde bewerkingen vele malen uitgevoerd worden. De analoge machine schiet hier duidelijk te kort wat betreft geheugenfaciliteiten maar biedt voor het analoge deelprobleem een zeer gunstige rekensnelheid.

Afhankelijk van het soort hybride rekenprobleem wordt de uitvoering van de rekenbewerkingen op digitale en analoge machine op twee manieren geregeld.

a) Analoge en digitale machine voeren om de beurt een rekenbewerking uit; dit noemt men alternerend hybride rekenen. Deze vorm van hybride rekenen wordt toegepast voor die rekenproblemen waarbij de rekenbewerking in de ene machine pas uitgevoerd kan worden met behulp van de resultaten van de voorgaande bewerking op de andere machine. De tijd, die de analoge machine in de houdtoestand verkeert, is afhankelijk van de benodigde tijd, die de uitlezing en verwerking van de data van de voorgaande rekenbewerking vergt in een programmadeel. De start van de volgende rekenbewerking vindt pas plaats als dit programmadeel voltooid is. Varieert de duur van dit programmadeel dan varieert ook de aangegeven cyclusduur.

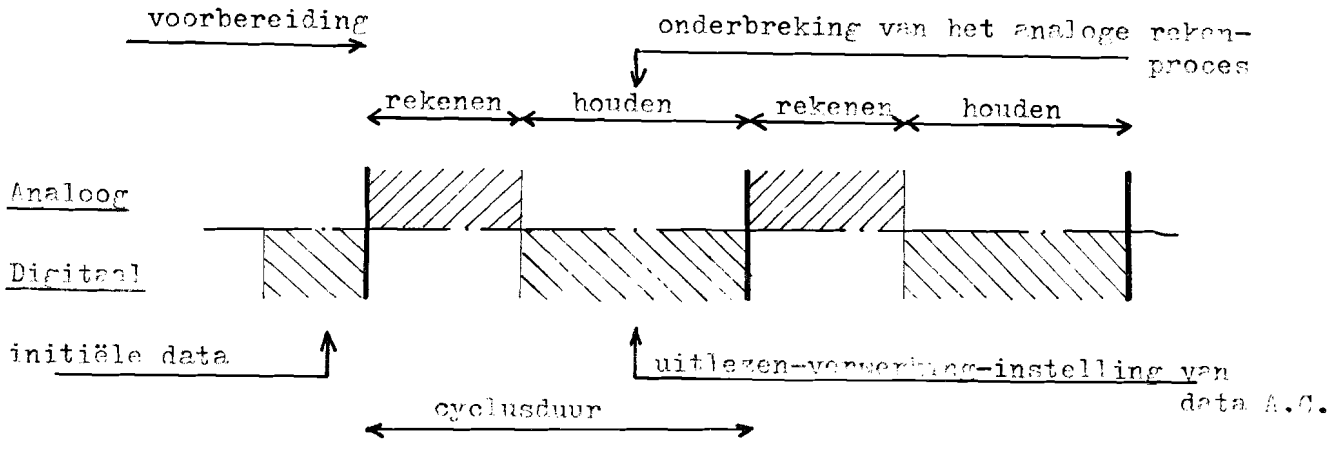


Fig. 1 : Alternerend hybride rekenen.

De analoge machine verkeert afwisselend in de reken- resp. houd-toestand. Na een analoge rekenbewerking slaat de digitale machine de resultaten van deze bewerking op, voert zonodig hiermede berekeningen uit en stelt de analoge begindata voor de volgende rekenbewerking in. Deze alternerende rekenvorm wordt o.a. toegepast bij het oplossen van stelsels niet-lineaire differentiaalvergelijkingen.

b) Analoge en digitale machine zijn gelijktijdig in het rekenproces betrokken. Men noemt dit <u>parallel</u> hybride rekenen.

Bij deze vorm van hybride rekenen zal de gegevens-uitwisseling tussen de twee partners op vaste tijdstippen en/of bij vaste gebeurtenissen moeten plaatsvinden. De ontwerper van het programma voor de sturing van het rekenproces wordt nu geconfronteerd met synchronisatieproblemen van analoge en digitale machine. Om dit op te lossen is het noodzakelijk, dat in de hardware van de digitale machine (of interface) voorzieningen zijn getroffen, om op instelbare tijdstippen interruptaanvragen te kunnen doen. Via deze interrupt wordt het digitale programma gestart dat voor de data-verwerking zorg draagt. Het tijddiagram voor analoge en digitale zijde ziet er dan als onderstaand uit.

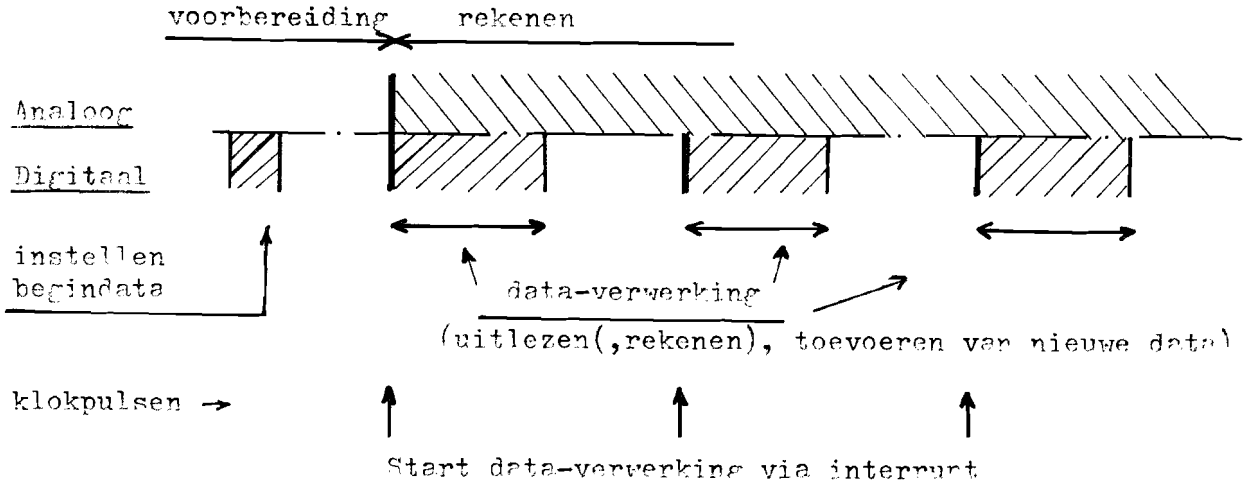


Fig. 2 : Parallel hybride rekenen.

Iedere uitgelezen data is gekoppeld aan een vaste tijd ten opzichte van de start van de rekenbewerking. De programma-duur is bepaald door het aantal interrupts en hun onderling tijdsinterval.

3 HYBRIDE HARDWARE - SOFTWARE

3.1 Hardware

In zijn algemeenheid omvat een hybride rekeneenheid de in onderstaand schema weergegeven componenten.

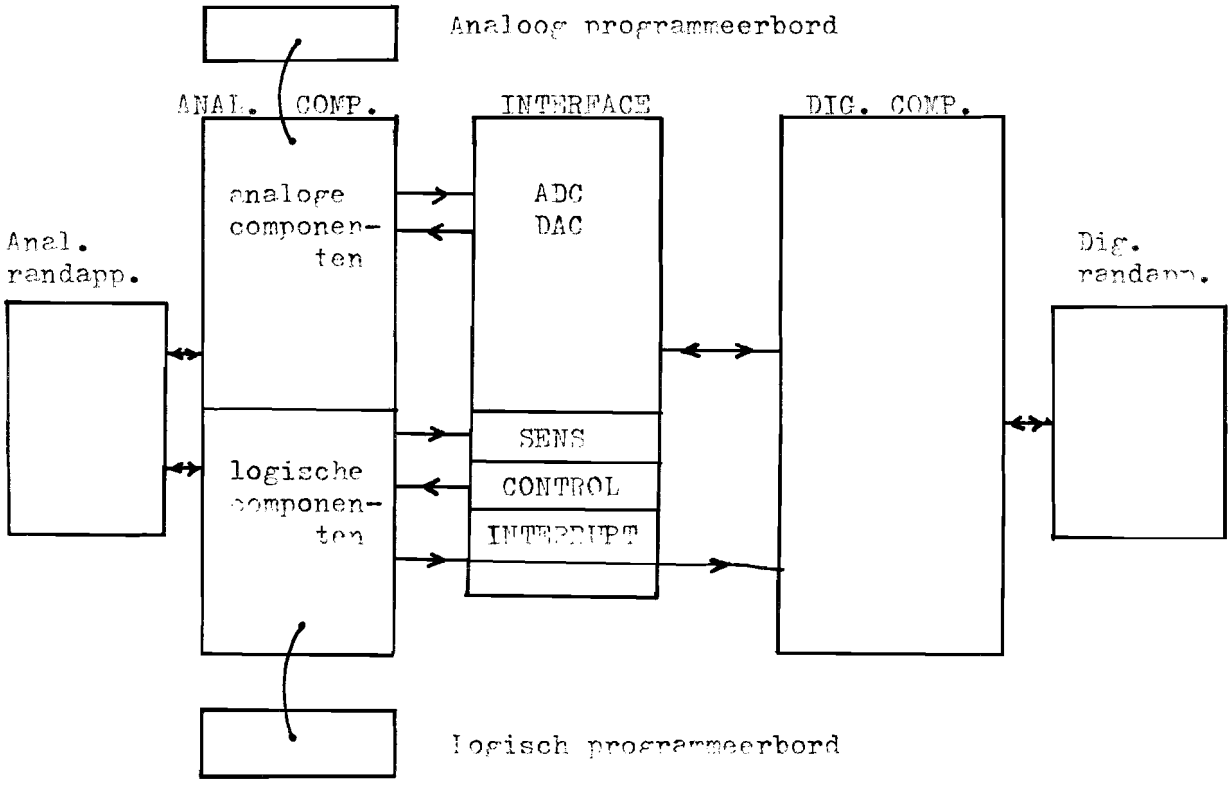


Fig. 3: Hybride rekeneenheid.

Bij de moderne hybride systemen zijn zowel analoge als digitale machine volwaardige rekenmachines. De analoge machine is qua uitvoering reeds volledig voorbereid op de koppeling met de digitale machine. Componenten als electronische schakelaars, digitaal instelbare coëfficienteenheden en parallelle logica laten zeer snelle omschakelingen in

de analoge schakelschema's toe. In dit opzicht voldoen de veel trager in te stellen servopotmeters niet meer. De digitale machine zal over hardware floating point rekenfaciliteiten moeten beschikken voor snelle en nauwkeurige berekeningen. Interrupt afhandeling moet snel kunnen gebeuren. Hiervoor is vaak speciale hardware vereist, waarbij ook de prioriteit, waarop de interrupt-aanvraag gebeurt, kan worden ingesteld.

In de interface treft men de volgende componenten aan:

a) een snelle analoog-digitaal converter, gekoppeld aan een multiplexer voor meerdere analoge ingangskanalen.
b) een aantal digitaal-analoog converters
c) een aantal registers die de hardware voor de sturing van de interface en conversie-apparatuur vormen. Via programma-instructies kunnen zowel lees- als schrijfoperaties op deze registers uitgevoerd worden. De mogelijkheid voor interrupt-aanvraag is ook in deze hardware ondergebracht.
d) een intervaltimer voor het genereren van pulsen. Via deze pulsen kunnen interrupt aanvragen worden gestart, waardoor synchronisatie van het hybride programma mogelijk is.

3.2 Software

De graad van ontwikkeling van software pakketten voor hybride installaties is veelal afhankelijk van de volgende factoren:

a) de hardware die de hybride installatie omvat. Dure faciliteiten als b.v. het automatisch "patchen" van het analoge rekenschema blijken economisch nog moeilijk te motiveren (zie lit. 7).
b) de gebruikersdoeleinden van de installatie; het aantal potentiële gebruikers, de mate waarin van de installatie gebruik wordt gemaakt, onderwijsfaciliteiten, enz.

Naast de gebruikelijke software voor de digitale machine, nodig voor de ontwikkeling en uitvoering van programma's, is er ook een pakket soft-

ware specifiek gericht op de toepassing bij hybried rekenen. Dit speciaal voor hybride gebruiksdoeleinden ontwikkeld pakket omvat in grote lijnen de volgende componenten:

I Software, waarin de communicatie tussen analoge en digitale machine wordt geregeld; een aantal routines, die vaak in assemblertaal geschreven zijn, zijn aanroepbaar in een hogere programmeertaal als BASIC of FORTRAN.

II Testprogramma's voor regelmatige controle van de analoge rekenmachine-componenten. Tevens wordt hierin ook betrokken de hardware van de interface.

III Utility-programma's voor de instelling van de analoge rekenelementen en statische test van de analoge rekenschakeling.

Bij grote hybriede installaties die de hardware voor het automatisch opzetten van analoge schema's bezitten zijn software pakketten in ontwikkeling die - uitgaande van de differentiaalvergelijkingen die het probleem beschrijven - zelf analoge rekenschema's genereren en aanbrengen op de analoge machine. In deze hybride configuraties is het streven om het totale hybride gebeuren te kunnen programmeren in een "hybride " programmeertaal. (lit. 6)

4 DE HYBRIDE INSTALLATIE IN DE GROEP ER

4.1 De vakgroep meten en regelen van de afdeling Electrotechniek van de Technische Hogeschool Eindhoven beschikt over een hybride computer die uit onderstaande delen is opgebouwd.

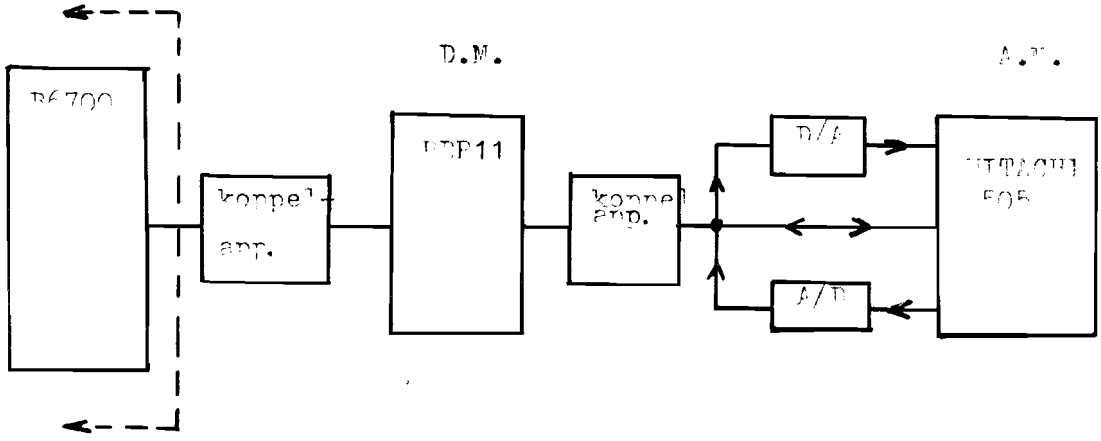


fig.4:hybride installatie in E R

De "kleine" configuratie PDP11-HITACHI 505 is aanwezig in de groep ER. Voor het uitgebreide digitale rekenwerk is een koppeling gelegd met de digitale B6700 computer in het rekencentrum van de TH Eindhoven. In de "grote" configuratie B6700-PDP11-HITACHI 505 vervult de PDP11 minicomputer dan de rol van databuffer en besturingsorgaan van het hybride reken proces Hierbij is vooral de regeling van de tijdcritische communicatie tussen analoge en digitale partner van belang.

Tussen de PDP11 en B6700, resp. de PDP11 en HITACHI 505 is de benodigde hardware voor de koppeling verwerkt in een koppelunit (zie lit.11 en lit.25). Voor het begrip van de gemaakte software is een korte beschrijving van de hybride systeemcomponenten in de kleine configuratie gewenst.

4.2 De digitale minicomputer PDP11

De PDP 11 is een minicomputer, die qua opbouw in feite bestaat uit

een snel datakanaal - "Unibus" - met daarop aangesloten processor , geheugen en randapparatuur.

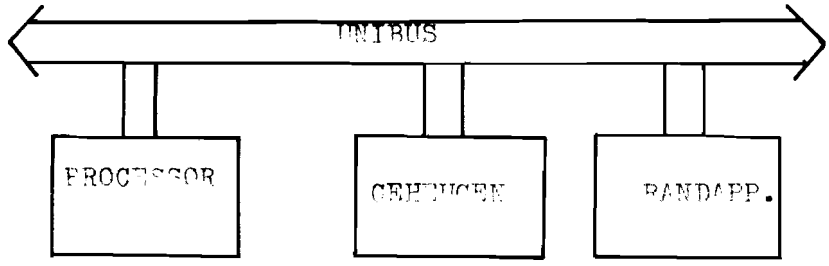


fig.1: PDP 11-Unibus.

De Unibus omvat 56 bidirectionele lijnen te weten:

16 data lijnen
18 adres lijnen
22 controle en synchronisatie lijnen.

De geheugenlocaties alsmede de registers in de randapparatuur vormen de adresruimte van de machine. De machine is een twee-adres type en heeft een woordlengte van 16 bits. De processor beschikt over 8 "general purpose" registers. Van deze registers is één register (nr. 7) in gebruik als programmateller. Register 6 wordt gebruikt als stackpointer. Door de toepassing van een hardware stack, d.i. een tijdelijk in gebruik zijnde aantal geheugenplaatsen, kunnen (geneste) interrupts en reentrant routines vrij eenvoudig uitgevoerd worden.

De 16 in gebruik zijnde adreslijnen geven een adrescapaciteit tot 32 K.woorden. De laatste 4 K. van de adresruimte is gereserveerd voor de registers in de randapparatuur. Het data - transport over de Unibus is geregeld in een master-slave relatie en wordt asynchroon uitgevoerd. Voor het verkrijgen van de masterpositie kan ieder randapparaat een aanvraag voor bus-mastership indienen. De prioriteit van een aanvraag kan via hardware op een bepaald niveau ingesteld zijn. De prioriteit van de processor wordt via software (aan te geven in het processor statuswoord PSW) bepaald. Aanvragen voor servicing door randapparaten kunnen via het plaatsen van een interrupt geëffectueerd worden. Is het prioriteitsniveau van deze interrupt hoger dan van het lopende programma, dan wordt dit onderbroken. De programmateller en processor status worden automatisch

gered op de stack en het programma wordt verder gestart op een adres dat via een hardware wijzer is vastgelegd in het interrupt vektor adres. Tevens wordt het processor status woord geladen met de prioriteit van het nieuwe programma. Is het nieuw gestarte programma afgehandeld dan kan via een return from interrupt (RTI) instructie het onderbroken programma weer hervat worden op het punt van onderbreking. Voor het programmeren in assembler-taal is een instructieset van 55 instructies aanwezig. Deze omvat

13 enkel operand instructies
7 dubbel operand instructies
22 programma controle instructies
8 conditie code instructies
5 diverse instructies.

Er zijn 8 adresseringsmodes; 4 directe en 4 indirecte. (zie lit.19)

4.3 Het hybride koppelapparaat tussen PDP11 en HITACHI 505

De functie van het hybride koppelapparaat omvat de besturing van de analoge machine en de apparatuur van DA- en AD-conversie. Om sturing via programmering mogelijk te maken bevat het koppelapparaat een aantal registers die tot de device adres-ruimte van de PDP11 behoren. De verdeling van de adressen is in onderstaande tabel aangegeven.

173600	status register
173602	synchronisatie register
173604	fouten register
173606	MODE A.C.
173610	klokpuls interval register
173612	versterker select register
173614	vrij adres
173616	vrij adres
173620	digitaal input register
173622	digitaal output register
173700	DAC/ADC kanaal 0
173702	DAC/ADC kanaal 1

173736	DAC/ADC kanaal 15
173776	laatste adres.

De hardware bevat verder een interrupt control moduul die het aanvragen van interrupts verzorgt. Er zijn twee interruptmogelijkheden aangebracht.

a) De voortgang van het hybride programma vereist het uitvoeren van een bepaald programma (INT A).

b) In het hybride koppelapparaat wordt een fout geconstateerd (INT B).

Het prioriteitsniveau van INT A is 6, van INT B 7. Voor de tijdcritische communicatie tussen de PDP11 en de HITACHI 505 dienen de tijdstippen, waarop de interrupts aangevraagd worden, instelbaar te zijn. Het tijdsinterval wordt afgeleid van een programmeerbare klok.

4.4 De analoge computer HITACHI 505

In deze machine worden de rekengrootheden gerepresenteerd door spanningen van -100 tot +100 volt. De machine bezit de volgende analoge componenten:

108	versterkers - bandbreedte 100 KHz - versterking 1 x
36	integratoren
16	compensatoren
12	multiplicatoren
36	servopotentiometers
72	potentiometers
16	vrije relais

Deze analoge componenten zijn verdeeld over twee consoles. Op een derde console kan de rekenmode en integratietijdconstante ingesteld worden. Voor indicatie zijn een digitale en analoge voltmeter aanwezig. De digitale voltmeter (Fluke) is via software te bedienen. Voor afstandbesturing zijn ingebouwd:

1) een adresselektor: hiermee kunnen de uitgangen van versterkers,

comperatoren of midden contacten van relais gekozen worden.

2) Een rekenmode instelling. De modes RESET, COMPUTE, HOLD, POTSET en ALLRESET zijn op afstand instelbaar.

4.5 Conversie - apparatuur

1) Analoog/Digitaal conversie

De 16 analoge inputkanalen worden via een multiplexer aangesloten op één ADC.

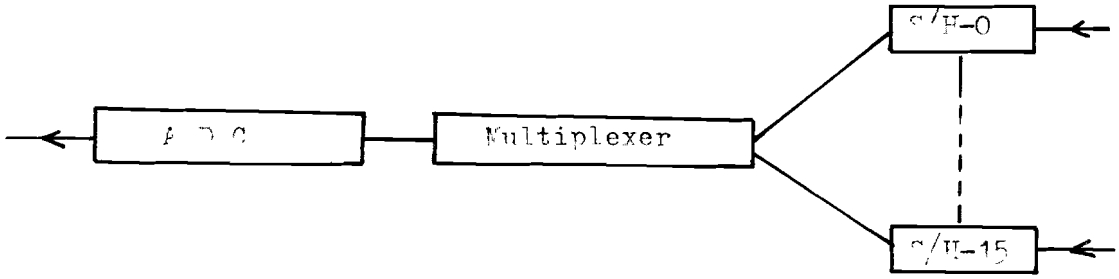


fig. 6: analoog-digitaal conversie-app.

Bij analoog-digitaal conversie worden de sample/hold versterkers van alle kanalen gelijktijdig in hold gezet waarna hun uitgangs waarden sequentieel geconverteerd worden

2) Digitaal/Analoog conversie

De 16 analoge output kanalen zijn ieder voorzien van een DAC. De kanalen zijn dubbel gebufferd om alle kanalen gelijktijdig te kunnen converteren.

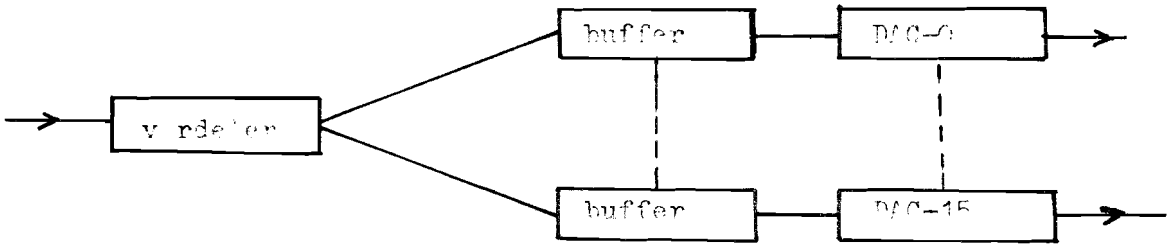


fig.7: digitaal-analoog conversie-app.

4.6 Randapparatuur

Als in- en uitvoermedium voor programma's staan ter beschikking een LT33-teletype en een display terminal Tektronic 4010. Voor opslag van programma's beschikt men over een dual Dec-tape. Per tape is de geheugencapaciteit 564 blokken van 256 woorden.

4.7 Systeem-software

De systeem-software is ondergebracht in het RT11 operating system. Dit operating systeem is opgezet voor programmaontwikkeling op interactieve basis voor één gebruikter. Voor programmaontwikkeling beschikt dit systeem over de systeemprogramma's "text editor", "MACRO-assembler", "linker", "peripheral interchange", "on line debugging program".

De verwerking van programma's geschiedt onder supervisie van de RT11-monitor. Voor de communicatie tussen geheugen en randapparatuur zijn op tape een aantal device-handlers opgenomen. Is een device-handler nodig voor het uitvoeren van data-transport dan wordt de handler van de tape in het geheugen geladen en voor de uitvoering van het transport benut. Programmering is op verschillend niveau mogelijk. Naast de PDP11-assemblertaal kan de gebruiker ook gebruik maken van de hogere taal BASIC. Hiertoe bevat het operating system een aan de speciale file-structuur van RT11 aangepaste BASIC-interpreter voor conversationele programmering. Sinds April '75 kan onder RT 11 ook in FORTRAN geprogrammeerd worden.

5 HYBRIDE COMMUNICATIE-PAKKET IN BASIC/RT11

5.1 Algemeen

De hybride routines die de communicatie tussen de PDP11 en de HITACHI 505 regelen zijn geschreven in de PDP11 assembler taal. De routines zijn qua opbouw en toepassing gericht op de invoeging in de BASIC-interpreter, die onder het RT11-operating system functioneert. Vanaf de teletype kan dan in BASIC op interactieve basis met de hybride machine gewerkt worden. Het aanroepen van de routine in BASIC geschiedt via de "CALL"-statement. Achter de naam van de routine kan ook nog een lijst van argumenten meegegeven worden. Vorm van statement:

CALL "NAME" (argument 1, argument 2, ..., ...).

Voor de interface van de variabelen in de argument list met de routine-instructies in assembler-code wordt verwezen naar het rapport :

"Assembler routines in BASIC" (zie lit. 21) .

Functioneel is het hybride communicatiepakket op te delen in:

a) routines die de hybride run voorbereiden en uitvoeren.

b) routines voor bediening van de mode van de analoge machine.

c) routines voor in/uitlezen van zowel analoge als digitale in/uitgangskanalen van de analoge machine.

d) routines voor graphic display op de Tektronic 4010-display-terminal.

5.2 Organisatie van een hybride rekenprogramma

Wanneer men een parallelle hybride rekenbewerking wil uitvoeren-nadat het analoge rekenschema is aangebracht en gecontroleerd - dan moeten er in de tijd gezien de volgende acties plaatsvinden.

a) De analoge machine moet de initiële data toegevoerd moeten krijgen.

b) De start van de analoge rekenbewerking wordt vastgelegd op een

tijdstip bepaald door de klok.

c) Met vaste tijdsintervallen worden da analoge uitgangskanalen van de A.C. uitgelezen en de data van de ingangskanalen ingesteld/gewijzigd. De benodigde tijd voor deze bewerking zal men gezien het tijdcritische karakter zo kort mogelijk willen houden.

d) Na een vooraf aangegeven aantal tijdsintervallen stopt de analoge machine met rekenen.

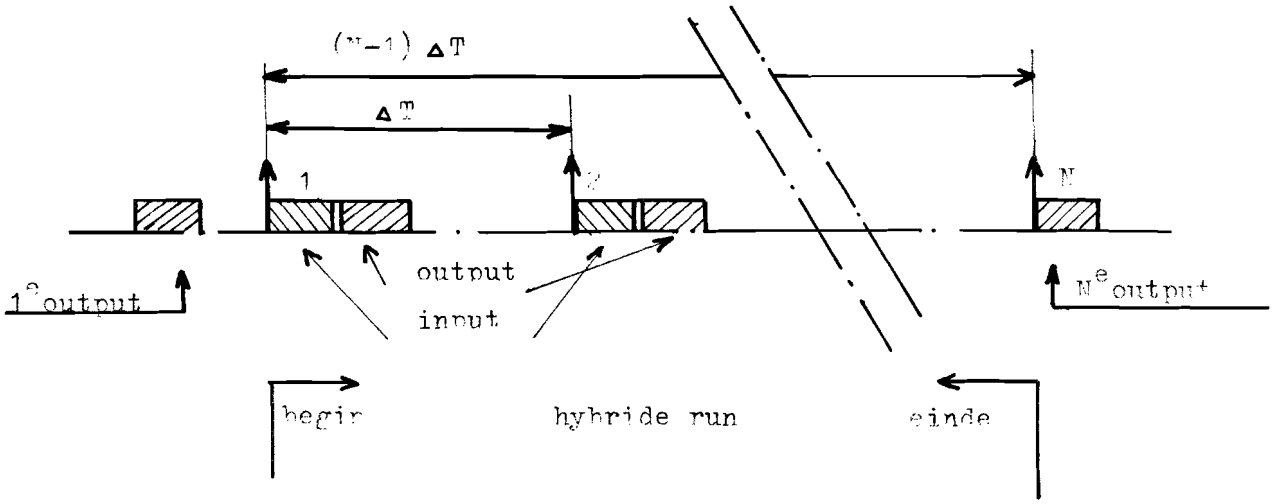


fig.8: tijddiagram hybride run

output: data van digitale naar analoge machine.
input : data van analoge naar digitale machine.

De klokpulsen die met vaste tijdsintervallen worden gegenereerd vormen de synchronisatie-basis voor de afloop van de hybride run. Iedere tijdcritische bewerking wordt uitgevoerd zodra een klokpuls komt. In de hybride interface wordt dan een interrupt gegenereerd die een interrupt service routine start. In deze routine zijn de instructies opgenomen voor de in- en uitlezing van de analoge kanalen. Om het aantal instructies zo laag mogelijk te houden zijn in de voorbereidingsfase van de hybride run reeds de adressen van input/outputkanalen en -data vastgelegd. In fig.9 op blz.20 is het verloop van een [illegible] programma met de routines "LOOP","IN","OUT" en"END"aangegeven.

De routines "LOOP", "OUT" en "IN" zorgen voor de opbouw van een

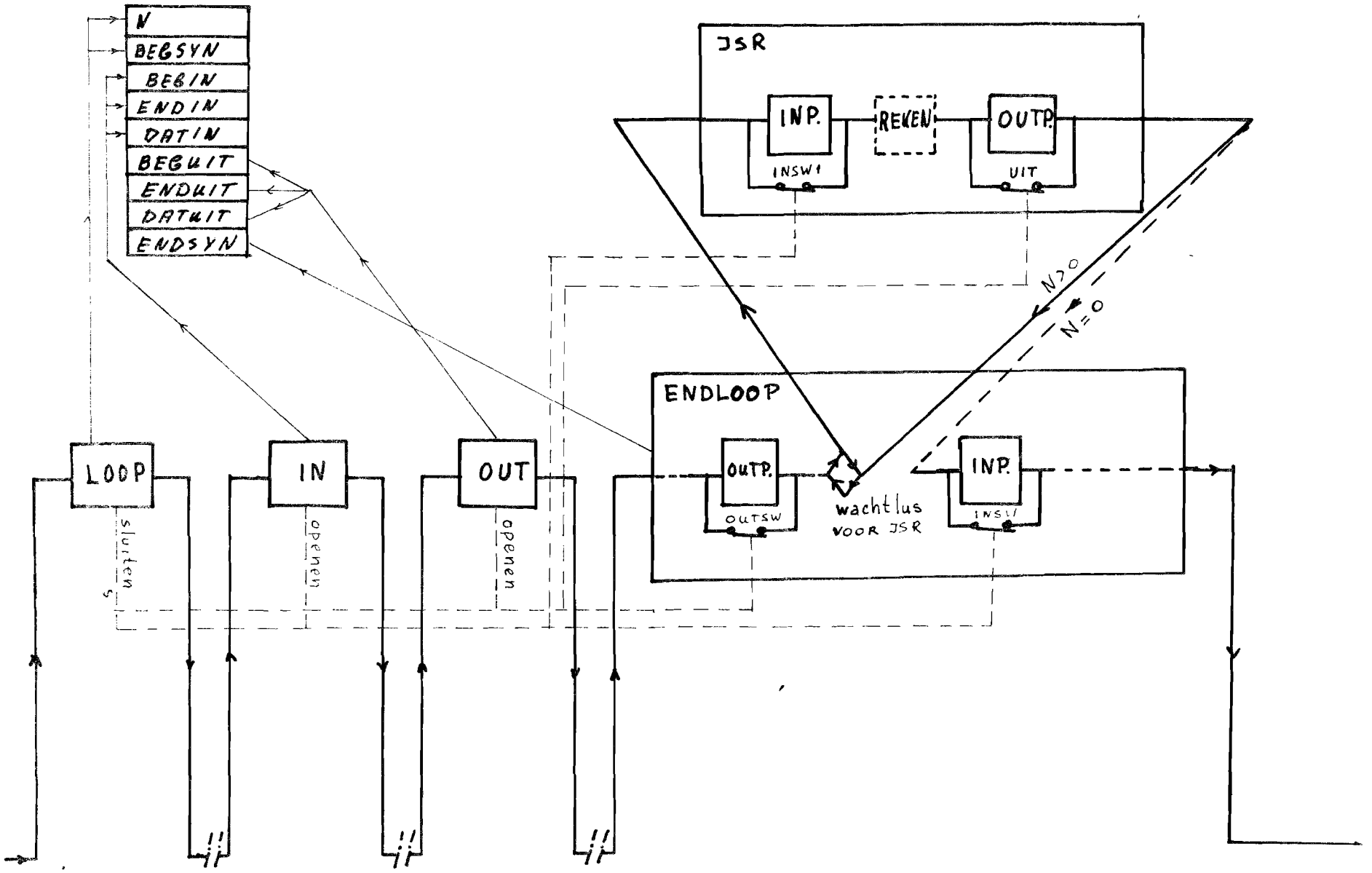


fig. 9: Verloop van een hybride run in een BASIC-programma.

tabel waarin de gegevens voor de uitvoering van de hybride run zijn opgeslagen. De tabel omvat de volgende elementen

N	:	aantal klokpulsen tijdens de run
BEGSYN	:	data voor het synchronisatieregister van de hybride interface bij de eerste klokpuls (zie flowchart I-1)
ENDSYN	:	dezelfde functie als BEGSYN maar nu bij de laatste klokpuls. Deze data bepaalt o.a. de mode waarin de A.C. na de klokpuls gaat
BEGIN	:	adres eerste inputkanaal
ENDIN	:	adres laatste inputkanaal
DATIN	:	adres inputdata
BEGUIT	:	adres eerste outputkanaal
ENDUIT	:	adres laatste outputkanaal
DATUIT	:	adres outputdata.

De routine "LOOP" (A.C. mode, N (,ACR)) levert de gegevens voor N en BEGSYN.
Het eerste argument specificeert de mode van de analoge machine bij de eerste klokpuls. Men dient een 3/4 HC-naam op te geven waarvan de eerste letter een C(ompute), of een H(old) of een R(eset) is.

Het argument N (aantal klokpulsen) kan als een numerieke expressie opgegeven worden. Voor het derde element kan ieder als argument toelaatbaar symbool gebruikt worden. Bij opgave van een derde element wordt de start van de hybride run met het A.C. ready signaal gesynchroniseerd.

De routines "IN" en "OUT" zijn belast met de opgave van adressen van input/outputkanalen en -data. In feite zijn "IN" en "OUT" in assembler-code in één routine gecombineerd. De argument list bestaat uit drie elementen:

(array, kanaalnummer eerste kanaal, kanaalnummer laatste kanaal).

Er is aangenomen dat de in, resp. uit te lezen kanalen, numeriek en volgorde geplaatst zijn. De array is de opslagplaats van de data die van de geheugenadressen naar de kanalen of van de kanalen naar de geheugenadressen wordt overgebracht.

Opgave van de kanaalnummers kan als numerieke expressie plaatsvinden.

De instelling van de lengte van het tijdsinterval ΔT wordt uitgevoerd door de routine TINT (fractie,macht) Het klokregister van de programmeerbare klokeenheid wordt hierin op de op te geven waarde ingesteld. De lengte van het interval is

10 "macht" x "fractie" x 10 usec.

In de routine END LOOP, verder aangeduid met "ENDL" wordt de hybride run gestart en uitgevoerd. Voorafgaande aan de eigerlijke start wordt eerst nog het element ENDSYN van de tabel ingevuld. In de argument list van "ENDL" wordt hiertoe de mode opgegeven waarin de A.C. na de laatste klokpuls dient te gaan. Is ENDSYN van data voorzien dan worden de analoge uitgangskanalen op de beginwaarden ingesteld (1e output), vervolgens worden de interrupt-vektor adressen geladen met de adressen van de interrupt service routines "ISR" en "ERR". "ISR" zorgt voor de tijdcritische in- en uitlezing van de analoge kanalen. "ERR" geeft foutmelding bij een ADC of DAC-overload en/of timing fout en breekt de hybride run af. Timing fout betekent dat het opgegeven tijdsinterval te kort is. . Het synchronisatie-register van het hybride koppelapparaat wordt geladen met de data opgeslagen in BEGSYN. Het foutenregister wordt geladen zodat een mogelijke fout ook inderdaad een foutmelding geeft. Het programma gaat nu in een wachtlus totdat de eerste klokpuls optreedt. Bij het optreden van de eerste klokpuls wordt in de hybride interface een interrupt aangevraagd die de interrupt service routine "ISR" in uitvoering brengt. In deze "ISR"-routine wordt input en output gedaan in deze volgorde. Tussen deze twee handelingen kan een BASIC-routine opgenomen worden. Deze routine zal dan in assembler geschreven en in de BASIC-interpreter gelinkt moeten zijn. Aan het einde van "ISR" wordt N met één verlaagd en op nulwaarde getest. Via een RTI-instructie wordt de interrupt service routine verlaten en keert het programma terug naar de wachtlus in de routine "ENDL". Bij de volgende klokpuls herhaalt de procedure zich. Is de "ISR"-routine (N-1)-maal doorlopen dan wordt na afloop van "ISR" niet teruggesprongen naar de wachtlus in "ENDL" maar naar het adres volgend op de wachtlus. Het synchronisatieregister van de hybride interface wordt nu geladen met de

inhoud van ENDSYN. Zodra de volgende klokpuls komt wordt de hybride run gestopt en - zo gewenst - de analoge uitgangskanalen gelezen (laatste input).

In het programma zijn een viertal "switches" opgenomen die de volgende functies hebben. De switches $INSW_1$ en INSW worden geopend bij de uitvoering van de routine "IN". Wenst men tijdens de hybride run geen input-routine "IN" is niet in het programma opgenomen-dan blijven de schakelaars $INSW_1$ en INSW dicht en worden de programmadelen die normaliter met de uitvoering van de input-handling belast zijn overgeslagen. Via dezelfde constructie kan men via het wel of niet opnemen van de routine "OUT" aangeven of men wel of geen output wenst. Nu vervullen de schakelaars OUTSW en OUT dienovereenkomstige functies. In de routine "LOOP" worden de schakelaars gesloten,zodat een vaste beginpositie gewaarborgd is.

In appendix I zijn de routines "LOOP", "IN", "OUT", "ENDL", "TINT" en de twee interrupt service routines "ISR" en "ERR" in flowchart vorm weergegeven met een meer gedetailleerde toelichting. De programma tekst van de routines is eveneens in appendix I opgenomen.

In het voorgaande is geschetst hoe de uitvoering van een hybride run met een aantal routines tot stand komt. De rekenvorm die gekozen is, is parallel hybried rekenen; de analoge computer staat tijdens de volledige run in de rekenmode. Alternerend hybried rekenen is met behulp van de besproken routines eveneens ook mogelijk. Beschouwt men namelijk een hybride run met de lengte van één tijdinterval als onderdeel van een alternerende hybride run dan zal periodieke herhaling van de run met de duur van een tijdinterval leiden tot de alternerende rekenwijze. De routines LOOP-(OUT-IN) en ENDL worden programmatisch herhaald. Tussen iedere cyclus van de routines kan men nu in BASIC berekeningen uitvoeren.

5.3 Programma-prioriteit tijdens de hybride run

De aanvraag voor een interrupt bij een klokpuls geschiedt op prio-

riteitsniveau 6(dit is de hardware prioriteit). De software prioriteit van de interrupt service routine "ISR" is 7 (hoogsteprioriteit). De aanvraag voor een interrupt bij een fouttoestand in de hybride interface gebeurt op niveau 7. Dit betekent dat als de interrupt service routine "ISR" in uitvoering is, deze niet onderbroken wordt bij het optreden van een fout. Pas als "ISR" volledig afgehandeld is wordt de eventuele aanvraag voor de interrupt service routine "ERR" gehonoreerd. De voordelen van deze methode zijn

a) volledige afhandeling van "ISR" betekent dat alle eventueel opgetreden fouten tijdens "ISR" gemeld worden
b) kortere interrupt service routine "ISR" doordat de bewaking van de programma-teller geen extra instructies vergt.

Als in BASIC een routine als "ENDL" wordt aangeroepen, dan wordt bij het begin van de routine een J(ump) S(ub) R(outine) instructie uitgevoerd.

JSR PC "ROUTINE"

De inhoud van de programma-teller wordt op de stack geplaatst en de programma-teller wordt geladen met het adres van deuit te voeren routine in dit geval "ENDL".Tijdens de uitvoering van "ENDL" kunnen voor twee interrupt service routinesinterrunt aanvragen gedaan worden. Wanneer de interrupt service "ERR" wordt aangevraagd, dan zal na afhandeling van de routine de hybride run afgebroken worden en het BASIC-programma worden voortgezet. De programma teller zal dus bij de terugkeer naar BASIC met het juiste adres geladen moeten worden. Bij iedere klokpuls wordt een interrupt aanvraag voor de interrupt service routine "ISR" gedaan. De fout-corditie waarbij een interrupt aanvraag voor "ERR" gedaan wordt, kan gedurende het totale tijdsverloop van "ENDL" zowel tijdens de afhandeling van "ISR", als daarbuiten optreden. Vindt dit plaatstijdens de uitvoering van "ISR", dan is het terug-keeradres naar het BASIC-programma, gezien t.o.v. het laatst gebruikte stack-adres, anders dan bij de toestand waarbij rechtstreeks uit "ENDL" een aanvraag voor "ERR" gedaan wordt. Om nu ingewikkelde zoek-procedures te vermijden, met in achtneming van de mogelijke voorgeschiedenis, is de software prioriteit van de interrupt service routine "ISR" zodanig gekozen dat onderbreking door een interrupt aanvraag voor interrupt service routine "ERR" niet mogelijk is In fig.10 wordt de prioriteit van het programma in de routine "E[illegible]" aangegeven.

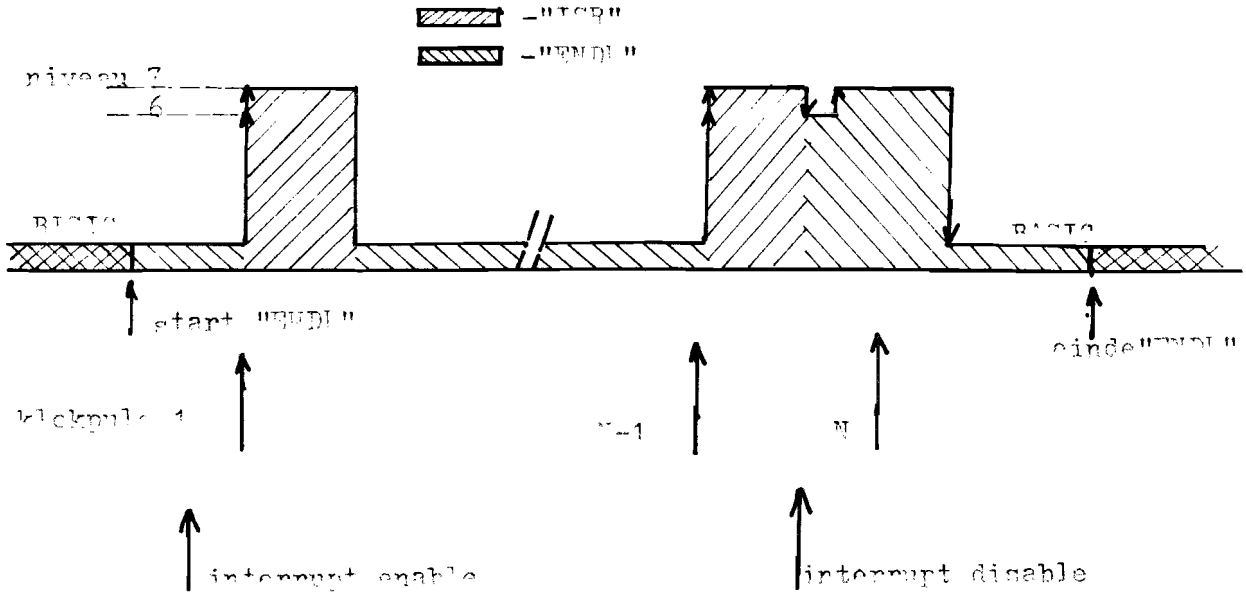


fig.10: programma prioriteit tijdens hybride run.

Na de voorlaatste klokpuls wordt na afhandeling van de interrupt service routine "ISR" niet meer teruggesprongen naar de wachtlus in "END!". Het optreden van de laatste klokpuls wordt dan gedetekteerd door te testen of een bepaald bit in het statusregister van het hybride koppelapparaat wordt opgezet. Doordat nu geen interrupt-aanvraag met hoge prioriteit plaatsvindt, wordt de prioriteit van het lopende programma op 7 gehandhaafd. Zodra via het controle bit in het statusregister geconstateerd is dat de laatste klokpuls is geweest, wordt de hybride run afgesloten met de laatste input. De routine "END!" is afgehandeld en via een RTS-PC instructie wordt het BASIC-programma voortgezet. Na afhandeling van de interrupt service routine "ISR" bij de voorlaatste klokpuls wordt de prioriteit van "END!" even teruggezet op 6 om een eventuele interrupt-aanvraag voor de interrupt service routine "ISR" te effectueren.

5.4 Tijdsduur van de interrupt service routine "ISR"

De tijd - nodig voor de uitvoering van routine "ISR" - bepaalt met welke maximale frequentie de in- en uitlezing van de analoge data

kan plaatsvinden. Door de benodigde tijd voor de uit te voeren instructies op te teller is de totale tijdsduur van "ISR" te bepalen. Zonder rekenroutine is de duur van "ISR" afhankelijk van twee factoren:

1. het wel of niet plegen van input of output.
2. het aantal gebruikte in- en uitgangskanalen.

De executie tijd van "ISR" is gelijk aan 80+Nx27+Mx18 usec. indien men de voor de PDP 11-instructies opgegeven executie tijden hanteert

N: aantal input-kanalen
M: aantal output-kanalen.

Als controle op bovengenoemde formule is met behulp van een programma nagegaan bij welk tijdsinterval een foutmelding wordt gegeven als functie van het aantal kanalen dat bij de bewerking betrokken is. In het programma wordt het tijdsinterval tussen twee klokpulsen steeds met 10 usec. vergroot. De spreiding in de gemeten tijd bedraagt dus maximaal 10 usec. In fig. 11.is het resultaat van de meting weergegeven voor drie situaties

1. in- en output
2. alleen input
3. alleen output.

In iedere situatie wordt het aantal kanalen van een tot zestien opgevoerd. De gemeten tijd nodig voor het uitvoeren van "ISR" is ca. 10 % korter dan de theoretisch bepaalde waarde. Men dient bij dit resultaat wel te bedenken dat de spreiding van de executie tijd per instructie minstens 10% kan bedragen.

5.5 Routines voor directe besturing van de mode van de analoge computer

Programmering van de mode-besturing is opgenomen in de routines "RESE" (reset), "COMP" (compute), "HOLD" (hold), "POTS" (potset) en "ALLR" (allreset).

De rekentoestand, waarin de analoge machine gestuurd wordt is afhankelijk van de data in register MODE AC van het bijgeleverde koppelapparaat. De tabel geeft de data-inhoud van het register MODE AC voor de verschillende modes.

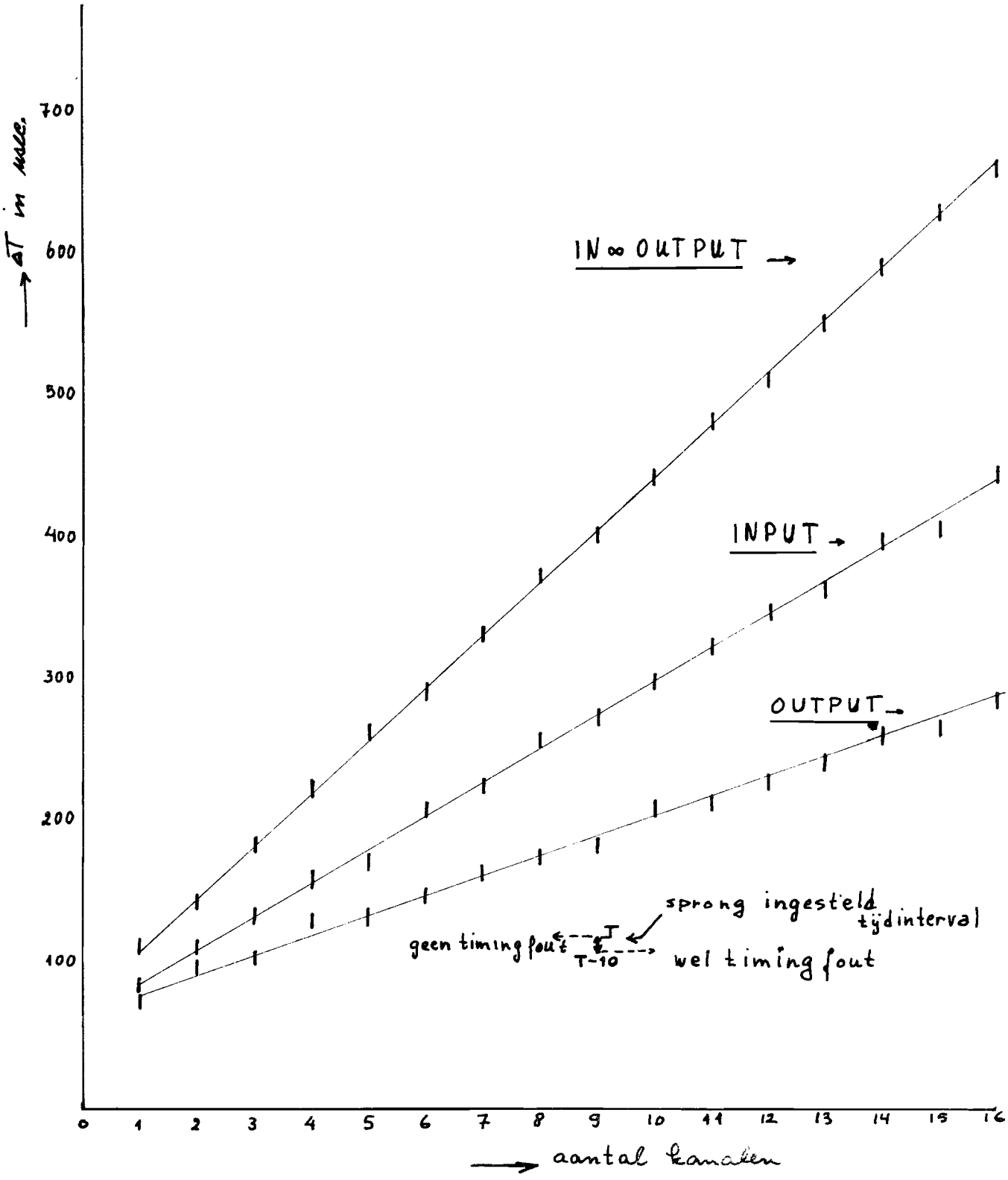


fig.11: Duur van "ISR" als functie van het aantal in- en outputkanalen.

Data	Mode
4	reset
1	compute
2	hold
10	potset
20	allreset

Uitvoering van een statement als CALL "RESE" zet de analoge machine in de mode reset.

5.6 Routines voor het in/uitlezen van de analoge kanalen

5.6.1 Routine "PUT"

De routine "PUT" (data, kanaalnummer) plaatst de in "data" opgegeven waarde op het onder "kanaalnummer" gespecificeerde ingangskanaal van de analoge machine. Bij de opgave van de gewenste spanningswaarde op het analoge kanaal dient men rekening te houden met de omzetting van digitale naar analoge waarde. Een in "data" opgegeven decimale waarde van ± 16384 komt overeen met een analoge ingangsspanning van ± 100 volt. Wil men dus dat de analoge ingangsspanning op een bepaald kanaal U-volt bedraagt, dan dient de decimale waarde van het getal onder "data" gelijk te zijn aan 163,84 x U. Het "kanaalnummer" kan als numerieke expressie worden opgegeven. In de routine wordt getest op een mogelijke "DAC-overload". Treedt deze fout op dan wordt dit aan de gebruiker gemeld en het lopende BASIC-programma wordt afgebroken met de melding ? Arg. at line ..... Doorgaan van het programma wordt niet als zinvol gezien omdat men duidelijk een foutief startsignaal aanbrengt.

5.6.2 Routine "GET"

De routine "GET" (datanaam , kanaalnummer) leest het onder "kanaalnummer" opgegeven analoge uitgangskanaal uit en plaatst de betreffende data in de onder "datanaam " opgegeven variabele. Hierbij kan de onder

"data-naam" opgegeven variabele een numerieke scalar of een array-element zijn. Het "kanaalnummer" kan weer als numerieke expressie worden opgegeven. Wil men de analoge spanningswaarde uit laten printen, dan zal men weer rekening moeten houden met de gepleegde analoog-digitaal conversie. Stel "data-naam" = B1. Dan is de analoge spanningswaarde op het uitgelezen kanaal $\frac{B1}{163,84}$ volt. In de routine wordt getest op een mogelijk optreden van "ADC overload". Zo ja, dan wordt dit aan de gebruiker gemeld maar het BASIC-programma wordt niet afgebroken. Dit blijft ter beoordeling van de gebruiker zelf. Het kan namelijk zijn dat bij wel afbreken van het programma het uitprinten van voor de gebruiker zinvolle data van andere uitgangskanalen wordt afgebroken.

5.7 Routines voor de digitale in- en outputkanalen

5.7.1 Routine "CNTR"

Voor de digitale besturing van componenten in de analoge machine beschikt de analoge machine over 16 digitale outputkanalen. Deze 16 kanalen bestaan uit de 16 bits van een woord. Via de routine "CNTR" kan een bepaald bit van dit woord 1 of 0 gemaakt worden.
Aanroep van de routine: CALL "CNTR" (data, kanaalnummer). Het argument "data" bepaalt het hoog of laag zijn; het "kanaalnummer" geeft aan voor welk digitaal output-kanaal dit geldt. "Data" zowel als "kanaalnummer" kunnen als numerieke expressie worden opgegeven. Voor "data" geldt dan wel dat de numerieke expressie in waarde gelijk moet zijn aan 1 of 0.

5.7.2 Routine "SENS"

Voor het inlezen van data in digitale vorm zijn in het hybride koppelapparaat 16 digitale inputkanalen aanwezig. Ook nu is de informatie over de toestand van deze kanalen opgeslagen in een 16 bits-woord. Via de routine "SENS" kan voor een bepaald bit (= kanaalnummer) worden nagegaan of dit hoog dan wel laag is. De waarde 1 of 0 wordt opgeslagen

in het argument "data-naam" van "SENS". CALL "SENS" (X, 5) leest digitaal kanaal 5 en bergt in de BASIC-variabele X de waarde 0 of 1 op.

6 HET GEBRUIK VAN DE TEKTRONIC 4010-TERMINAL ALS GRAPHIC DISPLAY IN BASIC

Om via BASIC-statements gebruik te kunnen maken van de graphic mode van de display-terminal zijn de routines "PLOT" en "ALFA" ontwikkeld. Bij de opzet van deze routines is gebruik gemaakt van de reeds bestaande software voor het gebruik van de Tektronic 4010 (zie lit.20). De routine "PLOT" (plot mode, X-coordinaat, Y-coordinaat) voert afhankelijk van de waarde van het argument "plot mode" de volgende functie uit.

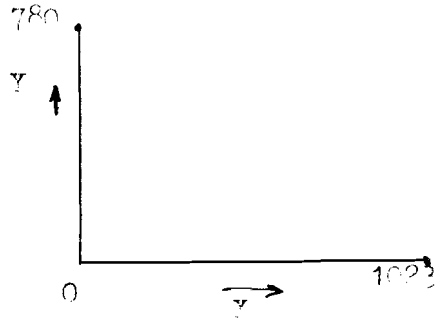
a) "plot mode" = 0
In deze uitvoering wordt de terminal in de lineaire interpolatie mode gezet en de grafische cursor geplaatst in het onder X en Y opgegeven beginpunt.

b) "plot mode" > 0
Op het beeldscherm wordt nu een lijn getrokken van de voorgaande coordinaten naar de nieuw opgegeven coordinaten. Aan een eerste aanroep met "plot mode" > 0 dient steeds een aanroep van "PLOT" met "plot mode" = 0 vooraf te gaan.
Het programma

```
1 LET  A = 0    LET  B = 1
2 CALL "PLOT" (A, X0, Y0)
3 CALL "PLOT" (B, X1, Y1)
4 CALL "PLOT" (B, X2, Y2)
```

tekent twee rechten van het punt X_0, Y_0 via X_1, Y_1 naar X_2, Y_2.

Bij gebruikmaking van de graphic mode van de display is het beeldscherm opgedeeld in 1024 X en 781 Y-eenheden.



De gebruiker zal hiermee bij de opgave van X- en Y-coordinaat rekening moeten houden.

c) "plot mode" < 0

In het opgegeven punt wordt op het beeldscherm een punt met verhoogde intensiteit getekend.

Terugkeer van de graphic mode naar de alfanumerieke mode is mogelijk via de routine "ALPA"(tekst-string). In deze routine kan als "optional" een tekst-string als argument meegegeven worden. De display van deze tekst-string start vanaf het punt dat in de voorgaande "PLOT"-routine is opgegeven. Tekst bij X-en/of Y-as kan op deze wijze worden bijgeschreven.

7 GEBRUIK VAN DE FLUKE-DIGITALE VOLTMETER VIA PROGRAMMERING

Om in de hybride installatie nauwkeurige spanningsmetingen te kunnen doen, is een digitale voltmeter ingebouwd. De absolute fout van deze meter bedraagt op het 10 V en 100 V schaalbereik bij een ingangsspanning Vi

$|\Delta Vi| \leq 0{,}04 \times 10^{-2} \times Vi + 0{,}001 \times 10^{-2} \times F.S.$ volt

Naast het gebruik als instrument met visuele indicatie, lag het voor de hand om deze nauwkeurige meetbron ook via software te willen bedienen en uit te lezen. Zodra deze voorziening gerealiseerd is kan men de digitale voltmeter in een programma o.a. gebruiken voor:

a) het uitlezen van versterkers op de analoge machine.
b) het controleren van de juistheid van instellingen van servopotentiometers.
c) het uitvoeren van metingen in testprogramma's van de analoge componenten van de analoge machine.

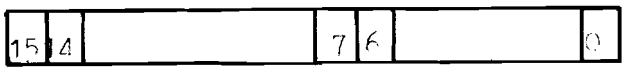
Caron (zie lit. 4) heeft indertijd al aangetoond dat een uitspraak over juiste werking van o.a. de A/D - converter met de daaraan gekoppelde S/H versterkers en de D/A converters pas gedaan kan worden bij het gebruik van een meetbron van voldoende nauwkeurigheid. Metingen van gelijkspanningen via de A/D converter, voldeden niet wegens het geringe oplossende vermogen van de A/D converter.

7.1 Koppeling aan de Uni-bus

Om de Fluke digitale voltmeter via programmering te kunnen bedrijven, is het meetinstrument via een interface gekoppeld aan de Uni-bus van de PDP11-minicomputer. In deze interface bevinden zich een aantal registers die via PDP11-assembler instructies te adresseren zijn. Deze registers omvatten:

a) Status register; bevat informatie over de bedrijfstoestand van de D.V.M. (Digitale Volt Meter). De relevante bits met hun functies

zijn hieronder benoemd.



adr. = 166070

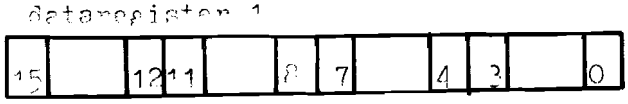
bit 6: interrupt enable bit.

bit 7: interrupt (of klaar) bit; Als dit bit geset wordt is de D.V.M. klaar met de uitvoering van een meting en kan de data uitgelezen worden.

bit 14:overload-bit; wordt geset bij overload.

b) Data-registers. Deze registers bevatten de vijf digits van de meetwaarde, opgeslagen in een BCD-code.

dataregister 1



adres 166074

5e dec. 4e dec. 3e dec. 2e dec.

dataregister 2

adres 166076

1e dec.

Per 4 bits van dataregister 1 zijn de 2e t/m 5e digit-waarde opgeslagen. In de bits 0 t/m 3 van dataregister 2 is de 1e digit van de meetwaarde opgeslagen. De BCD-codering van de digitwaarden wordt via programmering omgezet naar twee integer getallen die de cijfers van de meetwaarde voor en achter de comma representeren.

c) Register voor instelling van de meetcondities.

adres 166062

bit 2 t/m 7; meetbereik van 0,1 volt tot 10000 volt steeds stijgend met een factor 10.

bit 8: wisselspanningsmeting

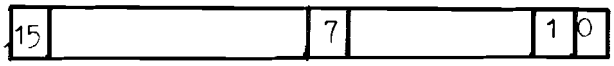
bit 9: kΩ- meting

bit 10: Ω - meting

bit 11: externe referentie-bron
bit 12: filter ingeschakeld

Als de bitwaarde 0 is, is het opgegeven bereik (instelling) bij die bit gekozen.
Zijn de bits 2 t/m 7 allemaal 0 dan wordt het juiste meetbereik door de meter zelf bepaald. (automatic range)

d) Register voor uitlezing van de externe instelfuncties.



adres 166072

bit 0: instelling remote. Bit = 1 als op het bedieningspaneel van de meter de knop remote ingedrukt is.
bit 1: synchronisatie van de sample-puls uit de interne of externe bron. Als voor gebruik in programma's de knop SAMPLE-RATE in de stand extern staat is de bitwaarde 1.
bit 7: in dit bit is de polariteit van de gemeten spanning aangegeven. Bij negatieve spanningswaarde is dit bit hoog. Dit is geen externe instelfunctie maar wordt tijdens de meting zelf bepaald door het meetinstrument.

7.2 Meting via de routine RDVM

In de onder BASIC aan te roepen routine RDVM is de meting via programmering geregeld. Uitvoering van het statement CALL "RDVM" (D_1, D_2) bepaalt de meetwaarde, vastgelegd in de twee integer getallen D_1 en D_2. De keuze om de meetwaarde in twee integer getallen vast te leggen is gedaan omdat een woordlengte van 16 bit niet toereikend is voor de representatie van een getal met vijf digits met waarden tussen 0 en 9. Max. integer getal $2^{15}-1 = 32767$. Het zestiende bit is voor het teken van het integer getal gereserveerd. D_1 bevat de twee of drie cijfers voor de komma, D_2 de drie cijfers achter de komma. Het teken van D_1 en D_2 wordt tijdens de meting bepaald uit de bitwaarde van bit 7 van het register op adres 166072 (zie vorig punt).

De meetwaarde in volts bedraagt dus $D_1 + 0{,}001 \times D_2$ volt.

Bij de uitvoering van de meting is gezien de grootte van de spanningswaarden op de analoge machine gekozen voor een meetbereik van 100 V - DC - met filter. Via de inschakeling van het meetfilter wordt de invloed van stoorsignalen gereduceerd; de meettijd per meetwaarde is echter vrij lang ca. 0,5 sec. In de routine wordt getest of de gebruiker de meter in de vereiste bedieningsstand heeft gezet. Is dit niet het geval, wat niet of foutief meten tot gevolg zal hebben, dan wordt aan de gebruiker een melding op de teletype gegeven. Deze boodschap vermeldt dat de remote-knop ingedrukt moet zijn en de knop SAMPLE-RATE op extern geplaatst. Voor gebruiker-aanwijzigingen, listing en flowchart van routine RDVM zie de diverse bijlagen.

7.3 Koppeling van de DVM aan de analoge machine

De digitale voltmeter kan op de analoge machine extern met de select-bus gekoppeld zijn. Aan de select-bus van de A.C. kan intern een versterker uitgang of een middencontact van een relais gekoppeld zijn.

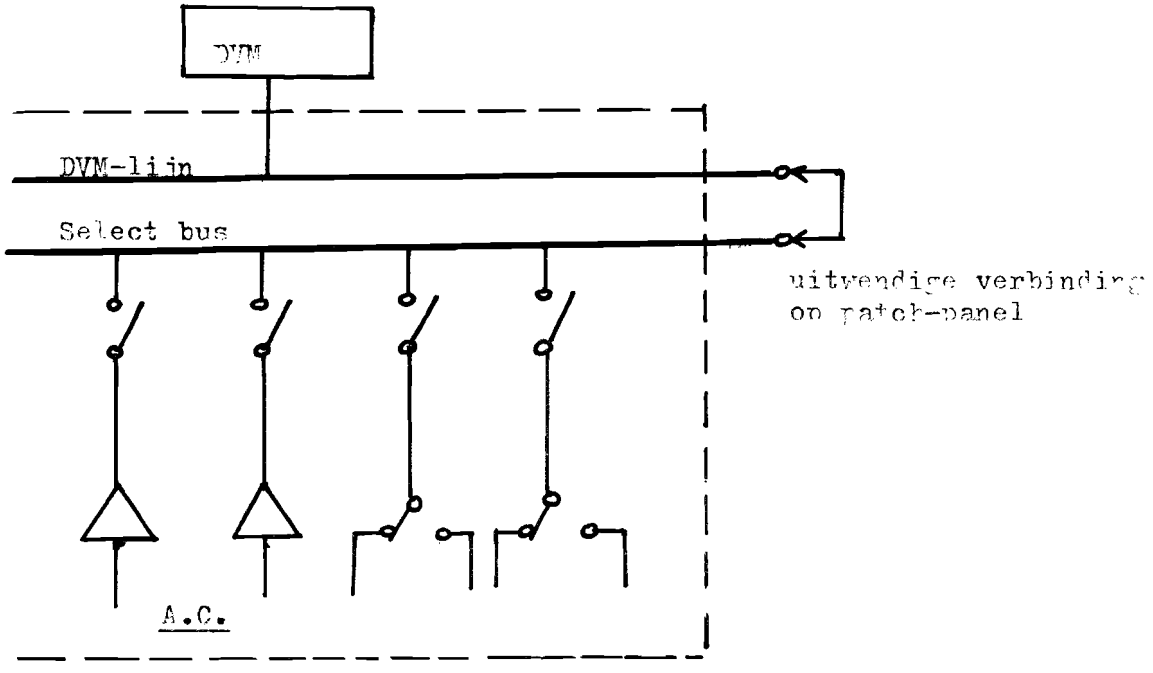


fig. 14: koppeling DVM aan de A.C.

De koppeling van een component aan de select-bus wordt tot stand gebracht via de adres selector unit. Deze unit heeft tot taak het selecteren van een versterkersuitgang of middencontact van een relais. Na selectie wordt de component intern op de select-bus van de analoge machine geschakeld. De adres selector unit bevindt zich in de interface tussen analoge en digitale machine. Het adres van de gekozen component wordt in het amplifierselect register (adres 173612) gebracht en de hardware in de amplifier select eenheid schakelt het overeenkomstige relais dat de geadresseerde component met de select-bus verbindt. Wil men de select-bus intern weer vrij maken dan is het voldoende om in het status register van de hybride interface bit 5 te resetten.

Programmatisch is het selecteren van een versterker of middencontact van een relais verwerkt in de routine "SLEC"(adres). "Adres" is een van de toegelaten adresnummers op de twee consoles van de analoge machine. Voor console A is dit 0 t/m 69, voor console B 100 t/m 169. In de routine SLEC is een pauze opgenomen om de schakeltijd van het keuze-relais te overbruggen. Het vrijmaken van de select-bus geschiedt via de aanroep van "SLEC" zonder argument. Hierdoor kan de hybride gebruiker de select-bus gebruiken zonder dat er een element aangekoppeld is via een voorafgaande "SLEC"-routine.

Voor het selecteren en instellen van servopotmeters op console B is de routine "SPOT" geschreven. De loper van een geselecteerde servopotentiometer is intern gekoppeld aan de DVM-lijn van de analoge machine. "SPOT" (adres, waarde) zet de onder adres opgegeven potmeter op de onder "waarde" opgegeven instelling. De adressering wordt qua hardware weer uitgevoerd door de adres selector eenheid. Nadat het adres van de in te stellen potmeter in het amplifier-select register is gezet, wordt de potmeter geselecteerd en ingesteld op de via kanaal 0 van de DAC toegevoerde analoge waarde. Bij het opgeven van "waarde" in de argumentenlijst dient men te bedenken dat een decimale waarde van 16382 overeenkomt met een analoge ingangsspanning van 100 volt. Alleen positieve waarden zijn geoorloofd. Via de routine "RDVM" kan de ingestelde waarde uitgelezen worden en aldus op zijn juistheid worden beschouwd.

8 HYBRIDE COMMUNICATIE-ROUTINES IN FORTRAN

8.1 Algemeen

Het hybride communicatie-pakket dat voor gebruik in BASIC-RT11 is ontwikkeld is zodanig aangepast dat ook in de FORTRAN-taal met de communicatie-routines gewerkt kan worden. Het RT11 - operating system is om in FORTRAN te kunnen werken uitgebreid met een FORTRAN-compiler en een FORTRAN-library. In deze library is naast specifieke reken en systeem routines een pakket routines opgenomen die nodig kunnen zijn voor de verwerking van de gecompileerde code. De hybride communicatie-routines zijn niet toegevoegd aan de FORTRAN-library maar in een aparte library opgenomen. Tijdens het linken van de object-modulen worden de routines, die nodig zijn voor de uitvoering van het programma, uit de library gehaald en opgenomen in de load module. De gecompileerde code van een FORTRAN source-programma is dan ook te zien als een aaneenrijging van aanroepen van routines uit de FORTRAN-library en routines uit andere object-modulen of libraries.

Bij gebruik van FORTRAN-RT11 vervalt het interactieve element dat bij BASIC-RT11 mogelijk was. Programma's on line veranderen is niet meer direct mogelijk. Bij programmawijzigingen zal men steeds weer de weg compileren - linken moeten doorlopen.

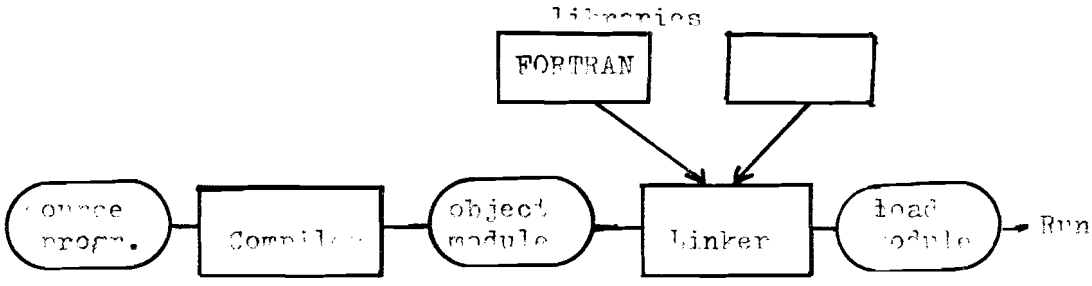


fig.13: opbouw van een FORTRAN load module

De opzet van assembler routines, onder FORTRAN aanroepbaar, kan in vergelijking met de assembler routines, aanroepbaar in BASIC,

vaak eenvoudiger zijn. In de "FORTRAN" assembler routine kunnen in vergelijking met de "BASIC" assembler routine de volgende functies vervallen.

a) syntax check op linkerhaak, comma en rechterhaak in een argumentlist. Deze check vindt in FORTRAN reeds in de compilatie-fase plaats.

b) omzetting van floating point waarden naar integer waarden. In FORTRAN kunnen de variabelen waarvoor dit gevorst is als integer worden gedeclareerd.

De adressering van de argumenten van een subroutine is bepaald door een wijzer in register R5 van de PDP11. Bij aanroep van een routine wijst R5 naar de volgende tabel.

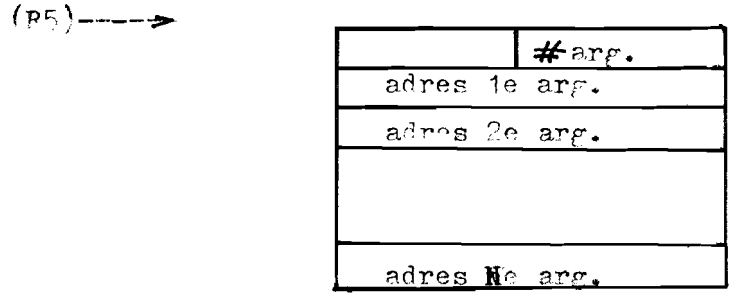


fig. 14: adressering van argumenten van een FORTRAN subroutine.

Bij aanroep van een assembler routine in BASIC is de adressering van argumenten in de routine via een offset-waarde t.o.v. het begin van de symbol tabel vastgelegd. Offset-waarde en beginadres van de symbol tabel zijn vastgelegd in register 1 resp. register 5 van de PDP11. Adressering en het toekennen van een waarde aan de variabele vereist dan ook in een "BASIC"-assembler routine meer instructies. Resumerend kan men zeggen dat assembler routines voor gebruik in FORTRAN veelal korter en eenvoudiger zijn wat betreft instructie-set, dan voor de overeenkomstige BASIC assembler routines.
De BASIC-interpreter beschikt over een systeem routine die in geval van een foutief argument een foutmelding op de teletype geeft met daarin opgenomen in welk regelnummer zich de fout bevindt.
FORTRAN-RT11 beschikt niet over deze mogelijkheid. Bovendien bevatten

niet alle statements een regelnummer. In de assembler routines, waar de check op een juiste waarde van het argument wordt uitgevoerd, wordt bij een foutief argument een branch naar een foutmeldingsroutine gemaakt. In deze routine wordt dan de aanroep van een FORTRAN subroutine voorbereid en uitgevoerd. Deze FORTRAN subroutine geeft dan op de teletype een foutmelding:

"Hybride fout # n"

n is een getal dat de gebruiker informeert in welke hybride routine de fout aanwezig is. Het getal n wordt als argument aan de FORTRAN subroutine meegegeven. In appendix III blz.16 en 17 is de waarde van n voor de diverse routines aangegeven.


```
fout: BR      FOUT

FOUT:  MOV n, ERRHYB
       JMP ERRARG (of JSR PC ERRARG)

ERRARG: MOV  TABEL,R5      ; link adres
        JSR PC, HYBERR     ; FORTRAN routine
        RTS PC

TABEL: .BYTE  0,1          ; één argument
       .WORD  ERRHYB       ; adres van argument

ERRHYB: .WORD  0
```

HYBERR is de FORTRAN subroutine met als argument een integer variabele met de waarde van n. Afhankelijk van de waarde van n kan men het programma stoppen of vanuit de routine HYBERR weer terugkeren naar de routine ERRARG. De volgende instructie in deze routine is een JSR PC - instructie waardoor in het oorspronkelijke programma, de routine verlaten wordt waar-

in de fout optrad. In de interrupt-service routine ERR, die aangeroepen kan worden bij een fout tijdens de uitvoering van een tijdcritische hybride run, dient na de foutmelding weer teruggesprongen te worden naar de routine zelf. Bij de label FOUT wordt dan niet JMP- maar JSR PC- ERRARG uitgevoerd.

8.2 Samenstelling van de library

De hybride communicatie routines zijn ondergebracht in een library-file HYBLIB. De library-vorm heeft als voordeel dat tijdens de link-fase van een FORTRAN-programma alleen die object-modules in de load file worden opgenomen die de in dat programma aan te roepen routines bevatten. In het link commando kan HYBLIB als een input file opgegeven worden.
De samenstelling van de routine-set is vrijwel gelijk aan die van het pakket dat bij BASIC gebruikt wordt. Er zijn slechts twee kleine verschilpunten. Bij de routines LOOP en MEDI is als een argument de mode van de analoge machine aangegeven. In BASIC kan men in de routine opgeven de letter C, H of R. In FORTRAN wordt de integer variabele MODE ongegeven die vooraf geinitialiseerd moet zijn op de waarde 1 (compute)
of 2 (hold)
of 3 (reset).
Andere waarden geven een foutmelding. Verder is bij de digitale in/out routines SENS en CNTR de variabele "waarde" als logical variabele geharteerd.

CNTR (waarde, kanaalnummer)

"waarde" is geinitialiseerd als true of false.

In de library zijn verder nog opgenomen de routines voor de foutmeldirs ERRARG en HYBERR. Voor flowchart en listing zie appendix I.

Het gebruik en de toepassing van de hybride routines is dezelfde als bij BASIC. De tijdcritische in- en uitlezing van analoge kanalen wordt programmatisch weer uitgevoerd als

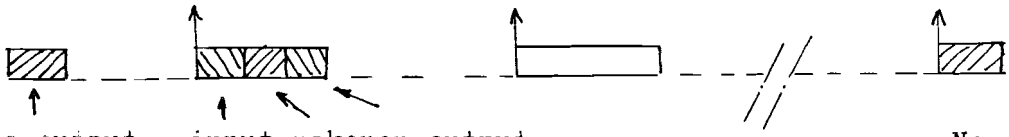
CALL LOOP
CALL OUT -voorbereidingsfase
CALL IN

CALL ENDI - uitvoeringsfase

In de routine ENDI worden de tijdcritische bewerkingen verricht in de interrupt service routine ISR. Het nadeel van deze methode is dat er in de tijdcritische uitvoeringsfase van de routine ENDI net als in BASIC niet gerekend kan worden; tenzij in een assembler routine. In de volgende paragraph wordt aangegeven hoe dit bezwaar ondervangen zou kunnen worden.

8.3 Tijdcritisch rekenen in FORTRAN

In FORTRAN zou rekenen wel tijdcritisch kunnen plaatsvinden in de FORTRAN-taal zelf. Om dit te kunnen realiseren zou een andere programma-organisatie van het hybride rekenproces nodig zijn. De voorbereidingsfase kan dan vervallen en de uitvoeringsfase wordt opgesplitst in een aantal subroutines.



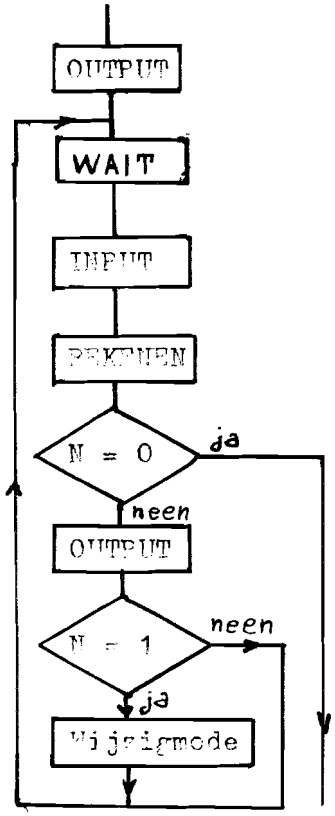
Bovenstaande tijdcritische bewerking zou als volgt in een FORTRAN-programma kunnen worden ondergebracht.

```
      CALL OUTPUT ;  1e output
      DO 10  I = 1,N
      CALL WAIT  ;  test voor klokpuls
      CALL INPUT
```

REKEN-STATEMENTS

```
      IF  (N.EQ.0)  GO TO 10
```

```
      CALL  OUTPUT
      IF  (N.EQ.1 ) mode = hold
10    CONTINUE
```



OUTPUT is een assembler routine die de analoge ingangskanalen van data voorziet.

INPUT is een assembler routine die de analoge uitgangskanalen inleest.

WAIT is een assembler routine die test op het optreden van een klokpuls en dan het programma vervolgt.

De INPUT en OUTPUT routine zullen in deze opzet wel meer instructies bevatten dan de interrupt-service routine ISR voor de data-verwerking van de analoge in- en uitgangskanalen. De uitvoering van de rekenstatements geschiedt echter veel sneller dan in een BASIC-programma, dat iedere statement van uit de source-tekst nog moet interpreteren.

9 TESTPROGRAMMA VOOR DE ANALOGE MACHINE EN INTERFACE

In de huidige hybride installaties wordt voor meerdere doeleinden test-software ontwikkeld. Hierdoor is men in staat om

a) dagelijks een routine check uit te voeren, alvorens andere programma's uit te voeren. Uit de testgegevens kan men mogelijke defecten afleiden.

b) Door het vastleggen van de testgegevens in teststaten is men in staat zich een indruk te vormen van het verloop van componenten.

In de groep ER was voor de hybride configuratie IBM 360 - HITACHI 505 een testpakket geschreven in DOS-FORTRAN IV (zie lit. 4). Een redesign van dit pakket lag voor de hand. Bij deze redesign zijn de volgende punten van belang.

I De gewijzigde hybride configuratie PDP11 - HITACHI 505.

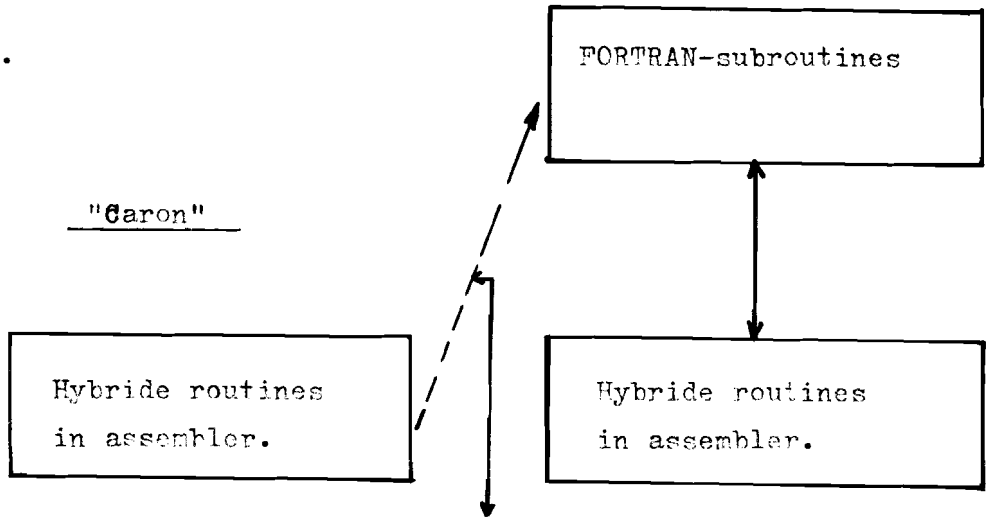
In het bestaande testpakket is via een aantal assembler routines de communicatie tussen de analoge en digitale machine geregeld. Voor de PDP11 - HITACHI 505 combinatie dienen deze routines aan de gewijzigde interface aangepast te worden. Voor deze aanpassing doen zich de volgende mogelijkheden voor:

a) laat de oorspronkelijke routines qua aanroep en argumenten intact en wijzig alleen de opbouw van de in assembler geschreven routine.

b) maak gebruik van de reeds aanwezige set hybride communicatie routines die onder FORTRAN aanroepbaar zijn. Dit zou betekenen dat een hybride routine in het Caron-pakket vervangen zou worden door een of meerdere routines uit de nieuwe set.

Aan beide methoden kleven nadelen. Methode a eist het schrijven van een stel nieuwe routines in assemblertaal. Methode b zal qua programmering veel wijzigingen vergen in het bestaande testprogramma. Een aangepaste vorm van methode b lijkt het meest bruikbaar. De oorspronkelijke set routines blijft qua aanroep en argumenten gelandhaafd maar is qua programmeerniveau een laag omhoog geschoven. In feite bestaat de nieuwe routine nu uit een FORTRAN-subroutine die de aanroep verzorgt van een of meer routines van het reeds onder FORTRAN-RT11 opgebouwde hybride communicatie

pakket.



De routines blijven qua naam en argumenten hetzelfde.

II De mogelijkheid om tests van een enkele component onder te brengen in interactieve testprogramma's in BASIC-RT11.

In het Caron-testpakket is voor de test van separate componenten de stand-alone feature ingebouwd. Dit was nodig omdat men in FORTRAN niet interactief kon werken. In de nieuwe hybride configuratie staat de BASIC-RT11 interpreter ter beschikking waarmee relatief snel een programma opgezet, uitgevoerd en zonodig gewijzigd kan worden. Korte, snel aan te passen, BASIC testprogramma's zouden de taak van de stand-alone test in het Caron pakket kunnen overnemen.

III Het gebruik van de Fluke digitale voltmeter.

Caron heeft bij de analyse van de meet-nauwkeurigheid in meerdere tests vastgesteld, dat er geen beslissende uitspraak is te doen over de nauwkeurigheid van de geteste component(en). In hoofdzaak is dit terug te voeren tot de grote meetfout die kan optreden bij de meting van analoge spanningswaarde via de ADC met S/H-O- versterker. De digitale voltmeter heeft een veel kleinere absolute meetfout. Meting van de spanningswaarde, waar mogelijk, via dit meetinstrument zal daarom een duidelijke verbetering van de testresultaten geven.

Appendix betreffende "Caron"-adaptaties

Nieuwe programma structuur
Listing nieuwe "hybride" routines
Wijzigingen in testschema's t.g.v. gebruik van Fluke-DVM.

10 SLOTBESCHOUWING

Met de toevoeging van de in het hybride communicatie-pakket beschreven routines aan de BASIC-RT11 interpreter is geprogrammeerde bediening van de analoge computer HITACHI 505 in een hogere computer-taal mogelijk geworden. Deze feature zal zonder meer de bedieningsflexibiliteit ten goede komen. Voor eenvoudige hybride programma's zonder tijdcritisch rekenwerk is gebruik van BASIC-RT11 toereikend. Het direct kunnen verwerken en modificeren van source-programma's in BASIC geeft de gebruiker een stuk flexibiliteit die bij de niet-interactieve taal FORTRAN-RT11 ontbreekt. De tijd, nodig voor programma-voorbereiding en test zal bij BASIC-. ten opzichte van FORTRAN-programma's, dan ook korter zijn. De executietijd en het geheugengebruik zal daarentegen bij FORTRAN door de directe verwerking van in object-code gestelde programma's voordeliger zijn. De kortere executietijd maakt tijdcritisch rekenen in FORTRAN mogelijk. Daartoe is een uitbreiding van het hybride communicatiepakket in FORTRAN noodzakelijk met de in punt 8.3 omschreven routines. De vraag in welke mate de nu ontwikkelde routines zullen voldoen, kan pas beantwoord worden indien ze functioneren in hybride gebruikersprogramma's. Ten aanzien van de testprogrammatuur is in dit stadium nog weinig te zeggen. De aanpassing van het "Caron"-pakket is gaande. Met de inpassing van de Fluke-digitale voltmeter zal een grotere nauwkeurigheid van o.a. de ADC- en DAC's-test bereikt kunnen worden. Bij het testen van de beschreven routines is men meermalen gehandicapt geweest door

a) nog onduidelijke fouten in de systeem software, waardoor het gebruikte operating system defect raakt.

b) storingen in de hardware van de analoge machine en diens interface.

Ten aanzien van dit laatste punt zal de testprogrammatuur een oplossing moeten bieden. Voor punt a zal de invoering van versie 2 van het RT11-operating system - zoker t.a.v. de detectie van de aard van de fout die de storing deed optreden - betere mogelijkheden bieden om tot een oplossing van dit probleem te geraken.

11 LITERATUUR

1- Beckert D,Liebig H en Wiesenthal P., Die Programmierung hybrider Rechenanlagen auf der Grundlage formeller Sprachen, Proceedings AICA-'70 blz 262 t/m 271.

2- Bekey and Karplus, Hybrid Computation, New-York, John Wiley&Sons, 1968.

3- Boers K. en Neggers K.A.M.,Een automatisch diagnostiek systeem voor een hybride computerinstallatie, afstudeerverslag T.H.E., jan 1972.

4- Caron P.M.,Testroutines voor de analoge rekenmachine en interface van de groen ER, afstudeerverslag T.H.E., sept. 1973.

5- Elzas M.S., PL1 or towards a unique language for all continuous system simulation, Proc. AICA-'73 blz. 65-70.

6- Elzas M.S., Present state of H.I.F.I.P.S.,a hybrid interactive formula interpreting programming system, Proc. AICA-'70 blz.690-695.

7- Franklin M.A. and Strauss J.C.,Automated programming of analog hybrid computers-a review, Simulation jan-1972 blz. 11-19.

8- Hambury J.N.& Barney G.C., The hybrid in control,Proc. AICA 1970 blz. 709-718.

9- Hambury J.N.& Barney G.C.,The components of hybrid computation, The computer bulletin, febr. 1970, blz. 31-36.

10- Hambury J.N.,Ironside J. and Barney G.C., An economical display system, The computer bulletin, sept. 1969, blz. 314-322.

11- Holst P.A., Hybrid- a user developed hybrid interpretive language Simulation Mei 1972, blz. 179-187.

12- Holtz I.F.E.M., Het hybride koppelapparaat tussen de PDP 11/20 en de Hitachi 505, afstudeerverslag T.H.E. sept. '73.

13- Hieber L., Anforderungen an Betriebssysteme von Grossrechenanlagen mit angekoppelten Analogrechner,Proc. AICA1970, blz. 719-723.

14- Verckhoffs C.J.H., De toepassing van de hybride rekenmachine bij het oplossen van technische en wetenschappelijke rekenproblemen, Informatie, juni 1973, blz. 310-319.

15- Korn en Korn, Electronic analog and hybrid computers,McGraw-Hill 1972.

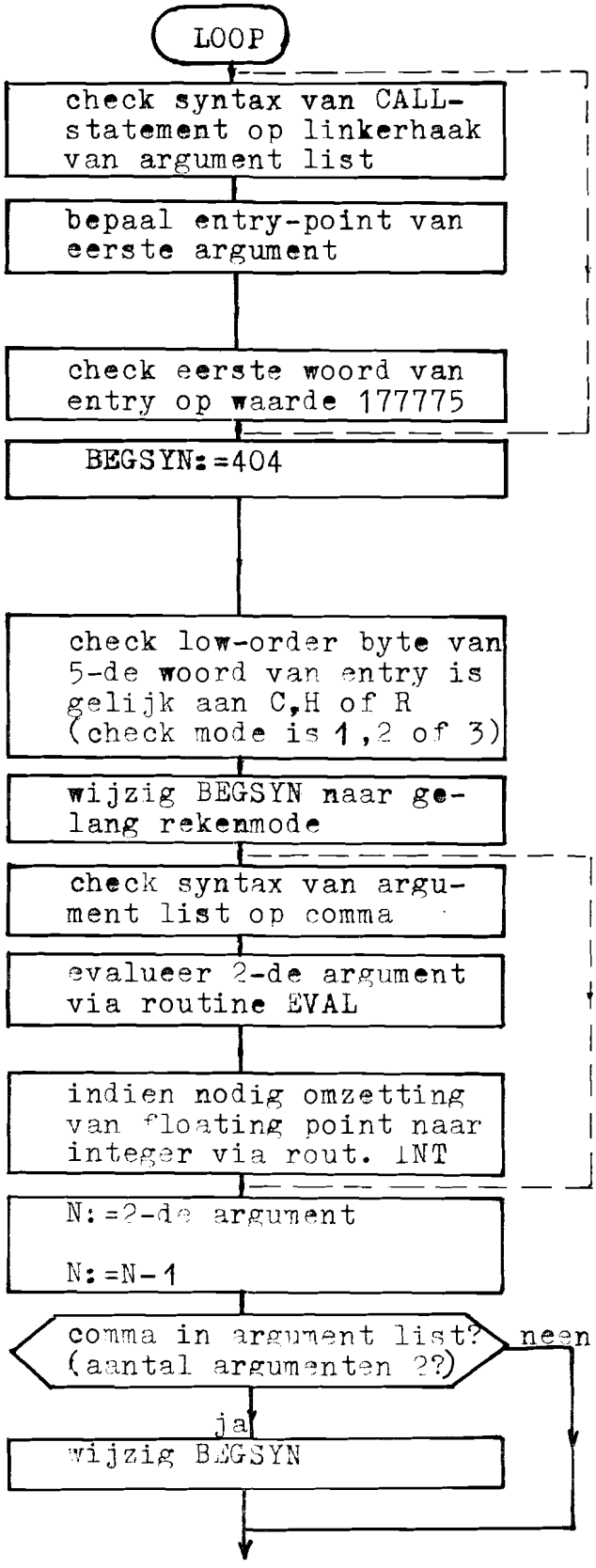
15- Manuels- BASIC/RT11-DEC-11-LBACA-A-D
16- FORTRAN/RT11-DEC-11-LFLRA-A-D
-LFCOA-A-D
-LRFPA-A-DN1
-LRFPA-A-D
17- RT11-System-DEC-11-ORUCA-A-D
18- Hitachi analog computer 505 operation manuel
Hitachi analog computer programming manuel
19- Diverse PDP11-handboeken
20- Tektronic 4010 computer display terminal
Tektronic 4010 software
21- Mulleneers J.J.M., assembler routines in BASIC, stageverslag T.H.E. juli 1974.
22- Schmidt W.E., Important considerations in procuring and implementing a hybrid computer, Proc. AICA 1970, blz. 273-278.
23- Schwarze K., Automatisches Skalieren und statischer Test mit dem hybriden Interpretor HOI, Angewandte Informatik, Teil 3 1972, blz. 127-140.
24- Zuidervaart J.C., De hybride rekenmachine, stand van zaken en toekomstige ontwikkelingen, Informatie juni 1973, blz. 304-310.
25- Woensdregt D.M.P., De koppeling van een PDP 11/20 computer aan het BURROUGHS B6700 systeem van de T.H.E., afstudeerverslag T.H.E. juni '73.

Deze appendix omvat de flowcharts+listings van de diverse hybride routines voor gebruik in BASIC en FORTRAN.

Het verloop van de routines, die in BASIC gebruikt worden, is als uitgangspunt voor de flowcharts gekozen. In dezelfde flowcharts is echter ook het verloop van de routines in FORTRAN aangegeven. De onderbroken lijnen, alsmede de tekst tussen haken in de diverse blokken geldt dan ook voor het verloop van de routines voor gebruik in FORTRAN.

Bladwijzer

LOOP(mode, aantal klokpulsen N (,A.C.R.))



Opmerkingen:

;Reg.1 bevat een wijzer naar linkerhaak-teken

;Reg.1 bevat een wijzer naar de offset van argument in user area. Reg.5 bevat indirect beginadres van user area.
;entry bevat 5 woorden, eerste woord=177775

;synchr. register van hybride interface geeft aan: enable interrupt op klokpuls zodra SYNREG met de inhoud van BEGSYN wordt geladen
;mode bij eerste klokpuls compute,hold of reset

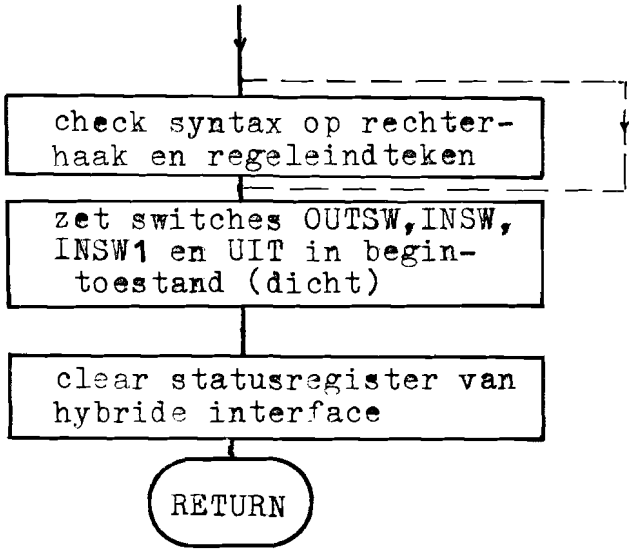
;bit 6,7 of 5 wordt geset in BEGSYN

;Reg.1 bevat wijzer

;waarde van 2-de argument wordt opgeslagen in floating point acc. FAC(1) en FAC(2)

;aantal klokpulsen in hybride run is N

;start hybride run op A(nalogc) C(omputer) R(eady) signaal



;via routines OUT en IN
kunnen switchen weer open
gezet worden

;voorgeschiedenis wordt
uitgeschakeld

```
        .PAGE
        .TITLE  HELP ,HYPRIDE & LEERZAME ROUTINES
        .SBTTL  CONSTANT SECTION
        .CSECT HYFUN2

        R0=%100                                                 BASIC
        R1=%101
        R2=%102
        R3=%103
        R4=%104
        R5=%105
        SP=%106
        PC=%107

        ;DUMMY CONSTANTES
        ERRREG  =173604
        SYNREG  =173602
        STTREG  =173600
        ERRVEC  =174
        ISRVEC  =170
        ERRPR   =340
        PS      =177776
        ISRPR   =340

        ;EINDE DUMMY'S


TABEL:
N:      .WORD   0               ;ACTUAL AANT
BEGIN:  .WORD   0               ;ADRES EERSTE INPUTKAN
ENDIN:  .WORD   0               ;ADRES LAATSTE INPUTKAN
DATIN:  .WORD   0               ;ADRES INPUTDATA
BEGUIT: .WORD   0               ;ADRES EERSTE OUTPUTKAN
ENDUIT: .WORD   0               ;ADRES LAATSTE OUTPUTKAN
DATUIT: .WORD   0               ;ADRES OUTPUTDATA
BEGSYN: .WORD   104004          ;SYNCHRO.REG VOOR EERSTE TP
ENDSYN: .WORD   104006          ;     ,,    ,,  ,,  LAATSTE TP
BEGERR: .WORD   102000334       ;ERROR.REG VOOR EERSTE TP
REKENX: .WORD   0               ;P.M.


        .PAGE
        .SBTTL  ROUTINE LOOP
        .GLOBL  LOOP
        .GLOBL  N,BEGIN,ENDIN,DATIN
        .GLOBL  BEGUIT,ENDUIT,DATUIT
        .GLOBL  BEGSYN,ENDSYN,BEGERR
        .GLOBL  REKENX                          ;P.M.
        .GLOBL  .LPAR,.RPAR,.COMMA,ERRARG,ERRSYN,INI,.EOL

;SUBROUTINE LOOP (MODEAC,N,ACn)
;MODEAC:        VARIABELE ALS NAAM IS
;               CX->AC IN COMPUTE
;               HX->AC IN HOLD
;               LX->AC IN RESET
;N:     AANTAL WORDS,WORDT INTEGER GEMAAKT
;ACn:           INDIEN HIER IETS OPGEGEVEN
;               WORDT,WORDT HET BEGIN MET ACn
;               GESYNCHRONISEERD
```

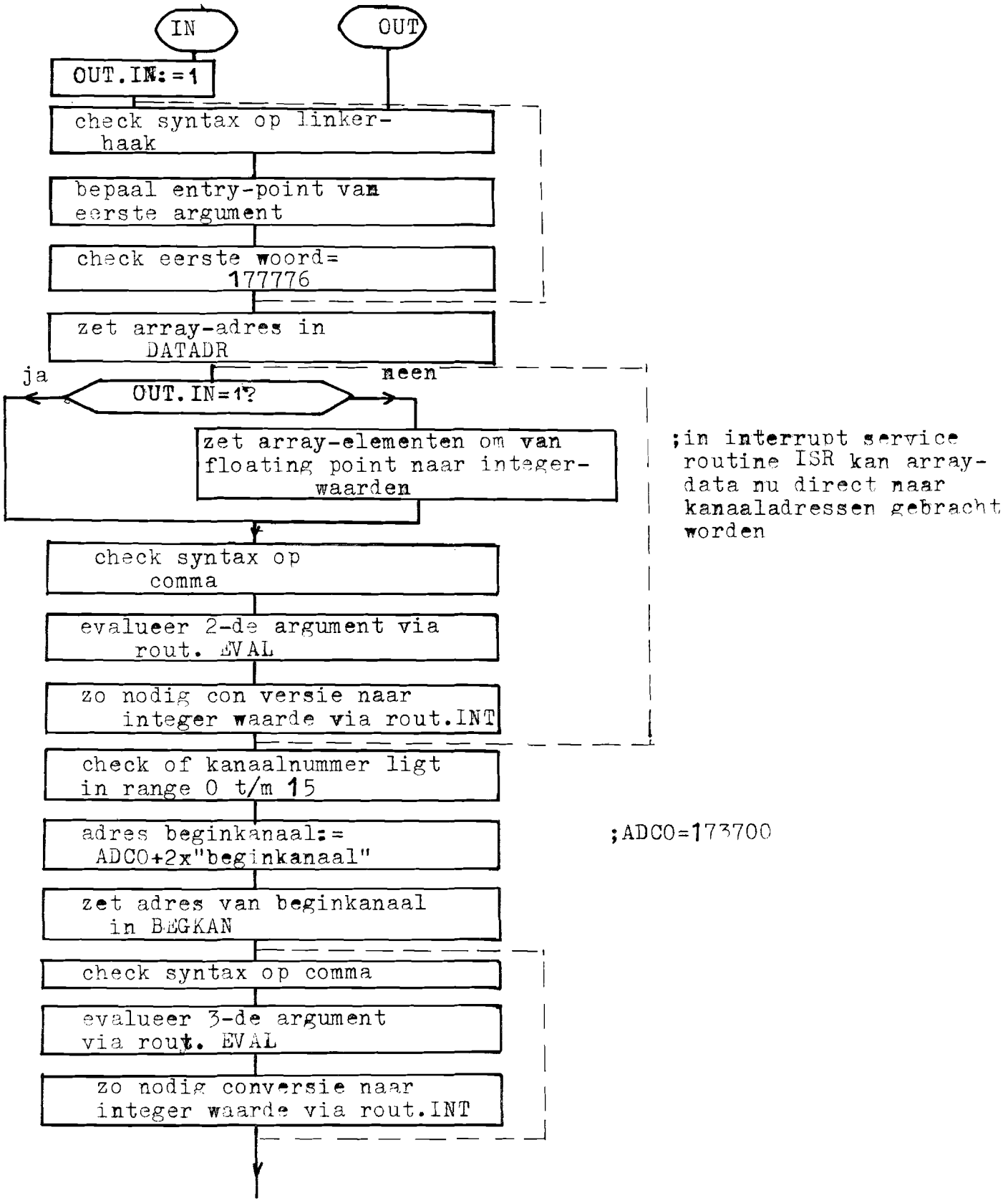
```
LOOP:   CMPB    (R1)+,#.LPAR
        BNE     LSYN
        MOVB    (R1)+,R2
        BMI     LSYN
        SWAB    R2
        BISB    (R1)+,R2
        ADD     (R5),R2

        CMP     #177775,(R2)+
        BNE     LARG
        MOV     #10404,BEGSYN
        MOV     6(R2),L2
        BIC     #200,R2
        CMPB    R2,#10103       ;ASCII C?
        BNE     1$
        BIS     #10100,BEGSYN
        BR      VARG
1$:     CMPB    R2,#10110       ;ASCII H?
        BNE     2$
        BIS     #10200,BEGSYN
        BR      VARG
2$:     CMPB    R2,#10122       ;ASCII R?
        BNE     LARG
        BIS     #1040,BEGSYN

;VOLGENDE ARGUMENT OPHALEN

VARG:   CMPB    (R1)+,#.COMMA
        BNE     LSYN
        JSR     PC,FVAL
        BCS     LARG            ;NOT NUMERIC
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT
1$:
        MOV     42(R5),N
        DEC     N

;VOLGENDE ARGUMENT

NARG:   CMPB    (R1),#.COMMA
        BEQ     1$
3$:     CMPB    (R1)+,#.RPAR
        BNE     LSYN
        CMPB    (R1),#.EOL
        BNE     LSYN
        MOV     #417,OUTSW
        MOV     #411,INSW
        MOV     #410,INSW1
        MOV     #413,OUT
        CLR     STEPC
        RTS     PC              ;BACK TO BASIC!!
1$:     INC     R1
        TST     (R1)+
        BIS     #10000,BEGSYN
        BR      3$

;FOUTUITGANGEN NAAR BASIC

LARG:   JMP     ERRARG
LSYN:   JMP     ERRSYN
```

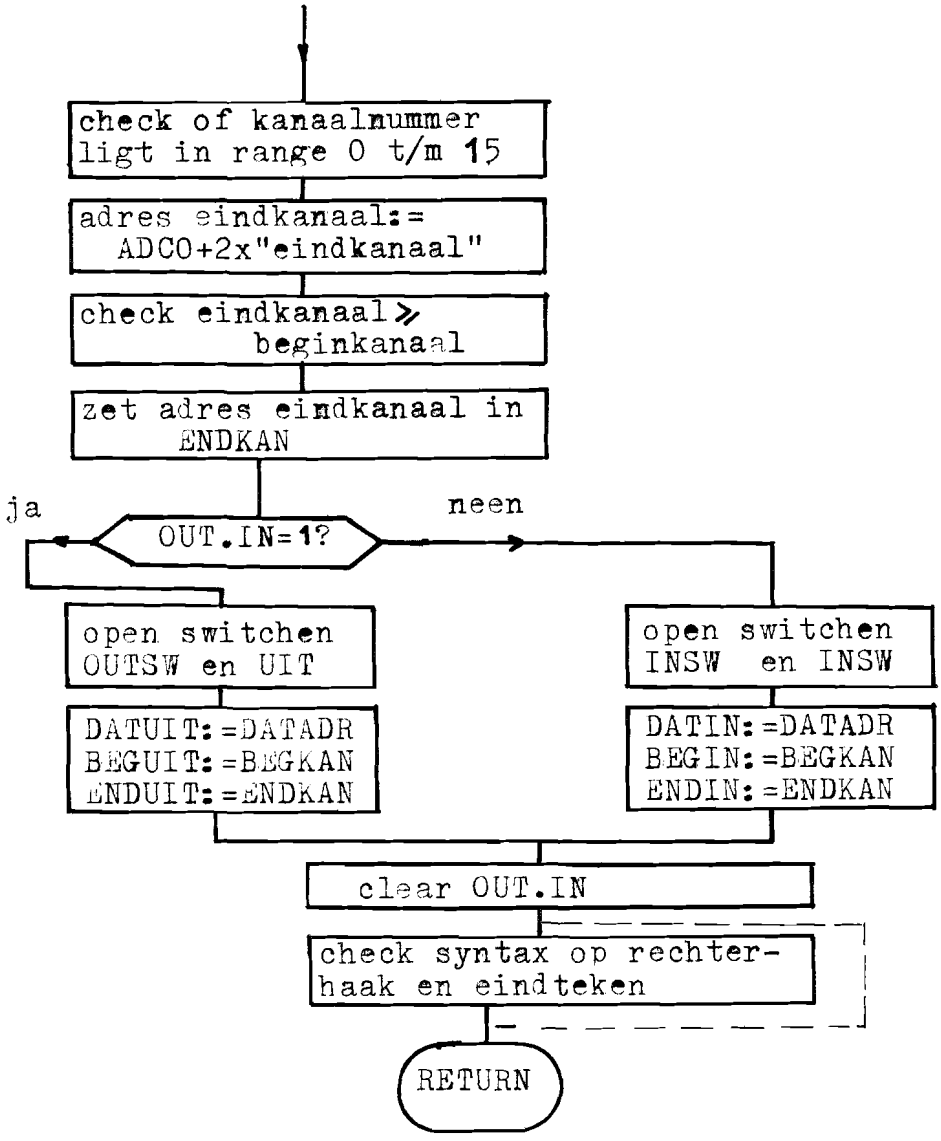
```
        .PAGE
        .TITLE  HELP ,HYBRIDE & LEEGAME ROUTINES
        .SBTTL  CONSTANT SECTION
        .CSECT HYFUN2

        R0=%0
        R1=%1
        R2=%2
        R3=%3
        R4=%4
        R5=%5
        SP=%6
        PC=%7

        ;DUMMY CONSTANTES
        ERRREG  =177684
        SYNREG  =177682
        STTREG  =177680
        ERRVEC  =174
        ISRVEC  =170
        PRIOR   =340
        PS      =177776
        ISRPR   =340

        ;EINDE DUMMY'S


TABEL:
N:      .WORD   0               ;ACTUAL RUN
BEGIN:  .WORD   0               ;ADRES EERSTE INPUTKAN
ENDIN:  .WORD   0               ;ADRES LAATSTE INPUTKAN
DATIN:  .WORD   0               ;ADRES INPUTDATA
BEGUIT: .WORD   0               ;ADRES EERSTE OUTPUTKAN
ENDUIT: .WORD   0               ;ADRES LAATSTE OUTPUTKAN
DATUIT: .WORD   0               ;ADRES OUTPUTDATA
BEGSYN: .WORD   177404          ;SYNCHRO.REG VOOR EERSTE TP
ENDSYN: .WORD   177430          ;    ,,    ,,  ,,  LAATSTE TP
BEGERR: .WORD   020034          ;ERR.REG VOOR EERSTE TP
BEKENX: .WORD   0               ;P.M.


        .PAGE
        .SBTTL  ROUTINE LOOP
        .GLOBL  LOOP,OUT,IN,ENDL,ISR,ERR,ERRARC,ERRHYB
        .GLOBL  N,BEGIN,ENDIN,DATIN
        .GLOBL  BEGUIT,ENDUIT,DATUIT
        .GLOBL  BEGSYN,ENDSYN,BEGERR
        .GLOBL  BEKENX                  ;P.M.
```

FORTRAN

```
;SUBROUTINE LOOP (MODEAC,N,ACR)
;MODEAC:          VARIABELE, ALS WAARDE IS
;                 1->AC IN COMPUTE
;                 2->AC IN HOLD                          FORTRAN
;                 3->AC IN RESET
;N:       AANTAL RUNS,WORDT INTEGER GEMAAKT
;ACR:             INDIEN HIER IETS OPGEGEVEN
;                 WORDT,WORDT HET BEGIN MET ACR
;                 GESYNCHRONISEERD


LOOP:    MOV     #^O404,BEGSYN
         MOV     (R5)+,RC
         MOV     @(R5)+,R2
         CMPB    R2,#^O1 ;COMP?
         BNE     1$
         BIS     #^O100,BEGSYN
         BR      VARG
1$:      CMPB    R2,#^O2 ;HOLD?
         BNE     2$
         BIS     #^O200,BEGSYN
         BR      VARG
2$:      CMPB    R2,#^O3 ;RESET?
         BNE     LARG
         BIS     #^O40,BEGSYN

;VOLGENDE ARGUMENT OPHALEN

VARG:    MOV     @(R5)+,N
         BMI     LARG
         DEC     N
;VOLGENDE ARGUMENT

NARG:    CMPB    R0,#2
         BNE     1$                ;AANTAL ARGUMENTEN IS 3
3$:      MOV     #416,OUTSW
         MOV     #410,INSW
         MOV     #407,INSW1
         MOV     #412,UIT
         CLR     SITREG
         RTS     PC
1$:      BIS     #^O10000,BEGSYN
         BR      3$

;FOUTUITGANG

LARG:    MOV     #1,ERRHYB
         JMP     ERRARG



; ----------------------------
```

OUT en IN(array,beginkanaal,eindkanaal)




check of kanaalnummer ligt in range 0 t/m 15
adres eindkanaal:= ADC0+2x"eindkanaal"
check eindkanaal ≥ beginkanaal
zet adres eindkanaal in ENDKAN
ja
OUT.IN=1?
neen
open switchen OUTSW en UIT
DATUIT:=DATADR
BEGUIT:=BEGKAN
ENDUIT:=ENDKAN
open switchen INSW en INSW
DATIN:=DATADR
BEGIN:=BEGKAN
ENDIN:=ENDKAN
clear OUT.IN
check syntax op rechter-haak en eindteken
RETURN

```
;BASICROUTINES "OUT" & "IN"
;       OUT (ARRAY,BEGINKANAAL,EINDKANAAL)
;       IN  (  "  ,     "      ,     "     )
```

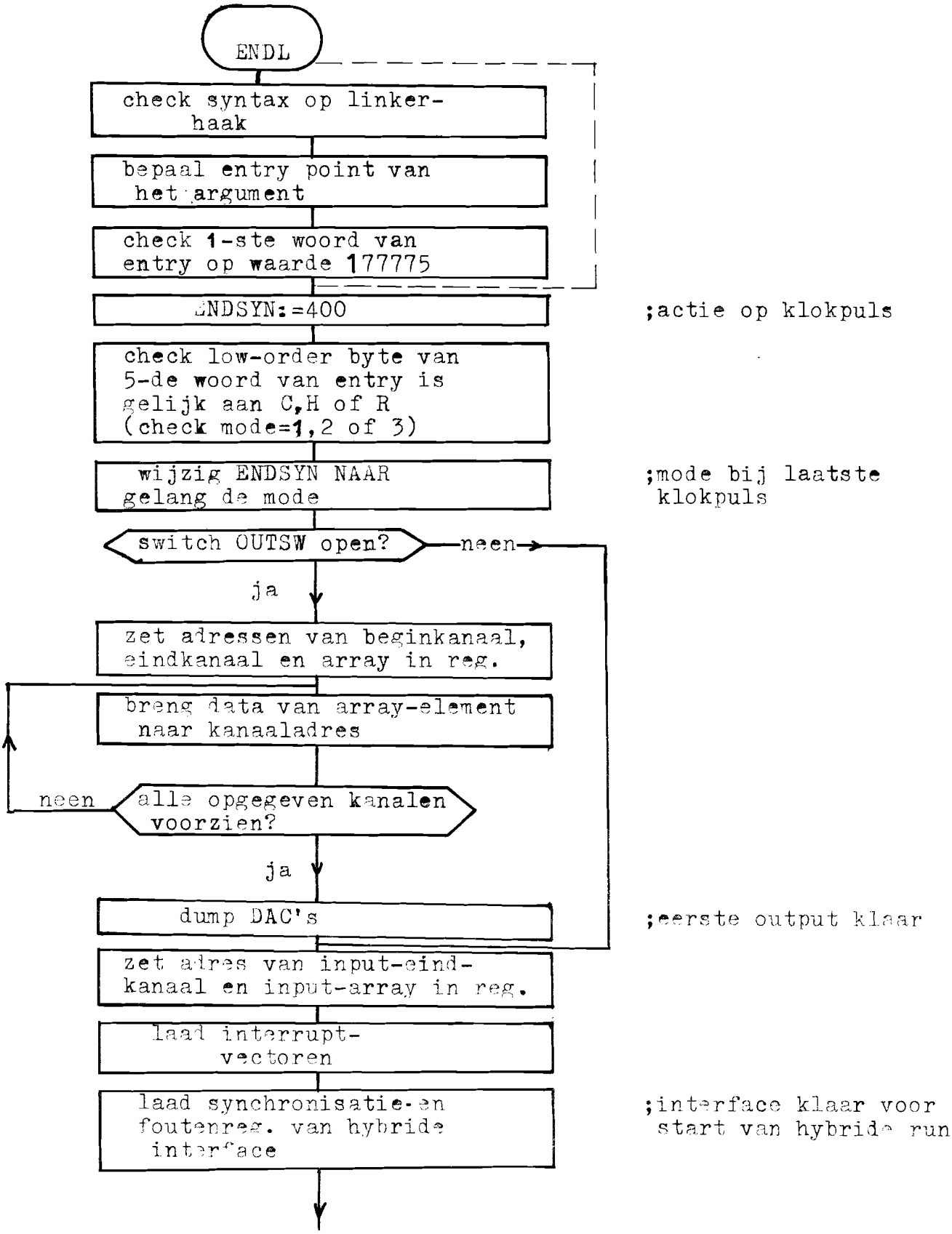


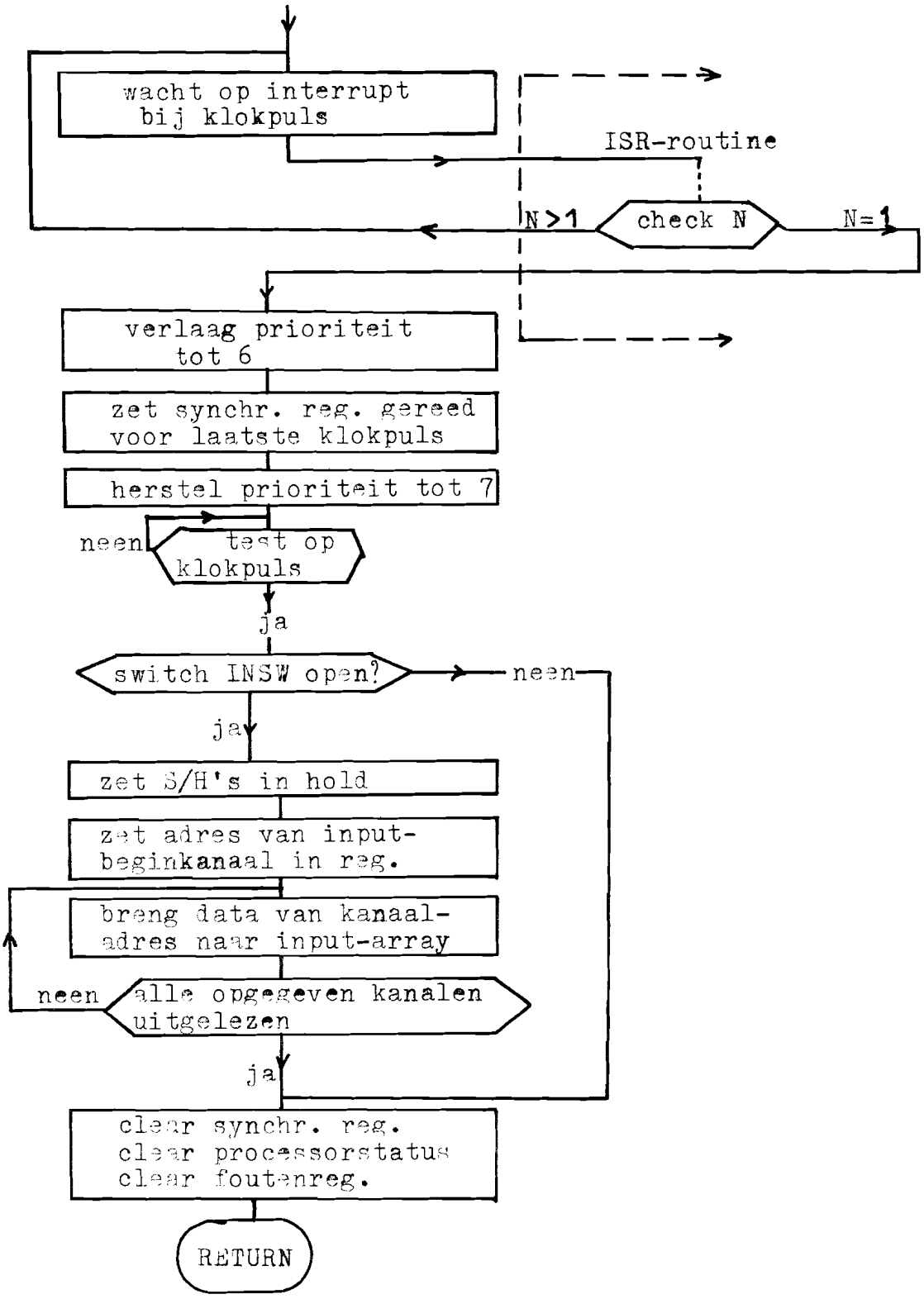
```
        R0=%0
        R1=%1
        R2=%2
        R3=%3
        R4=%4
        R5=%5
        R6=%6

LSY:    JMP     FOUTSYN
LAR:    JMP     FOUTARG
        SP=%6
        PC=%7
SS1:    .WORD   0
SS2:    .WORD   0


IN:     INC     OUT.IN
OUT:    CMPB    (R1)+,#.LPAR
        BNE     LSY
        MOVB    (R1)+,R2
        BMI     LSY
        SWAB    R2
        DISB    (R1)+,R2
        ADD     (R5),R2
        CMP     #^O177776,(R2)+ ;CHECK ARRAY
        BNE     LAR
        MOV     (R2)+,R3
        ADD     #4,R3   ;R3 CONTAINSADDRES ARRAY-ELEMENT
        MOV     R3,DATADR

        TST     OUT.IN
        BNE     NARG1

        TST     (R2)+
        MOV     (R2),SS2        ;SECOND SUBSCRIPT
        TST     -(R2)
2$:     MOV     (R2),SS1        ;FIRST SUBSCRIPT
1$:     MOV     (R3)+,FAC1(R5)  ;INTEGER MAKEN VAN ARRAY-WAARDEN
        MOV     (R3),FAC2(R5)
        JSR     PC,INT
        MOV     FAC2(R5),(R3)+
        DEC     SS1
        BPL     1$              ;VOLGEND ELEMENT
        DEC     SS2
        BPL     2$              ;VOLGENDE KOLOM

;VOLGENDE ARGUMENT

NARG1:  CMPB    (R1)+,#.COMMA
        BNE     LSY
        JSR     PC,EVAL
        BCS     LAR             ;NOT NUMERIC
        TST     FAC1(R5)
        BPL     1$
        JSR     PC,INT
```

```
1$:
        MOV     FAC2(R5),R3
        BIT     #177760,R3          ;KANAALNUMMER <16
        BNE     LAR
        ASL     R3
        ADD     #ADC0,R3            ;R3 ADRES BEGINKANAAL
        MOV     R3,BEGKAN

        ;VOLGENDE ARGUMENT

NARG2:  CMPB    (R1)+,#.COMMA
        BNE     LSY

        JSR     PC,FVAL
        BCS     LAR                 ;NOT NUMERIC
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT
1$:
        MOV     FAC2(R5),R2
        BIT     #177760,R2          ;KANAALNUMMER <16
        BNE     LAR
        ASL     R2
        ADD     #ADC0,R2
        CMP     R2,BEGKAN           ;BEGINKANAAL < EINDKANAAL
        BMI     LAR
        MOV     R2,ENDKAN

        TST     OUT.IN
        BEQ     OUTL

        MOV     #240,INS+1
        MOV     #240,INS+

        MOV     DATADR,DATIN
        MOV     BEGKAN,BEGIN
        MOV     ENDKAN,ENDIN
        BR      TRUG

OUT.IN: .WORD   0
DATADR: .WORD   0
BEGKAN: .WORD   0
ENDKAN: .WORD   0

OUTL:

        MOV     #240,OUTS+

        MOV     #240,UIT
        MOV     DATADR,DATUIT
        MOV     BEGKAN,BEGUIT
        MOV     ENDKAN,ENDUIT

TRUG:   CLR     OUT.IN
        CMPB    (R1)+,#.APAR
        BNE     SY
        CMPB    (R1),#.EOL
        BNE     SY

        RTS     PC          ;BACK TO BASIC
SY:     JMP     ERRSYN
```

```
;FORTRAN ROUTINES "OUT" & "IN"
;       OUT (ARRAY,BEGINKANAAL,EINDKANAAL)
;       IN  (  "  ,     "     ,     "     )                    FORTRAN
LAR:    MOV     #2,ERRHYB               ;FOUT IN OUT
        TST     OUT.IN
        BEQ     1$
        MOV     #3,ERRHYB               ;FOUT IN IN
1$:     JMP     ERRARG

ADC0=173700



IN:     INC     OUT.IN
OUT:    MOV     (R5)+,R0
        MOV     (R5)+,DATADR            ;ADRES EERSTE ARRAY ELEMENT
;VOLGENDE ARGUMENT

NARG1:  MOV     @(R5)+,R3
        BIT     #177760,R3      ;KANAALNUMMER <16
        BNE     LAR
        ASL     R3
        ADD     #ADC0,R3        ;R3 ADRES BEGINKANAAL
        MOV     R3,BEGKAN

        ;VOLGENDE ARGUMENT

NARG2:  MOV     @(R5),R2
        BIT     #177760,R2      ;KANAALNUMMER <16
        BNE     LAR
        ASL     R2
        ADD     #ADC0,R2
        CMP     R2,BEGKAN       ;BEGINKANAAL < EINDKANAAL
        BMI     LAR
        MOV     R2,ENDKAN

        TST     OUT.IN
        BEQ     OUTL

        MOV     #240,INSW1
        MOV     #240,INSW


        MOV     DATADR,DATIN
        MOV     BEGKAN,BEGIN
        MOV     ENDKAN,ENDIN
        BR      TRUG


OUT.IN: .WORD   0
DATADR: .WORD   0
BEGKAN: .WORD   0
ENDKAN: .WORD   0

OUTL:

        MOV     #240,OUTSW


        MOV     #240,UIT
        MOV     DATADR,DATUIT
        MOV     BEGKAN,BEGUIT
        MOV     ENDKAN,ENDUIT

TRUG:   CLR     OUT.IN
```

ENDL(mode)


ENDL
check syntax op linker-haak
bepaal entry point van het argument
check 1-ste woord van entry op waarde 177775
ENDSYN:=400
;actie op klokpuls
check low-order byte van 5-de woord van entry is gelijk aan C,H of R (check mode=1,2 of 3)
wijzig ENDSYN NAAR gelang de mode
;mode bij laatste klokpuls
switch OUTSW open?
neen
ja
zet adressen van beginkanaal, eindkanaal en array in reg.
breng data van array-element naar kanaaladres
neen
alle opgegeven kanalen voorzien?
ja
dump DAC's
;eerste output klaar
zet adres van input-eind-kanaal en input-array in reg.
laad interrupt-vectoren
laad synchronisatie-en foutenreg. van hybride interface
;interface klaar voor start van hybride run


wacht op interrupt bij klokpuls
ISR-routine
N>1
check N
N=1
verlaag prioriteit tot 6
zet synchr. reg. gereed voor laatste klokpuls
herstel prioriteit tot 7
neen
test op klokpuls
ja
switch INSW open?
neen
ja
zet S/H's in hold
zet adres van input-beginkanaal in reg.
breng data van kanaal-adres naar input-array
neen
alle opgegeven kanalen uitgelezen
ja
clear synchr. reg.
clear processorstatus
clear foutenreg.
RETURN

```
;--------------------------------
```

BASIC

```
        .PAGE
        .SBTTL  ROUTINE ENDLOOP
        .GLOBL  ISR,ERR,ENDL
        .GLOBL  N,BEGIN,ENDIN,DATIN
        .GLOBL  BEGUIT,ENDUIT,DATUIT
        .GLOBL  BEGSYN,ENDSYN,BEGERR
        .GLOBL  REKENX  ;P.M.

;BASIC STATEMENT CALL"ENDL"(MODE A.C.)
E1:     .WORD   4



ENDL:
        CMPB    (R1)+,#.LPAR
        BNE     LSYN
        MOVB    (R1)+,R2
        BMI     LSYN
        SWAB    R2
        BISB    (R1)+,R2
        ADD     (R5),R2

        CMP     #177775,(R2)+
        BNE     LARG
        MOV     #^O400,ENDSYN
        MOV     6(R2),R2
        BIC     #200,R2
        CMPB    R2,#^O103       ;ASCII C?
        BNE     1$
        BIS     #^O133,ENDSYN
        BR      EOL
1$:     CMPB    R2,#^O110       ;ASCII H?
        BNE     2$
        BIS     #^O200,ENDSYN
        BR      EOL
2$:     CMPB    R2,#^O122       ;ASCII R?
        BNE     LARG
        BIS     #^O40,ENDSYN


;---

EOL:
OUTSW:  BR      NOOUTP  ;OUTSW=NOP AT FIRST OUTPUT
        MOV     BEGUIT,R0       ;FIRST OUTPUT
        MOV     ENDUIT,R1
        MOV     DATUIT,R2
1$:     CLR     (R2)+   ;DATA IS STORED AS INTEGER
        MOV     (R2)+,(R0)+     ;ARRAY ELEMENT 2 WORDS
        CMP     R0,R1
        BLE     1$      ;R0<=R1:OUTPUT
        CLR     SYNREG
        MOV     #2,SYNREG       ;DUMP DAC'S
```

```
;---EERST OUTPUTCYCLUS GEDAAN
;---INPUTREGISTERS NOG VULLEN

NOOUTP: MOV     ENDIN,R3
        MOV     R4,R1
        MOV     DATIN,R4

;--

        MOV     #ISR,ISRVEC
        MOV     #ERR,ERRVEC
        MOV     #ISRPR,ISRVEC+2
        MOV     #ERRPR,ERRVEC+2


        MOV     BEGSYN,SYNREG   ;BEVEG INTERFACE IN BEGIN
        MOV     BEGERR,ERRREG   ;TOESTAND :INTERRUPTS KOMEN
                                ;VANAF NU

BUSY:   BR      BUSY    ;PARKEERLUS TIJDENS HYP.RUN

COMBET: BIC     #200,PS ;PRIORITEIT=6
        MOV     ENDSYN,SYNREG
        BIS     #200,PS ;HERSTEL PRIORITEIT=7   ;LAATSTE I.P.MOET NOG
1$:     TSTB    STTREG          ;AFGEWERKT WORDEN
        BPL     1$

INSW:   BR      NOINPT  ;INSW=NOP AT LAST INPUT
        MOV     #1,SYNREG
        MOV     BEGIN,R0
1$:     CLR     (R4)+
        MOV     (R0)+,(R4)+

        CMP     R0,R3
        BLE     1$



NOINPT: CLR     SYNREG
        CLR     ERRREC
        CLR     PS



BACK:   MOV     R1,R4
        RTS     PC      ;BACK TO BASIC!!!!!!!
```

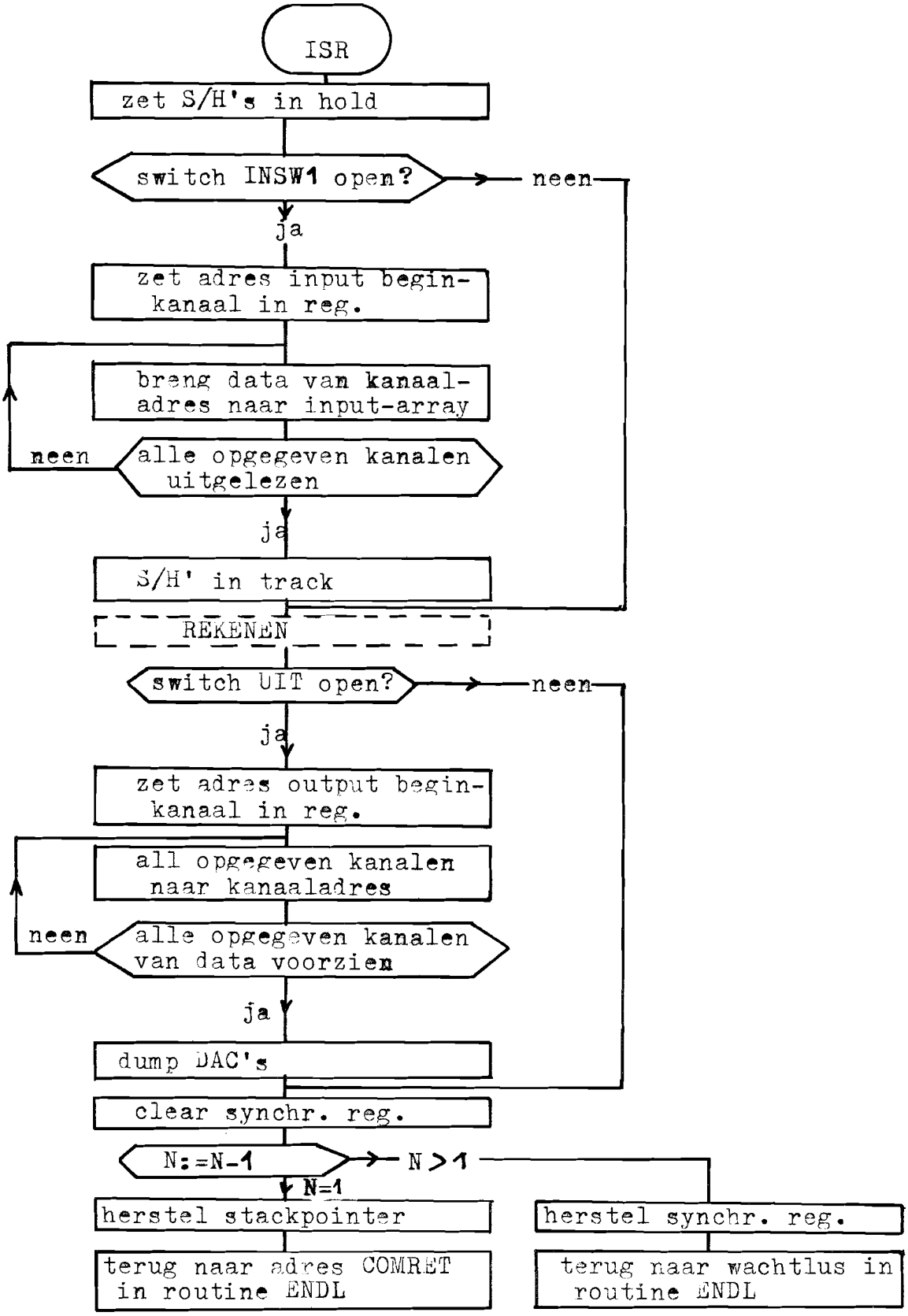
```
;------

                                                        FORTRAN
        .PAGE
        .SBTTL  ROUTINE ENDLOOP

;FORTRAN STATEMENT CALL ENDL (MODE A.C.)
E1:     .WORD   0
LARG1:  MOV     #4,ERRHYP
        JMP     ERRALG



ENDL:   MOV     (R5)+,R0
        MOV     @(R5),R2

        MOV     #^O400,ENDSYN
        CMPB    R2,#^O1
        BNE     1$
        BIS     #^O100,ENDSYN
        BR      EOL
1$:     CMPB    R2,#^O2
        BNE     2$
        BIS     #^O200,ENDSYN
        BR      EOL
2$:     CMPB    R2,#^O3
        BNE     LARG1
        BIS     #^O40,ENDSYN


;---

EOL:


OUTSW:  BR      NOOUTP  ;OUTSW=NOP AT FIRST OUTPUT
        MOV     BEGUIT,R0       ;FIRST OUTPUT
        MOV     ENDUIT,R1
        MOV     DATUIT,R2
1$:     MOV     (R2)+,(R0)+
        CMP     R0,R1
        BLE     1$      ;R0<=R1:OUTPUT
        CLR     SYNREG
        MOV     #2,SYNREG       ;DUMP DAC'S

;---FIRST OUTPUTCYCLUS GEDAAN
;---INPUTREGISTERS NOG VULLEN

NOOUTP: MOV     ENDIN,R3
        MOV     R4,E1
        MOV     DATIN,R4

;--
```

FORTRAN

```
        MOV     #ISR,ISRVEC
        MOV     #ERR,ERRVEC
        MOV     #ISRPR,ISRVEC+2
        MOV     #ERRPR,ERRVEC+2


        MOV     BEGSYN,SYNREG   ;BRENG INTERFACE IN BEGIN
        MOV     BEGERR,ERRREG   ;TOESTAND :INTERRUPTS KOMEN
                                ;VANAF NU

RUST:   BR      RUST    ;PARKEERLUS TIJDENS HYB.RUN

COMRET: BIC     #200,PS ;PRIORITEIT=6
        MOV     ENDSYN,SYNREG
        BIS     #200,PS ;HERSTEL PRIORITEIT=7   ;LAATSTE T.P.MOET NOG
1$:     TSTB    STTREG          ;AFGEWERKT WORDEN
        BPL     1$

INSW:   BR      NOINPT  ;INSW=NOP AT LAST INPUT
        MOV     #1,SYNREG
        MOV     BEGIN,R0
1$:     MOV     (R0)+,(R4)+

        CMP     R0,R3
        BLE     1$



NOINPT: CLR     SYNREG
        CLR     ERRREG
        CLR     PS



BACK:   MOV     R1,R4
        RTS     PC
```

:----------

Interrupt service routine ISR



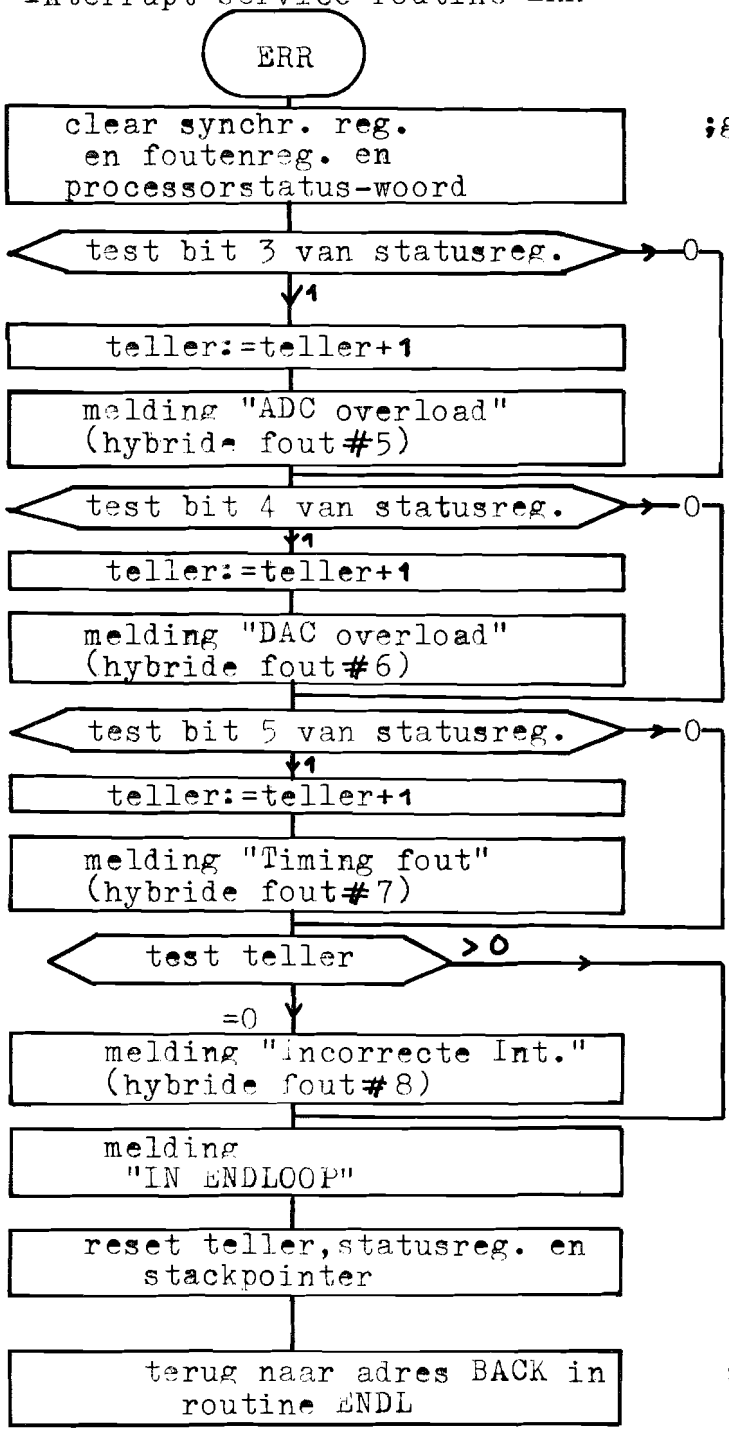
BASIC

```
;------                              ---------------
        .PAGE
        .SBTTL  INTRPT SERVICE ROUTINE
        .GLOBL  ISR
        .GLOBL  N,BEGIN,ENDIN,DATIN
        .GLOBL  REQUIT,ENDQUIT,DATQUIT
        .GLOBL  REGSYN,ENDSYN,REGERR
        .GLOBL  RERFN                           ;P.M.


ISR:    MOV     #1,SYNREG
INSW1:  BR      RERFN   ;VOINP SWITCH,INSW1=NOP AT INPUT

        MOV     BEGIN,R0
1$:     CLR     (R4)+
        MOV     (R0)+,(R4)+

        CMP     R0,R3   ;R0<=R3:INPUT
        BLE     1$
        CLR     SYNREG          ;SYNC AMPS IN TRACK


RERFN:;P.M.


QUIT:   BR      KLAAR   ;QUIT=NOP AT OUTPUT
        MOV     BEGQUIT,R0
1$:     CLR     (R2)+
        MOV     (R2)+,(R0)+

        CMP     R0,R1
        BLE     1$
        CLR     SYNREG
        MOV     #2,SYNREG       ;DUMP DAC'S


KLAAR:  CLR     SYNREG
        DEC     N       ;AANTAL TP'S N+1
        BNE     RET
        ADD     #10,SP  ;FAKE QUIT
        JMP     COMRET
RET:    MOV     REGSYN,SYNREG
        RTI
```

FORTRAN

```
        .PAGE
        .SBTTL  INTRPT SERVICE ROUTINE


ISR:    MOV     #1,SYNREG
INSW1:  BR      REKEN   ;NOINP SWITCH,INSW1=NOP AT INPUT

        MOV     BEGIN,R0
1$:     MOV     (R0)+,(R4)+

        CMP     R0,R3   ;R0<=R3:INPUT
        BLE     1$
        CLR     SYNREG          ;S/H AMPS IN TRACK


REKEN:;P.M.


UIT:    BR      KLAAR   ;UIT=NOP AT OUTPUT
        MOV     BEGUIT,R0
1$:     MOV     (R2)+,(R0)+

        CMP     R0,R1
        BLE     1$
        CLR     SYNREG
        MOV     #2,SYNREG       ;DUMP DAC'S


KLAAR:  CLR     SYNREG
        DEC     N       ;AANTAL TP'S N
        BNE     RET
        ADD     #104,SP ;FAKE RTI
        JMP     COMRET
RET:    MOV     BEGSYN,SYNREG
        RTI
```

Interrupt service routine ERR



```
;ERROR ROUTINE
        .PAGE
        .SBTTL ERROR IN HYBRID INTERFACE
        .GLOBL ERR,MSG
TELLER: .WORD   0

ERR:    CLR     SYNREG
        CLR     ERRREG

        CLR     R5
        BIT     #6,STTREG
        BEQ     1$
        INC     TELLER
        JSR     R1,MSG
        .ASCII  '"ADC OVERLOAD'
        .BYTE 0
        .EVEN
1$:     BIT     #10,STTREG
        BEQ     2$
        INC     TELLER

        JSR     R1,MSG
        .ASCII  '"DAC OVERLOAD'
        .BYTE 0
        .EVEN
2$:     BIT     #20,STTREG
        BEQ     3$
        INC     TELLER

        JSR     R1,MSG
        .ASCII  '"TIMING FOUT'
        .BYTE 0
        .EVEN
3$:
        TST     TELLER
        BNE     4$
        JSR R1,MSG
        .ASCII  '"INCORRECTE INT'
        .BYTE 0
        .EVEN
4$:     JSR     R1,MSG
        .ASCII  'IN ENDLOOP"'
        .BYTE   0
        .EVEN
        CLR     TELLER
        CLR     STTREG

        ADD     #104,SP
        JMP BACK
```

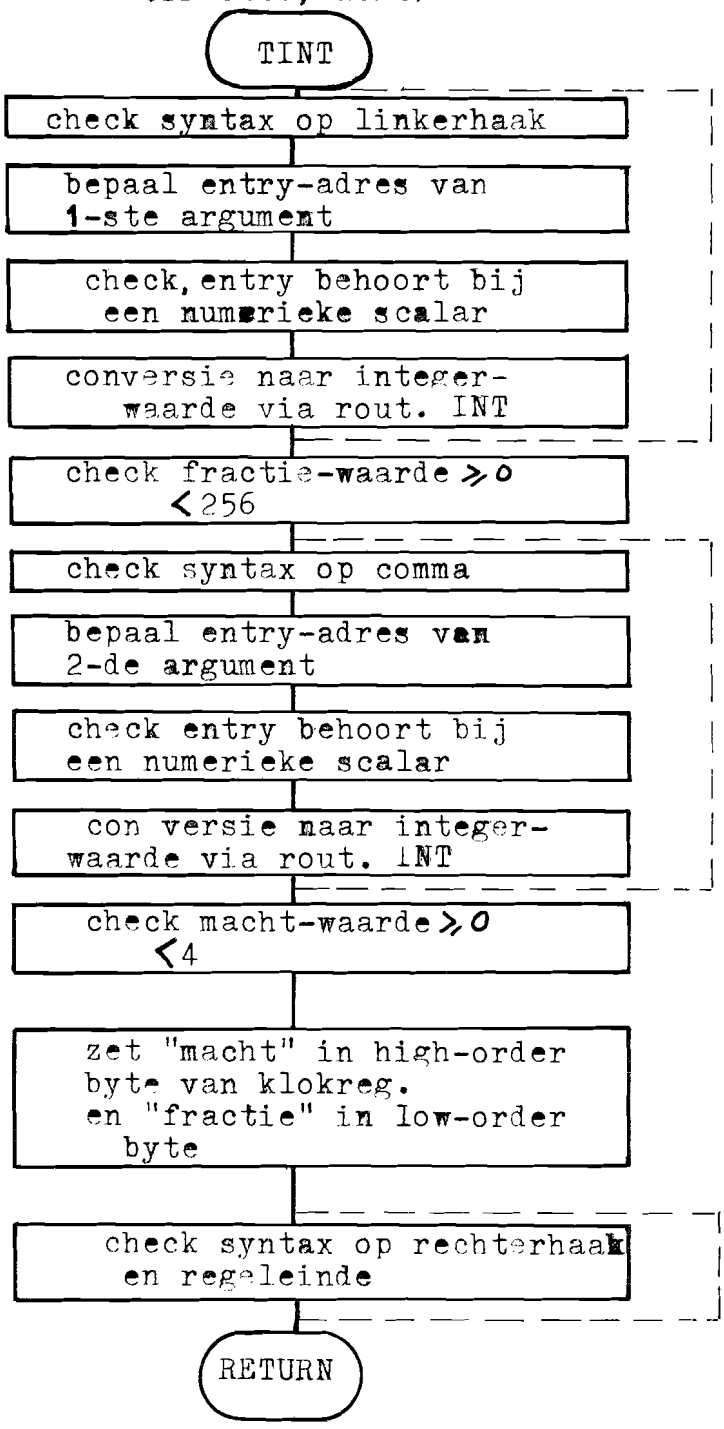
FORTRAN

```
;ERROR ROUTINE
        .PAGE
        .SBTTL ERROR IN HYBRID INTERFACE
TELLER: .WORD   0

ERR:    CLR     SYNREG
        CLR     ERRREG

        CLR     PS
        BIT     #4,STTREG
        BEQ     1$
        INC     TELLER
        MOV     #5,ERRHYB
        JSR     PC,ERRARG
1$:     BIT     #10,STTREG
        BEQ     2$
        INC     TELLER
        MOV     #6,ERRHYB
        JSR     PC,ERRARG
2$:     BIT     #20,STTREG
        BEQ     3$
        INC     TELLER
        MOV     #7,ERRHYB
        JSR     PC,ERRARG
3$:
        TST     TELLER
        BNE     4$
        MOV     #10,ERRHYB
        JSR     PC,ERRARG
4$:     CLR     TELLER
        CLR     STTREG

        ADD     #104,SP
        JMP BACK
```

```
;
;
;
;
GETERA: JMP     ERRARG
GETERS: JMP     ERRSYN
;
;
;
;
        .PAGE
        .SBTTL  SET MODE ANALOG COMPUTER
        .GLOBL  RESE,COMP,HOLD,POTS,ALLR
;
;
MODEAC=173606
;
;
;RESET
RESE:
        CMPB    (R1)+,#.EOL     ;CHECK EOL TOKEN
        BNE     GETERS
        MOV     #4,MODEAC
        RTS     PC
;
;COMPUTE
COMP:
        CMPB    (R1)+,#.EOL     ;CHECK EOL TOKEN
        BNE     GETERS
        MOV     #2,MODEAC
        RTS     PC
;
;HOLD
HOLD:
        CMPB    (R1)+,#.EOL     ;CHECK EOL TOKEN
        BNE     GETERS
        MOV     #1,MODEAC
        RTS     PC
;
;POTSET
POTS:
        CMPB    (R1)+,#.EOL     ;CHECK EOL TOKEN
        BNE     GETERS
        MOV     #10,MODEAC
        RTS     PC
;
;ALLRESET
ALLR:
        CMPB    (R1)+,#.EOL     ;CHECK EOL TOKEN
        BNE     GETERS
        MOV     #20,MODEAC
        RTS     PC
;
;
```

```
; HYBRID USER ROUTINES PACKAGE
;
        .TITLE  HURPAK HYBRID USER ROUTINES PACKAGE
        .SBTTL  DEFINITION SECTION
        .CSECT  HYFUN1
;
        .GLOBL  ERRARG,ERRHYB;
;
;                                                       FORTRAN
R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7
STTREG=173600

;
GETERA: MOV     #25,ERRHYB
        JMP     ERRARG

;
;
;
;
        .PAGE
        .SBTTL  SET MODE ANALOG COMPUTER
        .GLOBL  RESE,COMP,HOLD,POTS,ALLR
;
;
MODEAC=173686
;
;
;RESET
RESE:
        MOV     #4,MODEAC
        RTS     PC
;
;COMPUTE
COMP:
        MOV     #2,MODEAC
        RTS     PC
;
HOLD:
        MOV     #1,MODEAC
        RTS     PC
;
;POTSET
POTS:
        MOV     #10,MODEAC
        RTS     PC
;
;ALLRESET
ALLR:
        MOV     #20,MODEAC
        RTS     PC
;
```

TINT(fractie,macht)



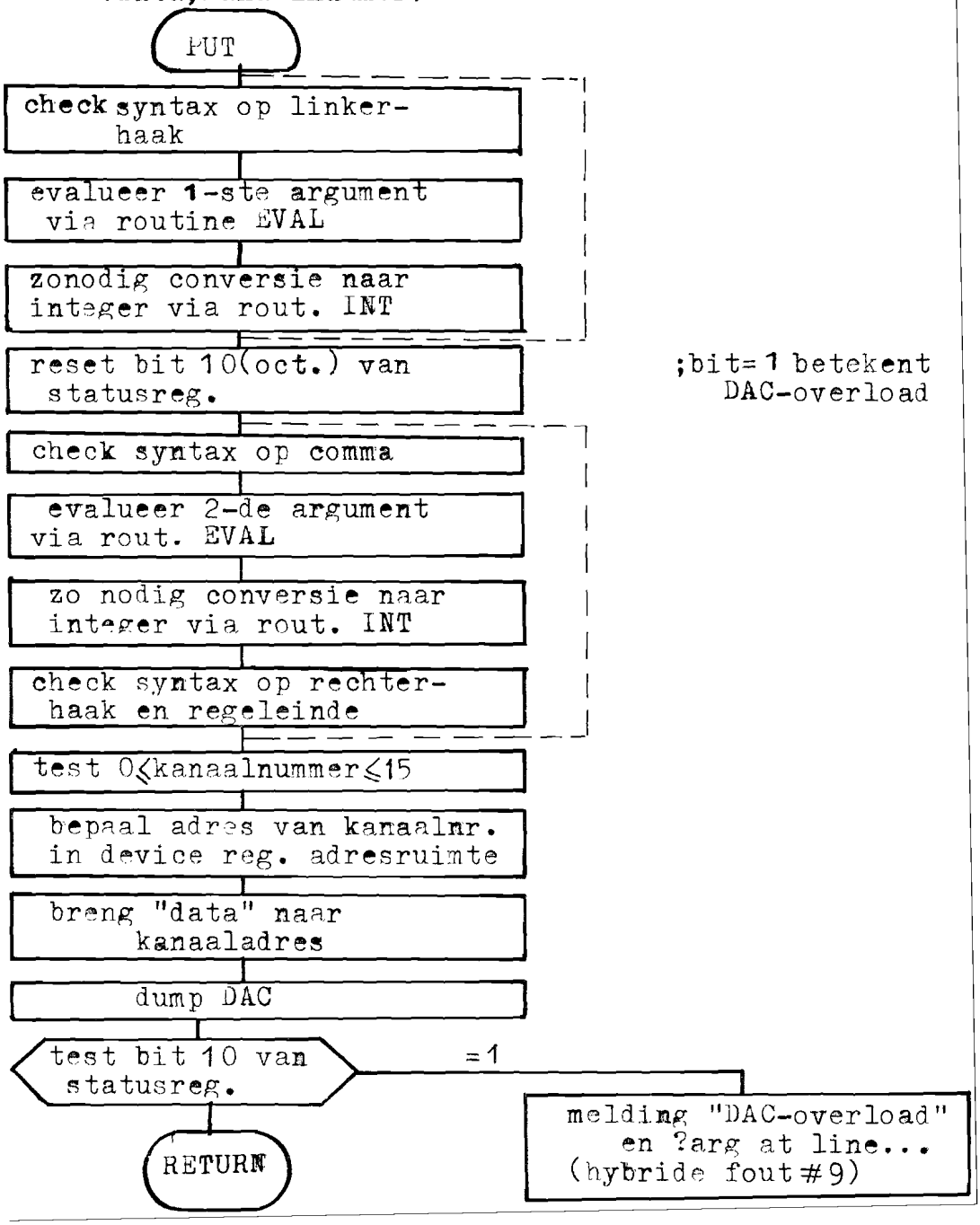
BASIC

```
;ROUTINE "TINT"(FLACTIF,MACHT)

        .PAGE
        .SBTTL INSTELLEN KLOKREGISTER
        .GLOBL TINT
        TICREG=177610
FTERR:  JMP     FOUSYN
FTERRA: JMP     FOUARG

TINT:   CMPB    (R1)+,#.LPAR
        BNE     FTERR
        MOVB    (R1)+,R2
        BMI     FTERR
        SWAB    R2
        BISB    (R1)+,R2
        ADD     (R5),R2
        CMP     #^O177775,(R2)+
        BNE     FTERRA
        MOV     (R2)+,FAC1(R5)
        MOV     (R2),FAC2(R5)
        JSR     PC,INT
        MOV     FAC2(R5),R3
        BIT     #^O177400,R3    ;INTEGER <256
        BNE     FTERRA

;VOLGEND ARGUMENT

        CMPB    (R1)+,#.COMMA
        BNE     FTERR
        MOVB    (R1)+,R2
        BMI     FTERR
        SWAB    R2
        BISB    (R1)+,R2
        ADD     (R5),R2
        CMP     #^O177775,(R2)+
        BNE     FTERRA
        MOV     (R2)+,FAC1(R5)
        MOV     (R2),FAC2(R5)
        JSR     PC,INT
        MOV     FAC2(R5),R2
        BIT     #^O177774,R2    ;INTEGER <4
        BNE     FTERRA
        SWAB    R2      ;MACHT IN HIGH-ORDER BYTE
        ADD     R3,R2
        MOV     R2,TICREG
        CMPB    (R1)+,#.RPAR
        BNE     FTERR
        CMPB    (R1)+,#.EOL
        BNE     FTERR
        RTS     PC      ;BACK TO BASIC
```

FORTRAN

```
;ROUTINE "TINT"(FRACTIE,MACHT)

        .PAGE
        .SBTTL INSTELLEN KLOKREGISTER
        .GLOBL TINT
        TICREG=173610

ETERA:  MOV     #27,ERRHYB
        JMP     ERRARG


TINT:   MOV     (R5)+,R0
        MOV     @(R5)+,R3
        BIT     #^O177400,R3    ;INTEGER <256
        BNE     ETERA

;VOLGEND ARGUMENT
        MOV     @(R5),R2
        BIT     #^O177774,R2    ;INTEGER <4
        BNE     ETERA
        SWAB    R2      ;MACHT IN HIGH-ORDER BYTE
        ADD     R3,R2
        MOV     R2,TICREG
        RTS     PC
```

PUT (data,kanaalnummer)

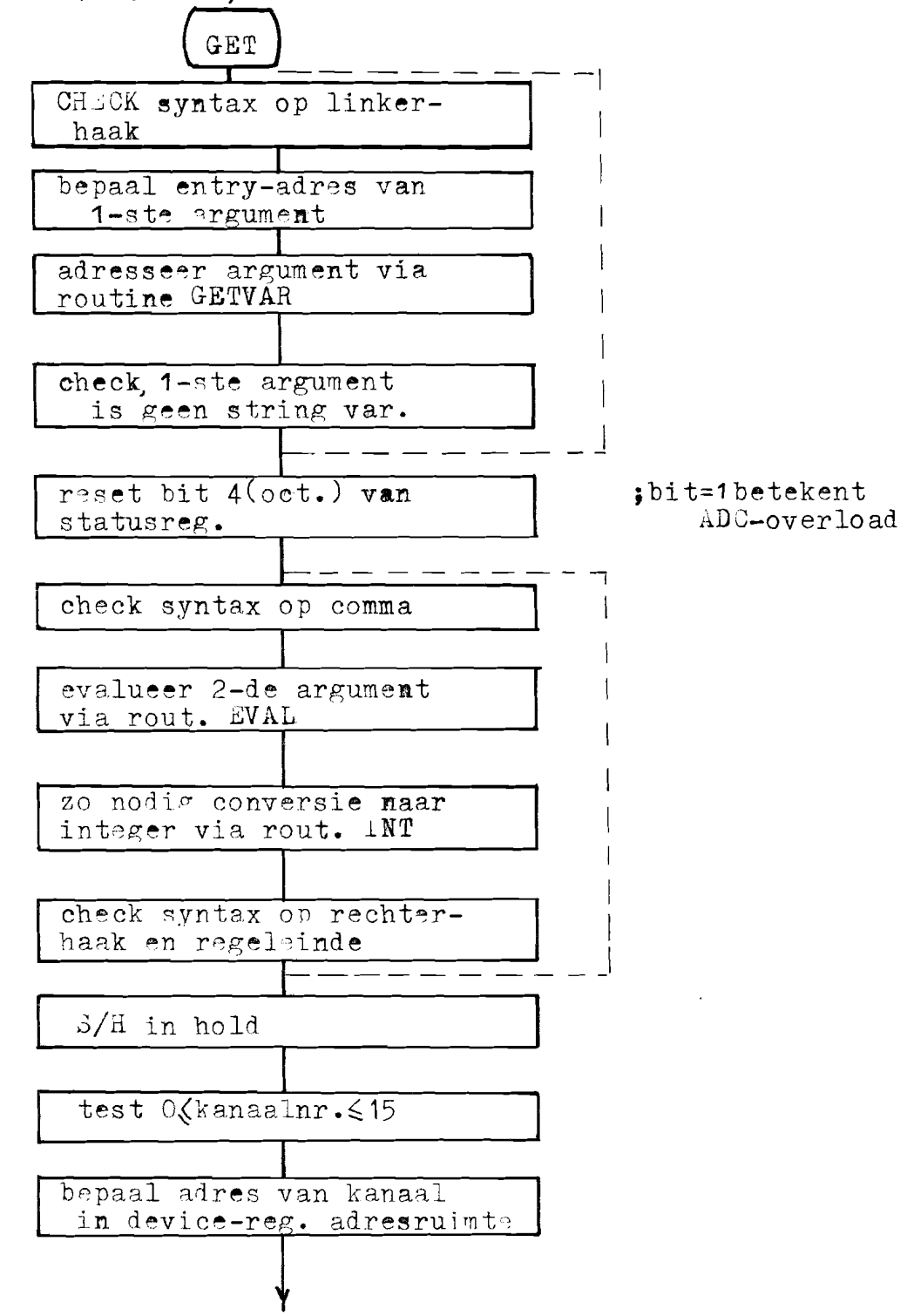


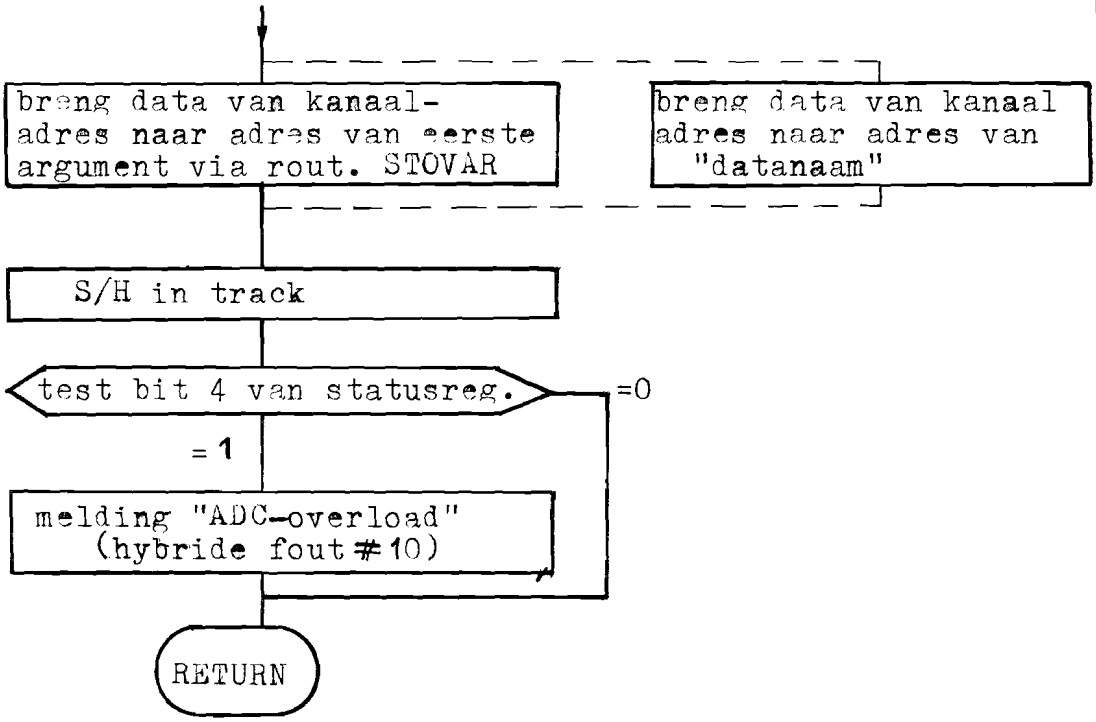
```
;
;
        .PAGE
        .SBTTL  WRITE ONE DAC                          BASIC
        .GLOBL  PUT
;
;CALLING STATEMENT IN BASIC IS AS FOLLOWS
;CALL"PUT"(DATA,CHANNELADDRESS)
;
;

TERS:   JMP     ERRSYN
TERA:   JMP     ERRARG
M1:     .WORD   0
;
DAC0=173700
PUT:    CMPB    (R1)+,#.LPAR    ;CHECK STARTING'('
        BNE     TERS
        JSR     PC,EVAL
        BCS     TERA    ;NOT NUMERIC
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT
1$:     MOV     FAC2(R5),M1     ;M1 CONTAINS DATA
        BIC     #1B,STTREG
;
; NEXT ARGUMENT, CHANNEL NUMBER
;
        CMPB    (R1)+,#.COMMA
        BNE     TERS
        JSR     PC,EVAL
        BCS     TERA    ;NOT NUMERIC
        TST     FAC1(R5)
        BEQ     2$
        JSR     PC,INT
2$:                     ;CHANNEL NUMBER IN FAC2
        CMPB    (R1)+,#.RPAR    ;CHECK CLOSING ')'
        BNE     TERS
        CMPB    (R1)+,#.EOL     ;CHECK EOL TOKEN
        BNE     TERS
;
;
        MOV     FAC2(R5),R3
        BIT     #177760,R3      ;KANAALNR <16
        BNE     TERA
        ASL     R3
        ADD     #DAC0,R3        ;DACADDRESS IN R3
        MOV     M1,(R3) ;DATA IN DAC BUFFER
        CLR     SYNREG
        MOV     #102,SYNREG     ;DUMP DATA
        BIT     #1B,STTREG
        BNE     3$
;
;
;
        .IFF    PC
3$:     JSR     R1,MSG
        .ASCII  '"DAC OVERLOAD"'
        .BYTE   0
        .EVEN
        BR      TERA
```

FORTRAN

```
;
        .PAGE
        .SBTTL  WRITE ONE DAC
        .GLOBL  PUT
;
;CALLING STATEMENT IN FORTRAN IS AS FOLLOWS
;CALLPUT(DATA,CHANNELADDRESS)
;
;


TERA:   MOV     #26,ERRHYB
        JMP     ERRARG
M1:     .WORD   0
;
DAC0=173700
PUT:    MOV     (R5)+,R0
        MOV     @(R5)+,M1       ;M1 CONTAINS DATA
        BIC     #10,STTREG
;
; NEXT ARGUMENT, CHANNEL NUMBER
;
        MOV     @(R5),R3
        BIT     #177760,R3      ;KANAALNR <16
        BNE     TERA
        ASL     R3
        ADD     #DAC0,R3        ;DACADDRESS IN R3
        MOV     M1,(R3) ;DATA IN DAC BUFFER
        CLR     SYNREG
        MOV     #102,SYNREG     ;DUMP DATA
        BIT     #10,STTREG
        BNE     3$
;
;
;
        RTS     PC
3$:     MOV     #11,ERRHYB
        JSR     PC,ERRARG
        RTS     PC
```

GET (datanaam,kanaalnummer)



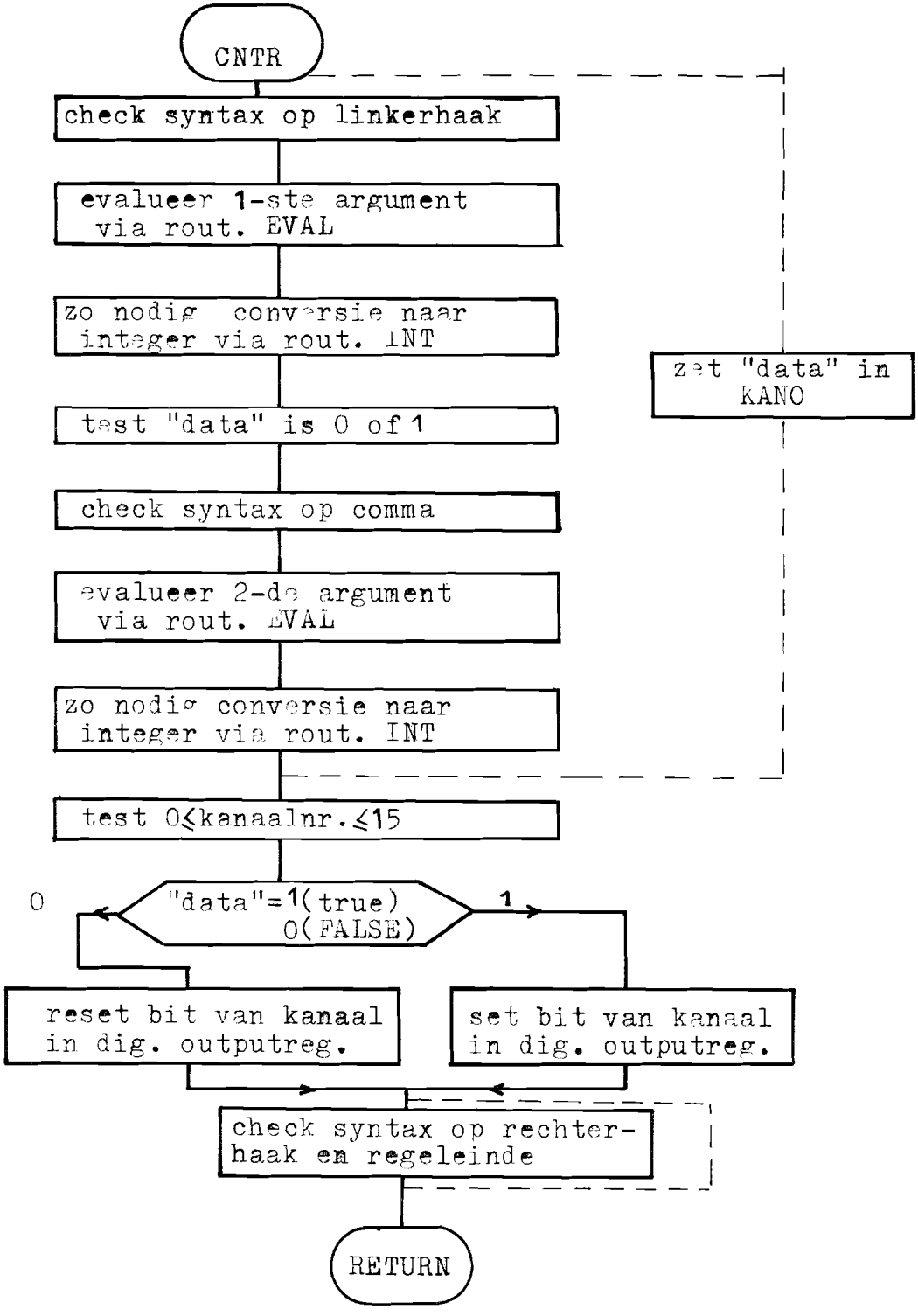

breng data van kanaal-
adres naar adres van eerste
argument via rout. STOVAR
breng data van kanaal
adres naar adres van
"datanaam"
S/H in track
test bit 4 van statusreg.
=0
=1
melding "ADC-overload"
(hybride fout #10)
RETURN

```
-;
;
        .PAGE                                   BASIC
        .SBTTL  READ ONE ADC CHANNEL
        .GLOBL  GET;
;CALLING STATEMENT IN BASIC IS AS FOLLOWS:
;CALL"GET"(DATAADDRESS,CHANNEL NUMBER)
;
;
SYNREG=173682
ADC0=173700
;
;
GET:    CMPB    (R1)+,#.LPAR    ;CHECK STARTING'('
        BNE     GETERS
        MOVB    (R1)+,R2
        BMI     GETERS
        SWAB    R2
        BISB    (R1)+,R2
        ADD     (R5),R2
        JSR     PC,GETVAR       ;GET DATA ADDRESS
        CMP     @VARSAV(R5),#-1 ;CHECK STRING
        BEQ     GETERA
        BIC     #4,STTREG
;
;NEXT ARGUMENT,CHANNEL NUMBER
;
        CMPB    (R1)+,#.COMMA
        BNE     GETERS
        JSR     PC,EVAL
        BCS     GETERA
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT
1$:                     ;CHANNEL NUMBER IN FAC2
        CMPB    (R1)+,#.RPAR    ;CHECK CLOSING ')'
        BNE     GETERS
        CMPB    (R1)+,#.EOL     ;CHECK EOL TOKEN
        BNE     GETERS
;
;
        MOV     #1,SYNREG       ;S/H IN HOLD
        MOV     FAC2(R5),R3
        BIT     #177760,R3      ;<16
        BNE     GETERA
        ASL     R3
        ADD     #ADC0,R3        ;ADC-CHANNELADDRESS IN R3
        MOV     (R3),FAC2(R5)   ;RESULT IN FAC
        JSR     PC,STOVAR       ;STORE RESULT IN DATAADDRESS
        CLR     SYNREG
        BIT     #4,STTREG
        BNE     2$
        RTS     PC
2$:     JSR     R1,MSG
        .ASCII  '"ADC OVERLOAD"'
        .BYTE   3
        .EVEN
        RTS     PC
;
```

FORTRAN

```
;
;
        .PAGE
        .SBTTL  READ ONE ADC CHANNEL
        .GLOBL  GET;
;CALLING STATEMENT IN FORTRAN IS AS FOLLOWS:
;CALL GET (DATAADDRESS,CHANNEL NUMBER)
;
;
SYNREG=173602
ADC0=173700
;
;
GET:    MOV     (R5)+,R0
        MOV     (R5)+,R0
        BIC     #4,STTREG
;
;NEXT ARGUMENT,CHANNEL NUMBER
;
;
        MOV     #1,SYNREG       ;S/H IN HOLD
        MOV     @(R5),R3
        BIT     #177760,R3      ;<16
        BNE     GETERR
        ASL     R3
        ADD     #ADC0,R3        ;ADC-CHANNELADDRESS IN R3
        MOV     (R3),(R0)

        CLR     SYNREG
        BIT     #4,STTREG
        BNE     2$
        RTS     PC
2$:     MOV     #12,ERRHYB
        JSR     PC,ERRARG
        RTS     PC
;
;
```

CNTR (data,naam)



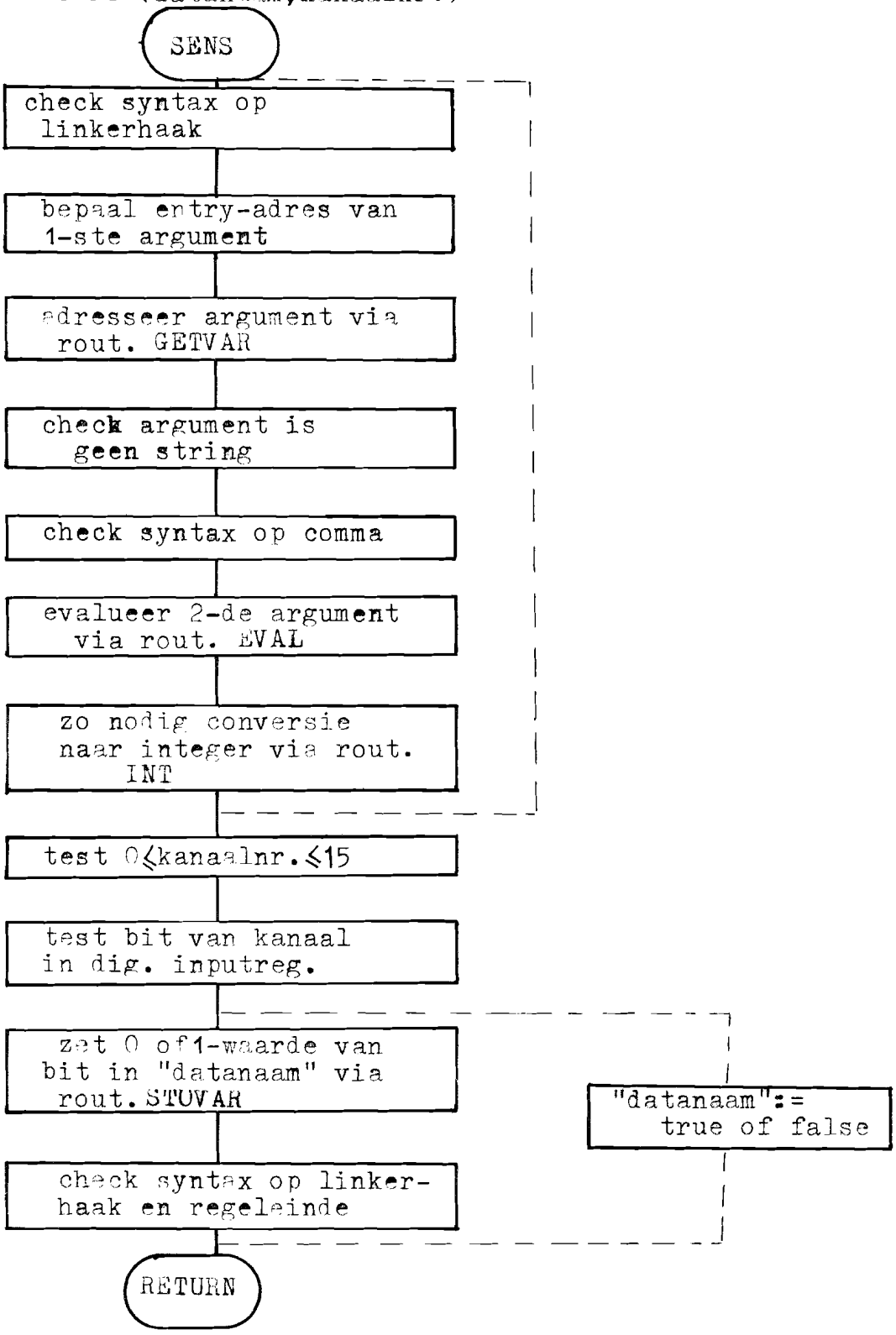
BASIC

```
.SBTTL  ROUTINE CNTR
.GLOBL  CNTR
                        ;CALLING SEQUENSE IN BASIC
                        ;CALL "CNTR"(DATA,CHANNELNUMBER)
DIGOUT= 173622
KAND:   .WORD   0

KAN0:   .WORD   0

CNTR:   CMPB    (R1)+,#.LPAR

        JSR     PC,EVAL                 ;GET FIRST ARGUMENT
        BCS     FALC                    ;DATA NOT NUMERIC
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT

1$:     BIT     #177776,FAC2(R5)
        BNE     FALC
        MOV     FAC2(R5),KAND

        CMPB    (R1)+,#.COMMA
        BNE     FSYN
        JSR     PC,EVAL                 ;GET SECOND ARGUMENT
        BCS     FALC
        TST     FAC1(R5)
        BEQ     2$
        JSR     PC,INT
2$:     MOV     FAC2(R5),R2
        BIT     #177760,R2              ;CHANNEL NUMBER <16
        BNE     FALC
        MOV     #1,KAND
3$:     TST     R2
        BEQ     4$
        ASL     KAND
        DEC     R2
        BR      3$
4$:     TST     KAND
        BEQ     5$
        BIS     KAND,DIGOUT     ;CHANNEL BIT =1
        BR      6$
5$:     BIC     KAND,DIGOUT     ;CHANNEL BIT =0
6$:

        CMPB    (R1)+,#.RPAR
        BNE     FSYN
        CMPB    (R1),#.EOL
        BNE     FSYN
        RTS     PC              ;RETURN TO BASIC
```

FORTRAN

```
.SBTTL  ROUTINE CNTL
.GLOBL  CNTL
                        ;CALLING SEQUENSE IN FORTRAN
                        ;CALL CNTL(DATA,CHANNELNUMBER)
DIGOUT= 173682
KANO:   .WORD   0

KANO:   .WORD   0

CNTL:
        MOV     (R5)+,R2
        MOVB    @(R5)+,KANO



        MOV     @(R5),R2
        BIT     #177760,R2              ;CHANNELNUMBER <16
        BNE     FA1
        MOV     #1,KANO
3$:     TST     R2
        BEQ     4$
        ASL     KANO
        DEC     R2
        BR      3$
4$:     TST     KANO
        BEQ     5$
        BIS     KANO,DIGOUT     ;CHANNEL BIT =1
        BR      6$
5$:     BIC     KANO,DIGOUT     ;CHANNEL BIT =0
6$:     RTS     PC
```

SENS (datanaam,kanaalnr.)



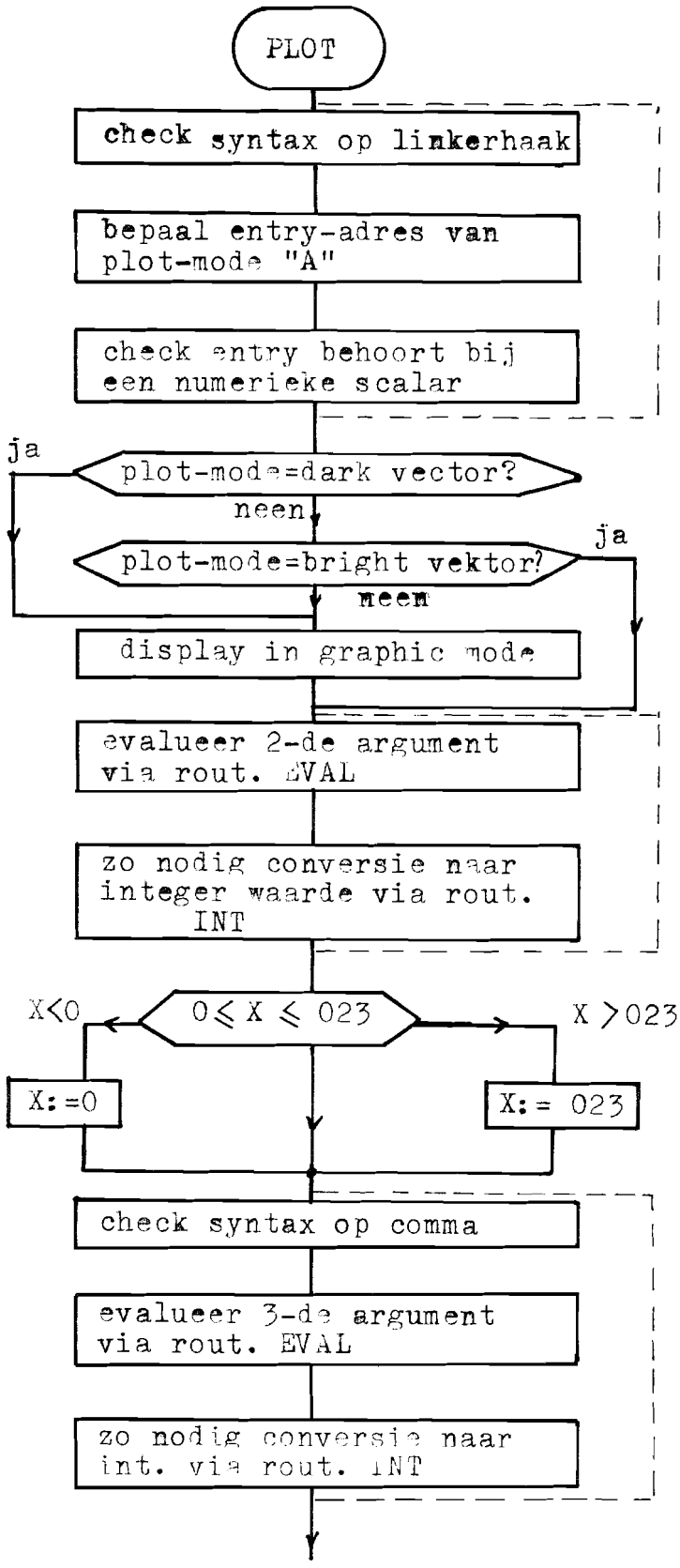
BASIC

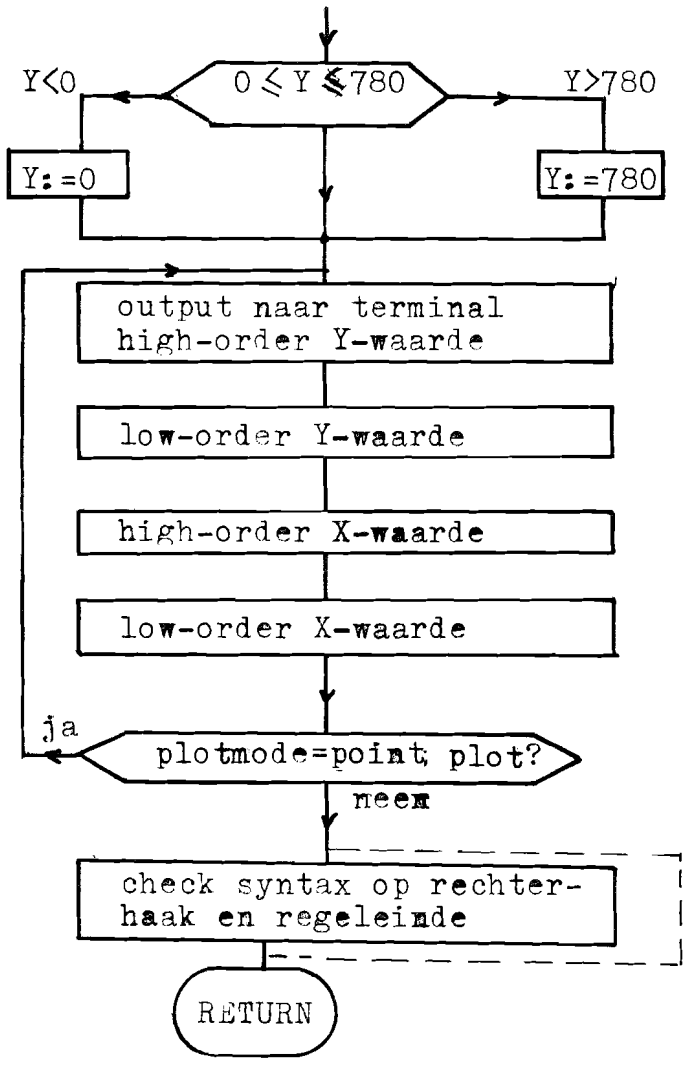
```
        .SBTTL  ROUTINE SENS
.GLOBL  SENS,GETVAR,STOVAR      ;CALLING SEQUENCE IN BASIC
                                ;CALL "SENS"(DATAADRES,CHANNELNUMBER)
VARSAV=22
DIGIN=173622
KAN:    .WORD   0

SENS:   CMPB    (R1)+,#.LPAR    ;CHECK CORRECT STARTING
        BNE     FSYN
        MOVB    (R1)+,R2
        BMI     FSYN
        SUB     R2
        PUSH    (R1)+,R2
        ADD     (R5),R2
        JSR     PC,GETVAR       ;ADRES DATA
        CMP     @VARSAV(R5),#-1 ;CHECK NO STRING
        BEQ     FARG

        CMPB    (R1)+,#.COMMA   ;CHECK COMMA
        BNE     FSYN
        JSR     PC,EVAL
        BCS     FARG            ;DATA NOT NUMERIC
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT
1$:     MOV     FAC2(R5),R2
        BIT     #177760,R2      ;CH.NUMBER <16?
        BNE     FARG
        MOV     #1,KAN
2$:     TST     R2
        BEQ     3$
        ASL     KAN             ;CHANNEL-BIT MOVES ONE PLACE
        DEC     R2
        BR      2$
3$:     CLR     FAC2(R5)
        BIT     KAN,DIGIN
        BEQ     4$              ;CHANNEL-BIT =0
        INC     FAC2(R5)        ;CHANNEL-BIT =1
4$:     JSR     PC,STOVAR       ;0 OR 1 IN DATA ADRES

        CMPB    (R1)+,#.RPAR
        BNE     FSYN
        CMPB    (R1),#.EOL
        BNE     FSYN
        RTS     PC              ;RETURN TO BASIC
FSYN:   JMP     FARSYN
FARG:   JMP     FARGAC
```

FORTRAN

```
.TITLE FDIGIO    DIGITAAL IN EN UIT
.SBTTL  BEGIN SECTION
.CSECT  HYFUN3

.GLOBL FRRARG,FRRHYB,HYDERR
;REGISTER DEFINITIONS
;
R0=     %0
R1=     %1
R2=     %2
R3=     %3
R4=     %4
R5=     %5
STACK=  %6
PC=     %7

        .SBTTL  ROUTINE SENS
.GLOBL  SENS    ;CALLING SEQUENCE IN FORTRAN
                        ;CALL SENS(DATAADRES,CHANNELNUMBER)

DIGIN=177682
KAN:    .WORD   0

SENS:   ADD     #4,R5
        MOV     @(R5),R2
        BIT     #177760,R2          ;CH.NUMBER <16?
        BNE     FARG
        MOV     #1,KAN
2$:     TST     R0
        BEQ     3$
        ASL     KAN                 ;CHANNEL-BIT MOVES ONE PLACE
        DEC     R2
        BR      2$


3$:     CLRB    @-(R5)
        BIT     KAN,DIGIN
        BEQ     4$                  ;CHANNEL-BIT =0
        MOVB    #377,@(R5)          ;CHANNEL-BIT =1
4$:     RTS     PC

FARG:   MOV     #23,FRRHYB
        JMP     ERROR
```

PLOT (A, X, Y)


PLOT
check syntax op linkerhaak
bepaal entry-adres van plot-mode "A"
check entry behoort bij een numerieke scalar
ja
plot-mode=dark vector?
neen
ja
plot-mode=bright vektor?
neen
display in graphic mode
evalueer 2-de argument via rout. EVAL
zo nodig conversie naar integer waarde via rout. INT
X<0
0 ≤ X ≤ 023
X > 023
X:=0
X:= 023
check syntax op comma
evalueer 3-de argument via rout. EVAL
zo nodig conversie naar int. via rout. INT



;high- en low-order waarde zijn bepaald door de vijf bovenste resp. vijf onderste bits van coordinaat-waarde, aangevuld met twee controle bits.

```
;
.TITLE  PLOT & OTHER ROUTINES
.SBTTL  BEGIN SECTION
.CSECT  HYFUN3
.GLOBL PLOT,EVAL,INT,.LPAR,.RPAR,.COMMA,.EOL
.GLOBL ERROR,ERRSYN,CHOUT
;
;REGISTER DEFINITIONS
;
R0=     %0
R1=     %1
R2=     %2
R3=     %3
R4=     %4
R5=     %5
STACK=  %6
PC=     %7
TTKS=   177560
TTKB=   177562
TTPS=   177564
TTPB=   177566
FAC1=   40
FAC2=   42
;
        ;


;       CHOUT
;
;       THIS ROUTINE IS CALLED TO OUTPUT
;       AN ASCII CHARACTER TO THE 4010
;       GRAPHIC COMPUTER TERMINAL
;
;       TO CALL  PUT THE CHARACTER IN REG 0
;       AND EXECUTE A
;
;       JSR     R5,CHOUT
;
;       WILL RETURN  WITH REG 0 UNCHANGED
;
;
CHOUT:  TSTB    TTPS            ;CHECK FOR PUNCH READY
        BPL     CHOUT           ;WAIT FOR READY
        MOVB    R0,TTPB         ;MOVE IN BYTE
        RTS     R5              ;RETURN
;
```

```
.SBTTL  ROUTINE PLOT
;        PLOT
;                                              BASIC
;        THIS ROUTINE IS CALLED TO PLOT
;        IN VECTOR, OR POINT
;        PLOT MODE DEPENDING ON THE
;        VALUE OF A AS DESCRIBED BELOW.
;
;        IF
;        A = 0  INITIALIZE AND DARK VECTOR TO X,Y
;
;        A > 0  BRIGHT VECTOR TO  X,Y
;
;        A < 0  POINT PLOT TO X,Y

;        CALLING SEQUENCE IN BASIC "PLOT"(A,X,Y)
WRSY:   JMP     PARSYN


PLOT:   CMPB    (R1)+,#.LPAR    ;CHECK CORRECT STARTING
        BNE     WRSY
        MOVB    (R1)+,R2
        BMI     WRS
        SUB     R2
        BISB    (R1)+,R2
        ADD     (R5),R2
        CMP     #177775,(R2)+   ;CHECK NUMERIC SCALAR
        BNE     WRA
        MOV     (R2)+,FAC1(R5)
        MOV     (R2),FAC2(R5)
        JSR     PC,INT
        MOV     FAC2(R5),R0             ;PLOT MODE IN R0;
;

        MOV     R0,IPTMOD               ;CHECK REG R0,IPTMODE R0,IPTMOD
         MOV        R0,-(STACK)         ;SAVE R0 ON STACK
         BEQ        IPTDV               ;JUMP IF INIT. AND DARK VECT.
         BPL        IPTNRM              ;JUMP IF NORMAL VECTOR
         MOV        #037,R0
         JSR        R5,CHOUT
IPTDV:   MOV        #035,R0             ;OUTPUT A GS TO INITIALIZE
         JSR        R5,CHOUT            ;SET MODE
;
IPTNRM: CMPB    (R1)+,#.COMMA
        BNE     WRS
        JSR     PC,EVAL         ;GET SECOND ARGUMENT
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT          ;X-COORDINAAT INTEGER
1$:     MOV     FAC2(R5),R0      ;MOVE X COORD TO REG 0
         BPL        IPT1$               ;JUMP IF GEQ 0
         CLR        R0                  ;IF NEG SET TO 0
IPT1$:   CMP        R0,#1024.           ;CHECK FOR ON SCREEN
         BMI        IPT12               ;JUMP IF IN RANGE
         MOV        #1023.,R0           ;SET TO EDGE IF TOO HIGH
IPT12:   MOV        R0,IPTX             ;SAVE X VALUE
        CMPB    (R1)+,#.COMMA
        BNE     WRS
        JSR     PC,EVAL                 ;GET THIRD ARGUMENT
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT                  ;Y-COORDINAAT INTEGER
1$:     MOV     FAC2(R5),R2
```

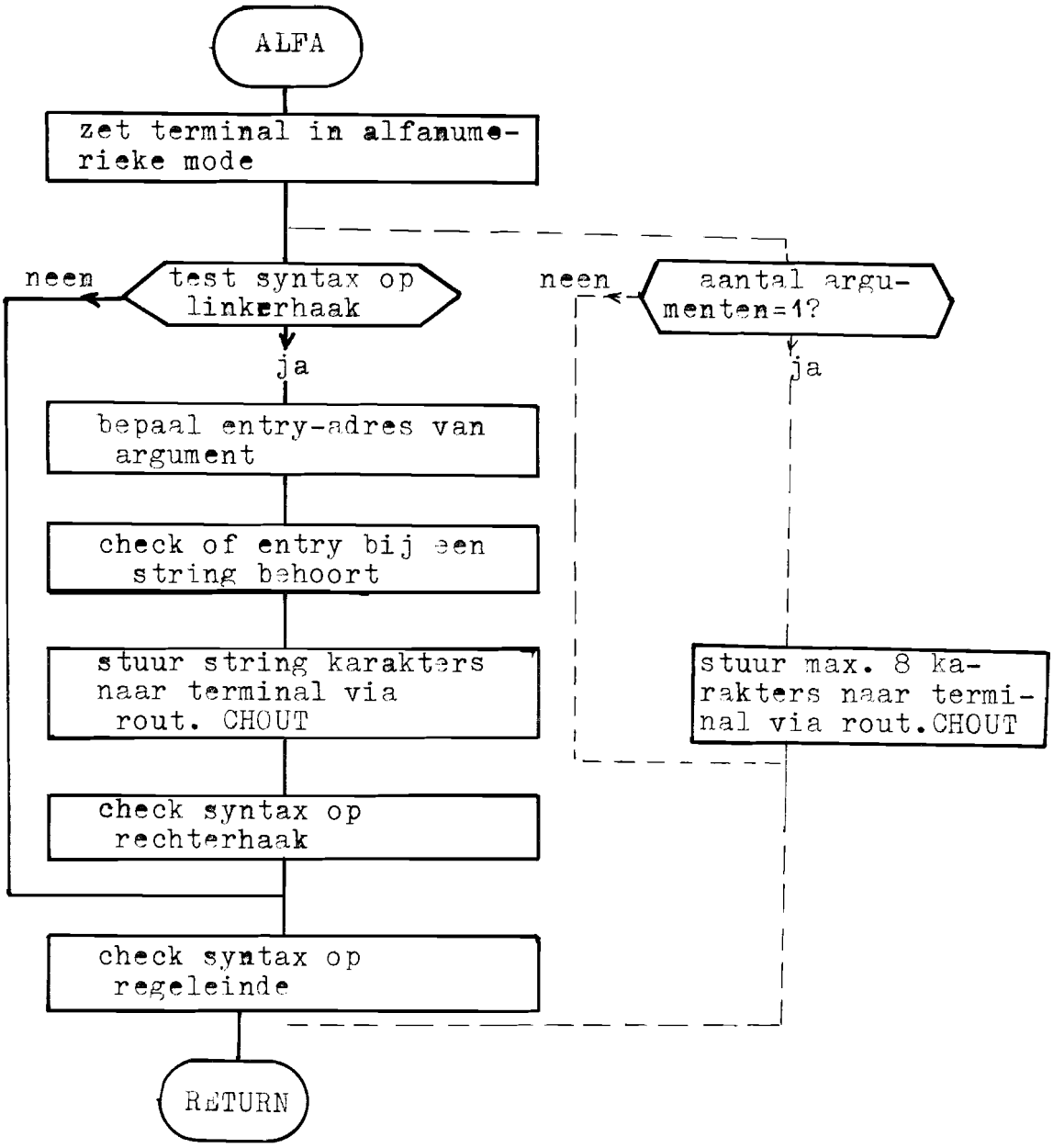
```
        BPL     TPT14           ;JUMP IF GEQ 0
        CLR     R0              ;CLEAR REG 0
TPT14:  CMP     R0,#781.        ;CHECK FOR TOO LARGE Y
        BMI     TPT16           ;JUMP IF IN RANGE
        MOV     #780.,R0        ;MOVE TO EDGE OF SCREEN
TPT16:  MOV     R0,TPTY         ;SAVE Y VALUE

TPTPNT: MOV     TPTY,R0         ;GET Y VALUE
        ROL     R0              ;MOVE UPPER 5 BITS
        ROL     R0              ;TO UPPER BYTE
        ROL     R0
        SWAB    R0              ;SWAP UPPER AND LOWER BYTES
        BIC     #177740,R0      ;MASK OFF EXTRA
        BIS     #000040,R0      ;SET IN HI Y TAG
        JSR     R5,CHOUT        ;OUTPUT HI Y
        MOV     TPTY,R0         ;GET Y COORD
        BIC     #177740,R0      ;MASK TO LOW 5 BITS
        BIS     #000140,R0      ;AND SET LOW Y TAG
        JSR     R5,CHOUT        ;SHIP OUT LOW Y BYTE
;
        MOV     TPTX,R0         ;GET X COORD
        ROL     R0              ;AND ADJUST LIKE Y
        ROL     R0
        ROL     R0
        SWAB    R0              ;SWITCH BYTES
        BIC     #177740,R0      ;MASK OFF EXTRA
        BIS     #000040,R0      ;SET HI X TAG
        JSR     R5,CHOUT        ;OUTPUT HI X BYTE
        MOV     TPTX,R0         ;GET X COORD
        BIC     #177740,R0      ;LEAVE ONLY LOW BITS
        BIS     #000100,R0      ;SET IN LOW X BITS
        JSR     R5,CHOUT        ;OUTPUT LOW X BYTE
        TST     TPTMOD          ;CHECK FOR POINT PLOT
        BPL     RTN             ;RETURN
        CLR     TPTMOD          ;CLEAR AND DRAW VECTOR
        BR      TPTPNT
RTN:    MOV     (STACK)+,R0     ;RESTORE R0 AND EXIT
        CMPB    (R1)+,#.RPAR
        BNE     WRS
        CMPB    (R1),#.EOL
        BNE     WRS
        RTS     PC              ; RETURN TO BASIC


;
TPTX:   .WORD   0
TPTY:   .WORD   0

TPTMOD: .WORD   0

WRS:    JMP     ERRSYN
WRA:    JMP     ERRARG
```

FORTRAN

```
.TITLE  PLOT
.SPTIL  BEGIN SECTION
.CSECT  HIRUNS
.GLOBL PLOT
;
;REGISTER DEFINITIONS
;
AC=     %0
A1=     %1
A2=     %2
A3=     %3
A4=     %4
A5=     %5
STACK=  %6
PC=     %7
TTKS=   177430
TTKB=   177432
TTPS=   177434
TTPB=   177436
        ;


;       CHOUT
;
;       THIS ROUTINE IS CALLED TO OUTPUT
;       AN ASCII CHARACTER TO THE ADM3
;       GRAPHIC COMPUTER TERMINAL
;
;       TO CALL  PUT THE CHARACTER IN REG 0
;       AND EXECUTE A
;
;       JSR     A5,CHOUT
;
;       WILL RETURN WITH REG 0 UNCHANGED
;
;
CHOUT:  TSTB    TTPS            ;CHECK FOR PUNCH READY
        BPL     CHOUT           ;WAIT FOR READY
        MOVB    AC,TTPB         ;MOVE IN BYTE
        RTS     A5              ;RETURN
;

.SBTTL  ROUTINE PLOT
;       PLOT
;
;       THIS ROUTINE IS CALLED TO PLOT
;       IN VECTOR, OR POINT
;       PLOT MODE DEPENDING ON THE
;       VALUE OF A AS DESCRIBED BELOW.
;
;       IF
;       A = 0  INITIALIZE AND DARK VECTOR TO X,Y
;
;       A > 0  BRIGHT VECTOR TO  X,Y
;
;       A < 0  POINT PLOT TO X,Y

;       CALLING SEQUENCE IN FORTRAN PLOT(A,X,Y)
```

```
PLOT:   MOV     (R5)+,R0
        MOV     @(R5)+,R0               ;PLOT MODE IN R0
;
                                                        FORTRAN
        MOV     R0,TPTMOD               ;CHECK REG R0
        MOV     R0,-(STACK)             ;SAVE R0 ON STACK
        BEQ     TPTDV                   ;JUMP IF INIT. AND DARK VECT.
        BPL     TPTNRM                  ;JUMP IF NORMAL VECTOR
        MOV     #037,R0
        JSR     R5,CHOUT
TPTDV:  MOV     #035,R0                 ;OUTPUT A GS TO INITIALIZE
        JSR     R5,CHOUT                ;SET MODE
;
TPTNRM: MOV     @(R5)+,R0       ;MOVE X COORD TO REG 0
        BPL     TPT10                   ;JUMP IF GEQ 0
        CLR     R0                      ;IF NEG SET TO 0
TPT10:  CMP     R0,#1024.               ;CHECK FOR ON SCREEN
        BMI     TPT12                   ;JUMP IF IN RANGE
        MOV     #1023.,R0               ;SET TO EDGE IF TOO HIGH
TPT12:  MOV     R0,TPTX                 ;SAVE X VALUE
        MOV     @(R5)+,R0       ;MOVE Y COORD TO REG 0


        BPL     TPT14                   ;JUMP IF GEQ 0
        CLR     R0                      ;CLEAR REG 0
TPT14:  CMP     R0,#781.                ;CHECK FOR TOO LARGE Y
        BMI     TPT16                   ;JUMP IF IN RANGE
        MOV     #780.,R0                ;MOVE TO EDGE OF SCREEN
TPT16:  MOV     R0,TPTY                 ;SAVE Y VALUE

TPTPNT: MOV     TPTY,R0                 ;GET Y VALUE
        ROL     R0                      ;MOVE UPPER 5 BITS
        ROL     R0                      ;TO UPPER BYTE
        ROL     R0
        SWAB    R0                      ;SWAP UPPER AND LOWER BYTES
        BIC     #177740,R0              ;MASK OFF EXTRA
        BIS     #000040,R0              ;SET IN HI Y TAG
        JSR     R5,CHOUT                ;OUTPUT HI Y
        MOV     TPTY,R0                 ;GET Y COORD
        BIC     #177740,R0              ;MASK TO LOW 5 BITS
        BIS     #000140,R0              ;AND SET LOW Y TAG
        JSR     R5,CHOUT                ;SHIP OUT LOW Y BYTE
;
        MOV     TPTX,R0                 ;GET X COORD
        ROL     R0                      ;AND ADJUST LIKE Y
        ROL     R0
        ROL     R0
        SWAB    R0                      ;SWITCH BYTES
        BIC     #177740,R0              ;MASK OFF EXTRA
        BIS     #000040,R0              ;SET HI X TAG
        JSR     R5,CHOUT                ;OUTPUT HI X BYTE
        MOV     TPTX,R0                 ;GET X COORD
        BIC     #177740,R0              ;LEAVE ONLY LOW BITS
        BIS     #000100,R0              ;SET IN LOW X BITS
        JSR     R5,CHOUT                ;OUTPUT LOW X BYTE
        TST     TPTMOD                  ;CHECK FOR POINT PLOT
        BPL     RTN                     ;RETURN
        CLR     TPTMOD                  ;CLEAR AND BRIGHT VECTOR
        BR      TPTPNT
RTN:    MOV     (STACK)+,R0             ;RESTORE R0 AND EXIT
        RTS     PC                      ; RETURN


;
TPTX:   .WORD   0
TPTY:   .WORD   0
```

ALFA (tekst-string)

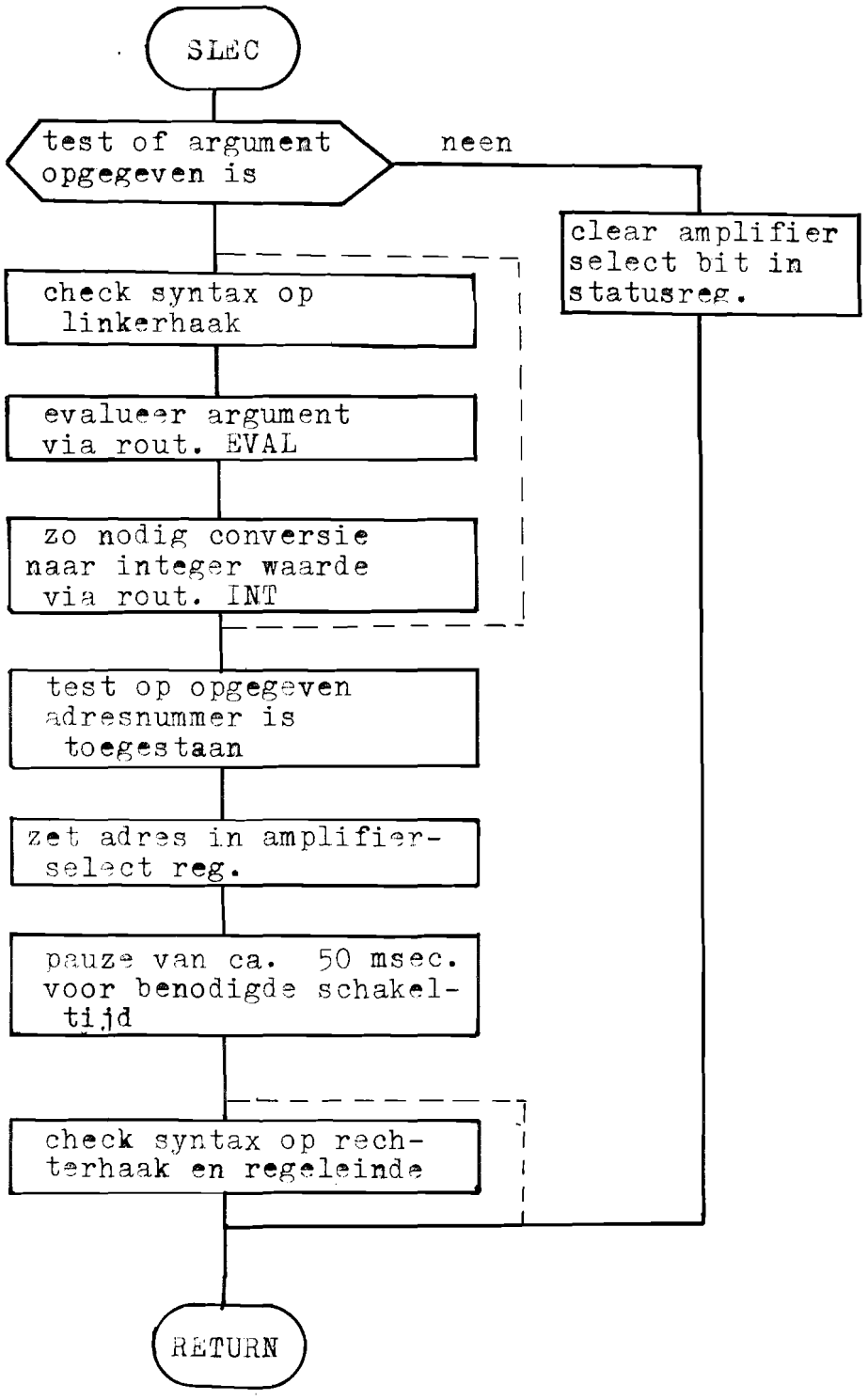


```
.SBTTL  ROUTINE ALFA
.GLOBL  ALFA
;                       CALLING SEQUENCE IN BASIC CALL "ALFA"
;                 ARGUMENT MAY BE SPECIFIED AS ("STRING")
;                       ALFA GIVES RETURN TO ALFANUMERIC MODE
;                       AFTER PLOT CALLS



ALFA:   MOV     #37,R0
        JSR     R5,CHOUT                ;ALFANUMERIC MODE
        CMPB    (R1),#.LPAR
        BNE     3$
        INC     R1
        MOVB    (R1)+,R2
        BMI     ES
        SWAB    R2
        BISB    (R1)+,R2
        ADD     (R5),R2         ;R2: ENTRY POINT STRING
        CMP     #177777,(R2)+           ;CHECK ARGUMENT IS STRING
        BNE     EA
        CMP     #177777,(R2)            ;NUL-STRING?
        BEQ     2$
        MOV     (R2),R2         ;R2: ADRES STRING
        MOVB    (R2)+,R3                ;R3: NUMBER OF CHARACTERS=N
        BMI     EA
        ADD     #2,R2
1$:     MOVB    (R2)+,R0
        JSR     R5,CHOUT                ;CHAR. TO DISPLAY
        DEC     R3              ;N:=N-1
        BNE     1$
2$:     CMPB    (R1)+,#.RPAR
        BNE     ES

3$:     CMPB    (R1)+,#.EOL
        BNE     ES



        RTS     PC                      ;RETURN TO BASIC
ES:     JMP     ERRSYN
EA:     JMP     ERRARG
```

FORTRAN

```
TPTMOD:   .WORD     0

.SBTTL  ROUTINE ALFA
.GLOBL  ALFA
;                       CALLING SEQUENCE IN FORTRAN CALL ALFA
;               ARGUMENT MAY BE SPECIFIED AS STRING(REAL*8)
;                       ALFA GIVES RETURN TO ALFANUMERIC MODE
;                       AFTER PLOT CALLS



ALFA:   MOV     #37,R0
        JSR     R5,CHOUT                ;ALFANUMERIC MODE
        CMPB    (R5)+,#1
        BNE     3$
        INC     R1
        MOV     (R5),R2         ;R2: ADRES STRING
        MOV     #10,R3          ;8 CHARACTERS :N

1$:     MOVB    (R2)+,R0
        JSR     R5,CHOUT                ;CHAR. TO DISPLAY
        DEC     R3              ;N:=N-1
        BNE     1$
3$:
        RTS     PC
        .END
```

SLEC (adres)


SLEC
test of argument opgegeven is
neen
clear amplifier select bit in statusreg.
check syntax op linkerhaak
evalueer argument via rout. EVAL
zo nodig conversie naar integer waarde via rout. INT
test op opgegeven adresnummer is toegestaan
zet adres in amplifier-select reg.
pauze van ca. 50 msec. voor benodigde schakel-tijd
check syntax op rechterhaak en regeleinde
RETURN

BASIC

```
          .PAGE
          .TITLE  FLUKE  ROUTINES
          .SBTTL  CONSTANT SECTION
          .CSECT  HYFUN4
          R0=%100
          R1=%101
          R2=%102
          R3=%103
          R4=%104
          R5=%105
          SP=%106
          PC=%107
;ADRESSEN VAN FLUKE INTERFACE
          STREG=   166072
          DATA1=   166074
          DATA2=   166076
          INSREG=  166062
          LAMPS=   166072
;ANDERE ADRESSEN
          ASLEEG=  173612
          STTREG=  173600
          SYNLEG=  173602
          DACP=    173732
          MODFAC=  173646

;ANDERE CONSTANTEN
          VARSAV=  22
          FAC1=    40
          FAC2=    42
 ;
;GLOBL DEFINITIES

          .GLOBL  SLEC,.DVM,SPOT,TENA
          .GLOBL  .LPAR,.COMMA,.RPAR,.EOL
          .GLOBL  FNUSYN,FNDARG,GETVAR,STOVAR,MSC
          .GLOBL  EVAL,INT
```

BASIC

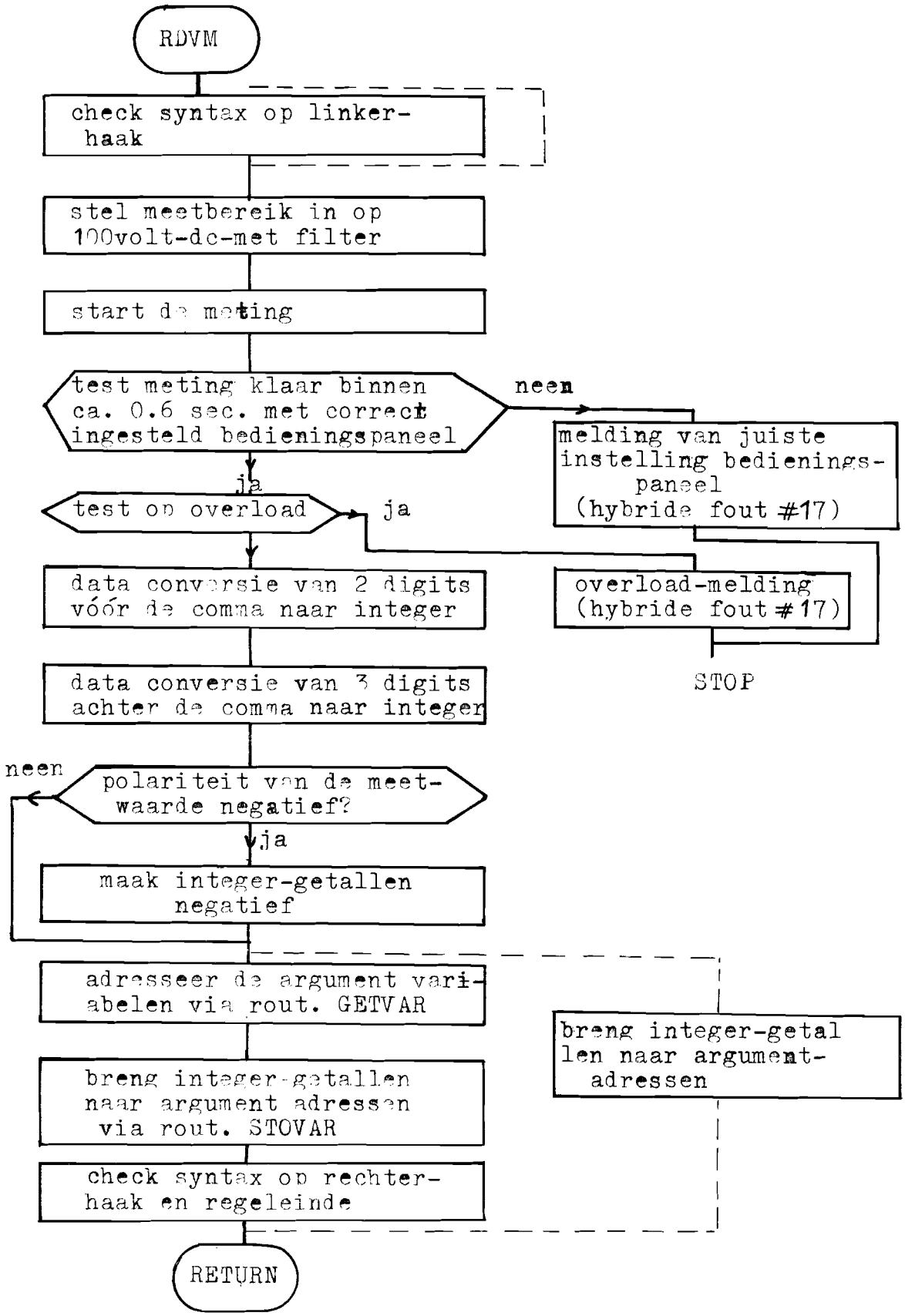
```
        .PAGE
        .SBTTL SELECT-AMPLIFIER-ROUTINE
;
;AANROEP IN BASIC CALL "SLEC"(ADRESNR.)
;ARGUMENT KAN ALS NUMERIEKE EXPRESSIE OPGEGEVEN WORDEN
;SLEC CHECKT NUMMER VAN ADRES EN ZET,INDIEN ADRES CORRECT,
;DIT IN HET AMPLIFIER SELECT REGISTER EN GEEFT DAN EEN PAUZE
;VAN CA. 150 MSEC.
;"SLEC" ZONDER ARGUMENT MAAKT SELECT-BUS VRIJ
SLEC:   CMPB    (R1),#.EOL
        BNE     SAMP
        BIC     #40,SITREG
        RTS     PC

SAMP:   CMPB    (R1)+,#.LPAR            ;CHECK LINKERHAAK
        BNE     SE
        JSR     PC,EVAL                 ;EVALUEER EXPRESSIE
        BCS     AE                      ;ARG. NIET NUMERIEK
        TST     FAC1(R5)
        BPL     1$
        JSR     PC,INT                  ;INTEGER
1$:     MOV     FAC2(R5),R2             ;R2 BEVAT ADRESNR.
        BMI     AE                      ;ADRES NEGATIEF
        CMP     R2,#252                 ;ADRES < 170 ?
        BPL     AE
        CMP     R2,#144                 ;ADRES < 100
        BPL     2$                      ;99<ADRES<170
        CMP     R2,#136                 ;ADRES< 70

        BPL     AE
2$:     MOV     R2,ASLREG               ;ADRESNR. NAAR REG
        MOV     #77777,R2
3$:     DEC     R2                      ;PAUZE VAN CA, 150 MSEC.
        BPL     3$                      ;DEC+BPL="5USEC."
        CMPB    (R1)+,#.RPAR
        BNE     SE
        CMPB    (R1),#.EOL
        BNE     SE
        RTS     PC                      ;TERUG NAAR BASIC
```

```
        .PAGE
        .TITLE  FFLUKE  ROUTINES
        .SBTTL  CONSTANT SECTION
        .CSECT  HYFUN4
        R0=%0
        R1=%1
        R2=%2
        R3=%3
        R4=%4
        R5=%5
        SP=%6
        PC=%7
;ADRESSEN VAN FLUKE INTERFACE
        STREG=   166070
        DATA1=   166074
        DATA2=   166076
        INSREG=  166062
        RAPEG=   166072
;ANDERE ADRESSEN
        ASLREG=  173612
        STTREG=  173600
        SYNREG=  173602
        DAC0=    173700
        MODFAC=  173606

;ANDERE CONSTANTEN
        VARSAV=  22
        FAC1=    40
        FAC2=    42
 ;
;GLOBL DEFINITIES

        .GLOBL  SLEC,ADV4,SPOT,PENX,FRFARG,FARHYB
```

FORTRAN

```
        .PAGE
        .SBTTL SELECT-AMPLIFIER-ROUTINE
;
;AANROEP IN FORTRAN CALL SLEC(ADRESNR.)
;ARGUMENT KAN ALS NUMERIEKE EXPRESSIE OPGEGEVEN WORDEN
;SLEC CHECKT NUMMER VAN ADRES EN ZET,INDIEN ADRES CORRECT,
;DIT IN HET AMPLIFIER SELECT REGISTER EN GEEFT DAN EEN PAUZE
;VAN CA. 150 MSEC.
;"SLEC" ZONDER ARGUMENT MAAKT SELECT-BUS VRIJ
SLEC:   CMPB    (R5)+,#1
        BEQ     SAMP
        BIC     #40,@TTREG
        RTS     PC

SAMP:   INC     R5
        MOV     @(R5)+,R2               ;R2 BEVAT ADRESNR.
        BMI     AF1                     ;ADRES NEGATIEF
        CMP     R2,#252                 ;ADRES < 170 ?
        BPL     AF1
        CMP     R2,#144                 ;ADRES < 100
        BPL     2$                      ;99<ADRES<170
        CMP     R2,#106                 ;ADRES< 70

        BPL     AF1
2$:     MOV     R2,@SLEREG              ;ADRESNR. NAAR REG
        MOV     #77777,R2
3$:     DEC     R2                      ;PAUZE VAN CA, 150 MSEC.
        BPL     3$                      ;DEC+BPL="5USEC."
        RTS     PC
AF1:    MOV     #20,@#HYP
        JMP     @#ADAG


        .PAGE
        .SBTTL  ROUTINE TENX
;ROUTINE TENX WORDT GEBRUIKT BIJ HET UITLEZEN VAN DE DVM
;TENX VERMENIGVULDIGT EEN GETAL ,OPGESLAGEN IN R2 MET EEN FACTOR 10
;EINDRESULTAAT IS OPGESLAGEN IN R2


TENX:   ASL     R2                      ;2*
        MOV     R2,R3
        ASL     R2
        ASL     R2                      ;R2=8*
        ADD     R3,R2                   ;R2=10*
        RTS     PC
```

RDVM (integer ,integer?)



BASIC

```
        .PAGE
        .SBTTL  ROUTINE TENX
;ROUTINE TENX WORDT GEBRUIKT BIJ HET UITLEZEN VAN DE DVM
;TENX VERMENIGVULDIGT EEN GETAL OPGESLAGEN IN R2 MET EEN FACTOR 10
;EINDRESULTAAT IS OPGESLAGEN IN R2

TENX:   ASL     R2                      ;2*
        MOV     R2,R3
        ASL     R2
        ASL     R2                      ;R2=8*
        ADD     R3,R2                   ;R2=10*
        RTS     PC

        .PAGE
        .SBTTL  READ DVM
;AANROEP IN BASIC CALL "RDVM"(D1,D2)
;D1 IS EEN INTEGER DIE DE TWEE DIGITS VOOR DE COMMA WEERGEEFT
;D2 IS EEN INTEGER DIE DE DRIE DIGITS ACHTER DE COMMA WEERGEEFT
;DE GEMETEN SPANNINGSWAARDE IS D1+0,001*D2 VOLT

MELD:   JSR     R1,MSG
        .ASCII  'BIJ GEBRUIK VAN FLUKE DVM EERST FUNCTION '
        .BYTE   15
        .BYTE   12

        .ASCII  'OP REMOTE EN DAN SAMPLE-RATE OP EXT.'

        .BYTE   15
        .BYTE   12
        .BYTE   0
        .EVEN
        BR      AF


T:      .WORD   0,0,0
D1:     .WORD   0
D2:     .WORD   0
C:      .WORD   0
SF:     JMP     ERRSYN
AF:     JMP     FLAG
```

BASIC

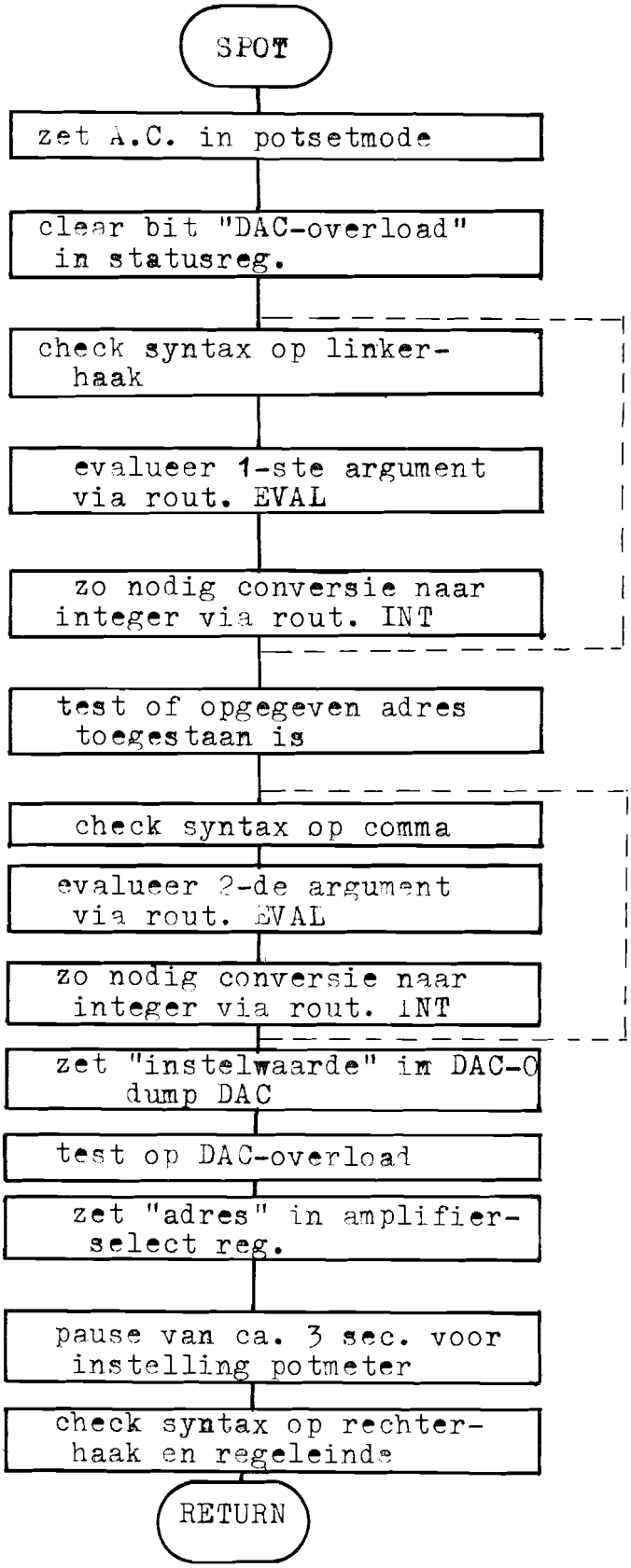
```
RDVM:
        CMPB    (R1)+,#.LPAR
        BNE     5F
        CLR     DATA1                   ;AFZETTEN VAN BUSY-BIT
        MOV     #7737,INSREG            ;INSTELLING MEETBEREIK OP
                                        ;100V-DC-MET FILTER
        CLR     RAREG                   ;START MEETING

        MOV     #167777,R2

1$:     DEC     R2

        BEQ     MELD                    ; CA. 12 USEC.
        TSTB    STAREG                  ;MEETING KLAAR?
        BPL     1$                      ;TESTTIJD CA.0.6 SEC
        BIT     #1,RAREG                ;TEST FUNCTION DVM OP REMOTE
        BEQ     MELD
        BIT     #2,RAREG                ;KNOP SAMPLE-RATE OP EXT.
        BEQ      MELD

        BIT     #40000,STREG            ;OVERLOAD TEST
        BEQ     2$
        JSR     R1,MSG
        .ASCII  '"DVM IN OVERLOAD"'
        .BYTE   15
        .BYTE   12
        .BYTE   3
        .EVEN
        BR      AF
2$:     MOV     DATA2,R2
        BIC     #177760,R2              ;1E DIGIT RESTERAT
        JSR     PC,TENX
        MOV     DATA1,R3
        BIC     #177760,R3              ;2E DIGIT RESTERAT
        ADD     R3,R2           ;R2=10D1+D2
        BIT     #20,DATA2
        BEQ     8$
        ADD     #144,R2                 ;R2=100+10D1+D2
8$:
        MOV     R2,D1
```

```
;
        MOV     DATA1,R2
        MOV     #2,R3
        MOV     #T,R4
3$:     ROR     R2
        ROR     R2
        ROR     R2
        ROR     R2
        MOV     R2,(R4)
        BIC     #177760,(R4)+           ;DIGIT IN (R4)
        DEC     R3
        BPL     3$
        MOV     -(R4),R0                ;R0 BEVAT 5E DIGIT
        MOV     -(R4),R2                ;R2 BEVAT 4E DIGIT
        JSR     PC,TENX                 ;R2:=10*4E DIGIT
        ADD     R2,R0
        MOV     -(R4),R2                ;R2 BEVAT 3E DIGIT
        JSR     PC,TENX
        JSR     PC,TENX
        ADD     R2,R0                   ;R0=100D3+10D4+D5
        MOV     R0,D2
;BEPALING TEKENBIT

        BIT     #200,RAREG
        BNE     4$
        BR      5$
4$:     NEG     D2                      ;TEKEN NEGATIEF
        NEG     D1

;OPBERGEN VAN D1 EN D2 IN ARGUMENT VARIABELEN

5$:     CLR     C
        MOV     #D1,R4
6$:     MOVB    (R1)+,R2
        BMI     5E
        SWAB    R2
        BISB    (R1)+,R2
        ADD     (R5),R2
        JSR     PC,GETVAR
        CMP     @VARSAV(R5),#-1         ;NO STRING

        BEQ     4E
        MOV     (R4)+,FAC2(5)
        JSR     PC,STOVAR               ;INT. WAARDE DIGIT
        CMP     C,#1                    ;IN ARGUMENT
        BEQ     7$
        CMPB    (R1)+,#.COMMA
        BNE     5E8
        INC     C
        BR      6$                      ;VOLGEND ARGUMENT
7$:
        CMPB    (R1)+,#.RPAR
        BNE     5E8
        CMPB    (R1),#.EOL
        BNE     5E2
        RTS     PC                      ;TERUG NAAR BASIC
```

```
        .PAGE
        .SBTTL  READ DVM
;AANROEP IN FORTRAN CALL RDVM(D1,D2)                    FORTRAN
;D1 IS EEN INTEGER DIE DE TWEE DIGITS VOOR DE COMMA WEERGEEFT
;D2 IS EEN INTEGER DIE DE DRIE DIGITS ACHTER DE COMMA WEERGEEFT
;DE GEMETEN SPANNINGSWAARDE IS D1+0,001*D2 VOLT


T:      .WORD   0,0,0
D1:     .WORD   0
D2:     .WORD   0
C:      .WORD   0


AF:     MOV     #21,@#177570
        JMP     @#ARG



RDVM:
        CLR     DATA1                   ;AFZETTEN VAN BUSY-BIT
        MOV     #7737,INSREG            ;INSTELLING MEETBEREIK OP

                                        ;100V-DC-MET FILTER
        CLR     MAREG                   ;START MEETING

        MOV     #167777,R2

1$:     DEC     R2


        BEQ     AF                      ; CA. 12 USEC.
        TSTB    STREG                   ;MEETING KLAAR?
        BPL     1$                      ;TESTTIJD CA.2.6 SEC
        BIT     #1,MAREG                ;TEST FUNCTION DVM OP REMOTE
        BEQ     AF
        BIT     #2,MAREG                ;KNOP SAMPLE-RATE OP EXT.
        BEQ      AF


        BIT     #40000,STREG            ;OVERLOAD TEST
        BEQ     2$
        BR      AF
2$:     MOV     DATA2,R2
        BIC     #177760,R2              ;1E DIGIT AFGESCHEIDEN
        JSR     PC,TENX
        MOV     DATA1,R3
        BIC     #177760,R3              ;2E DIGIT AFGESCHEIDEN
        ADD     R3,R2           ;R2=10D1+D2
        BIT     #20,DATA2
        BEQ     8$
        ADD     #144,R2                 ;R2=100+10D1+D2
8$:
        MOV     R2,D1
;
```

```
        MOV     DATA1,R2
        MOV     #2,R3
        MOV     #1,R4
3$:     ROR     R2
        ROR     R2
        ROR     R2
        ROR     R2
        MOV     R2,(R4)
        BIC     #177760,(R4)+           ;DIGIT IN (R4)
        DEC     R3
        BPL     3$
        MOV     -(R4),R0                ;R0 BEVAT 5E DIGIT
        MOV     -(R4),R2                ;R2 BEVAT 4E DIGIT
        JSR     PC,TENX                 ;R2:=10*4E DIGIT
        ADD     R2,R0
        MOV     -(R4),R2                ;R2 BEVAT 3E DIGIT
        JSR     PC,TENX
        JSR     PC,TENX
        ADD     R2,R0                   ;R0=100D3+10D4+D5
        MOV     R0,D2
;BEPALING TEKENBIT

        BIT     #200,MAREG
        BNE     4$
        BR      5$
4$:     NEG     D2                      ;TEKEN NEGATIEF
        NEG     D1

;OPBERGEN VAN D1 EN D2 IN ARGUMENT VARIABELEN

5$:     MOV     (R5)+,R0
        CLR     C
        MOV     #D1,R4
6$:     MOV     (R4)+,@(R5)+
        CMP     C,#1
        BEQ     7$
        INC     C
        BR      6$                      ;VOLGEND ARGUMENT
7$:     RTS     PC
```

SPOT (adres,instelwaarde)



BASIC

```
        .PAGE
        .SBTTL  SET POT.METER
AANROEP IN BASIC CALL "SPOT"(ADRES,WAARDE)

SPOT CONTROLEERT ADRESNR. EN INSTELWAARDE
GEEFT EEN PAUZE VAN CA.3SEC. NA HET DOORGEVEN
VAN ADRESNR. EN INSTELWAARDE

TMP:    .WORD   0

POT:    MOV     #10,MODDAC              ;POTSET MODE
        BIC     #10,STTREG              ;BIT VAN DAC-OVERLOAD
        CMPB    (R1)+,#.LPAR
        BNE     SER
        JSR     PC,EVAL
        BCS     AER
        TST     FAC1(R5)
        BEQ     1$
        JSR     PC,INT
1$:     MOV     FAC2(R5),R2             ;R2 BEVAT ADRESNR.
        BMI     AER
        CMP     R2,#376                 ;ADRES<254
        BPL     AER
        CMP     R2,#332                 ;ADRES>217
        BMI     AER
        MOV     R2,TEMP
        CMPB    (R1)+,#.COMMA

        BNE     SER
        JSR     PC,EVAL
        BCS     AER
        TST     FAC1(R5)
        BEQ     2$
        JSR     PC,INT

2$:     MOV     FAC2(R5),DACR
        BMI     AER                     ;NEGATIEVE INSTELWAARDE
        CLR     SINREG
        MOV     #2,SINREG               ;DUMP DAC
        BIT     #10,STTREG              ;TEST DAC OVERLOAD
        BNE     AER
        MOV     TEMP,ASLREG
        MOV     #23,R3                  ;PAUZE VAN 3SEC.
3$:     MOV     #77777,R2
4$:     DEC     R2
        BPL     4$                      ;20*32*10EXP3*5USEC.=3,2SEC.
        DEC     R3
        BPL     3$
        CMPB    (R1)+,#.RPAR
        BNE     SER
        CMPB    (R1),#.EOL
        BNE     SER
        RTS     PC                      ;TERUG NAAR BASIC
SER:    JMP     ERRSYN
AER:    JMP     ERRARG
```

FORTRAN

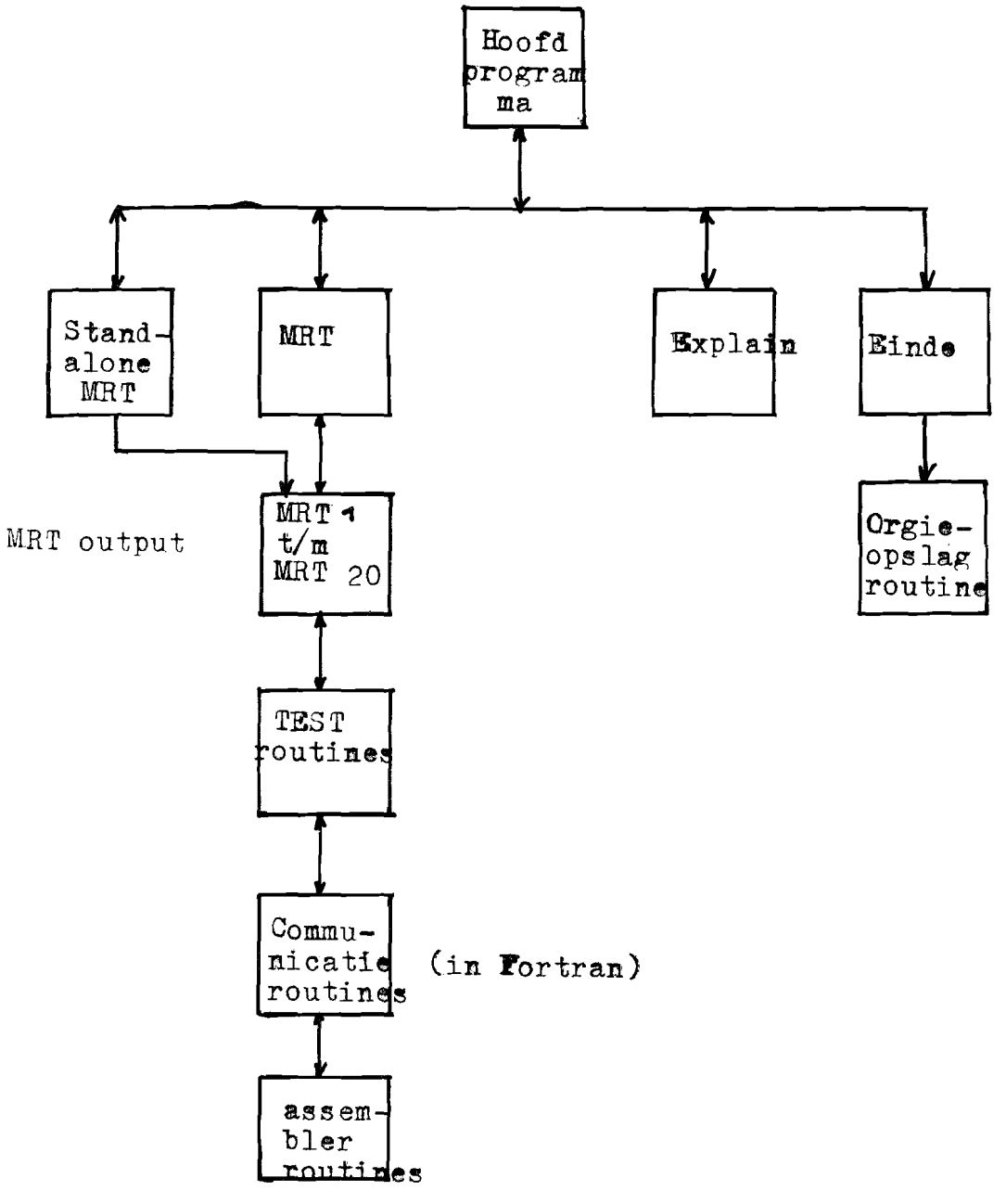
```
        .PAGE
        .SBTTL  SET POT.METER
AANROEP IN FORTRAN CALL SPOT(ADRES,WAARDE)

SPOT CONTROLEERT ADRESNR. EN INSTELWAARDE
GEEFT EEN PAUZE VAN CA.3SEC. NA HET DOORGEVEN
VAN ADRESNR. EN INSTELWAARDE

'EMP:   .WORD   0

)POT:   MOV     #10,MODEAC              ;POTSET MODE
        BIC     #10,STTREG              ;BIT VAN DAC-OVERLOAD
        MOV     (R5)+,R0
        MOV     @(R5)+,R2               ;R2 BEVAT ADRESNR.
        BMI     AER
        CMP     R2,#376                 ;ADRES<254
        BPL     AER
        CMP     R2,#332                 ;ADRES>217
        BMI     AER
        MOV     R2,TEMP

        MOV     @(R5)+,DAC0
        BMI     AER                     ;NEGATIEVE INSTELWAARDE
        CLR     SYNREG
        MOV     #2,SYNREG               ;DUMP DAC
        BIT     #10,STTREG              ;TEST DAC OVERLOAD
        BNE     AER
        MOV     TEMP,ASLREG
        MOV     #23,R3                  ;PAUZE VAN 3SERC.
1$:     MOV     #77777,R2
2$:     DEC     R2
        BPL     4$                      ;20*32*10EXP3*5USERRC.=3,2SERC.
        DEC     R3
        BPL     3$
        RTS     PC
)ER:    MOV     #22,ERRHYB
        JMP     ERRANG
```

```
ERRANG: MOV     #TABEL,R5
        JSR     PC,HYBERR
        RTS     PC
TABEL:  .BYTE   0,1
        .WORD   ERRHYB

ERRHYB: .WORD   0

FAR:    MOV     #24,ERRHYB
        JMP     ERRANG
        .END
*
```

```
        SUBROUTINE HYBERR(ERRNUM)
        INTEGER ERRNUM
        WRITE(7,100) ERRNUM
100     FORMAT('0HYBRIDE FOUT #',I2)
        IF (ERRNUM.LT.10.AND.ERRNUM.GT.4) GO TO 10
        STOP
10      RETURN
        END
```

In deze appendix zijn de volgende punten opgenomen:

Gewijzigde programma-structuur van "Caron"-testpakket

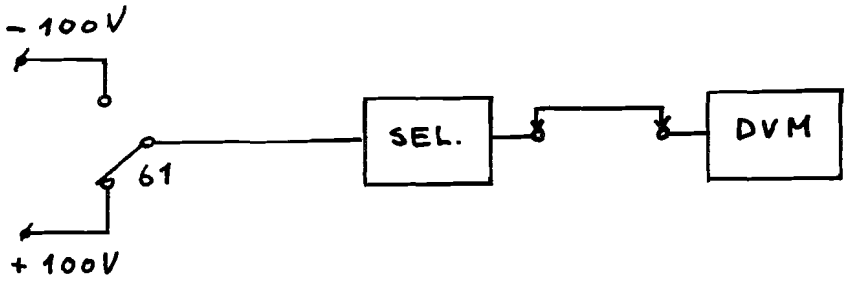


```
      SUBROUTINE INSEQ(N,BADRES,EADRES,AIN)
      DIMENSION AIN(N)
      INTEGER N,BADRES,EADRES,MODE,AANTAL,AIN

      MODE=1
      AANTAL=N/(EADRES-BADRES+1)

      CALL LOOP(MODE,AANTAL)
      CALL IN(AIN,BADRES,EADRES)
      MODE=3
      CALL ENDL(MODE)

      RETURN

C****** DEZE ROUTINE NEEMT OP ELKE TIMERPULS EEN SAMPLE
C****** VAN ALLE S/H KANALEN MET DE NUMMERS BADRES TOT EN MET
C****** EADRES. HET TOTALE AANTAL SAMPLES IS N

      END

      SUBROUTINE COMPII(N,ADRES,CIN)
      DIMENSION CIN(N)
      INTEGER N,ADRES,CIN,MODE
      MODE=1

      CALL LOOP(MODE,N)
      CALL IN(CIN,ADRES,ADRES)
      CALL ENDL(MODE)
      RETURN

C****** ROUTINE NEEMT OP ELKE TIMERPULS EEN SAMPLE VAN
C****** KANAAL 'ADRES' TOT ER N SAMPLES GENOMEN
C****** ZIJN. OPSLAG IN ARRAY CIN. MODE BLIJFT ONGEWIJZIGD.

      END


      SUBROUTINE INREP(N,ADRES,CIN)
      DIMENSION CIN(N)
      INTEGER N,ADRES,CIN,MODE
      MODE=1

      CALL LOOP (MODE,N)
      CALL IN(CIN,ADRES,ADRES)
      MODE=3
      CALL ENDL(MODE)
      RETURN

C*****  ROUTINE NEEMT OP ELKE TIMERPULS EEN SAMPLE VAN
C*****  KANAAL 'ADRES' TOT ER N SAMPLES GENOMEN ZIJN.
C*****  OPSLAG IN ARRAY CIN. MODE NA AFLOOP RESET.

      END

*
```

```
      SUBROUTINE AMPOV(NCONS,ADRES,N,BIN,OVER)
      DIMENSION BIN(N)
      INTEGER NCONS,ADRES,N,BIN,OVER,D1,D2,ADRES1
      OVER=1
      ADRES1=ADRES
      IF (NCONS.EQ.1) ADRES1=ADRES1+100
      CALL SLEC(ADRES1)
      DO 10 I=1,N
      CALL RDVM(D1,D2)
      BIN(I)=(D1+0.001*D2)*163.84
10    CONTINUE
      IF(BIN(N).GT.16382.OR.BIN(N).LT.-16384) OVER=-1
      RETURN

C****** DEZE ROUTINE LEEST EEN ADRES OP CONSOLE  A OF B
C****** (NCONS=1) N-MAAL. ALS LAATSTE SAMPLE OVERLOAD
C****** GEEFT DAN WORDT OVER -1.

      END

      SUBROUTINE TEST20(NCONS,ADRES,N,EIN)
      DIMENSION EIN(N)
      INTEGER NCONS,ADRES,N,EIN,D1,D2,ADRES1
      ADRES1=ADRES
      IF (NCONS.EQ.1) ADRES1=ADRES1+100
      CALL SLEC(ADRES1)
      CALL RDVM(D1,D2)
      DO 10 I=1,N
      EIN(I)=(D1+0.001*D2)*163.84
10    CONTINUE
      RETURN

C*****  ROUTINE WORDT GEBRUIKT VOOR HET METEN VAN DE
C*****  VOEDINGSSPANNINGEN -100 EN +100 VOLT

      END
```

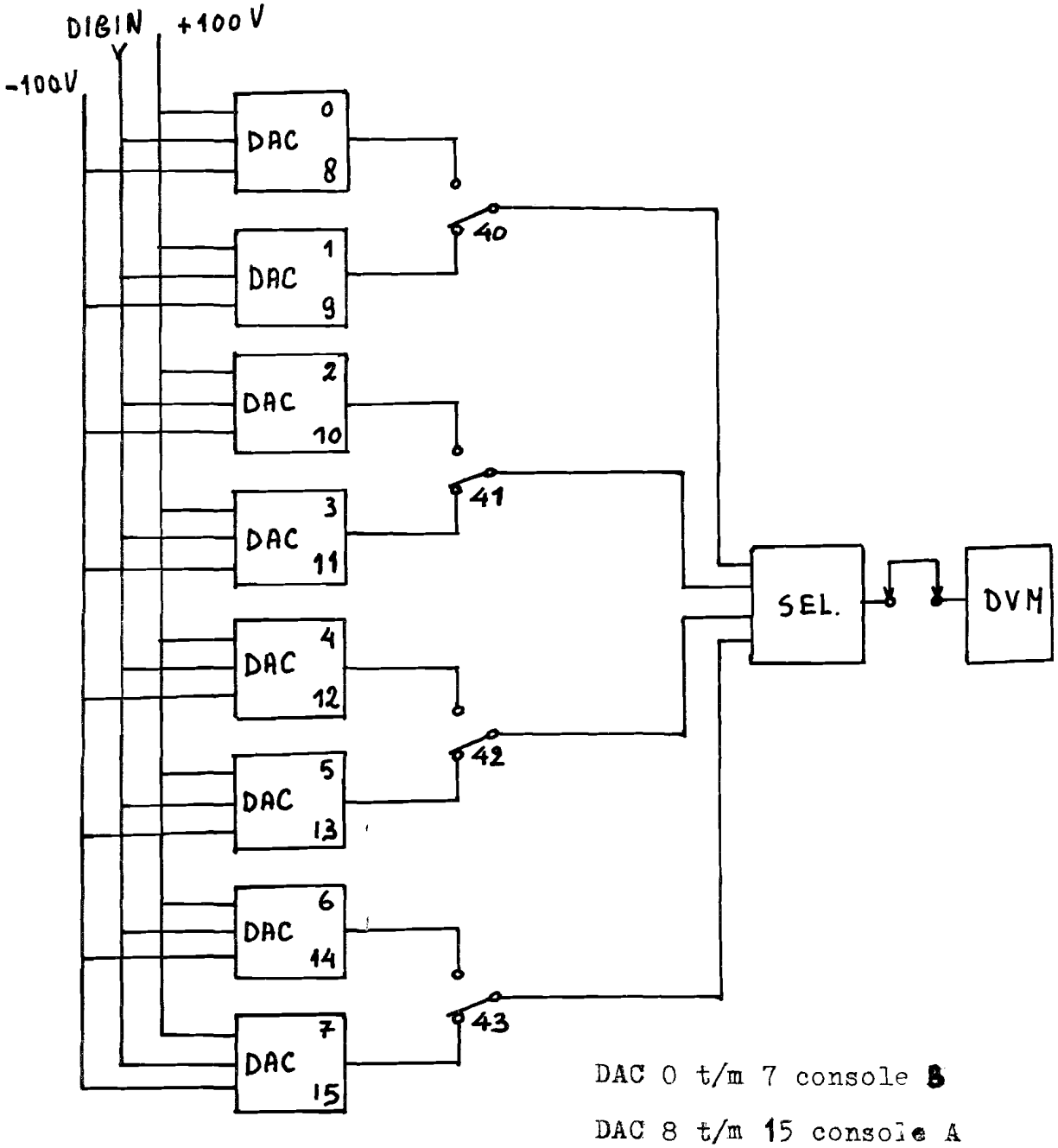
```
      SUBROUTINE DDOUT(FIN)
      DIMENSION FIN(8)

      LOGICAL*1 FIN
      DO 10 I=1,4
      CALL CNTR(FIN(I),I-1)
10    CONTINUE
      DO 20 I=9,12
      CALL CNTR(FIN(I-4),I-1)
20    CONTINUE
      RETURN

C****** ROUTINE ZET DE DIGITALE OUTPUT-KANALEN AFHANKELIJK
C****** VAN DE WAARDEN VAN DE ELEMENTEN VAN ARRAY FIN

      END

      SUBROUTINE DDIN(GIN)
      DIMENSION GIN(8)

      LOGICAL*1 GIN

      DO 10 I=1,4
      CALL SENS(GIN(I),I-1)
10    CONTINUE
      DO 20 I=9,12
      CALL SENS(GIN(I-4),I-1)
20    CONTINUE
      RETURN

C*****  ROUTINE LEEST DE DIGITALE INPUTKANALEN EN PLAATST
C*****  DE WAARDE (.TRUE. OR .FALSE.) IN ARRAY GIN

      END
```

```
      SUBROUTINE DUMPIE(DIS1,DIS2,IN1,IN2)
      INTEGER DIS1,DIS2,IN1,IN2

      CALL PUT (IN1,DIS1)
      CALL PUT (IN2,DIS2)
      RETURN

C****** ROUTINE ZET DE BITPATRONEN, DIE OVEREENKOMENMET DE
C****** GETALLEN IN1 EN IN2 OP DE DAC-KANALEN DIS1 EN DIS2

      END



      SUBROUTINE TIMERP(EXP,MULT)
      INTEGER EXP,MULT

      CALL TINT(MULT,EXP)
      RETURN

C****** TIMER VAN INTERFACE OP MULT*10EXP*10(-5) SEC.

      END


      SUBROUTINE MODEAC(MODE)
      INTEGER MODE

      IF (MODE.EQ.0) CALL RESE
      IF (MODE.EQ.1) CALL COMP
      IF (MODE.EQ.2) CALL HOLD
      IF (MODE.EQ.3) CALL POTS
      IF (MODE.EQ.4) CALL ALLR
      RETURN

C*****  BEDIENING VAN DE MODE VAN DE A.C.

      END
```



1 -Meting van de referentie-spanningen

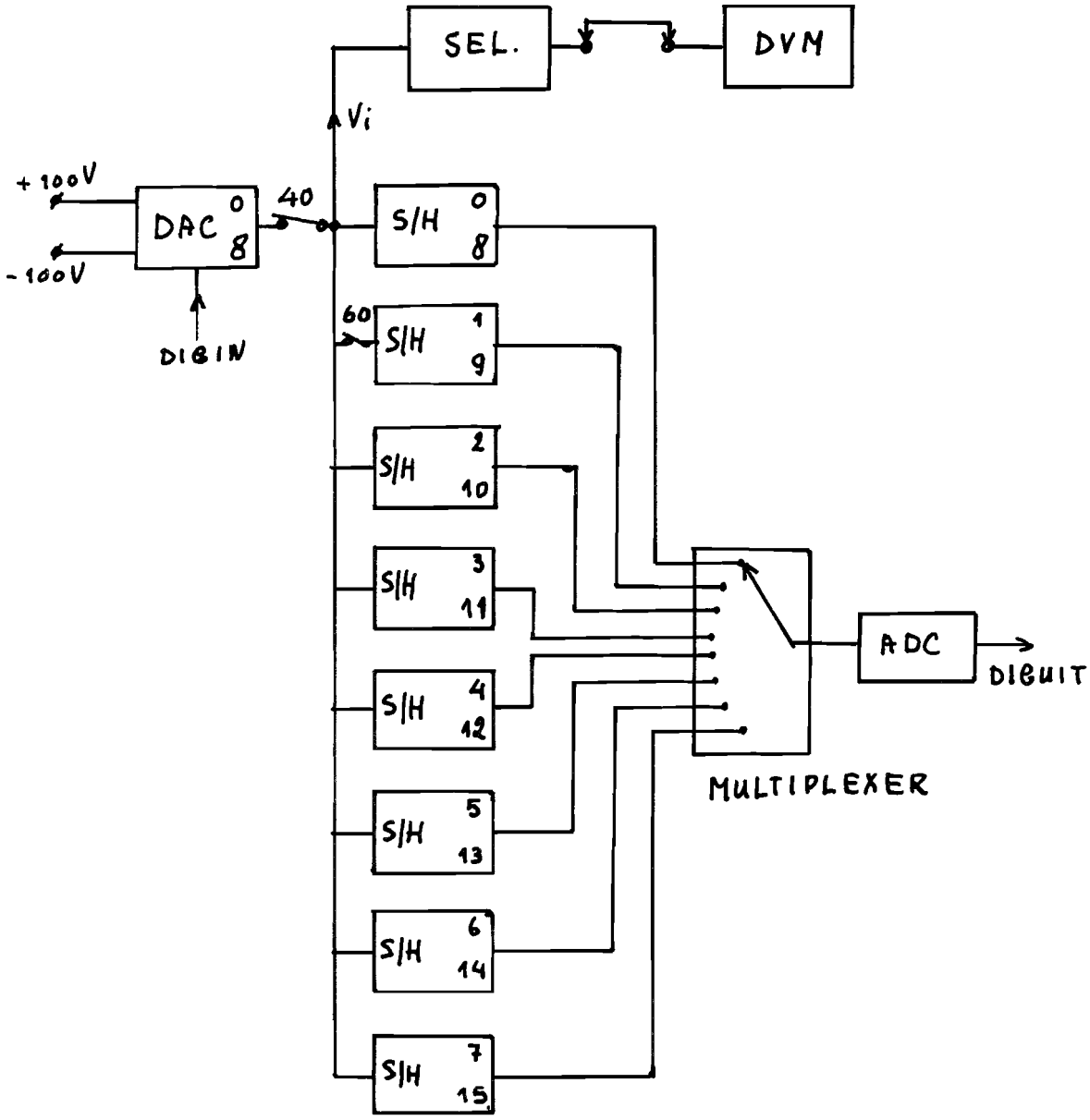
2- DAC-test



DAC 0 t/m 7 console B

DAC 8 t/m 15 console A

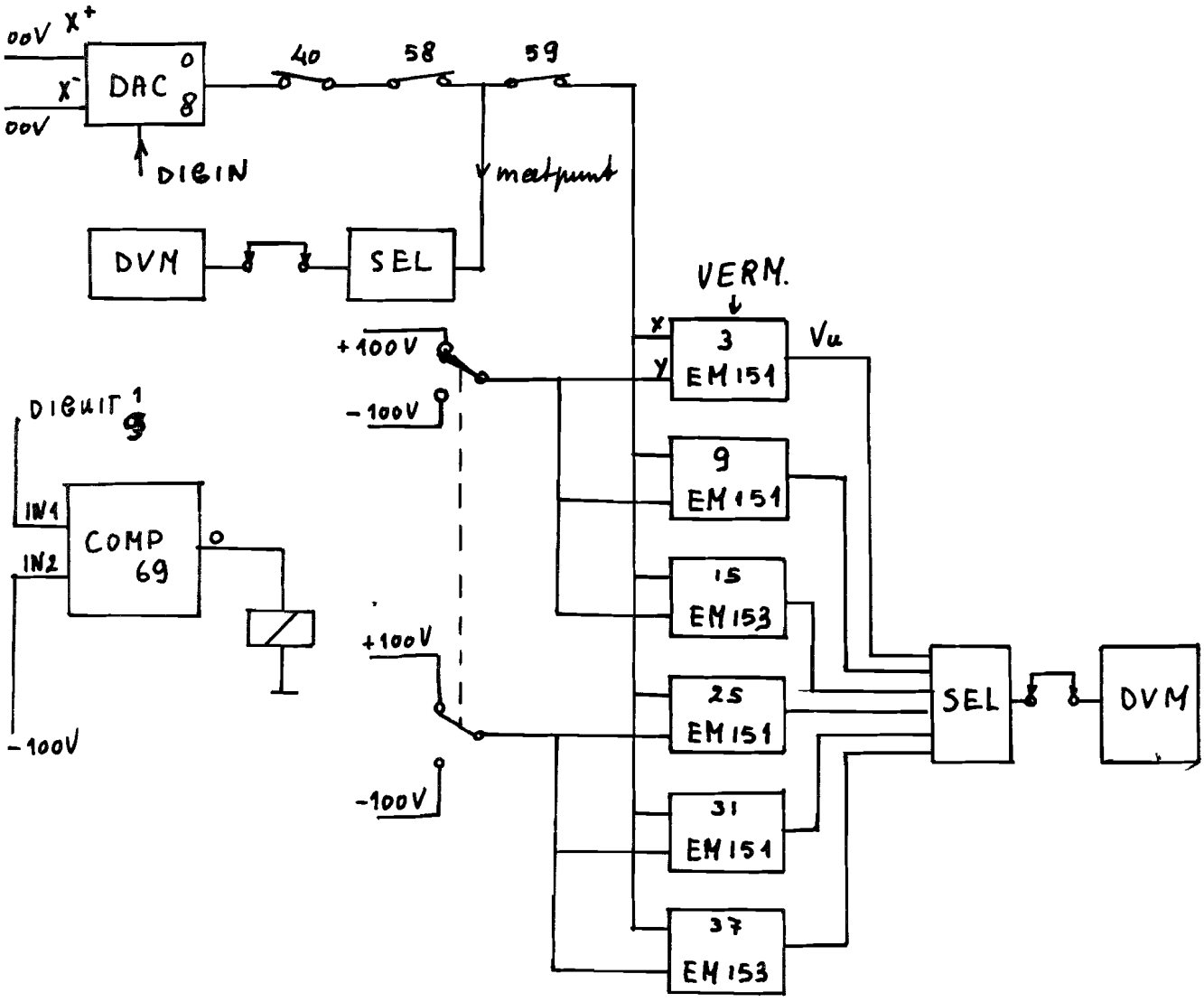
3- S/H versterkers met de ADC



DAC 0 en S/H 0 t/m 7 op console B

DAC 8 en S/H 8 t/m 15 op console A

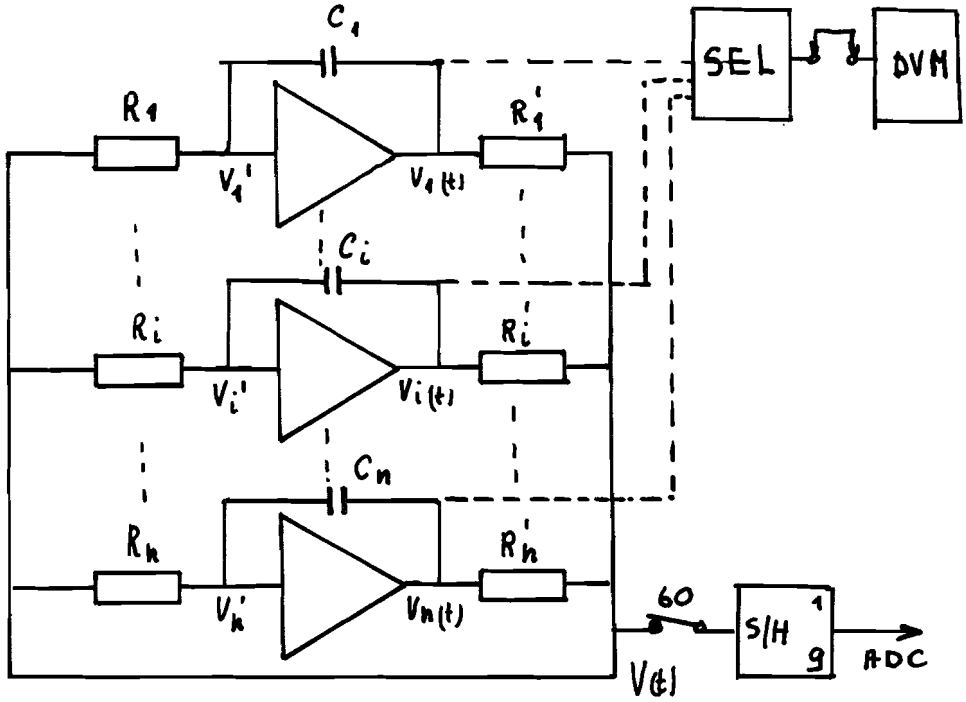
4- Test voor de vermenigvuldigers



Op console B DAC 0 en Diguit 1

Op console A DAC 8 en Diguit 9

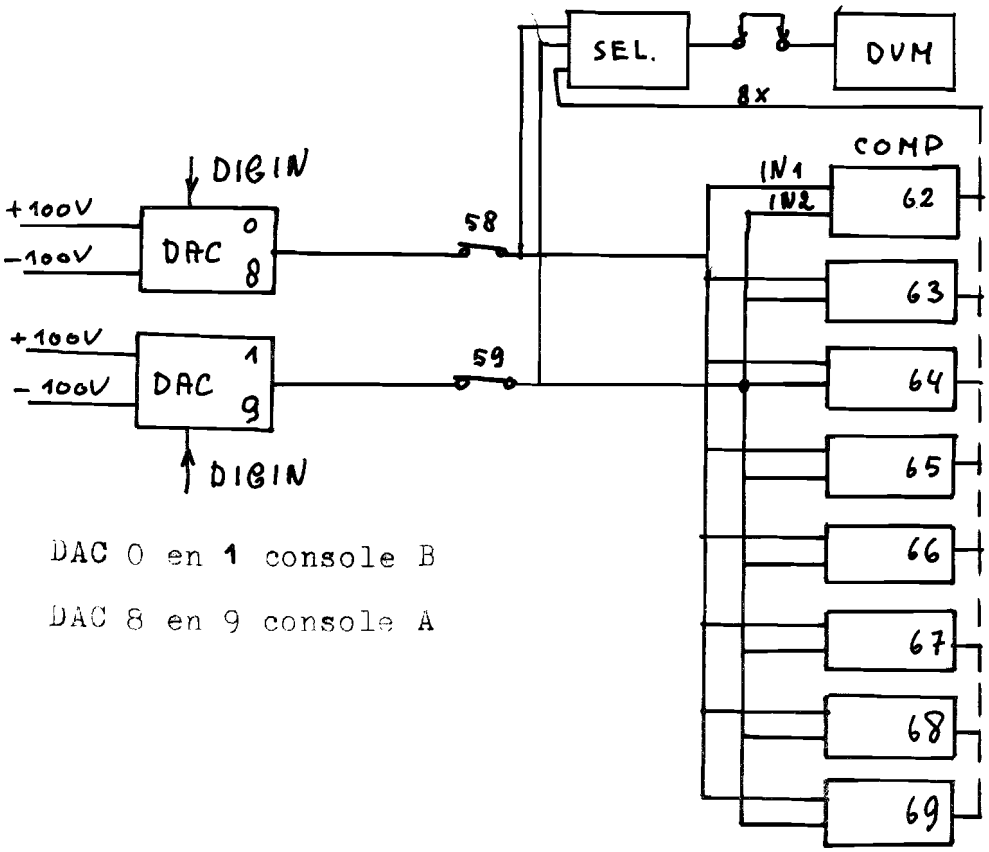
5- Test voor de integratoren



Op console A S/H 9

Op console B S/H

6- Test voor de comperatoren



DAC 0 en 1 console B

DAC 8 en 9 console A

7-Test voor QUAD- en DUAL-versterkers.

In het testschema is de selektor met S/H-O en ADC vervangen door de combinatie selektor-DVM.

8- Offset en ruis-test voor alle versterkers.

In principe geldt hier hetzelfde als bij punt 7. De totale duur van de test wordt dan echter erg lang, zodat hier een aanpassing van het testprogramma gewenst is.

BIJLAGE bij het verslag:

SOFTWARE VOOR EEN HYBRIDE COMPUTER

door J.J.M. Mulleneers

Gebruik van de routines in BASIC.

De routines zijn opgenomen in de volgende BASIC-interpreters:
HYBAS4 en HYBAS0.
Bij de opzet van HYBAS0 is van de overlay faciliteit gebruik gemaakt.
Hierdoor is meer geheugenruimte beschikbaar voor programma's.
De beschikbare geheugenruimte is bij HYBAS4 ca. 4K9 en bij HYBAS0 ca. 7K5 woorden.

Gebruik van de routines in FORTRAN.

De routines zijn opgenomen in een library file HYBLTB .
Deze file kan tijdens de link procedure in de input files opgegeven worden.

Naam: LOOP

Doel: Een aantal gegevens, nodig voor de start en uitvoering van de hybride rekenbewerking, opslaan in een tabel.

Aanroep: CALL LOOP (mode, N(,A.C.R.))

Argumenten: mode ;in BASIC variabele met beginletter C,H of R. In FORTRAN mode te initialiseren op 1(compute), 2(hold) of 3(reset). Bij de start van de hybride run gaat de Analoge Computer in de mode C(ompute), H(old) of R(eset).

N ;numerieke expressie, die aangeeft hoeveel klokpulsen in de hybride run zullen voorkomen.

A.C.R. ;"optional argument". Wordt op deze plaats in de argument list een willekeurige BASIC-of FORTRAN-symbool opgegeven, dan wordt de start van de hybride run met het signaal A(naloge) C(omputer) R(eady) gesynchroniseerd.

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen, +
hybride fout #1.

Opmerkingen: De duur van de hybride run is (N-1).Δ^m msec. Hierbij is Δ^m het onder de routine TINT gespecificeerd tijdinterval.
Data type van de argumenten in FORTRAN: integer.

Naam: IN

Doel: De gegevens, die nodig zijn om tijdens de hybride run snel en efficient de tijdcritische analoge output te verwerken, opslaan in een tabel.

Aanroep: CALL IN(array,beginkanaal,eindkanaal)

Argumenten:
- array ; naam van array waarin de analoge output na AD-conversie wordt opgeslagen.
- beginkanaal ; 15 ⩾ numerieke expressie ⩾ 0, zijnde het nummer van het eerste analoge uitgangskanaal.
- eindkanaal ; 15 ⩾ numerieke expressie ⩾ "beginkanaal", zijnde het nummer van het laatste analoge uitgangskanaal. De gebruikte kanalen dienen numeriek achter elkaar te staan.

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen, + hybride fout #3.

Opmerkingen: het aantal gebruikte analoge kanalen is gelijk aan ("eindkanaal"-"beginkanaal"+1).
Data type van argumenten in FORTRAN: integer.

Naam: OUT

Doel: De gegevens, die nodig zijn om tijdens de hybride run snel en efficient de tijdcritische analoge input te doen, opslaan in een tabel.

Aanroep: CALL "OUT"(array,beginkanaal,eindkanaal)

Argumenten: array ; naam van array waarin de data voor de analoge input is opgeslagen.
beginkanaal; 15⩾ numerieke expressie ⩾0, zijnde het nummer van het eerste analoge ingangskanaal.
eindkanaal ; 15⩾ numerieke expressie ⩾"beginkanaal", zijnde het nummer van het laatste analoge ingangskanaal. De gebruikte kanalen dienen numeriek achter elkaar te staan.

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen, +
hybride fout#2.

Opmerkingen: het aantal gebruikte analoge kanalen is gelijk aan ("eindkanaal"-"beginkanaal"+1).
Data type van argumenten in FORTRAN: integer.

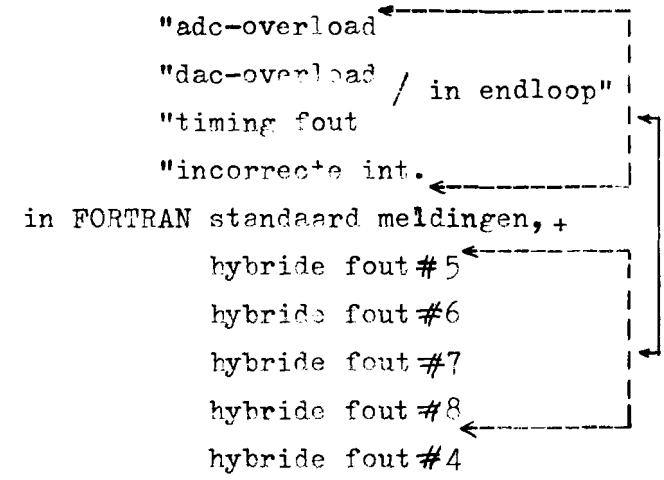
Naam: ENDL

Doel: Het starten en uitvoeren van de tijdcritische hybride rekenbewerking, waarbij de gegevens van de voorafgaande routines LOOP,OUT en IN de basis vormen.

Aanroep: CALL ENDL (mode)

Argument: mode ;in BASIC, variabele met beginletter C,H of R.
in FORTRAN mode te initialiseren op 1(compute), 2(hold) of 3(reset). Bij het einde van de hybride run gaat de Analoge Computer in de mode C(ompute), H(old) of R(eset).

Foutmeldingen: in BASIC standaard meldingen, +



Opmerkingen: Zodra een van de vier fouten met aparte melding optreedt (in FORTRAN fout 5 t/m 8), wordt de hybride run afgebroken en wordt het programma vervolgd met de uitvoering van de statements na de aanroep van de routine ENDL.
Data type van het argument in FORTRAN: integer.

Naam: TINT

Doel: Het instellen van het tijdsinterval tussen de opeenvolgende klokpulsen van de klok in de hybride interface.

Aanroep: CALL TINT (fractie, macht)

Argumenten: fractie ; numerieke scalar met waarden 1 t/m 255.
macht ; numerieke scalar met waarden 0 t/m 3.

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen, +
hybride fout # 23.

Opmerking: De lengte van het tijdsinterval bedraagt $10^{\text{"macht"}} \times \text{"fractie"} \times 10$ usec.
Data type van argumenten in FORTRAN: integer.

Naam: RESE
COMP
HOLD
POTS
ALLR

Doel: Het instellen van de mode van de analoge computer.

Aanroep: CALL RESE
CALL COMP
CALL HOLD
CALL POTS
CALL ALLR

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen.

Naam: GET

Doel: Het uitlezen van een analoog uitgangskanaal van de analoge machine.

Aanroep: CALL GET(datanaam, kanaalnummer)

Argumenten: datanaam ; variabele waarin de analoge uitgangs-
spanning na A/D conversie wordt opgeslagen.
kanaalnummer ; 15 ≥ numerieke expressie ≥ 0, zijnde het
nummer van het uit te lezen kanaal.

Foutmeldingen: in BASIC standaard meldingen, +
"adc-overload".
in FORTRAN standaard meldingen, +
hybride fout # 10
hybride fout # 21

Opmerkingen: De analoge uitgangsspanning in volt bedraagt
"datanaam"/163.84 .
Data type van argumenten in FORTRAN: integer.

Naam: PUT

Doel: Een analoog ingangskanaal instellen op een op te geven spanningswaarde.

Aanroep: CALL PUT(data, kanaalnummer)

Argumenten: data ; numerieke expressie die een maat is voor de analoge ingangsspanning.
kanaalnummer ; 15 ≥ numerieke expressie ≥ 0, zijnde het nummer van het analoge kanaal.

Foutmeldingen: in BASIC standaard foutmeldingen,+
"dac-overload".
in FORTRAN standaard foutmeldingen,+
hybride fout # 9
hybride fout # 22 .

Opmerkingen: De analoge ingangsspanning in volt bedraagt "data"/163.84 .
Data type van argumenten in FORTRAN: integer.

Naam: CNTR

Doel: Het instellen van een digitaal outputkanaal op hoog(1) of laag(0).

Aanroep: CALL CNTR(data,kanaalnummer)

Argumenten: data ; in BASIC numerieke expressie met waarde 0 of 1.
in FORTRAN logische variabele, die true of false is.

kanaalnummer ; 15 ≥ numerieke expressie ≥ 0, zijnde het nummer van digitale outputkanaal.

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen, + hybride fout # 20 .

Opmerking: Data type van argumenten in FORTRAN: integer (kanaalnr.)
logicalx1 (data) .

Naam: SENS

Doel: Het uitlezen van een digitaal inputkanaal.

Aanroep: CALL SENS(datanaam, kanaalnummer)

Argumenten: datanaam ; in BASIC variabele waarin de digitale waarde, 0 of 1, van het kanaal wordt opgeslagen.
in FORTRAN logische variabele die true of false wordt.

kanaalnummer ; 15 ≥ numerieke expressie ≥ 0, zijnde het nummer van het digitale input kanaal.

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen, + hybride fout #19 .

Opmerking: Data type van argumenten in FORTRAN: integer (kanaalnr.)
logicalx1 (datanaam)

Naam: PLOT

Doel: Het gebruik van de Tektronic 4010 display terminal in graphic mode.

Aanroep: CALL PLOT(A, X, Y)

Argumenten: A ;variabele, die op drie mogelijke waarden geinitialiseerd wordt.
A=0 d.w.z. initialisatie van de graphic mode in het punt met x en y als coordinaten (dark vector)
A>0 lijn tekenen naar opgegeven punt onder x,y (bright vektor)
A<0 tekenen van punt op het onder x,y opgegeven punt (point plot)

X ; x-coordinaat van het gewenste punt
$0 \leq X \leq 1023$

Y ; y-coordinaat van het gewenste punt
$0 \leq Y \leq 780$

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen.

Opmerkingen: Voorafgaand aan een eerste aanroep met A>0 dient steeds een statement met A=0 opgenomen te zijn.
Data type van argumenten in FORTRAN: integer .

Naam: ALFA

Doel: De Tektronic 4010 display terminal in alfanumerieke mode zetten na gebruik van de routine PLOT voor graphic mode.

Aanroep: CALL ALFA(tekst-string)

Argument: tekst-string ;in BASIC als string variabele op te geven.
in FORTRAN string in DATA-statement te declareren.
argument is "optional".

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen.

Opmerking: Data type van argument in FORTRAN: realx8 .

Naam: SLEC

Doel: Het selecteren van een versterkeruitgang of middencontact van een relais op de analoge computer.

Aanroep: CALL SLEC (adres)

Argument: adres ; numerieke expressie die het adresnummer van de te selecteren eenheid aangeeft. De waarde van de expressie dient te liggen in de volgende range 0 t/m 69 voor Console A of
100 t/m 169 voor Console B .

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen,
hybride fout # 16 .

Opmerkingen: Wordt bij een aanroep van SLEC geen argument opgegeven dan wordt de Select-bus van de analoge computer vrijgemaakt.
Data type van argument in FORTRAN: integer .

Naam: RDVM

Doel: Het meten en uitlezen van gelijkspanningen met de Fluke digitale voltmeter.

Aanroep: CALL RDVM (D1, D2)

Argumenten: D1 ; variabele waarin de getalwaarde is opgeslagen van de cijfers vóór de comma in de meetwaarde.
D2 ; idem als D1, echter nu voor de cijfers achter de comma.

Foutmeldingen: in BASIC standaard meldingen, +
bij gebruik van fluke dvm eerst function op remote en dan sample-rate op ext.
in FORTRAN standaard meldingen, +
hybride fout #17 .

Opmerkingen: De spanningswaarde in volt bedraagt D1 + 0.001X D2 .
Data type van argumenten in FORTRAN:integer .

Naam: SPOT

Doel: Het selecteren en instellen van een servopotentio-meter op de analoge machine.

Aanroep: CALL SPOT (adres, instelwaarde)

Argumenten: adres ; numerieke expressie die het adresnummer van de servopotentio-meter aangeeft.
instelwaarde; numerieke expressie die aangeeft op welke waarde de potmeter ingesteld moet worden.

Foutmeldingen: in BASIC standaard meldingen.
in FORTRAN standaard meldingen, +
hybride fout #18 .

Opmerkingen: 217 < adres <254
De instelwaarde dient als een getal tussen 0 en 16384 opgegeven te worden, bij een waarde van 16384 is weerstand volledig ingeschakeld.
Data type van argumenten in FORTRAN: integer .

Toelichting bij de foutmeldingen bij gebruik van de assembler-routines in FORTRAN.
De gebruiker kan een foutmelding als volgt krijgen:
HYBRIDE FOUT # N
Op de plaats van N wordt steeds een bepaald getal gezet.
Met behulp van onderstaande lijst kan nagegaan worden, in welke routine de fout is opgetreden en tengevolge waarvan.

N= 1 : argument fout in routine LOOP .

2 : " " " " OUT

3 : " " " " IN

4 : " " " " ENDL

5 : ADC-overload in routine ENDL

6 : DAC-overload " " "

7 : Timing fout " " " , tijdsinterval opgegeven in routine TINT is te kort.

8 : Incorrecte int." " " , hardware fout in hybride interface.

9 : DAC-overload in routine PUT

10 : ADC-overload " " GET

11 t/m 15 zijn niet gebruikt.

16 : argument fout in routine SLEC

17 : " " " " RDVM

18 : " " " " SPOT

N=19 : argument fout in routine SENS

20 : " " " " CNTR

21 : " " " " GET

22 : " " " " PUT

23 : " " " " TINT