Meteorologisch meetsysteem m.b.v. microcomputers

Keizer, L.

Award date:
1978

TECHNISCHE HOGESCHOOL EINDHOVEN

AFDELING DER ELEKTROTECHNIEK

vakgroep: Digitale Systemen

hoogleraar: prof.ir.A.Heetman

Afstudeerverslag:

Meteorologisch Meetsysteem m.b.v. Microcomputers.

L.Keizer

Uitgevoerd bij: ir.M.P.J.Stevens, THE
en ir.J.v.Gorp, KNMI.

Periode: 1 Juli 1977 t/m
27 April 1978.

Trefwoorden: universeel analoog systeem
microcomputer
bemonsteren
cyclische code
real time filter
communicatie netwerk
synchronisatie.

INHOUDSOPGAVE

1.0 SAMENVATTING

In dit verslag worden de mogelijkheden onderzocht, microcomputers in te zetten voor taken in een analoog meetnet. De problemen, die optreden bij analoge meetmethoden, in het bijzonder wanneer grote afstanden moeten worden overbrugd, maken een opzet met digitale systemen aantrekkelijk.

Aangezien het een analoog meetsysteem voor meteorologisch onderzoek betreft wordt veel aandacht besteed aan de meettechnische eisen, bij voorbeeld t.a.v. resolutie, signaal/ruisverhouding en vertraging.

Het blijkt dat bij het toepassen van een technisch eenvoudige methode, met dedicated processors, rekening moet worden gehouden met introductie van ruis.

De digitale transmissiekanalen worden beschermd tegen de invloed van storingen, door toepassing van foutcorrigerende codes en een retransmissiemechanisme. De combinatie blijkt in dit real time meetsysteem een betrouwbaar transport mogelijk te maken.

In de microcomputers, tussen weeropnemers en transmissiekanaal, kunnen programmatisch laagdoorlaatfilters gerealiseerd worden. Ook kunnen berekeningen aan meetwaarden van verschillende opnemers binnen een groep worden uitgevoerd. Een keuze van kanalen of filterkarakteristieken kan vanaf een centrale plaats worden gemaakt.

Voor de gebruikers van het meetsysteem, b.v. een minicomputer voor berekeningen of data-opslag, analoge registratie-apparatuur en grafische display's, dient een geheugenveld als interface. De

gebruikers hebben een privé-geheugen met, direct adresseerbaar, alle opnemers die in het systeem aanwezig zijn. Dit betekent dat ook relatief eenvoudige randapparatuur aan het meetsysteem kan worden aangesloten.

Het blijkt dat met de microcomputers een flexibel meetsysteem kan worden gerealiseerd, dat voldoet aan de gewenste analoge specificaties.

2.0 INLEIDING

Het afstudeerwerk, dat in dit verslag wordt beschreven, is niet, zoals gebruikelijk, een bijdrage tot een onderzoek dat aan de Technische Hogeschool wordt gedaan, maar de uitvoering van een opdracht van het Koninklijk Nederlands Meteorologisch Instituut in de Bilt. Het verslag beoogt een leidraad te zijn bij het realiseren van het project, waarvan de principes en mogelijkheden onderzocht werden.

Bij de Instrumentele Afdeling van het K.N.M.I. is men gestart met het ontwikkelen van microcomputers voor dataverwerking in grote en kleinere meetsystemen. De aanleiding tot deze ontwikkeling in digitale richting vormden de reeds gebruikelijke digitale verwerking van meetresultaten en het steeds groeiende aantal digitale weeropnemers. Een versnelling van deze omschakeling werd veroorzaakt door de problemen die optreden bij analoog transport over grote afstanden.

De opdracht luidde als volgt:

"Vastleggen van de eisen waaraan een meetsysteem voor meteorologisch onderzoek moet voldoen,ontwerpen van een systeem dat hieraan voldoet en onderzoeken in hoeverre en op welke wijze de taken binnen het geheel door microcomputers verricht kunnen worden."

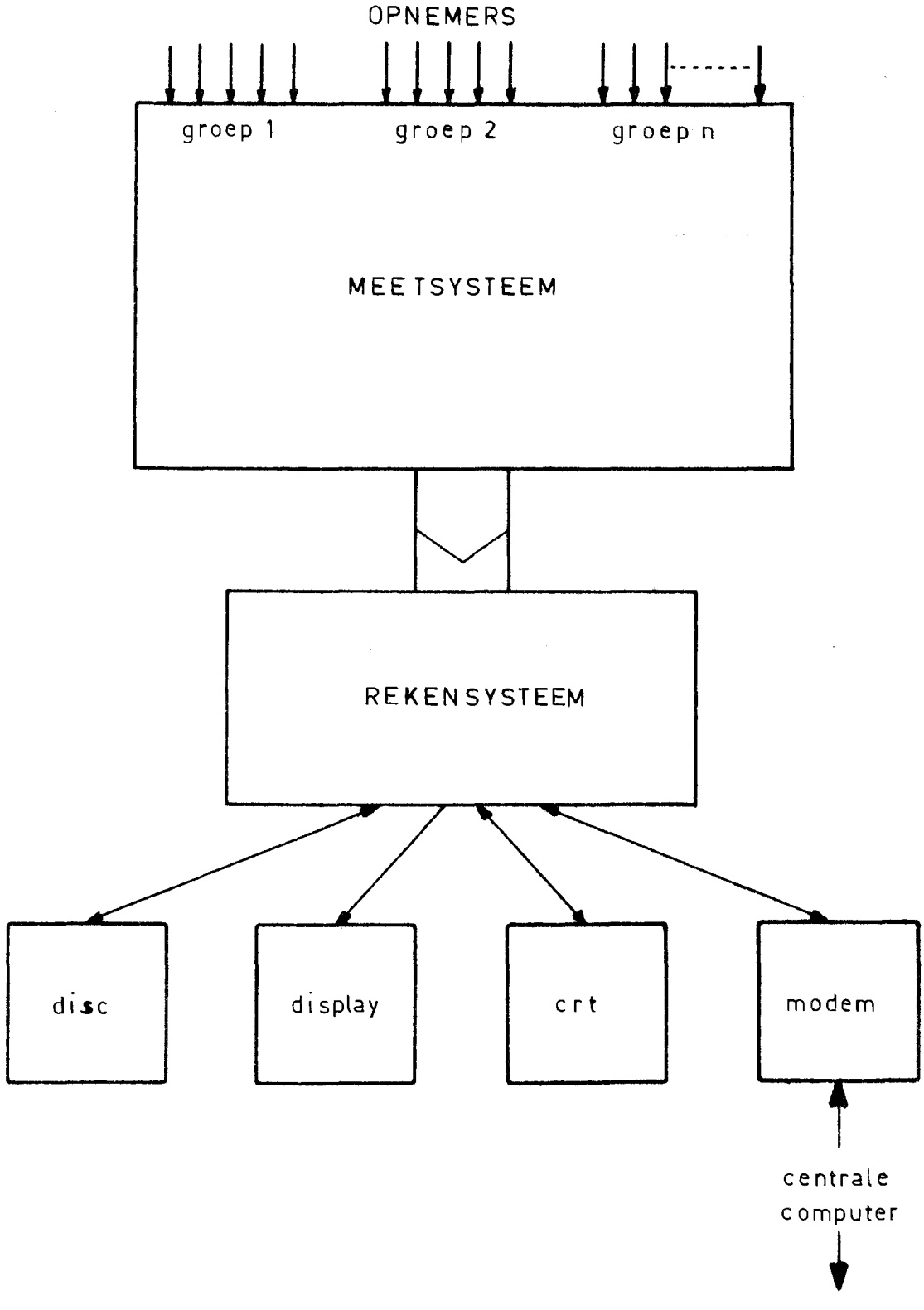
Gezien de vele digitale aspecten van het onderzoek, kon het plaats vinden in het kader van een afstudeerwerk bij de vakgroep Digitale Systemen aan de Technische Hogeschool Eindhoven.

3.0 BESCHOUWING MEETSYSTEEM

Een meetsysteem voor meteorologisch onderzoek bevat over het algemeen één of meer groepen opnemers. Men is namelijk geinteresseerd in de toestand van de atmosfeer op een bepaalde plaats en meet hiertoe een aantal grootheden, zoals windrichting, windsnelheid, temperatuur, relatieve vochtigheid et cetera. Voor een globaal overzicht is het voldoende één groep opnemers te gebruiken. Voor een nauwkeuriger inzicht (in een beperkt gebied), t.b.v. wetenschappelijk onderzoek, of voor een beter overzicht, b.v. bij vliegvelden, worden meerdere groepen ingezet.

De verwerking van de meetwaarden geschiedt in alle systemen centraal. Berekeningen voor ad hoc informatie voor de gebruiker (vliegvelden) worden uitgevoerd door kleinere rekenmachines (minicomputers). Ook worden deze ingezet bij berekening van resultaten en opslag van informatie, voor verdere verwerking op een grote centrale computer, t.b.v. wetenschappelijk onderzoek. Deze rekenmachine bevindt zich niet op de meetplaats, zodat informatietransport over grote afstand moet plaatsvinden. Dit laatste gebeurt nu nog "off-line".

We zien hier een duidelijke scheiding tussen het meetsysteem en het reken- annex opslag-systeem. Het is de taak van het meetsysteem de juiste informatie in verwerkbare vorm aan de rekenmachine aan te bieden. Aan de hand van dit uitgangspunt kunnen we nu een aantal wensen en eisen van gebruikers formuleren t.a.v. een nieuw ontwerp voor het meetsysteem.



Figuur 3.1: Schematische voorstelling totale systeem.

3.1. UNIVERSEEL: de ingangen van het meetsysteem dienen zo uitgevoerd te zijn dat alle huidige opnemers kunnen worden aangesloten, evenals de binnenkort te verwachten typen. Er verschijnen steeds betere opnemersystemen die hun informatie in digitale vorm aan het meetnet aanbieden. Een voorbeeld hiervan is de windvaan die, voor omzetting van positie naar digitale meetwaarde gebruik maakt van een codeschijf. Dit model is lichter (sneller) en vergt minder onderhoud dan zijn voorgangers met een potentiometer als omzetter.
Hetzelfde geldt voor de anemometer die pulsen afgeeft met een frequentie evenredig met de momentane windsnelheid. Voorts is de verwachting, dat zeker voor onderzoek nieuwe opnemers ontwikkeld zullen worden, die sneller en nauwkeuriger kunnen werken dan de huidige. Deze zullen voor een groot deel hun meetwaarden ook digitaal aanbieden.

3.2. NAUWKEURIG: de nauwkeurigheid van een meting wordt in de eerste plaats bepaald door het oplossend vermogen van de opnemer en verder beperkt door het transportkanaal. Elke trap in de keten tussen opnemer en rekenmachine kan aanleiding geven tot grotere onnauwkeurigheid, b.v. door filterwerking.

Het verlopen van karakteristieken van sensoren of kanaal en het optreden van fouten door storingen van binnen of buiten het systeem dragen bij tot de onbetrouwbaarheid van de meting. Voorbeelden van dergelijke storingen: statische ontladingen, 50 Hz inductie, schakelmanipulaties, overspraak en thermische ruis.

Het meetsysteem moet waar mogelijk voorkomen dat deze fouten doorwerken in de meetresultaten. Wanneer de fouten wel gedetecteerd, maar niet hersteld kunnen worden, dient dit aan de rekenmachine en eventueel aan de gebruiker gemeld te worden.

3.3. SNELHEID: in het algemeen wenst men voor wetenschappelijk

onderzoek over een breder frequentiespectrum te meten dan voor routinemetingen. Het toepassen van nog snellere (duurdere) opnemers heeft geen zin wanneer de fysische grootheid, die gemeten wordt, geen interessante componenten in het verbrede spectrum heeft. Zo is een bovengrens in de te verwachten ontwikkeling van sensoren aan te geven.

3.4. LOCALISEERBAARHEID v. FOUTEN: oncorrigeerbare fouten, die optreden door storingen of verloop van karakteristieken, werken wanneer ze niet opgemerkt worden door in de informatie, die wordt opgeslagen. Wanneer deze fouten wel gedetecteerd worden bij verwerking van de gegevens door de grote centrale rekenmachine kan tot ijking of herstel worden overgegaan.
Zeker bij een off-line verbinding (ponsband of magnetische tape of disc) zal dan lange tijd foute informatie worden opgeslagen. Er is dus behoefte aan het controleren van het meetsysteem tijdens bedrijf. Een foutmelding dient direct bij het optreden van een oncorrigeerbare fout gegenereerd te worden. Om het onderhoudswerk te verlichten is het tevens wenselijk dat de fout aan de hand van de melding localiseerbaar is.

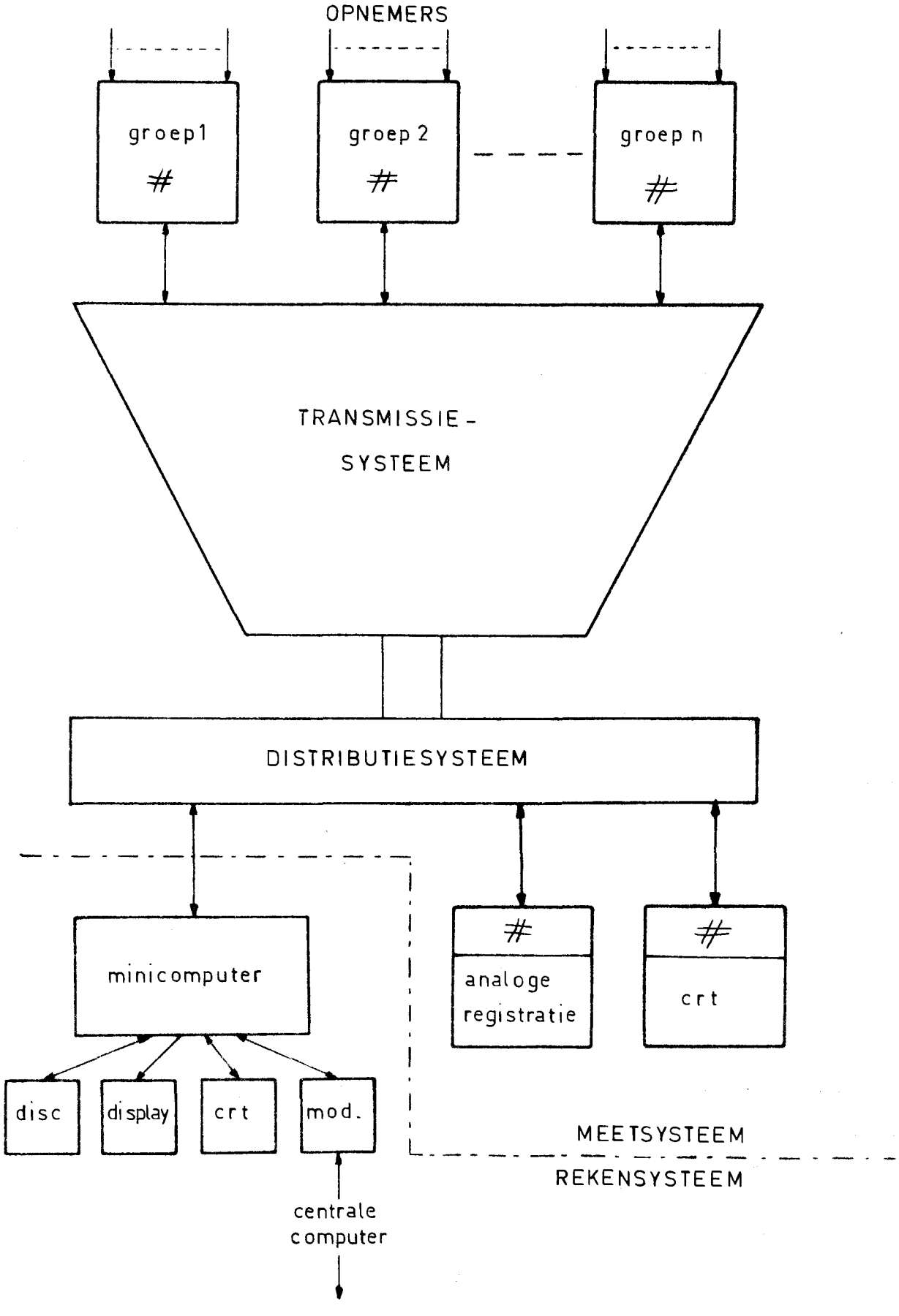
Het is niet de taak van het meetsysteem de meetwaarden te toetsen aan een theoretische verwachting en op grond daarvan meetwaarden te verwerpen of te corrigeren. Het is niet onmogelijk dat later blijkt dat de veronderstelling bij de toets onjuist was. De opgeslagen informatie kan niet meer gebruikt worden, omdat de oorspronkelijke meetwaarden verloren zijn gegaan. De terugkoppeling door deze tests dient via de rekenmachine te lopen.

3.5. BEDIENINGSGEMAK: een apparaat dat meetwaarden van het systeem betrekt moet zelf de gewenste kanalen en samplefrequenties in kunnen stellen. Aan de andere kant moet een opnemer ook met een eigen bemonsterfrequentie kunnen werken zonder consequenties in het systeem.

De wensen t.a.v. automatisch ijken, corrigeren van fouten bij transmissie en faciliteiten voor digitale opnemers maken toepassing van intelligente systemen in het meetnet noodzakelijk. De opkomst van microprocessoren maakt het mogelijk betaalbare programmeerbare deelsystemen in het geheel in te passen. Ze kunnen gebruikt worden als "dedicated processors" d.w.z. geprogrammeerd om één complexe taak uit te voeren. Het zijn dan zelfstandige eenheden met eigen geheugen, programmatuur, in- en outputmodules: microcomputers. Communicatie tussen de microcomputers geschiedt b.v. via een bussysteem of door serietransmissie. Voor elke verbinding dienen de voor en nadelen van de verschillende methoden onderzocht te worden.

Het geheel wordt op deze manier opgesplitst in deelproblemen, die met dedicated processors worden opgelost. Aangezien het een meetsysteem en geen rekensysteem betreft kan de koppeling van de deelsystemen beperkt blijven tot een synchronisatiemechanisme. Gegevens die worden uitgewisseld zijn "real-time meetwaarden" en geen data zoals in een rekensysteem. In het laatste geval heeft een proces bepaalde informatie nodig voor zijn voortgang. Bij een meetsysteem wordt informatie na zekere tijd ongeldig en overschreven door nieuwe.

Deze opzet met zelfstandige microcomputers maakt het gebruik van een gecompliceerd "operating system" overbodig. Het operating system bevindt zich in de minicomputer waarvan het meetsysteem een input is. Bij kleine opstellingen kan een microcomputer deze taak van de mini overnemen en een beperkt meetprogramma uitvoeren. In de volgende figuur is het meetnet nog eens schematisch weergegeven, met een # als aanduiding voor een intelligent deelsysteem. Tevens zijn nog twee randapparaten van het meetsysteem weergegeven, een CRT-terminal voor besturing van het systeem en eventuele foutmeldingen en een apparaat, dat analoge registraties van meetsignalen produceert.



Figuur 3.2: Schematische opzet met dedicated processors

Om tot een volledig ontwerp te komen worden in de volgende hoofdstukken deze onderwerpen nader onderzocht:

-de grootte van de DATASTROMEN door het systeem

-de SYNCHRONISATIE van het totale meetnet

-een TRANSMISSIESYSTEEM voor grote afstanden en

-de KOPPELING van transmissiesystemen met opnemers resp. gebruikerinterfaces.

4.0 DATASTROMEN.

Het uitgangspunt bij het ontwerp van een meetsysteem is de hoeveelheid te transporteren informatie. Wanneer de datastroom bekend is, kunnen berekeningen gedaan worden voor de gewenste capaciteit van transmissieketens en buffergeheugens. We hebben gezien dat het meetnet is opgebouwd uit groepen opnemers. Het is nu voldoende de maximale capaciteit van een groep te bepalen. Dit leidt direct tot de grootte van de datastromen in het gehele systeem.

4.1 GROEP. In tabel 4.1 is een overzicht te vinden van de situatie op een aantal meetstations. Het betreft hier een opstelling in een meetmast voor wetenschappelijk onderzoek, een meetstation voor routine metingen en een groep voor het "Energie Balans"-terrein, waar de warmteopslag in de aarde wordt onderzocht. Tussen haken is de toekomstvisie van de afdeling Meteorologisch Onderzoek van het K.N.M.I. t.a.v. het aantal meetpunten weergegeven. De snelheden in de tabel hebben betrekking op de bemonsterfrequenties van de opnemers.

We moeten ervan uitgaan, dat in de toekomst de behoefte ontstaat om betere opnemers sneller te kunnen bemonsteren. Tevens zal men door verbeterde opslagmogelijkheden meer (=gedetailleerder) informatie op willen slaan voor verwerking op de computer. Ook moet het mogelijk zijn één of meer opnemers met maximale snelheid te volgen op de centrale meetplaats, t.b.v. foutdetectie door visuele controle. Dit alles leidt tot een vergroting van de benodigde capaciteit.

	FREQUENTIE(Hz)	AANTAL			OPNEMER
		EB	MAST	ROUT	
TEMPERATUUR	10	-	6	2	fluctuatiemeter
	1	20	2	2	thermokoppel
	20	-	-(1)	-	sonisch
WINDRICHTING	2x20	-	-(3)	1	trivaan
	1	-	2(3)	1	vaan
WINDSNELHEID	10	-	-(3)	1	trivaan
	1	-	3	1	cupanemometer
ZICHT	10	-	-	-	
	1	-	1	1	
STRALING	10	-	-	-	
	1	7	-	1	
NEERSLAG	1	1	-	1	

Tabel 4.1: Opnemers op meetstations.

met: EB= energie balans terrein, onderzoek naar warmte opslag

MAST= een groep in de meetmast in Cabauw en

ROUT= weerstation (b.v. op vliegveld).

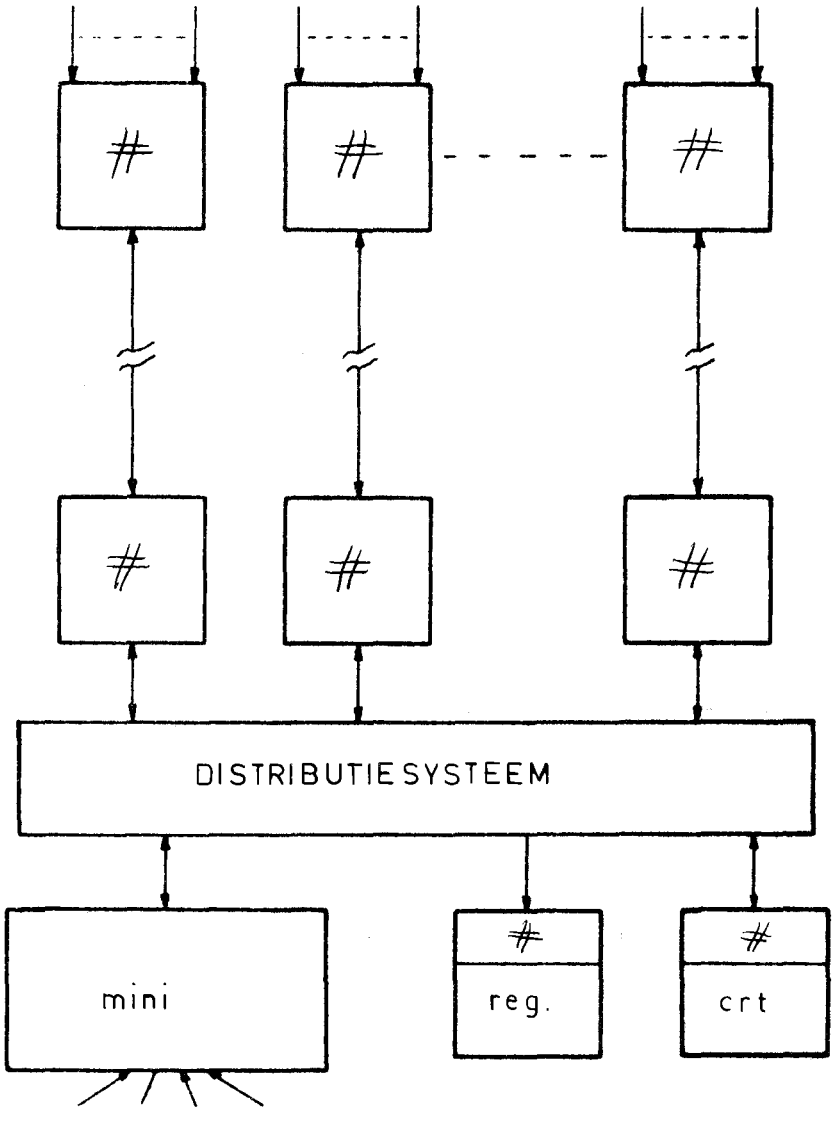
Uit de tabel blijkt dat voor een universeel inzetbare groep een capaciteit van 32 kanalen met een bemonsterfrequentie van 4 Hz en 16 kanalen van 32 Hz voldoende is. Dit komt neer op 640 meetwaarden per seconde.

4.2 MEETNET. Het aantal groepen per meetsysteem hangt af van het gebruiksdoel. Dit aantal varieert van 1, bij een routine-meetstation, tot ongeveer 5 bij vliegvelden en loopt op tot ongeveer 16 bij een groot project voor wetenschappelijk onderzoek. Een voorbeeld van het laatste is de meetmast die in gebruik is in Cabauw. Hier meet men op 11 niveaux, met elk 3 uithouders, tot 200 meter hoogte, de toestand van de atmosfeer. Per uithouder is plaats voor 16 meettoestellen, totaal 48 per groep. Samen met meetmasten op de tuiblokken, masten in de directe omgeving en opstellingen op het Energie Balans-terrein loopt het aantal groepen op tot 16.

Het ligt voor de hand als maximaal aantal groepen voor een meetnet nu 16 te nemen. Wanneer we ons tot dit aantal zouden beperken en een digitaal systeem hiervoor ontwerpen brengt de uitbreiding met slechts één groep grote problemen met zich mee. Voor een groot systeem zullen we in het vervolg rekenen met maximaal 32 groepen, terwijl het systeem te beginnen met 1 groep tot dit aantal uitbreidbaar moet zijn.

4.3 CONFIGURATIE. Gezien de grote afstanden die bij meetsystemen van deze soort overbrugd moeten worden (10-1000m), moet voor elke verbinding behalve de kabel ook een geschikte zend/ontvang-installatie toegepast worden. De kosten hiervan dwingen ons het datatransport vanaf groepen in serie te laten plaatsvinden. Een structuur met meerdere groepen op een kabel kan niet gebruikt worden, wegens de hoge transmissiesnelheden over grote afstand, die dan vereist zijn. Een voorbeeld: 600 meetwaarden van 12 bit levert een snelheid van 7.2 kbaud (zonder enige redundantie), bij aansluiting van meerdere groepen op een kabel moet dit ge-

tal vermenigvuldigd worden met het aantal groepen. Dergelijke snelheden zijn slechts met speciale apparatuur te bereiken. We komen nu tot de volgende configuratie van het meetsysteem:



Figuur 4.1: Configuratie meetsysteem.

4.4 DISTRIBUTIESYSTEEM. Voor het distributiesysteem wordt de ingaande datastroom nu maximaal 32*600 meetwaarden per seconde. Elk erop aangesloten randapparaat, zoals een minicomputer voor berekeningen, moet in principe deze 20.000 meetwaarden uit het meetnet kunnen betrekken. Het is duidelijk dat geen enkel systeem deze hoeveelheden per seconde kan verwerken of opslaan. Het meetsysteem biedt de mogelijkheid te kiezen uit 500 kanalen van 30 Hz en 1000 kanalen van 4 Hz. De gebruiker (= het randapparaat) kan de gewenste kanalen met eigen frequentie bemonsteren.

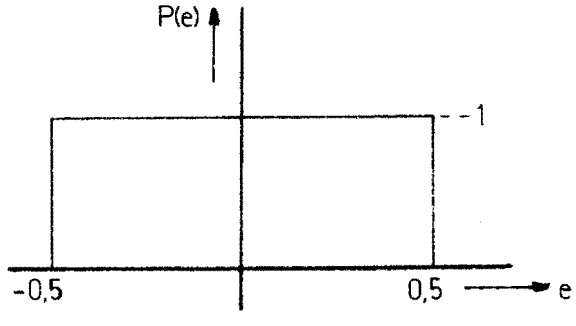
Dit is in principe dezelfde werkwijze als bij het huidige analoge meetnet. Daar wordt door de bekabeling een keuze gemaakt uit de voor A/D-omzetting gewenste kanalen, die met programmatisch in te stellen frequenties (Real-time-executive) worden bemonsterd. Een voordeel is nu dat door de software in de minicomputer ook iedere willekeurige opnemer in het meetnet gekozen kan worden.

Hoe een en ander gerealiseerd kan worden wordt beschreven in de hoofdstukken 5 en 7.

5.0 BEMONSTEREN EN SYNCHRONISEREN.

Wanneer een analoog signaal wordt omgezet in een datastroom, geschikt voor digitale verwerking, geschieden er twee dingen: het signaal wordt bemonsterd en het monster wordt gequantiseerd.

5.1 QUANTISERING. Quantiseringsruis is de fout die optreedt bij het quantiseren van een analoge waarde. Wanneer we een meetwaarde uitdrukken in n bits, verkrijgen we 2**n mogelijke meetwaarden. Laat het analoge signaal een naar de amplitude tussen -A en A volt begrensde functie zijn, voor de kansverdeling P(e) van de fout vinden we met als schaaleenheid A*2**(n-1) volt:



Figuur 5.1: Kansdichtheidsfunctie van quantiseringsfout

Het ruisvermogen is nu:

5.1.1 $$\sigma^2(e) = \int_{-\infty}^{\infty} [P(e)*(e-E(e))**2]\ de = \int_{-0,5}^{0,5} (e**2)\ de = 1/12$$

met E(e)=verwachting van e=0.
De signaal/ruisverhouding bedraagt, met signaalvermogen (A**2)/2

5.1.2 s/r = 10*log(1,5*(2**n)**2)

Voor een woordbreedte van 8 bit levert dit +50dB, voor 10bit +62dB en voor 12 bit +74dB. Deze waarden zijn van toepassing op dat signaal waarop in een rekensysteem "real-time" bereke-

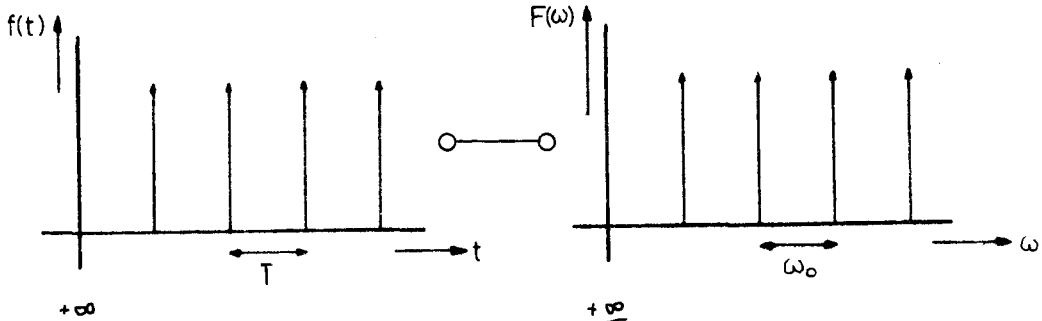
ningen worden uitgevoerd. B.v. bij berekening van een autocorrelatiespectrum. In dat geval zou een woordbreedte van 10 bit toereikend zijn.

Zijn we geinteresseerd in de momentane waarde van een grootheid dan moeten we de nauwkeurigheid t.g.v. quantisering uitdrukken in procenten. Voor 8 bit vinden we +0,20%, voor 10 bit +0,048% en voor 12 bit +0,012%. Voor metingen met hoge resolutie, zoals temperatuurbepalingen, is 10 bit onvoldoende. We zullen in het vervolg onder een meetwaarde een woord van 12 bit verstaan.

5.2 BEMONSTERING. Bemonsteren van een naar de amplitude en naar de frequentie begrensde tijdfunctie $f_\Omega(t)$, houdt in dat men de waarde van $f_\Omega(t)$ vaststelt op regelmatige tijdstippen. Laten we de bemonsterfrequentie $\omega_o = 2\pi/T$ stellen. In de literatuur(2) wordt de volgende stelling bewezen:

"Een periodieke reeks van δ-functies in de tijd transformeert in een periodieke reeks van δ-functies in het frequentiedomein"

Met de notatie $\circ\!\!-\!\!\circ$ voor de fouriertransformatie luidt de stelling in formule:



$$5.2.1 \quad \sum_{n=-\infty}^{+\infty} \delta(t-nT) \;\circ\!\!-\!\!\circ\; 2\pi/T \sum_{n=-\infty}^{+\infty} \delta(\omega - n\omega_o)$$

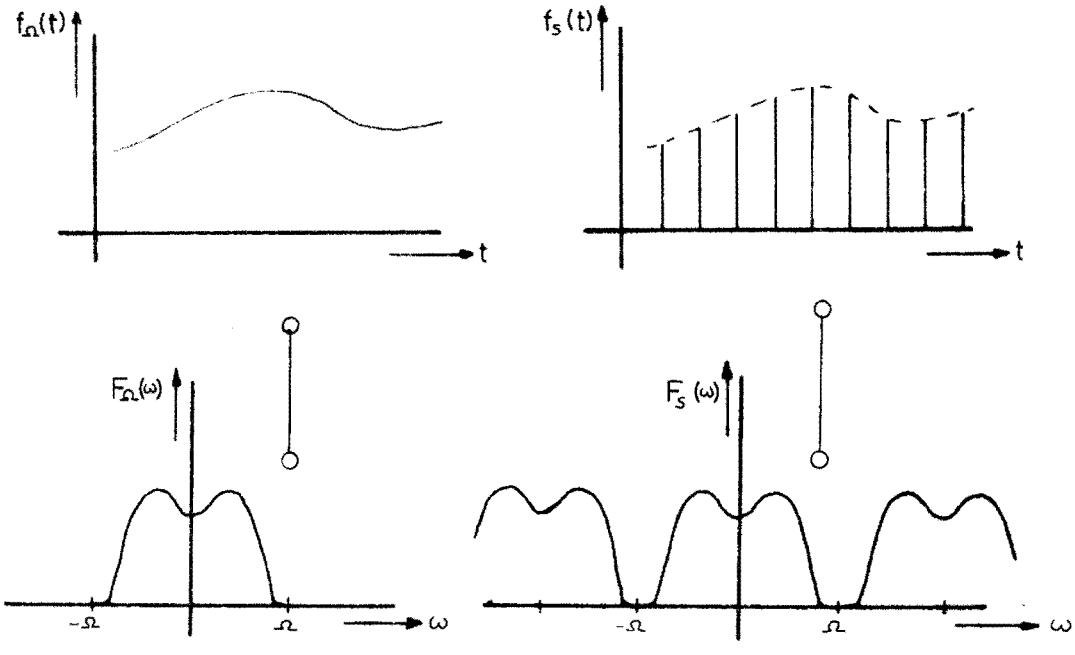
(met de volgende definities:
$\delta(t)=0$, voor $t=0$ en $\int_{-\infty}^{+\infty} \delta(t)\,dt = 1$
Voor de Fouriergetransformeerde $F(\omega)$:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)\, e^{j\omega t}\, dt \qquad)$$

Met behulp van genoemde stelling komen we tot het bemonsteringstheorema, via:

$$5.2.2 \quad f_s(t) = f_\Omega(t) \sum_{n=-\infty}^{+\infty} \delta(t-nT) = \sum_{n=-\infty}^{+\infty} f_\Omega(nT)\,\delta(t-nT)$$

$$5.2.3 \quad F_s(\omega) = \tfrac{1}{2\pi} F_\Omega(\omega) \otimes \omega_0 \sum_{n=-\infty}^{+\infty} \delta(\omega - n\omega_0) = \tfrac{1}{T} \sum_{n=-\infty}^{+\infty} F_\Omega(\omega - n\omega_0)$$



Figuur 5.2: Spectra.

Het originele signaal is uit het bemonsterde signaal $f_S(t)$ terug te winnen, afgezien van een konstante 1/T, door $f_S(t)$ te filteren met afsnijfrequentie Ω.

$$5.2.4 \quad F_s(\omega) * P_\Omega(\omega) = F_\Omega(\omega) \circ\!\!-\!\!\circ \tfrac{1}{T} f_\Omega(t)$$

Geen vervorming treedt op wanneer de spectra elkaar niet overlappen: $\omega_0 \geq 2\Omega$. Dit is het bemonsteringstheorema:

"De functie f(t) wordt volledig bepaald door zijn functiewaarden op de bemonsteringstijdstippen indien de bemonsteringsfrequentie $\omega_0 \geq 2\Omega$ met Ω de hoogst over te dragen frequentie van f(t)."

Dit kan nog eens duidelijk geillustreerd worden met de volgende formulering van Shannon:

$$5.2.5 \quad f_\Omega(t) = \sum f_\Omega(nT)\,\frac{\sin(\Omega t - n\pi)}{\Omega t - n\pi} \quad , \text{met} \quad \Omega = \pi/T$$

Hieruit zijn direct twee belangrijke consequenties voor het omzetten van analoge signalen in digitale af te leiden:

1. Voor bemonstering dient het signaal bandbegrensd te worden met een filter. Door bemonstering kunnen n.l. storingen buiten de beschouwde frequentieband bijdragen geven binnen deze band (overlappen).
2. Hoe sneller men de opnemers bemonstert, des te eenvoudiger kan dit filter zijn: de lobben komen verder uit elkaar te liggen en de afval/decade van het filter kan gering zijn.

De conclusie dat men een goedkoop filter toe kan passen door met extra hoge frequentie te bemonsteren zonder al deze monsters mee te sturen, dit om het transmissiesysteem niet onnodig te belasten, is onjuist. Volgens 5.2.5 zijn 2 Ωmonsters onafhankelijk van elkaar, de extra monsters zijn afhankelijk. We kunnen ons het bemonsterde signaal voorstellen als een sommatie van twee reeksen, de onafhankelijke en de afhankelijke monsters, zowel wat tijdfunctie als frequentiespectrum betreft. Bij weglaten van de afhankelijke monsters blijft het spectrum over van het met juist voldoende frequentie bemonsterde signaal. Een scherp filter zou nu nodig geweest zijn om overlapping te voorkomen.

5.3 DIGITAAL FILTER. Wel kunnen we het aantal te versturen monsters reduceren door na bemonstering een tweede filter digitaal te simuleren. Met behulp van de microcomputers bij de opnemers kunnen we een filterkarakteristiek met vrij kiesbare vorm berekenen. Een voorbeeld van een eenvoudig in software te implementeren "filter" is de berekening van een lopend gemiddelde. In Appendix A is een korte beschouwing gewijd aan een 6 dB/octaaf laagdoorlaatfilter.

Alle samplewaarden moeten in de berekening van het filter meegenomen worden, om het eenvoudige filter tussen opnemer en A/D-omzetter te kunnen toepassen. Voor opnemers die de informatie reeds in digitale vorm aanbieden geldt dezelfde redenering t.a.v. de samplereductie.

5.4 SYNCHRONISATIE. Wanneer een gebruiker een meetwaarde van het meetsysteem betrekt, moet zekerheid bestaan over het eigenlijke moment van bemonstering van de opnemer. Aangezien anders een fout ontstaat ten gevolge van onbekende verschuivingen van samplemomenten. Dit kan slechts bereikt worden wanneer het meetsysteem een bekende vertraging voor het signaal vormt.

Er zijn een aantal methoden die deze synchronisatie kunnen bewerkstelligen:

5.4.1 Meesturen van het tijdstip van bemonstering vanaf de opnemerplaats. Inderdaad is nu de vertraging bekend maar:
*bij het opnemersysteem moet een nauwkeurige "real time" klok aanwezig zijn,
*de overhead op het transmissiekanaal neemt enorm toe, (i.p.v. bij elke meetwaarde van 12 bit moet minstens een byte tijdinformatie meegestuurd worden: 40% overhead)
*en de gebruiker moet interpreteren (hij kan niet volstaan met het binnenhalen van meetwaarden, er moet ook controle van de tijdinformatie plaatsvinden).

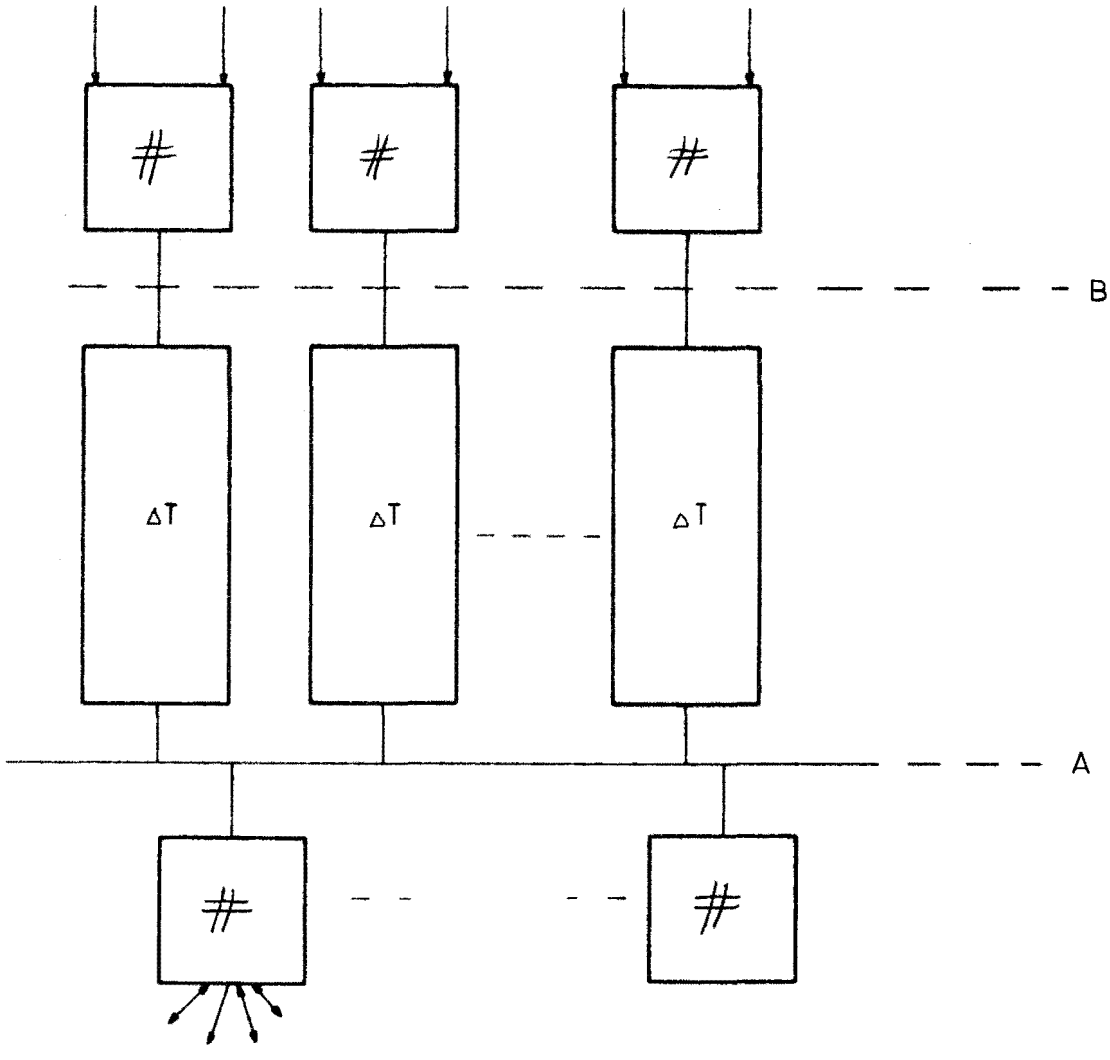
5.4.2 Zorgen voor een constante (en dus bekende) vertraging van elk kanaal door het meetsysteem. Dit houdt in dat het binnenhalen van een meetwaarde door de gebruiker, het transporteren van de meetwaarde door het transmissiesysteem en het bemonsteren van de opnemers gesynchroniseerd (vastgekoppeld) dient te verlopen. Dit is door een hardware mechanisme te bereiken. Er zijn een aantal bezwaren:
*synchroniseren van alle microcomputers is noodzakelijk, (dit vereist ingewikkelde protocollen voor datauitwisseling tussen de microcomputers)

*gebruikers en opnemersystemen dienen ook gesynchroniseerd te worden (dit verkleint het bedieningsgemak: de minicomputer wordt (tijd)afhankelijk van het meetsysteem. Bij nieuwe ontwerpen van opnemersystemen moet met deze koppeling rekening gehouden worden)

*door centraal synchroniseren wordt het systeem kwetsbaar.

Beide methoden 5.4.1&2 leiden tot technisch gecompliceerde en (dus) dure systemen.

De oplossing moet gezocht worden in een asynchrone koppeling van alle deelsystemen. Dit levert ook het grootste bedieningsgemak op: de gebruiker kan op willekeurige tijdstippen een meetwaarde binnenhalen met een bepaalde zekerheid over het tijdstip waarop de werkelijke bemonstering heeft plaatsgevonden. Door deze oplossing te kiezen kunnen we het meetsysteem opsplitsen in een aantal deelsystemen die zelfstandig zijn. We zullen aantonen dat de hierdoor geintroduceerde onnauwkeurigheden binnen elke gewenste grens gehouden kunnen worden.

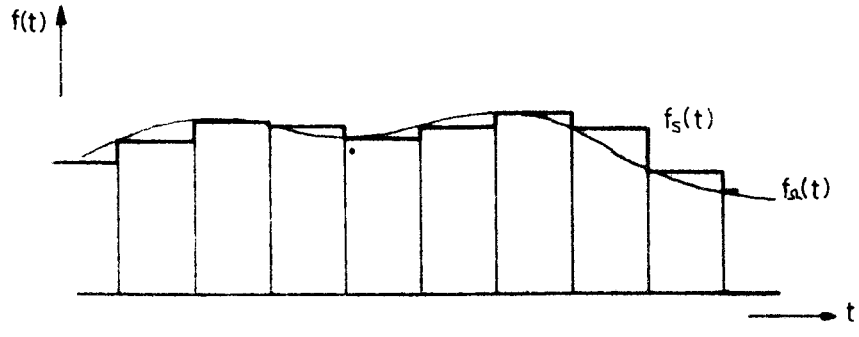


Figuur 5.3: Autonome deelsystemen.

Wanneer we voorlopig (zie hoofdstuk 6.0) aannemen dat het transmissiesysteem een constante vertraging voor de meetwaarden vormt, dan treden slechts asynchrone overgangen op bij de stippellijnen in figuur 5.4. Het opnemersysteem biedt meetwaarden aan, het transmissiesysteem haalt deze meetwaarden op en biedt ze een vaste tijd later weer aan de gebruiker aan en deze laatste haalt op zijn beurt (asynchroon) weer meetwaarden op. Bij A en B vindt eigenlijk een bemonstering van een reeds bemonsterd signaal plaats. Dit is in principe onmogelijk.

Volgens het bemonsteringstheorema dient de samplefrequentie groter te zijn dan twee keer de hoogste frequentie in het over te dragen signaal. Wanneer het te bemonsteren signaal zelf een bemonsterd signaal is, heeft dit een oneindig breed frequentiespectrum (figuur 5.2). Bemonstering hiervan kan alleen gebeuren met oneindig hoge samplefrequentie.

De tijdfunctie van een bemonsterd signaal in het meetsysteem heeft niet de vorm van figuur 5.2. De meetwaarde blijft geldig tot een nieuwe is geproduceerd:



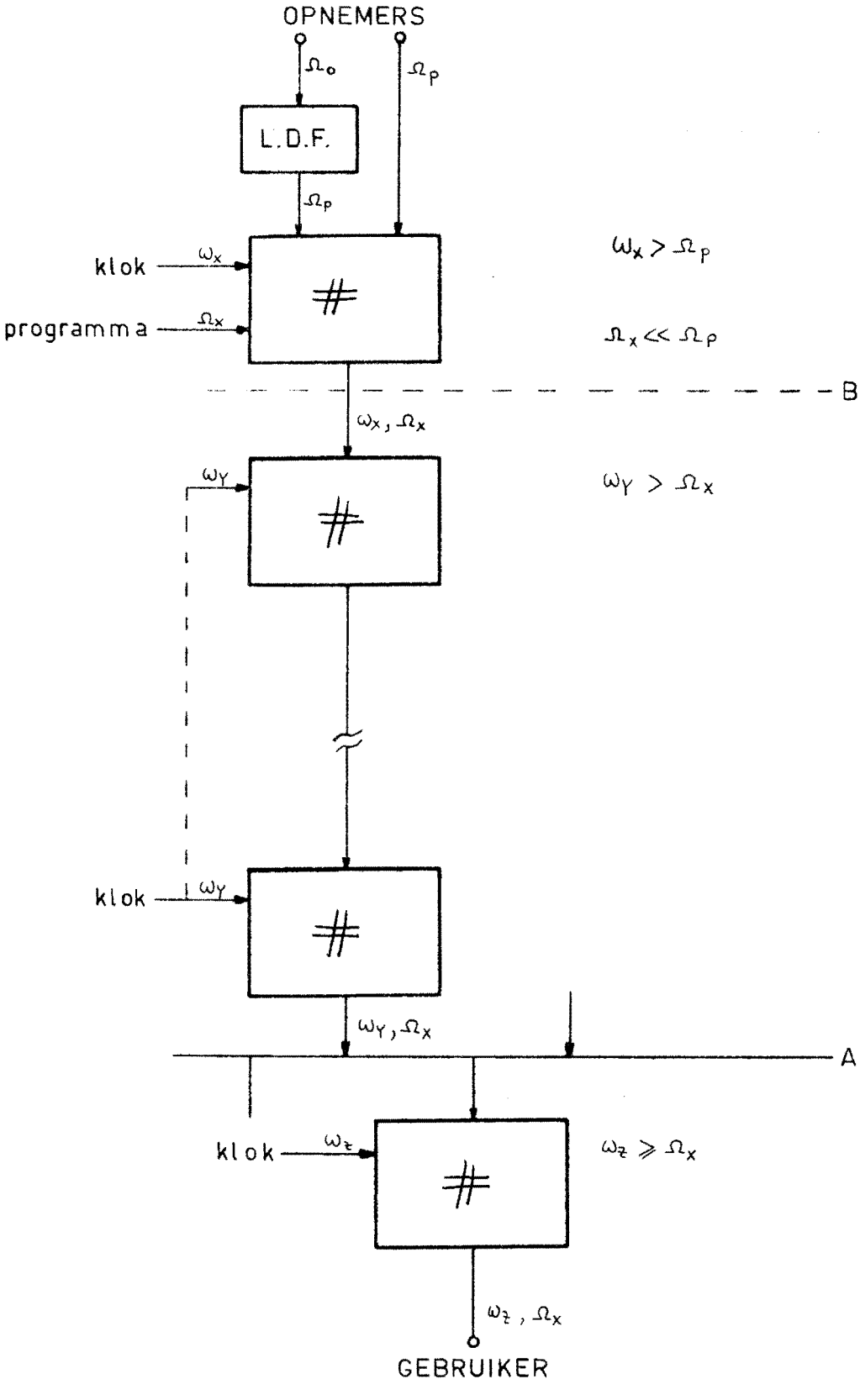
Figuur 5.4: Meetsignaal.

Toch blijft het onmogelijk dit signaal (naar de frequentie onbegrensd) correct te bemonsteren, maar we kunnen hier wel een goede benadering vinden.

We zullen voor het vervolg aan een digitaal meetsignaal, x, twee parameters toevoegen:

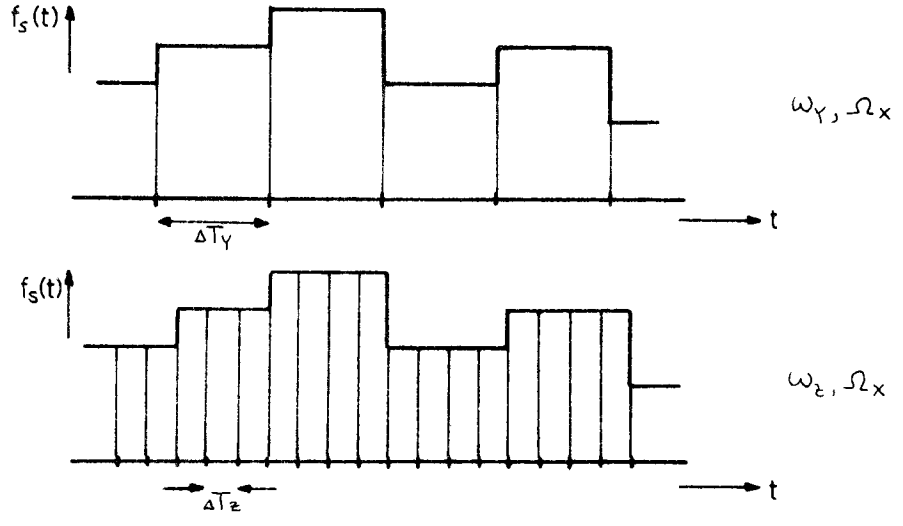
ω_x ;het signaal heeft sampletijd $T = 2\pi/\omega_x$

Ω_x ;het digitale signaal bevat informatie tot $\Omega_x/2$.



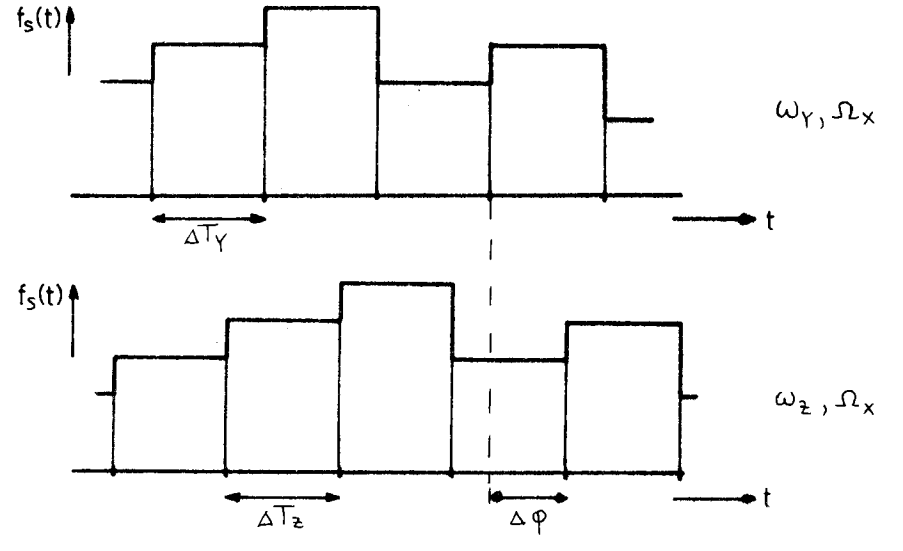
Figuur 5.5: Frequenties in het systeem.

Voor een asynchrone overgang zou een goede benadering zijn om (zie b.v. overgang A, figuur 5.5) $\omega_z \gg \omega_Y$ te kiezen. Er ontstaat slechts een geringe faseverschuiving van maximaal ΔT_z .



Figuur 5.6: $\omega_z \gg \omega_Y$.

5.4.3.OVERGANG A. Gezien de grootte van de datastroom bij overgang A is het onmogelijk hier met $\omega_z \gg \omega_Y$ te bemonsteren. We stellen hier $\omega_z \approx \omega_Y \geqslant \Omega_x$. Het bemonsteringsproces (met ω_z) is nu, door de gelijke frequentie, in zekere zin gecorreleerd met het "dataproductieproces" (met ω_Y). De fout die hierdoor ontstaat kunnen we geheel toeschrijven aan een fase onzekerheid $\Delta\varphi$.



Figuur 5.7: $\omega_z \approx \omega_Y$.

Dit is toelaatbaar voor overgang A, omdat het hier een onzekerheid in de faserelatie tussen twee groepen opnemers betreft.Bij een afstand tussen de groepen van 20 meter en ΔT_z maximaal 1/25 sec zou dit overeenkomen met een voortplantingssnelheid van 1000 km/uur van het gemeten verschijnsel.

5.4.4 OVERGANG B. Bij overgang B (figuur 5.5) kan, gezien de beperkte capaciteit van het transmissiekanaal, ook niet met $\omega_Y \gg \omega_X$ bemonsterd worden. Maar de oplossing $\omega_Y \approx \omega_X$ geeft hier geen uitkomst: de opnemers binnen een groep staan veel dichter bij elkaar, zodat de optredende fasefout niet getolereerd kan worden.

We kunnen de bemonsterfrequentie van de opnemers (ω_X) zodanig groot maken, dat met $\omega_X \gg \omega_Y > \Omega_X$ aanvaardbare resultaten verkregen worden. In Appendix B is een berekening van de signaal/ruisverhouding onder deze condities weergegeven. Het resultaat:

5.4.4.1 | s/r = 10*log(1,5*(p*q)**2), met $p=\omega_X/\Omega_X$ en $q=A/\Delta A$

voor een signaal van de vorm: $f(t)= x*A + \Delta A*f'(t)$, een fluctuatie rond een gemiddelde waarde. Programmatisch kunnen we nu in het opnemersysteem (p*q) voldoende groot maken. Voor signalen die zich gedragen als hierboven beschreven (ΔA), en dit zijn de meeste meteorologische grootheden, is q reeds groot (b.v. 40). Bij bekende Ω_X wordt dan de minimale samplefrequentie bepaald door de toelaatbare signaal/ruisverhouding. Voor p is bijvoorbeeld 25 komen we tot een s/rverhouding van 60 dB.

5.5 CONCLUSIES. Hier volgt nog een kort overzicht van de conclusies uit dit hoofdstuk en de hieruit voortvloeiende eisen voor de rest van het meetsysteem.

1. Eenvoudig filter te gebruiken voor analoge opnemers.

2. Opnemers met hoge frequentie te bemonsteren.

3. "Processing power" bij opnemers gebruiken voor berekening van filters of gemiddelden.

4. Transmissiesysteem moet constante vertraging voor de meetwaarden vormen.

5. De gebruiker bemonstert asynchroon de door het distributiesysteem aangeboden informatie.

6. Een gebruiker (b.v. minicomputer) kan de filterkarakteristieken (3.) programmatisch veranderen. De begrenzing vormt de eis $\omega_x \leq \omega_y$, de snelheid van het transmissiesysteem.

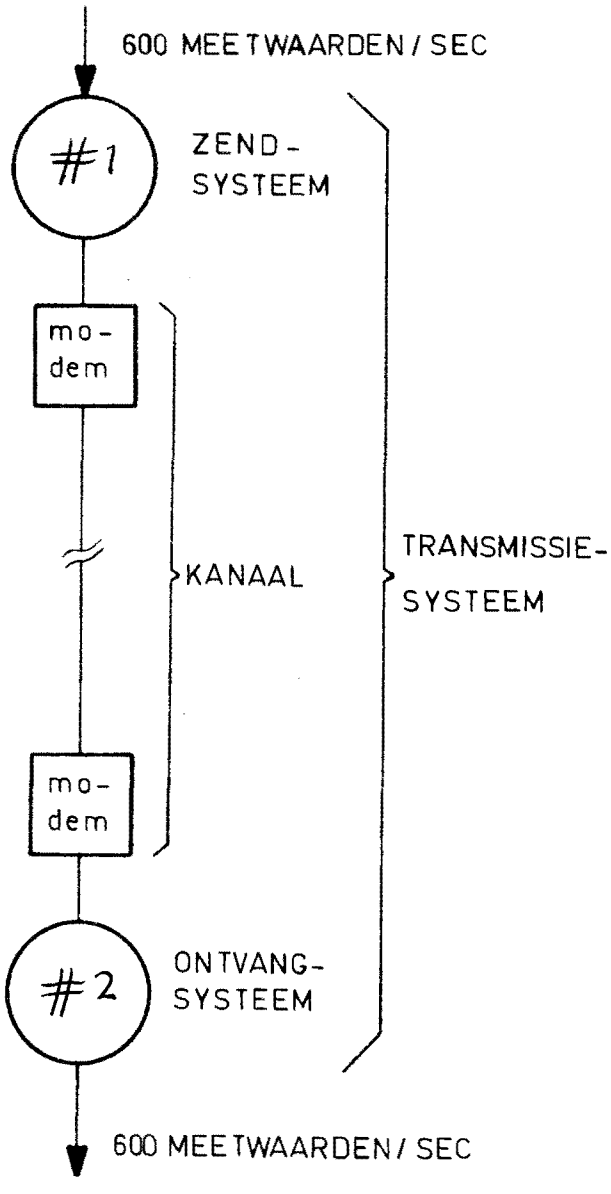
7. Om aan de theoretische eisen van bemonstering te voldoen, moet een langzamer randapparaat ($\omega < \omega_t$) toch ook met ω_t de meetwaarden aannemen en door berekening tot een juist spectrum geraken.

8. De organisatie van het distributiesysteem dient zodanig te zijn dat aan de gestelde voorwaarde voldaan blijft. D.w.z. de meetwaarden binnen een groep moeten zonder faseverschil doorgestuurd worden en tussen de groepen mag een geringe verschuiving toegestaan worden.

In het volgende hoofdstuk zal de eis voor het transmissiesysteem ΔT=constant worden uitgewerkt.

6.0 DATATRANSMISSIESYSTEEM.

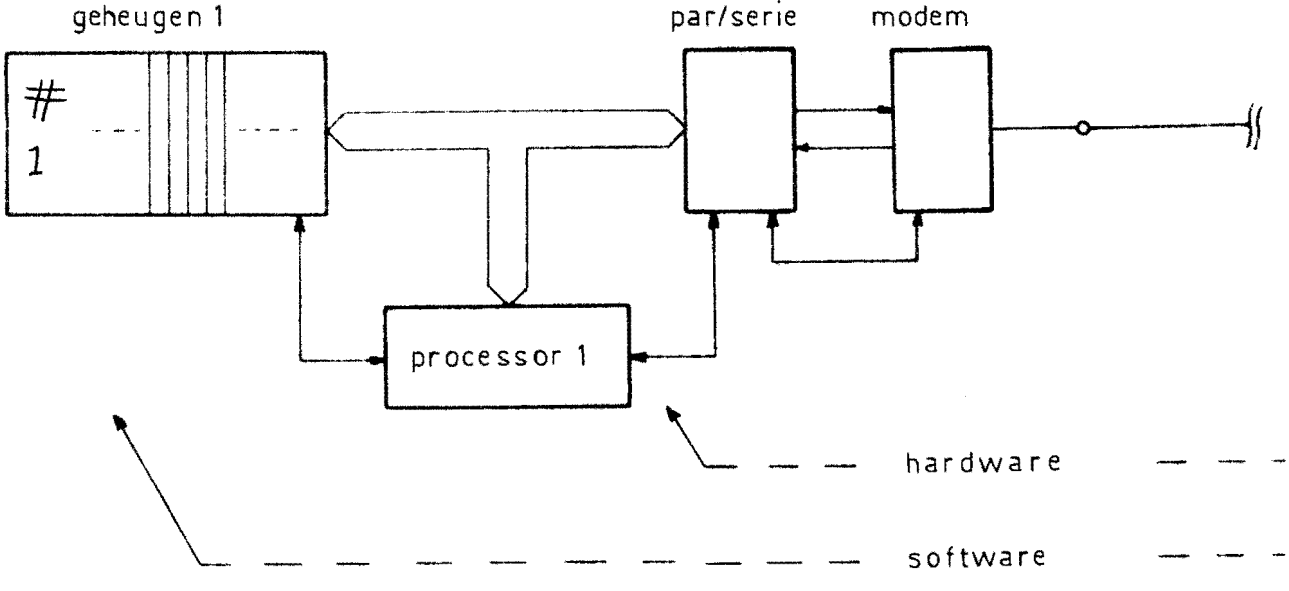
In het vorige zijn de eisen voor het transmissiesysteem geformuleerd. Ten aanzien van de capaciteit kunnen we nog het volgende opmerken; het heeft geen voordeel bij minder, dan maximaal mogelijk, aangesloten opnemers ook minder informatie te versturen. Dit houdt in dat we het systeem altijd de maximale hoeveelheid van 640 meetwaarden zullen laten transporteren. Dit vereenvoudigt de programmatuur voor de betrokken microcomputers en hun synchronisatie.



Figuur 6.1:
Schematische
voorstelling
transmissiesysteem.

Het zendsysteem, #1, bemonstert met twee frequenties, $\omega_x 1,2$, het opnemersysteem. Dit levert de 640 meetwaarden per seconde op. Deze waarden moeten correct overgebracht worden naar het ontvangsysteem #2, dit laatste dient nu weer de meetwaarden aan het distributiesysteem aan te bieden. In hoofdstuk 5.4 is als eis voor het transmissiekanaal gesteld een constante vertraging $\Delta T=c$ voor de meetwaarden. Dit vereist een strakke koppeling tussen zender en ontvanger.

6.1 TRANSMISSIEMETHODEN. Bij transport van informatie tussen twee computers hebben we te maken met zogenaamde protocollen; d.w.z. afspraken over de manier waarop data wordt uitgewisseld. Deze afspraken hebben zowel betrekking op de timing (inclusief "handshake") op hardware niveau, als op de gebruikte code en de manier van adressering. Bij elke overgang in een systeem leiden goede protocollen tot synchronisatie van de informatieuitwisseling. We zullen de mogelijkheden onderzoeken voor het serie-data transport tussen twee microcomputers.



Figuur 6.2: Data-uitwisseling: protocollen.

6.1.1 MODEM. Er zijn een tweetal standaardmethoden voor serie-transmissie: synchroon en asynchroon. Het onderscheid wordt gemaakt naar de manier waarop bitsynchronisatie tussen zender en ontvanger gerealiseerd wordt. In het synchrone geval wordt de zendklok meegestuurd, of met een extra kabel, of over dezelfde kabel in een "gat" in de frequentieband van het datasignaal. In het tweede geval wordt dit bereikt door synchronisatiebits mee te zenden, start- en stop-bits.

6.1.2 PARALLEL/SERIE. Wanneer nu bitsynchronisatie is bereikt moet nog uitgemaakt worden hoe deze bits tot het juiste woord omgezet kunnen worden (woordsynchronisatie). In het asynchrone geval zorgen de start- en stop-bits voor de herkenbaarheid van een woord in de datastroom. Bij synchroon bedrijf gebeurt dit door voor elk blok informatie één of meer "sync-karakters" te versturen. In beide gevallen moet de (digitale) serie-parallel omzetter deze synchronisatie verzorgen.

Het voordeel van de synchrone methode is de geringe overhead. Bij voorbeeld voor het versturen van 100 bytes van 8 bit:
asynchroon: 100 * (8 + startb + stopb) = 1000 bits
synchroon: (100 + 2*sync) * 8 = 816 bits.
Bij een bepaalde transmissiesnelheid kan met synchroon data-transport de meeste informatie per seconde verzonden worden.

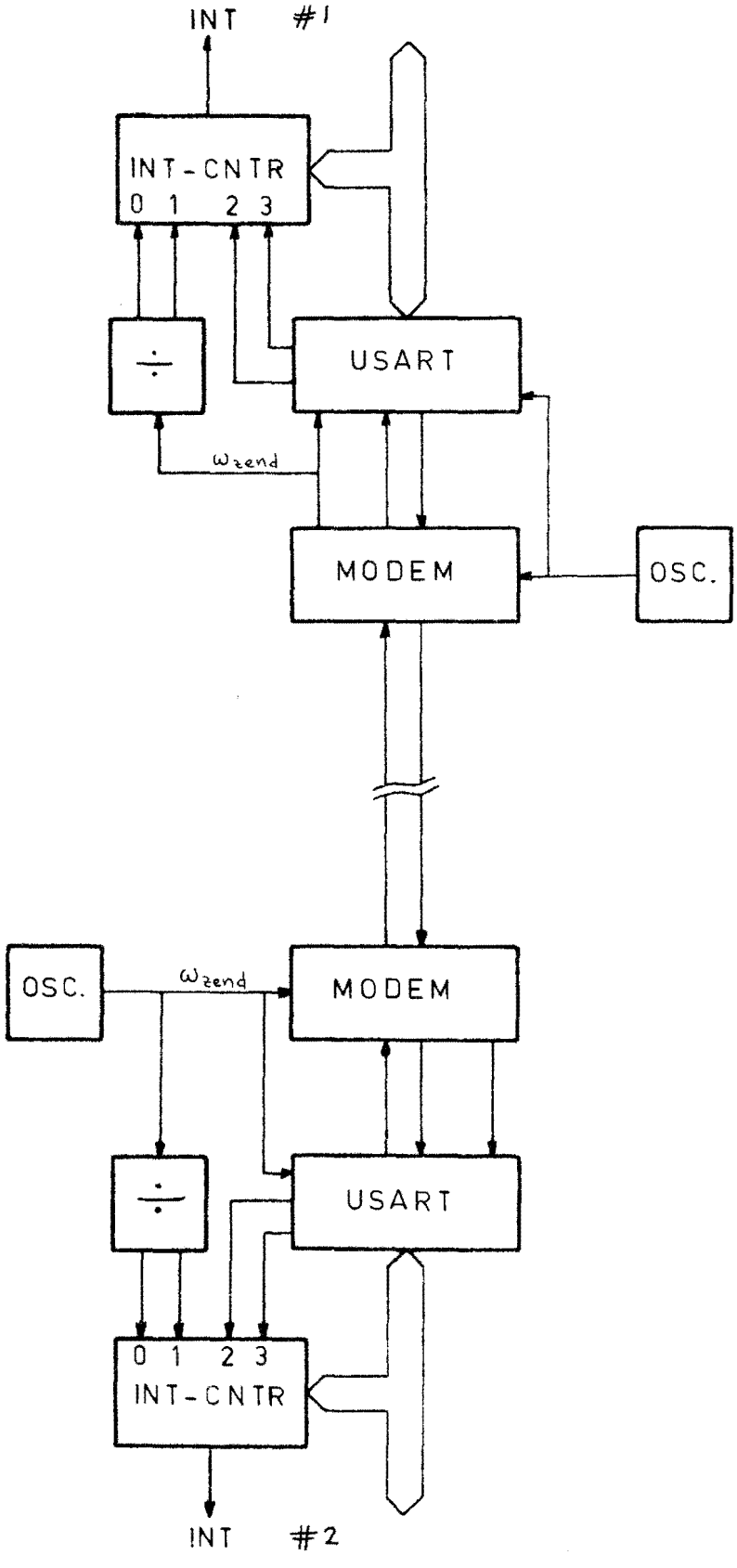
6.1.3 PROCESSOR. De synchronisatie tussen parallel/serie-omzetter en microprocessor kan ook op verschillende manieren verlopen. De gebruikelijke methode is het herhaaldelijk opvragen van de status van de omzetter tot weer een actie (byte van of naar de processor) kan plaatsvinden. De microcomputer kan in de tijd dat data ontvangen of verstuurd moet worden niets anders doen dan deze statuscontrole. In een meetsysteem moet voortdurend data verstuurd worden (real time-meetwaarden), zodat deze methode niet kan worden toegepast, de processor heeft geen tijd meer voor de andere besturingstaken.

Een andere methode is het verdelen van de taken van de microcomputer in een aantal korte acties, die gestart worden door gebeurtenissen buiten de processor: interrupts.
Door een interrupt wordt het op dat moment lopend programma onderbroken en wordt een specifieke interruptservice-routine gestart. Deze routine kan hardware (interruptcontroller) gekozen worden, of programmatisch: een algemene interruptroutine zoekt het interrumperend systeem (polling) door statuscontrole. De hardware methode werkt het snelst en interruptcontrollers zijn standaard bouwstenen in de meeste microcomputersystemen.

Bij meerdere randapparaten die kunnen onderbreken moet een prioriteitschema aangehouden worden. Sommige interrupts zijn urgenter dan andere, sommige mogen bij voorbeeld niet onderbroken worden. De verkrijgbare programmeerbare interruptcontrolers houden dit prioriteitsschema bij. In samenwerking met de serviceroutine wordt nu een snel reagerend regelsysteem verkregen.

6.1.4 GEHEUGEN. Na deze synchronisatie tussen parallel/serieomzetter en processor rest nog het mechanisme om informatie te herkennen. De software van het ontvangsysteem moet de gegevens op de goede plaats in het geheugen kunnen opbergen. Dit herkennen kan gebeuren doordat in een blok informatie een zogenaamde header wordt meegestuurd, de naam van het blok. Bij afspraak kunnen alle eropvolgende woorden herkend worden aan hun plaats in het blok.

6.2 REALISATIE. Op grond van voorgaande overwegingen ten aanzien van constante vertraging, synchroon transport en mogelijke protocollen, wordt het transmissiesysteem als volgt opgebouwd:



Figuur 6.3: Koppeling door interruptmechanisme.

Het transport tussen de USARTs (Universal (A)Synchronous Receiver Transmitters), een standaard component in microcomputersystemen, vindt synchroon plaats. De klok ω_{zend} wordt meegestuurd (zie Appendix C). Deze klok wordt tevens gebruikt om het bemonsteren van het opnemersysteem, door de zender #1, en het aanbieden van nieuwe data, door de ontvanger #2, aan de gebruiker, te synchroniseren : $\Delta T=c$. (Appendix E)

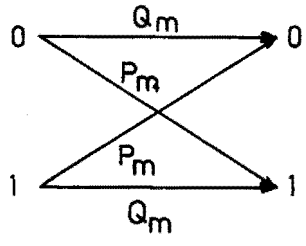
Na het vastleggen van deze configuratie van het systeem, moeten we methoden onderzoeken, die de betrouwbaarheid ervan kunnen verhogen.

6.3 CODERING. De enige manier om bij informatietransport opgetreden fouten te kunnen detecteren is het meesturen van extra (redundante) informatie. B.v., bij transport van een analoog signaal kunnen we gebruik maken van twee kabels, over de ene verzenden we het signaal +A maal versterkt, over het andere -A maal. Door aan ontvangstzijde het verschil te bepalen van beide signalen worden storingen ontstaan bij het transport gefilterd. Bij modems voor digitale transmissie wordt ook vaak van deze techniek gebruik gemaakt om de seriedata te versturen. Maar het digitale karakter van de informatie maakt nog een andere vorm van redundantietoevoeging mogelijk: codering.

N.l. door toevoeging van een aantal codebits aan een groep informatiebits kunnen bitfouten gedetecteerd en eventueel gecorrigeerd worden. Het rendement van het systeem neemt door deze redundantie af, dit effect komt nader aan de orde in 6.4.4. Bij voldoende beschikbare tijd kunnen geschikte codes tot een foutloze overdracht leiden. Voor gebruik van codering in het transmissiesysteem gaan we uit van een rekenmodel bestaande uit twee modems + kabel, die samen een Binary Symmetric Channel vormen. Deze "black box" kennen we een bepaalde bitfoutenkans toe om vervolgens wegen te zoeken om het effect van fouten te verminderen. Aan de hand van deze beschouwingen komen enkele gebruike-

lijke coderingstechnieken aan de orde.

6.3.1 MODEL. We stellen de kans dat in de black box een bitfout optreedt Pm en de kans dat een bit correct overkomt Qm. Voor een Binary Symmetric Channel vinden we dan:

gezonden 

ontvangen, met Qm>Pm, Qm=(1-Pm).

De kans Pw dat een woord van n bits een of meer fouten bevat is dan Pw= (1-Qm**n)= 1-(1-Pm)**n. We onderzoeken nu de invloed van Algebraische codes (ook wel cyclische, lineaire of polynoomcodes genoemd) op Pw. Dit zal geen essentiele beperking blijken te zijn, enkele van de meest gebruikte codes vloeien hier rechtstreeks uit voort. Ook blijken de gewenste methodes eenvoudig in het systeem te implementeren.

6.3.2 ALGEBRAISCHE CODES. Bij algebraische codes stelt men een woord van n bit voor als een polynoom van graad (n-1).

b.v. 100110111 <=> x**8 + x**6 + x**5 + x**4 + x**2 + x + 1.

De optellingen geschieden modulo 2. We beschouwen een woord van n bit ontstaan uit k informatiebits. Om tot een systematische code te geraken, informatie en checkbits gescheiden van elkaar, wordt als volgt een codewoord met een gegeven generatorpolynoom bepaald:

G(x)= generatorpolynoom (graad n-k)

W(x)= informatiewoord (graad k-1)

V(x)=codewoord= R(x) + W(x)*x**(n-k), met

R(x)= rest van de deling W(x)*x**(n-k)/G(x).

Op deze wijze verkrijgen we de checkbits op de minst significante n-k posities en de oorspronkelijke informatiebits op overige plaatsen (systematisch). De hierdoor ontstane codewoorden V(x) zijn deelbaar door G(x). Bij ontvangst van een woord V'(x) kan door deling door het bekende generatorpolynoom worden vastgesteld of V'(x) een geldig codewoord is, de rest van de deling is dan 0.

De kracht van een code kan berekend worden door na te gaan welke fouten in het ontvangen woord ongedetecteerd blijven. We zullen een fout in het ontvangen woord V'(x) aangeven door een foutpolynoom E(x):

V'(x)= V(x) + E(x).

[Een '1' in de foutvector inverteert (optelling is modulo 2) de bijbehorende bitpositie van V(x):

```
      1011  =   1001    + 0010
      V'(x) =   V(X)    + E(x)
x**3 + x + 1 = x**3 + 1  + x          ]
```

Aan ontvangkant wordt een fout in V'(x) alleen gedetecteerd als E(x) ondeelbaar is door G(x), een eigenschap van lineaire codes aangezien voor een lineaire code de som van twee codewoorden ook een codewoord is.

Wanneer G(x) meer dan een term =0 bevat worden enkele fouten gedetecteerd: E(x)= x**i, x**i ondeelbaar door G(x) als G(x) meer dan een term heeft.

6.3.3 PARITEITSCONTROLE. Als voorbeeld kiezen we G(x)= x + 1. De graad van dit polynoom is 1, dus bedraagt het aantal checkbits ook 1. Laat

W(x)= x**6 + x**3 + 1 <=> 1001001

dan is V(x)= x.W(x) + R(x) <=> 10010010 + 1=10010011.

De code detecteert elke foutvector E(x), die ondeelbaar is door x + 1 (die dus een oneven aantal termen ≠0 bevat, lit 4). Dit komt overeen met de "even-pariteitscontrole".

6.3.4 HAMMINGCODE. Wanneer G(x) deler is van x**e + 1, maar niet van x**c + 1, met c<e, dan noemt men e de exponent van G(x). Wanneer e de exponent is van G(x), detecteert een code, door G(x) gegenereerd, van lengte n<e alle enkele en dubbele fouten.

(bewijs:
enkele fouten : G(x) bevat meer dan 1 term, immers G(x)=x en G(x)=1 zijn onbruikbaar,
dubbele fouten: dan is E(x)= x**i + x**j , met 0<i<n en 0<j<n,
= (x**(i-j) + 1)*x**j, voor j<i.
Daar i-j<n<e is (x**(i-j) + 1) ondeelbaar door G(x) immers e is de exponent van G(x).)

Er is voor m (geheel) een generatorpolynoom G(x), met graad m, te vinden van exponent e=2**m -1. Waarbij e de grootst mogelijke waarde is: G(x) is een primitief polynoom. Deze genereert dus een enkele en dubbele fouten detecterende code voor codewoorden met lengte n< 2**m -1, waarvan m checkbits.

m	2**m -1	G(x)
1	1	1+x
2	3	1+x+x**2
3	7	1+x+x**3
4	15	1+x+x**4
⋮		
13	8191	1+x**9 +x**10 +x**12 +x**13
14	16383	1+x**4 +x**6 +x**13 +x**14
15	32767	1+x +x**15
16	65535	1+x**4 +x**13 +x**15 +x**16

Tabel 6.1: Primitieve polynomen (literatuur 5)

Het zal blijken dat bij gebruik van een primitief polynoom zelfs enkele fouten gecorrigeerd kunnen worden. Aan ontvangkant kan uit het ontvangen woord, mits het maximaal 1 fout bevat, het verzonden codewoord gedestilleerd worden. Als voorbeeld nemen we de generator G(x)= 1+x+x**3, voor een codewoord van 2**m-1=7 bit waarvan m=3 checkbits. Berekening van alle V(x) volgens de voorgeschreven methode levert:

W(x)	R(x)	W(x)	R(x)
6 5 4 3	2 1 0	6 5 4 3	2 1 0
0 0 0 0	0 0 0	1 0 0 0	1 0 1
0 0 0 1	0 1 1	1 0 0 1	1 1 0
0 0 1 0	1 1 0	1 0 1 0	0 1 1
0 0 1 1	1 0 1	1 0 1 1	0 0 0
0 1 0 0	1 1 1	1 1 0 0	0 1 0
0 1 0 1	1 0 0	1 1 0 1	0 0 1
0 1 1 0	0 0 1	1 1 1 0	1 0 0
0 1 1 1	0 1 0	1 1 1 1	1 1 1

Tabel 6.2: V(x) voor G(x)=1+x+x**3 .

Hier is ook het cyclische karakter van de code te zien, wanneer de bits van een codewoord over één positie cyclisch verwisseld worden ontstaat weer een geldig codewoord. Uit de tabel blijkt dat de checkbits R gevormd kunnen worden door de modulo 2 optelling van een groep informatiebits W:

```
R2 = W6 + W5 + W4
R1 =      W5 + W4 + W3
R0 = W6 +      W4 + W3
```

Wanneer nu aan ontvangkant een fout in V(x) is opgetreden kan door pariteitscontrole van:

```
Q2 = R2 + W6 + W5 + W4
Q1 = R1 +      W5 + W4 + W3
```

Q0 = R0 + W6 + W4 + W3

met de nu verkregen vector Q0.Q1.Q2 de plaats van de fout gevonden worden:

Q0.Q1.Q2	plaats
0 0 0	: geen fout
0 0 1	: R2
0 1 0	: R1
0 1 1	: W5 <---
1 0 0	: R0
1 0 1	: W6 <---
1 1 0	: W3 <---
1 1 1	: W4 <---

De vector Q, de syndroomvector genoemd, wijst naar de plaats van de fout in het ontvangen woord. Door inverteren van het betreffende bit is correctie verkregen. Deze manier van decoderen berust op het principe van Maximum Likelyhood Decoding, n.l. met de aanname dat de kans op n fouten in een woord groter is dan de kans op n+1 fouten. Voor het in 6.3.1 gestelde Qm>Pm is hieraan voldaan. We hebben nu een mechanisme verkregen dat ook bekend is onder de naam "Hamming enkele fouten corrigerende code".

6.3.5 CRC. De kracht van cyclische codes bij het optreden van foutenexplosies (bursts) zullen we nu onderzoeken. Onder een burst van lengte b verstaan we een willekeurig foutenpatroon, zoals tot uitdrukking komt in de vector E(x), waarbij het verschil van de graad van de eerste term $\neq 0$ en de graad van de laatste term $\neq 0$ gelijk is aan b-1.

* Wanneer een burst optreedt met lengte b<m wordt deze altijd gedetecteerd daar G(x) van de graad m is en dus E(x) ondeelbaar is door G(x).(De foutvector is n.l. te schrijven als E(x)=E'(x)*x**i, met E'(x) van de graad b-1. Daar x**i ondeel-

baar is door G(x) en E'(x) ook wordt deze burst gedetecteerd.)

* Voor b=m+1 (dit zijn 2**(b-2) mogelijke patronen) wordt slechts de realisatie E'(x)=G(x) niet gedetecteerd.

* Voor b>m+1 zijn van de 2**(b-2) mogelijke foutpatronen er 2**(b-2-m) deelbaar door G(x).

Samenvattend kunnen we zeggen dat de cyclische code alle bursts met lengte b<m+1 detecteert (behalve één) en de rest met kans:

1 - ongedet/mogelijk= 1 - 2**(b-2-m)/2**(b-2)= 1 - 2**(-m).

ALGEMEEN: Een codewoord van n bits lengte, waarvan m checkbits, heeft 2**n mogelijke waarden, waarvan 2**(n-m) toegestane codewoorden. Bij ontvangst van een codewoord, ook al is dit ontstaan uit een ander verzonden codewoord en een foutvector, kan geen fout opgemerkt worden. Wanneer dus het ene codewoord in het andere overgaat is foutdetectie onmogelijk, de detectiekans is dus

1 - ongedet/mogelijk= 1 - (2**(n-m)-1)/(2**n-1)~ 1 - 2**(-m).

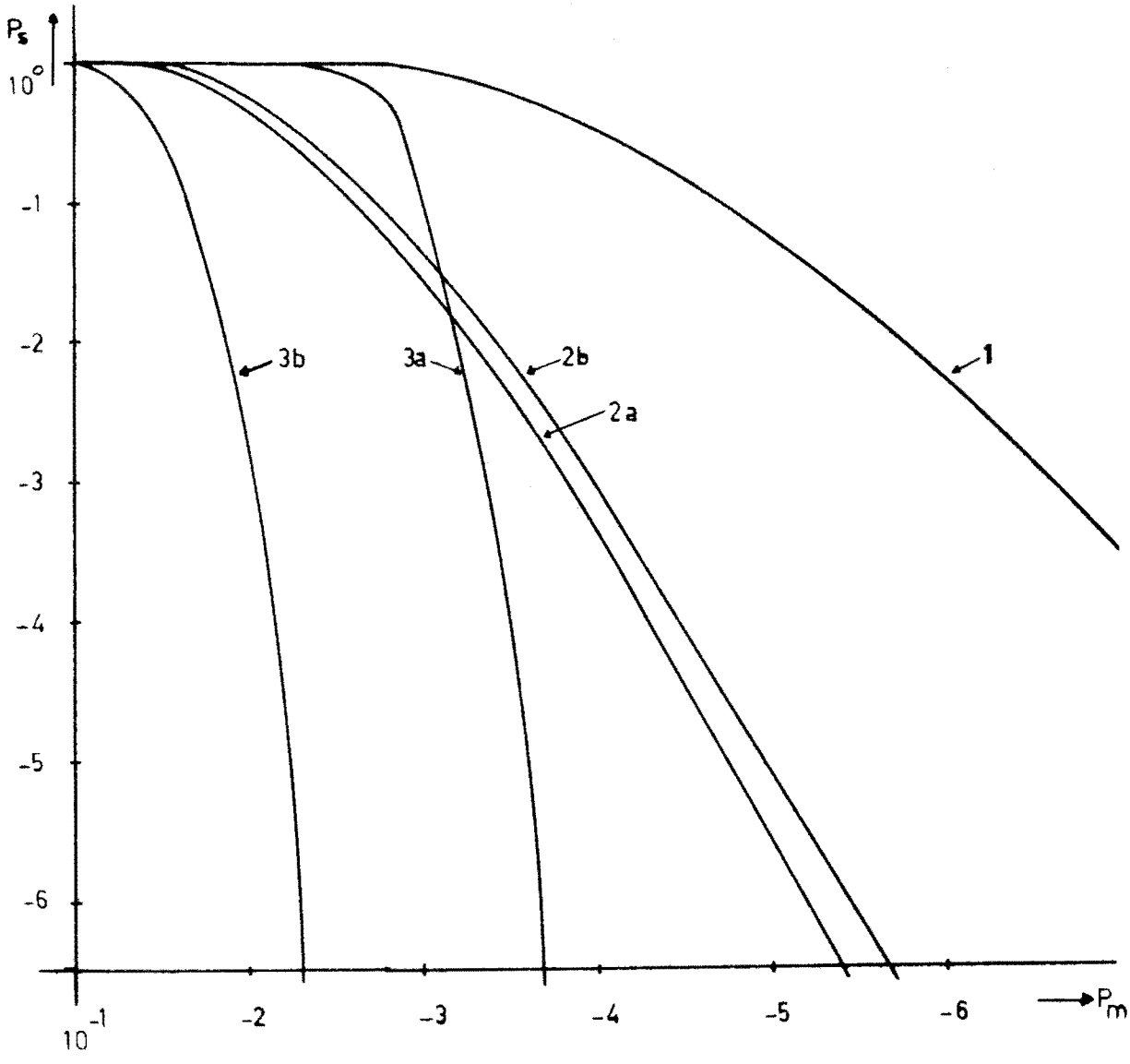
Wanneer we een cyclische code gebruiken om bursts in een blok informatie te detecteren, kiezen we een primitief polynoom, behorend bij de gewenste woordlengte n (in dit geval het aantal bits in een blok), en voegen een factor (1+x) toe. Deze factor neemt alle oneven fouten voor zijn rekening, zodat nu ook drievoudige fouten gedetecteerd worden naast de genoemde enkele en dubbele en de bursts. B.v. (1+x)*(1+x+x**15)=1+x**2+x**15+x**16, een gebruikelijke Cyclic Redundancy Check generator (analoog aan de Fire-codes: (1+x**c)*G(x), literatuur 4.).

6.4 TOEPASSING. Voor datatransmissie bestaat nu de mogelijkheid te kiezen uit foutcorrectie of foutdetectie. Foutdetectie heeft

slechts effect wanneer er een tweerichtingsverkeer mogelijk is tussen zender en ontvanger. De ontvanger moet bij een foutdetectie aan de zender om retransmissie vragen. Deze methode wordt in rekenmachinesystemen wel toegepast tussen computer en terminal.

Foutcorrigerende codes (forward correction) wordt veelal toegepast wanneer geen retransmissie kan plaatsvinden: b.v. bij satellieten en sommige invoersystemen bij rekenmachines.

Een andere strategie is het toepassen van foutcorrigerende codes voor regelmatig optredende foutpatronen (ruis) in combinatie met detectie en retransmissie. Door deze forward correctie kan het aantal nodige retransmissies beperkt worden. We kunnen de verschillende methoden met elkaar vergelijken, door de kans op een of meer fouten per seconde in een grafiek uit te zetten, tegen de bitfoutenkans Pm van het kanaal. We moeten de methoden per tijdseenheid vergelijken omdat we te maken hebben met een meetsysteem, hier moet een bepaalde hoeveelheid informatie per tijdseenheid verstuurd kunnen worden. Een maat voor de betrouwbaarheid is nu de foutkans per tijdseenheid.



Figuur 6.4: Kans op een of meer fouten per seconde.

De krommen in de figuur zijn als volgt tot stand gekomen, voor a woorden per seconde van 8 bit en f = 8*a:

6.4.1) DIRECT.

Woordfoutenkans: Pw = 1 - (1 - Pm)**8

Secondefoutkans: Ps = 1 - (1 - Pm)**(8*a)

Seinsnelheid : Fm = f

6.4.2a)CORRECTIE. Bij gebruik van het generatorpolynoom 1 + x + x**3 voor telkens 4 bit van een woord.

Woordfoutenkans:

$$Pw = 1 - (1 - Pm)^{**}7 - 7*Pm*(1 - Pm)^{**}6$$

Secondefoutkans:

$$Ps = 1 - ((1 - Pm)^{**}7 + 7*Pm*(1 - Pm)^{**}6)^{**}2a$$

Seinsnelheid :

$$Fm = 1{,}75*f$$

6.4.2b) CORRECTIE. Bij gebruik van generatorpolynoom $1 + x + x^{**}4$ voor telkens 8 bit (een woord).

Woordfoutenkans:

$$Pw = 1 - (1 - Pm)^{**}12 - 12*Pm*(1 - Pm)^{**}11$$

Seconde foutkans:

$$Ps = 1 - ((1 - Pm)^{**}12 + 12*Pm*(1 -Pm)^{**}11)^{**}a$$

Seinsnelheid :

$$Fm = 1{,}5*f$$

6.4.3) RETRANSMISSIE. We nemen hier aan dat 10 blokken nodig zijn voor de meetwaarden en 10 voor retransmissie (zie appendix D):

Blokfoutenkans :

$$Pb = 1 - (1 - Pw)^{**}(a/10),$$ met Pw:woordfoutenkans

Secondefoutkans:

$$Ps = 1 - \sum_{x=0}^{10} \binom{20}{x} *Pb^{**}x*Qb^{**}(20 - x),$$ met:

3a) Woordfoutenkans:

$$Pw = Pm^{**}8$$

Seinsnelheid :

$$Fm = 2*f,$$ zonder forward correction.

3b) Woordfoutenkans:

$$Pw = 1 - ((1 - Pm)^{**}7 + 7*Pm*(1 - Pm)^{**}6)^{**}2$$

Seinsnelheid :

$$Fm = 3{,}5*f,$$ met forward correction (2a)

We zien dat de combinatie 3b zeer goede resultaten oplevert. Dit was ook te verwachten aangezien hier de toegevoegde redundantie het grootst is (~70%). Toch heeft de combinatie enkele

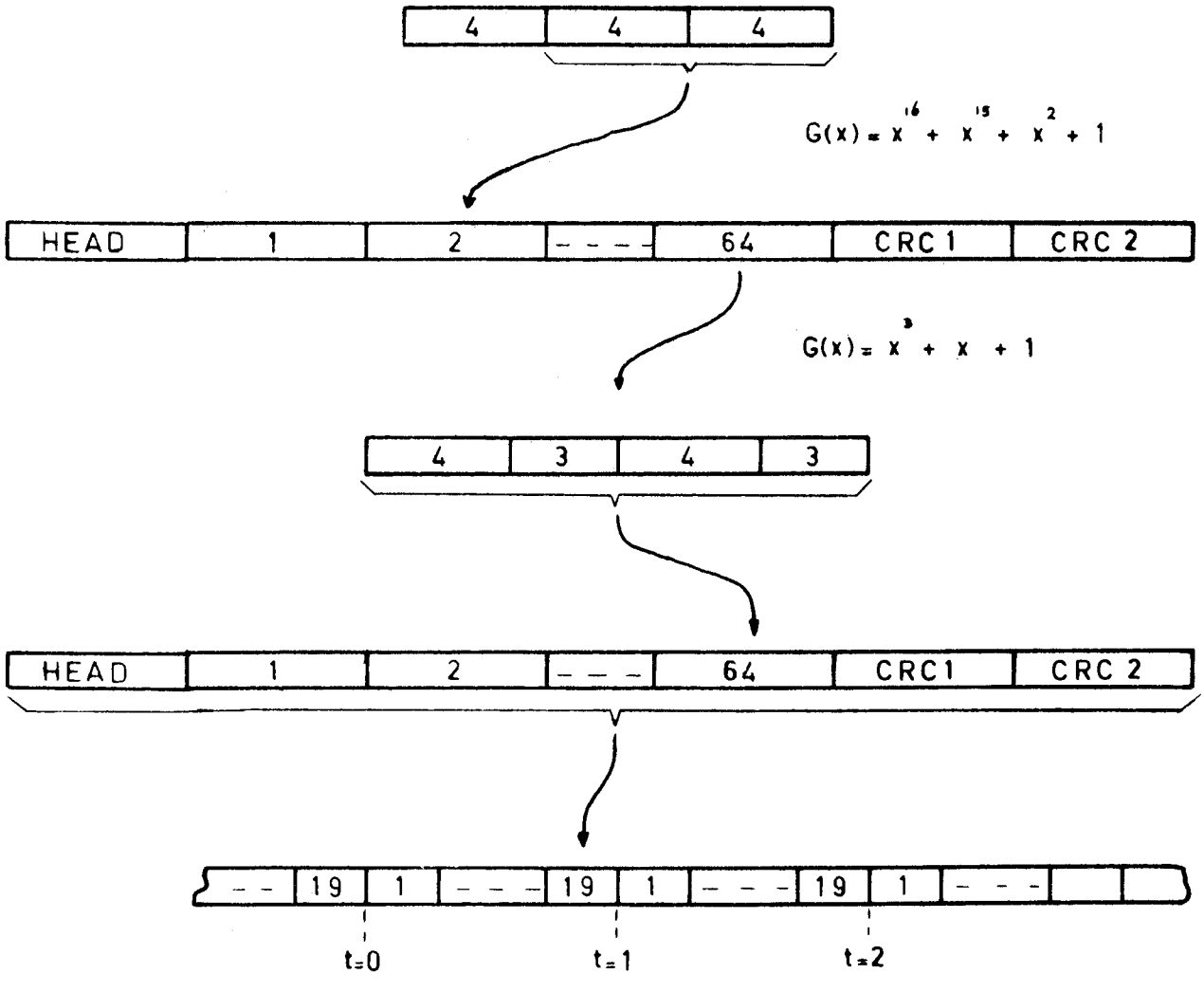
specifieke voordelen t.o.v. correctie of retransmissie apart: door correctie behoeven minder blokken herhaald te worden, dit betekent dat het retransmissiesysteem niet snel vast loopt door thermische ruis. Voorts geeft de retransmissiemethode de mogelijkheid foutenexplosies (statische ontlading, schakelmanipulaties) op te vangen, die door het correctiemechanisme niet hersteld kunnen worden.

6.4.4 NADELEN. In figuur 6.5 zien we dat de verwachte foutenkrommen door codering aanzienlijk gunstiger komen te liggen. Er zijn echter een aantal nadelen, vooral met betrekking tot toepassing in een meetsysteem. Ten eerste worden de woorden die ontvangen worden groter t.g.v. de toegevoerde checkbits. De hierdoor vergrote woordfoutenkans Pw is in de grafieken reeds verwerkt. De noodzaak om de transmissie snelheid te verhogen bij constante hoeveelheid meetwaarden per seconde en toenemende redundantie, heeft twee gevolgen: de bitfoutenkans Pm van modem + kabel t.g.v. thermische ruis neemt toe (anders gezegd; om dit te voorkomen moet de combinatie aan hogere eisen voldoen) en bij een storing van t seconde op de kabel zal een langere (meer bits) burst het gevolg zijn.

6.5 CONCLUSIE. We kunnen concluderen dat correctie en detectie + retransmissie in een meetsysteem, door de beperkte tijd die ter beschikking staat, de beste vooruitzichten op betrouwbaar transport bieden.

6.6 REALISATIE. In Appendix D wordt de optimale blokgrootte voor het transmissiesysteem bepaald: uitgaande van een zendsnelheid van 19,2 kbaud: 10 blokken met 64 meetwaarden en 9 blokken voor retransmissie. In het voorgaande (figuur 6.5) zijn we uitgegaan van woorden van 8 bit lengte. Eerder hadden we gesteld dat een meetwaarde 12 bit bevat. De meest significante bits van een meet waarde vertegenwoordigen de "gemiddelde waarde" van het signaal, deze zullen minder snel varieren dan de minder significante. Het

is daarom zinvol deze 4 bits niet mee te sturen behalve wanneer de waarde ervan veranderd is. Ze worden dan in een extra datablok voorzien van naam en plaats in het originele datablok, verstuurd. Schematisch volgt nu een overzicht van de zendprocedure.



Figuur 6.5: Zendprocedure met coderingsmechanisme.

De codeomzettingen kunnen op verschillende manieren gerealiseerd worden: hardware (b.v. schuifregisters, FPLA's of een prom-vertaler) of software (algoritme, table look up). Daar veel omzet-

tingen per seconde moeten plaatsvinden, ligt de snelle hardware oplossing voor de hand. Het nadeel van deze methode is het verliezen van een standaard hardware configuratie.
De Hamming codering (4,7) kan snel gebeuren door gebruik te maken van een tabel (zender: 16x7, ontvanger: 128x4). Het opzoeken kan binnen 10 μseconde plaatsvinden. Voor de CRC-bepaling is een tabel niet bruikbaar. Er bestaan programma's die voor elke byte ongeveer 100 μseconde nodig hebben (Intel Library).

Elke byte van een te versturen blok wordt opgesplitst in twee nibbles en deze worden via de tabel vertaald naar 7-bit woorden. De twee transmissieregisters van de USART kunnen hiermee gevuld worden. Voorts worden bij elke byte de CRC-woorden van het te versturen blok aangepast.

In Appendix E is een overzicht gegeven van de programmatuur, gebaseerd op de hardware-configuratie van figuur 6.3, nodig voor het transmissiesysteem.

7.0 OPNEMERSYSTEEM

In het meetsysteem moeten op drie plaatsen twee of meer microcomputers met elkaar kunnen communiceren. In het vorige hoofdstuk is de strakke koppeling binnen het transmissiesysteem besproken. Er blijven nog twee koppelingen over, die tussen het transmissiesysteem en respectievelijk het opnemersysteem en het distributiesysteem. In dit hoofdstuk zal het opnemersysteem met koppeling aan de orde komen.

In het algemeen moeten afspraken gemaakt worden, wanneer twee of meer parallelle processen gebruik maken van één gemeenschappelijke faciliteit. Dit kan een I/O-device zijn, een geheugenveld of b.v. een datapad. De toewijzing aan een bepaald proces kan betrekkelijk eenvoudig gebeuren, wanneer het een, door de hardware opbouw, ondeelbare actie betreft.

Het sturen van een lettercode naar een printer is een ondeelbare actie; het sturen van een pagina text betreft een aantal ondeelbare acties die begripsmatig samenhangen. Wanneer meerdere processen gebruik kunnen maken van een device, moeten afspraken leiden tot het ondeelbaar maken van deze samenhangende acties: het "Mutual Exclusion" probleem (M.E.).

7.1 OPNEMERSYSTEEM<->TRANSMISSIESYSTEEM. We zullen ons het opnemersysteem voorstellen als bestaande uit een aantal dedicated processors (zie 7.2), samen met het zendsysteem vormen ze een groep parallelle processen, die o.a. meetwaarden kunnen uitwisselen. De meeste microcomputersystemen hebben een 8 bit databus; t.a.v. 16 bit meetwaarden (12 bit + 4 bit index, zie appendix A) treedt dus het M.E.-probleem op wanneer uitwisseling van deze waarden geen ondeelbare actie is. Wanneer we I/O-communicatie tussen de microcomputers toepassen is dit in hardware op te los-

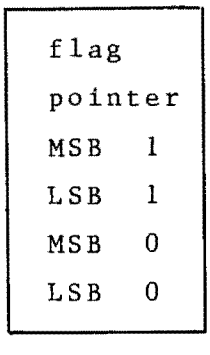
sen. Het bezwaar van deze methode is het aantal benodigde I/O-poorten.

Een andere mogelijkheid is te communiceren via een (8 bit) geheugen. We moeten dan programmatisch een protocol (specifiek voor een meetsysteem) uitvoeren. De volgende acties van respectievelijk zender en opnemer kunnen asynchroon plaatsvinden, zonder dat een meetwaarde verminkt wordt:

```
zender:    set flag            ; bezig
           load pointer        ; wijst naar vrij buffer
           load MSB            ; "gemiddelde"
           load LSB            ;
           clear flag          ; klaar

opnemer:   load pointer        ; wijst naar vrije buffer
           store MSB           ; in andere buffer
           store LSB           ;      "      "
           if flag: end        ; zender bezig: klaar
           else:change pointer; nieuw buffer
```

Er zijn geen onnodige wachttijden, de opnemer produceert meer dan de zender kan consumeren (zie 5.5). In het gemeenschappelijk geheugen bevinden zich nu voor elk meetkanaal twee databuffers:



Figuur 7.1: Buffer in gemeenschappelijk geheugen.

In Appendix A is het opnemer-gedeelte van dit protocol uitge-

werkt en vereenvoudigd. Voor het gemeenschappelijk geheugen kan nu de grootte en de byte-rate worden vastgesteld:

grootte = kanalen * buffergrootte/kanaal =
(32 + 16)* 6 = 288 bytes.

byte-rate= meetwaarden/sec * acties/meetwaarde * p =
600 * 5 * 25 = 75.000/sec
met p = 1/c, appendix A.

7.2 REALISATIE. Het multiprocessorsysteem van Intel, met een hardware toewijzingsstrategie (daisy chain) kan bovengenoemde byte-rate garanderen. Ook andere microcomputersystemen kunnen op deze wijze gekoppeld worden. We hebben nu een aantal dedicated microcomputers waarvan één het zendsysteem vormt.

7.3 OPNEMERSYSTEEM.

7.3.1 FOUTMELDINGEN. In appendix A is de 4 bit index van een meetwaarde aan de orde gekomen. Deze werd gegenereerd door een opnemer-testroutine en geplaatst in het privé geheugen van het opnemersysteem. Een andere mogelijkheid is om deze tests door een aparte microcomputer uit te laten voeren. De index wordt dan via het gemeenschappelijk geheugen aan de meetwaarde toegevoegd.

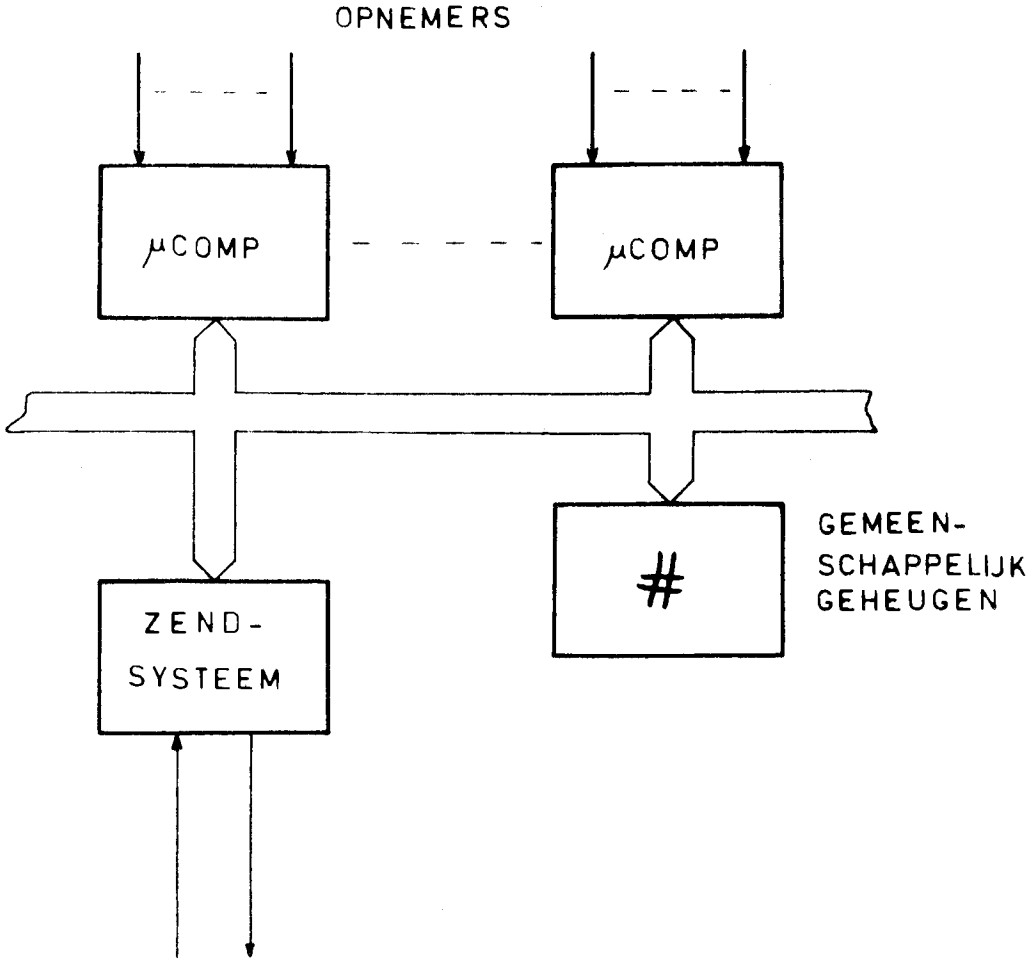
7.3.2 BEREKENINGEN. Berekeningen aan de uitgangssignalen van de opnemers, voor zover ze niet door de opnemersystemen zelf gedaan worden, kunnen door aparte microcomputers uitgevoerd worden. Als inputgrootheden voor deze berekeningen worden de momentane meetwaarden uit het gemeenschappelijk geheugen gebruikt. Het resultaat wordt weer in het gemeenschappelijk geheugen geplaatst als een van de 48 kanalen.

Het zendsysteem kent slechts 48 buffers, waaruit het 32 maal per seconde 16 waarden betrekt en 4 maal per seconde 32 waarden. De organisatie (programmatuur) van de opnemersystemen bepaalt

waarvoor de kanalen gebruikt worden.

7.3.3 OPSTARTGEGEVENS. In het tijdschema van zend en ontvangsysteem (figuur 6.5) is naast de data-blokken ruimte voor commando-blokken. Deze worden door systeem ,#1, opgemerkt en in het gemeenschappelijk geheugen geplaatst, op een locatie, gereserveerd voor het geadresserde opnemersysteem (appendix E). Via een I/O-actie (b.v. een interrupt) kan het zendsysteem ,#1, het opnemersysteem opmerkzaam maken op het informatie-blok. Omgekeerd kan ook een interrupt gebruikt worden als verzoek van het opnemersysteem om een blok te verzenden. De interruptroutine van #1 maakt dan de betreffende ack=0 (appendix E).

7.3.4 REALISATIE. Het aantal benodigde microcomputers in het opnemersysteem wordt in eerste instantie bepaald door het aantal aangesloten opnemers en voorts door de gewenste digitale filters. Wanneer, bij voorbeeld, één 6 dB/octaaf filter berekend moet worden voor elke aangesloten opnemer, kan een microcomputer 4000 meetwaarden per seconde produceren. Voor p=25 betekent dit 4000/600*25, ongeveer 26 % van de totale mogelijke capaciteit. Voor een vol bezette groep zijn dus minimaal 4 systemen nodig. Figuur 7.2 geeft een schematisch overzicht van het opnemersysteem.



Figuur 7.2: Schematische voorstelling opnemersysteem.

In het volgende hoofdstuk zal de koppeling van de ontvangst-systemen met de gebruikers worden beschreven.

8.0 DISTRIBUTIESYSTEEM

Gezien de laatste eis van 5.5 hebben we in het distributiesysteem, tussen transmissiesystemen en gebruikers, te maken met blokken meetwaarden, die een samenhangend geheel vormen. De transmissiesystemen hebben twee blokken meetwaarden aan te bieden: 1 blok van 32 kanalen, dat 4 keer/sec vernieuwd wordt, en 1 blok van 16 meetwaarden, dat 32 keer/sec verandert (Appendix E). De meetwaarden binnen een blok representeren bemonsteringen, die op het zelfde moment (Appendix B) plaats vonden, met een vertraging van één seconde. De blokken mogen niet opgesplitst worden om deze samenhang niet te verliezen.

8.1 GEMEENSCHAPPELIJK GEHEUGEN. Bij kleine systemen, wat betreft aantal groepen en gebruikers, kan een gemeenschappelijk geheugen als distributiesysteem gebruikt worden. De ontvangstsystemen gebruiken dit geheugen voor de beide buffers, A en B, zodat geen software blokverplaatsingen nodig zijn. De pointers, 0 en 1, wijzen nu de gebruikers waar de momentaan relevante informatie staat. Een geschikt protocol moet nu de uitwisseling regelen.

Deze oplossing vergt geen extra hardware, maar is slechts toepasbaar voor kleine systemen. Elke gebruiker moet in dat geval eenzelfde busstructuur als het ontvangsysteem hebben, m.a.w. ook een microcomputer zijn.

8.2 COMMUNICATIE NETWERK. Voor grote systemen, b.v. de meetopstelling in Cabauw, moet een andere methode worden gezocht. In dit geval blijft alle informatie in de geheugens van de microcomputers opgeslagen en vindt uitwisseling plaats via een gemeenschappelijk datapad.

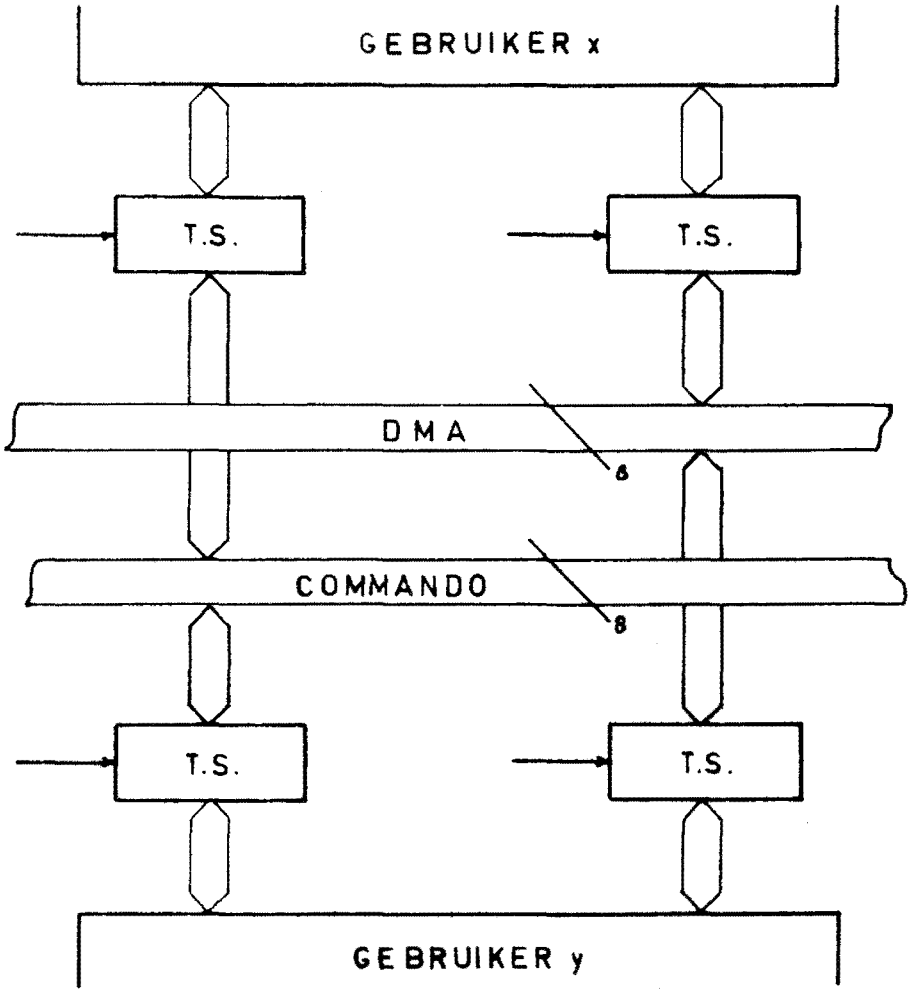
Het verkeer over dit datapad zal over het algemeen slechts in

één richting verlopen: van ontvangsysteem naar gebruiker. Buiten dit meetwaardentransport moet ook de mogelijkheid bestaan tot uitwisseling van andere gegevens, b.v. filtercorrecties, foutmeldingen of opstartgegevens en eventueel datablokken tussen gebruikers. We kunnen stellen dat elk apparaat, aangesloten op het distributiesysteem, met elke andere gebruiker moet kunnen communiceren. We moeten een verbindigsmogelijkheid zoeken, die in een "real time" omgeving aan deze eis voldoet.

In Appendix F wordt een matrixsysteem vergeleken met een 8 bits bussysteem, met als maat de verwachte doorlooptijd in het systeem. Het blijkt dat een bussysteem in dit concept de beste vooruitzichten biedt op een snel bloktransport.

8.3 BUSSTRUCTUUR. In principe gebeurt het uitwisselen van blokkken meetwaarden via een Direct Memory Access. Dit betekent dat de samenhang van de meetwaarden gegarandeerd is, aangezien de processor, in het ontvangsysteem, tijdens het DMA-transport (hold) geen pointers kan veranderen (Appendix E). Voor dit transport is een bus van 8 bit breedte nodig. De blokken zijn 16*2 en 32*2 woorden lang, voor de meetwaarden van 32 respectievelijk 4 Hz. Bij een snelheid van 1,25 μsec/woord (Intel 8257) betekent dit transporttijden van 40 en 80 μsec.

Het zal ook nodig zijn, dat tussen de verschillende systemen korte berichten, b.v. verzoeken om bepaalde gegevens, kunnen worden uitgewisseld. Om de wachttijden van deze commando's niet onnodig te verlengen is het nuttig een tweede databus te bestemmen als commandobus.



Figuur 8.1: Databussen.

Voor de toewijzing van deze bussen zijn een aantal methoden bekend. We zullen een aantal wensen formuleren, die deels specifiek zijn voor het meetsysteem en voortvloeien uit eerdere beschouwingen, en deels voortkomen uit technische overwegingen.

1* eenvoudig uitbreidbaar (tot een zeker maximum)

2* weinig extra hardware

3* snel (in verhouding tot transporttijden)

4* prioriteit te stellen aan meetwaarden en aan bepaalde gebruikers

5* niet kunnen blokkeren van laagste prioriteitsniveau

6* beperkt aantal verbindingsdraden

7* geen vitale delen (niet kwetsbaar)

8.4 BUSARBITER. We zullen de mogelijkheden onderzoeken om met een arbiterproces de toewijzing van de twee databussen, onder de gestelde condities, te regelen. De arbiter is een zelfstandig systeem en neemt een aparte plaats in het distributiesysteem in.

Om het aantal verbindingsdraden te beperken laten we de arbiter "colporteren", d.w.z. regelmatig alle gebruikers scannen om te zien of er een verzoek voor een van beide bussen is. Om een bus toe te kunnen wijzen moet aan vier voorwaarden zijn voldaan:

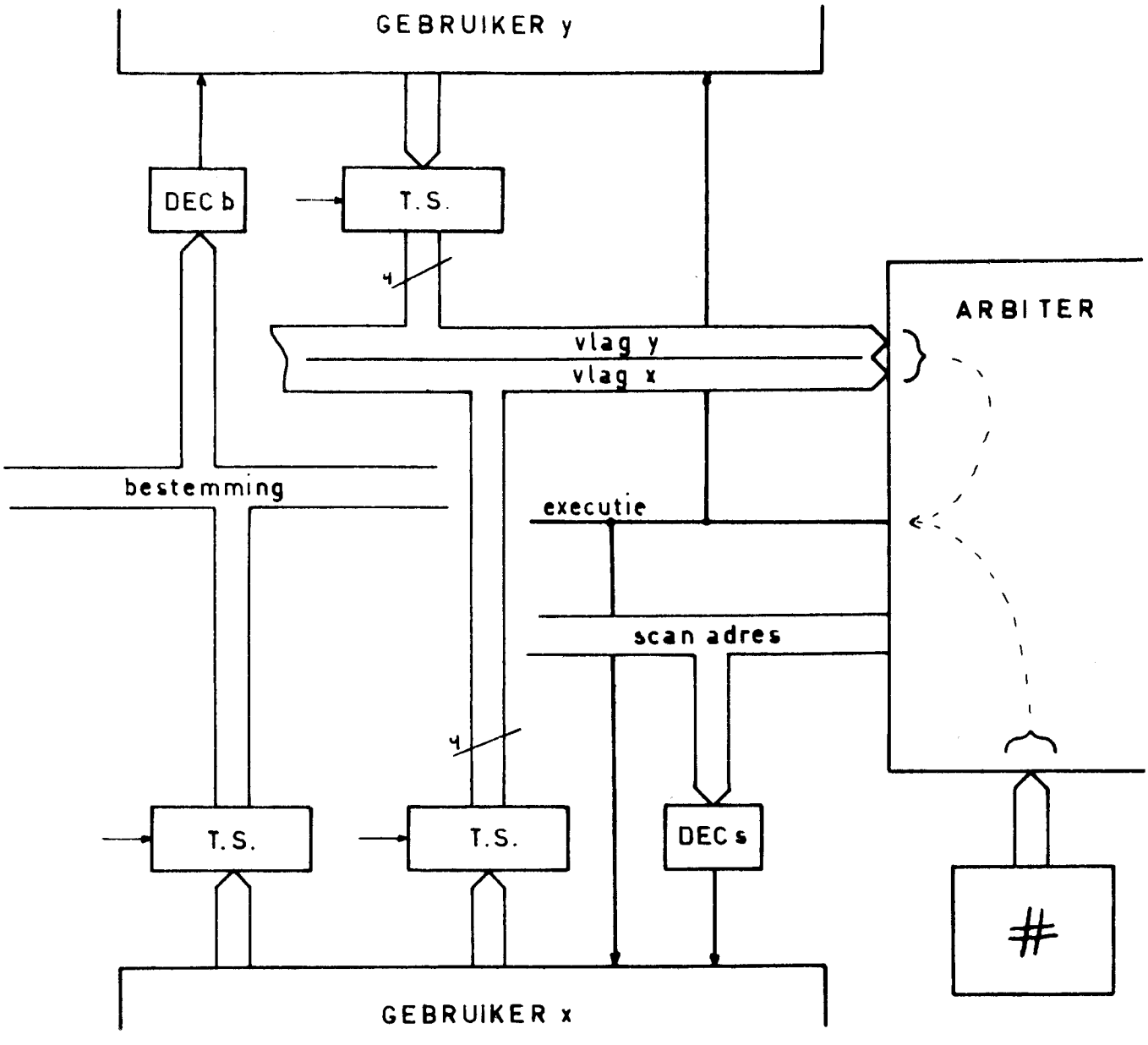
1. aanvrager aanwezig

2. bus onbezet

3. bestemming bereid

4. actie geoorloofd.

De arbiter moet in staat zijn deze condities te testen. Ter vergroting van de beslissingssnelheid wordt deze berichtgeving als volgt gerealiseerd:



Figuur 8.2: Besturing.

Door gebruik te maken van meerdere bussen, kan de arbiter na uitsturing van het scanadres direct de vlaggen van de aanvrager en die van de bestemming testen. De beslissing "wel of niet een executie-puls" is dan alleen nog afhankelijk van het arbiter-protocol. Het produceren van de vlaggen van de gebruiker vindt plaats, omdat de aanvrager een bestemmingsadres uitstuurt, dat door de bestemming gedecodeerd wordt.

Het voordeel van de extra bussen is tijdwinst: tijdens een DMA-transport kan reeds een nieuwe aanvrager worden opgezocht. Hierdoor wordt de verwerkingstijd van de aanvraag (de reactiesnelheid van de arbiter) niet altijd opgeteld bij de doorlooptijd van een transport door het systeem.

Dit alles wordt in hardware gerealiseerd: tri-statebuffers, adresdecoders en een decodering van de besturingssignalen, op elke interface tussen distributiesysteem en gebruiker.

De snelheid mogelijk door de extra hardware, in combinatie met een flexibele (programmeerbare) arbiter maakt dat aan de wensen geformuleerd in 8.3 kan worden voldaan, afgezien van 2* en 7*.

8.5 PRIORITEIT. De volgorde van de scanadressen bepaalt de prioriteitsstellingname in dit concept. Een eenvoudig telschema kan een rechtvaardige bustoewijzing realiseren:

1234a1234b1234c1234d1234Q1234a1234b1234c1234d1234R1234a1234b1234

Op deze manier worden de gebruikers in groepen ingedeeld, met verschillende prioriteit. Blokkeren van een lage prioriteit is uitgesloten. Wel moet een gebruiker met een lage prioriteit gemiddeld langer op een toewijzing wachten dan één uit een hogere groep.

De gebruiker bepaalt zelf zijn prioriteit door middel van de scanadresdecoder (eenvoudig te veranderen en uit te breiden).

Het telpatroon van de arbiter kan, bij het opstarten van dit systeem automatisch ingesteld worden, aangepast aan het aantal gebruikers en de groepen waarin ze zich bevinden (arbiter stuurt $\overline{\text{test}}$lijn=0 uit en alle geadresseerde gebruikers reageren met $\overline{\text{flag}}$=0). Op deze manier worden geen ongebruikte scanadressen opgevraagd. Dit impliceert een intelligente arbiter en leidt tevens tot de oplossing voor 7*:

8.6 KWETSBAARHEID. Wanneer de arbiter wordt uitgevoerd als een "standaard" microcomputer (met snel beslissingsprogramma) is het mogelijk de kwetsbaarheid van het systeem te verkleinen en daarmee de betrouwbaarheid te vergroten.

Het arbitersysteem bestuurt slechts de scanadres-bus en de besturingssignalen (execute, test en reset). Wanneer een fout van de arbitercomputer wordt gedetecteerd (watchdog) kan een transmissiesysteem de arbitrage overnemen en een foutmelding genereren. Dit overnemen gebeurt door de arbiter van de bussen af te koppelen (tri-state) en dan het arbiterprogramma uit te voeren. Nu valt, door een storing, niet het hele meetsysteem uit, maar slechts één groep. De foutmelding wijst ook direct de plaats van de storing aan.

8.7 INTERFACE. In Appendix F is een schema afgebeeld voor de hardware van de koppeling tussen een standaard microcomputersysteem en de distributie-bussen. Uitgangspunt is hier een bi-directionele parallel interface (8255) en een DMA-controller met vier request ingangen (8257). De interface produceert de vlaggen en onthoudt de status van de gebruiker. Voorts worden de besturingssignalen van de bus omgezet in signalen voor de microcomputer-bouwstenen. Een flowdiagram van een arbiterprogramma (op tabel look up-basis) voor een 8080 microprocessor is eveneens weergegeven. Met dit programma kan, met een cyclustijd van 0,5 µsec, deze snelheid gerealiseerd worden:

NOP: 25 µsec (geen executie-puls)

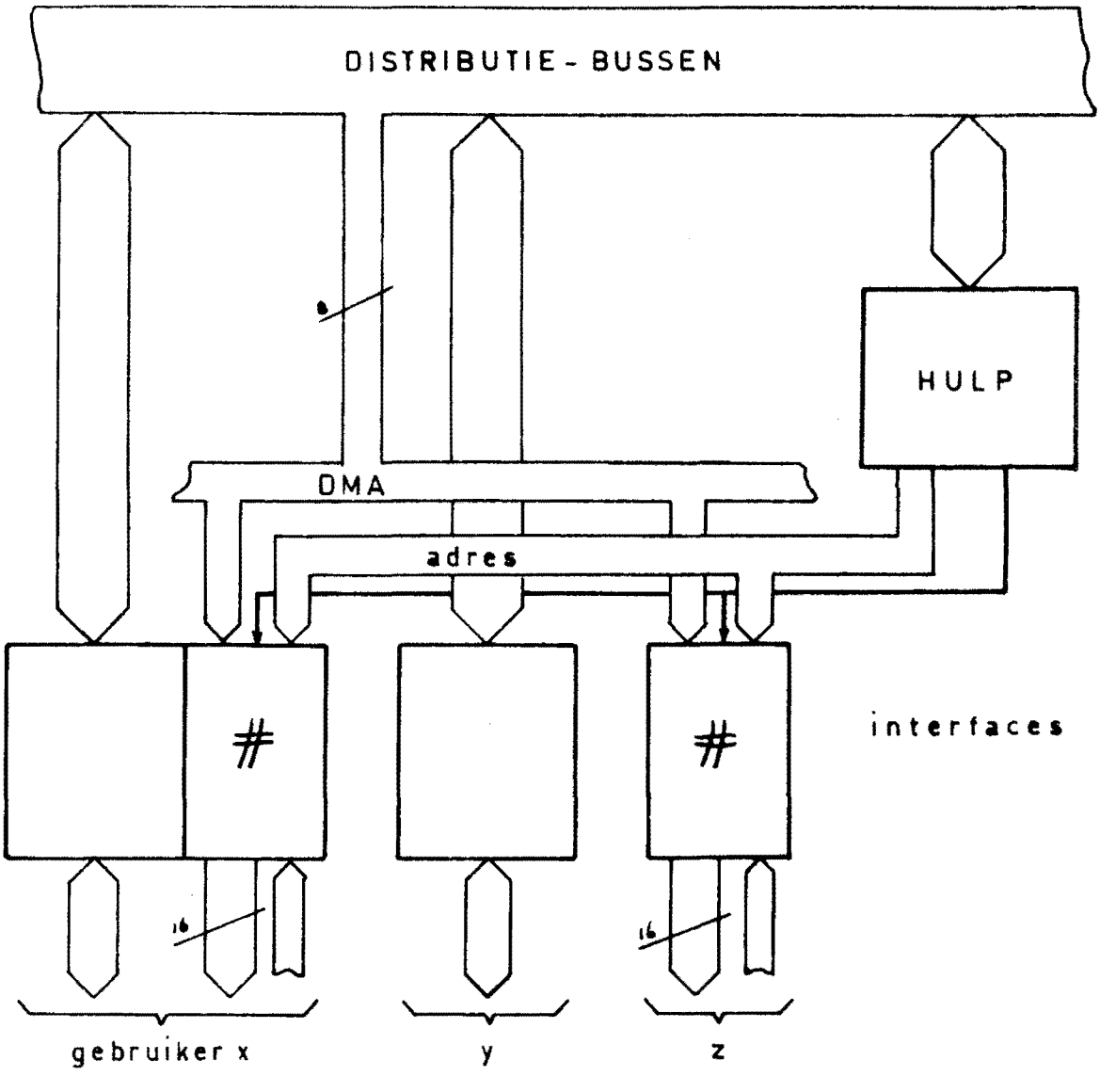
OP: 50 µsec (wel executie-puls).

In verhouding tot de transporttijden van 40 respectievelijk 80 µsec is dus met deze opzet een voldoende snelle arbiter te realiseren.

8.8 HULPSYSTEEM. In het vorige werden aan de gebruikers van het meetsysteem een aantal eisen opgelegd: ze worden geacht verzoe-

ken te doen om meetwaarden (vlaggen) en de meetwaarden te accepteren in blokvorm (schuifregister, DMA-controller). Dit veronderstelt een intelligent gebruikersysteem, met een volledige interface naar het distributiesysteem.

Door de opzet van vraag en antwoord worden blokken meetwaarden meerdere malen over de DMA-bus naar verschillende gebruikers gestuurd. Dit vormt een onnodige belasting van het distributiesysteem.
De volgende hulpschakeling komt aan beide bezwaren tegemoet:



Figuur 8.4: Hulpschakeling.

De schakeling fungeert als gebruiker van het distributiesysteem en vraagt regelmatig alle meetwaarden-blokken op. Via de DMA-bus en met adressen en stuursignalen van de hulpschakeling (normale DMA-acties) worden nu meerdere gebruiker RAM's tegelijk geladen. Een "parallelle" hardware handshake moet deze acties synchroniseren. De geheugens dienen, wanneer dit gewenst is, tevens als omzetting van 2*8 bit naar 1*16 bit woorden. Voor de geheugens kan een dynamisch type genomen worden (48*32x16 = 2kx16), aangezien ze regelmatig gevuld worden (in een systeem met 32 groepen worden 32*32=1024 blokken per seconde ontvangen, dat betekent dat elke miliseconde de accesstijd van het geheugen met 40 µsec wordt verlengd).

Voor de gebruiker betekent dit een prive geheugen met, direct adresseerbaar, alle opnemers die in het meetsysteem aanwezig zijn. Selecteren van een kanaal betekent nu adresseren van een geheugenlocatie.

8.9 CONCLUSIE. Door combinatie van een snel multi-bussysteem, ten koste van extra hardware, en een programmeerbare arbiter is een distributiesysteem verkregen, dat aan de eisen, gesteld in hoofdstuk 3 t.a.v. bedieningsgemak en meetnauwkeurigheid voldoet.

9.0 CONCLUSIES

De problemen, die optreden bij analoge meetmethoden, in het bijzonder wanneer grote afstanden moeten worden overbrugd, zijn door toepassing van digitale systemen op te lossen. Een extra voordeel van deze aanpak is een standaardisatie van hardware, gekoppeld aan flexibiliteit door het programmeerbare karakter van de bouwstenen.

In dit verslag is vooral de nadruk gelegd op de analoge eisen en de daaruit voortvloeiende consequenties voor het digitale systeem. Het blijkt mogelijk te zijn met toepassing van dedicated processors toch aan de meettechnische eisen te voldoen.

Bij realisatie van een systeem worden de kosten bepaald door de toegepaste microcomputers en de gewenste grootte van het meetsysteem.

LITERATUUROVERZICHT.

5.1 (1) J.S.Bendat,A.G.Piersol: Measurement and Analysis of Random Data.

5.2 (2) A.Papoulis: The Fourier Integral and its Applications, Mc Graw Hill, 1962.

5.3 (3) B.Soucek : Minicomputers in Dataprocessing and Simulation.

6.3 (4) W.Peterson: Errorcorrecting Codes, M.I.T. Press.

6.3 (5) H.v.d.Weg : Foutendetectie en diagnostiek in digitale circuits en systemen, THE, 1976.

A.0 (6) D.E.M.Taylor: art. Probabilistic trend detection, publ. Real Time Computing in Patient Management, Hill, 1976.

F.0 (7) W.Monhemius:Loketwachttijden, THE, 1.139.

APPENDIX A

A.0 LAAGDOORLAATFILTER m.b.v. MICROCOMPUTER. Bij het simuleren van een real-time laagdoorlaatfilter met een digitale rekenmachine moeten een groot aantal berekeningen, waaronder vermenigvuldigen en delen, in een beperkte tijd worden uitgevoerd. Het berekenen van een digitaal transversaal filter betekent b.v. voor iedere bemonstering met een aantal weegfactoren vermenigvuldigen. Het aantal weegfactoren is afhankelijk van de gewenste nauwkeurigheid van de filterkarakteristiek.

Het probleem dat zich voordoet wanneer microcomputers worden ingezet in een opnemersysteem, voor o.a. de berekening van filters is het ontbreken van macroinstructies voor vermenigvuldigen en delen. Zelfs optellen of aftrekken van getallen groter dan 16 bit is voor de meeste microprocessors geen standaard operatie. Door subroutines voor deze eenvoudige algebraische bewerkingen te programmeren zijn deze operaties toch uitvoerbaar, ze nemen echter veel rekentijd in beslag. We zullen nu de mogelijkheid onderzoeken of een 1e orde laagdoorlaatfilter te realiseren is zonder de noodzaak tot vermenigvuldigen.

Voor een 1e orde RC-filter geldt voor de impulsresponsie: $h(t)=(1/T1)*\exp(-t/T1)$, voor $t>0$ en $h(t)=0$ voor $t<0$, met $T1= RC$. Een laagdoorlaatfilter met $fo= 1/T1$, het -3 dB punt, en een afval van 6 dB/octaaf. Het uitgangssignaal $y(t)$, behorende bij het ingangssignaal $x(t)$, vinden we uit de convolutie van $x(t)$ en $h(t)$:

$$y(t)= x(t)\otimes h(t)= \int_{-\infty}^{t} x(\tau)*h(t-\tau)\, d\tau, \text{ daar } h(t)=0 \text{ voor } t<0.$$

Wanneer $x(t)$ een gedigitaliseerd signaal voorstelt, met bemonsterfrequentie $1/T$, kunnen we van de integraal overgaan op een sommatie met de volgende gelijkheid:

$$x(t)=\int_{-\infty}^{+\infty} x(\tau)*\delta(t-\tau)\ d\tau=\sum_{n=-\infty}^{+\infty} x(nT)*\delta(t-nT)T$$

$$y(t)=\int_{-\infty}^{t}\sum_{-\infty}^{1} x(nT)*\delta(\tau-nT)*T*h(t-\tau)\ d\tau=\sum_{n=-\infty}^{t/T} T*x(nT)*h(t-nT),\text{door}$$

verwisseling van de volgorde van sommeren en integreren. Nu is

$$y(kT)=\sum_{n=-\infty}^{k}(T/T1)*x(nT)*\exp(-kT-nT)/T1$$

$$y(kT+T)=\exp(-T/T1)\sum_{n=-\infty}^{k}(T/T1)*x(nT)\exp(-k-n)T/T1+(T/T1)*x(kT+T)$$

$$y(kT+T)=(\exp(-T/T1))*y(kT)+(T/T1)*x(kT+T)$$ (literatuur 6)

Het uitgangssignaal van het filter is de som van de vorige uitkomst, *exp(-T/T1), en de momentane ingangsspanning, *T/T1. De invloed van het verleden neemt exponentieel af (Exponentially mapped past).
De reeksontwikkeling van exp(-T/T1):

$$\exp(-T/T1)=1-T/T1+(1/2!)*(T/T1)**2-(1/3!)*\ldots.$$

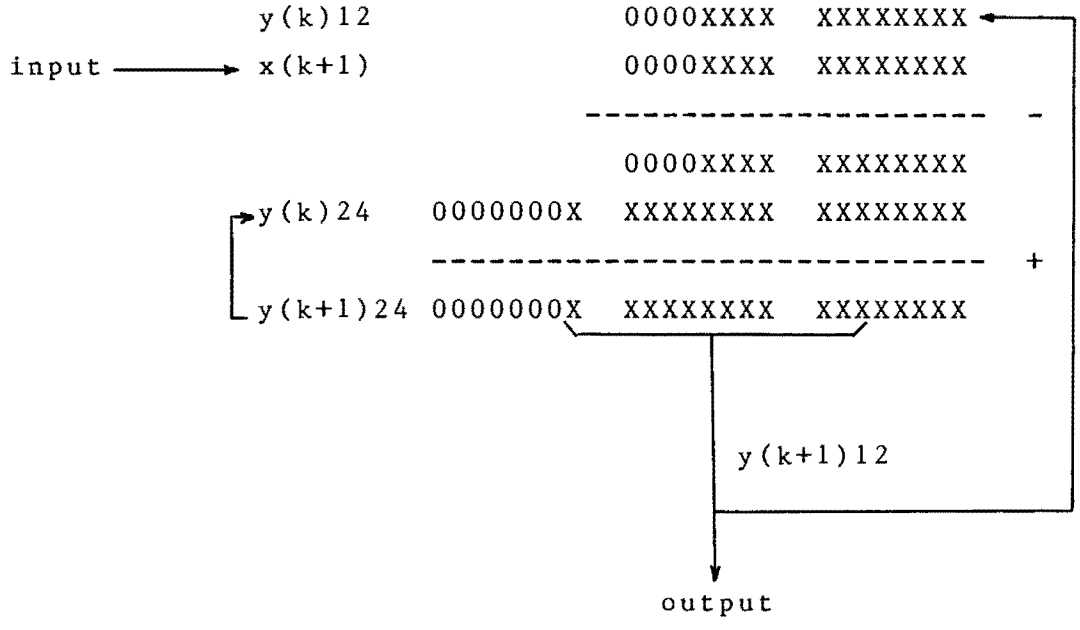
mogen we na de tweede term afbreken voor T/T1<<1. Herschrijving van het resultaat gevonden voor y(kT+T) levert nu de uitdrukking

A.0 $$y(k+1)=y(k)+c*[x(k+1)-y(k)],\ \text{met}\ c=T/T1<<1$$

We hebben nog te maken met één vermenigvuldiging, c*[. . .]. Vermenigvuldigen in het tweetallig stelsel met machten van het grondtal twee komt overeen met het "schuiven van de komma", het schuiven van registers. De beperking van c tot machten van twee heeft geen directe discretisering van mogelijke afsnijfrequenties tot gevolg. Daar c=T/T1 kan elke afsnijfrequentie ingesteld worden door een keuze van c, in de vorm 2**(-n), en een aanpassing van de sampletijd T, tussen To en 2*To.

A.1 PROGRAMMAVERLOOP. Voor de implementatie van het filter op een microcomputer kan een snel programma geschreven worden dat de 12 bit waarde van x(k+1) van de 12 bit waarde y(k) aftrekt, en dit verschil optelt bij de 24 bit representatie van y(k),

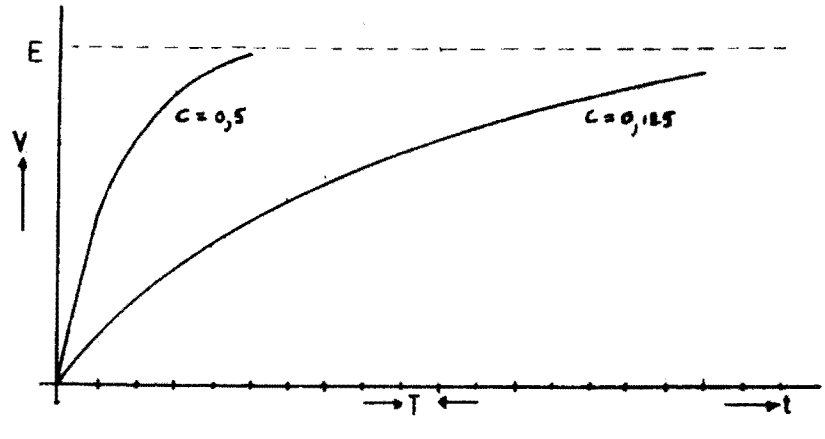
waardoor de vermenigvuldiging met c tot stand komt:



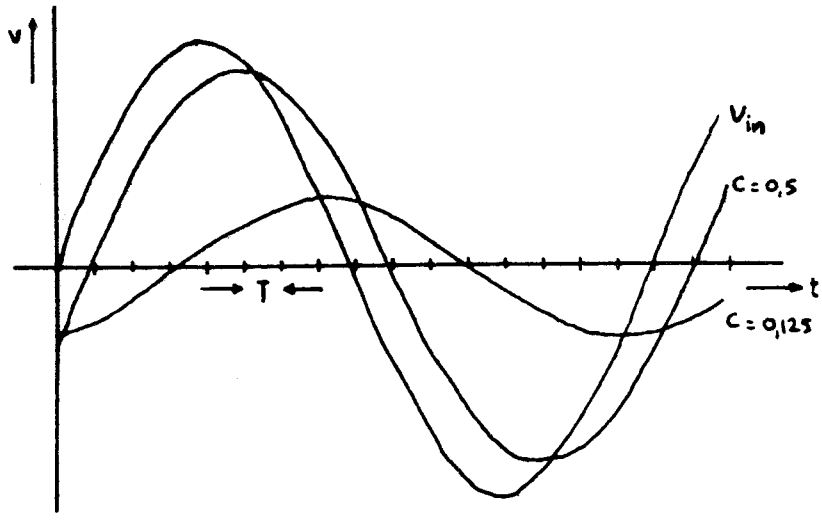
Figuur A.1: Programmaverloop, c=2**(-5).

De 12 bit representatie van y(k+1) ontstaat door de juiste 12 bit van het resultaat over te nemen. De n minst significante bits van y(k+1)24 moeten bewaard blijven, daar het filter anders nooit zijn eindwaarde kan bereiken.

Hier volgen nog twee simulaties van het filter uitgevoerd op een 8080-microcomputersysteem:



Figuur A.2: Stapresponsie, Vin= E*U(t).



Figuur A.3: Harmonisch ingangssignaal.

Opmerking: het in cascade schakelen van filters geschiedt door de y(k+1)12 van het eerste filter te gebruiken als ingangssignaal [x(k)12] van een tweede filter, etc. De benodigde rekentijd neemt, evenals de benodigde geheugenruimte, evenredig met het aantal filters in cascade toe. Voorts treedt, net als bij analoge filters, verschuiving van het -3dB punt op.

In het volgende is een subroutine weergegeven die gemiddeld 500 machine cycli (~250 μsec) nodig heeft om een 12 bit meetwaarde van een opnemer aan te nemen, een index van b.v. een testroutine toe te voegen, een filter te berekenen en een dataprotocol in een gemeenschappelijk geheugen (met het zendsysteem) te onderhouden.

A.2 CONCLUSIE. We mogen uit het voorgaande concluderen dat het mogelijk is met een microcomputer "real-time filters" te berekenen, onder de voorwaarde dat de bemonsteringsfrequentie hoger is dan de gewenste afsnijfrequentie. Onder dezelfde voorwaarde mocht een eenvoudig RC-filter tussen opnemer en digitaal systeem worden toegepast. Er geldt namelijk c=1/p (Appendix B) en p>>1.

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                    1 ;
                    2 ;
                    3 ;************************************************
                    4 ;
                    5 ; SUBROUTINE BEREKENT LAAGDOORLAATFILTER MET
                    6 ; C=2**(-5) EN COMMUNICEERT VIA HET PRIVEGEHEUGEN
                    7 ; MET EEN EVENTUELE TESTROUTINE EN VIA EEN GEMEEN-
                    8 ; SCHAPPELIJK GEHEUGEN MET HET ZENDSYSTEEM.
                    9 ;
                   10 ;
                   11 ; PRIVE GEHEUGENLOCATIES:
                   12 ;
                   13 ; YK12P: YK12P&YK12P+1 BEVATTEN VORIGE 12 BIT
                   14 ;        REKENWAARDE VAN FILTEROUTPUT.
                   15 ; YK24P: YK24P,+1&+2 BEVATTEN VORIGE 24 BIT
                   16 ;        REKENWAARDE VAN FILTER.
                   17 ; INDEX: WORDT GEVULD DOOR TESTROUTINE EN BEVAT
                   18 ;        EEN CODE IN DE 4 MSBS
                   19 ;
                   20 ; GEMEENSCHAPPELIJKE GEHEUGENLOCATIES:
                   21 ;
                   22 ; YK12G: YK12G&YK12G+1: 0E BUFFER
                   23 ;        YK12G+2&YK12G+3: 1E BUFFER
                   24 ;        IN EEN BUFFER STAAT DE VORIGE MEETWAARDE
                   25 ;        (INCLUSIEF INDEX) DIE VERSTUURD KON WOR-
                   26 ;        DEN
                   27 ; WYZER: BEVAT EEN INFORMATIEBIT VOOR HET TE GE-
                   28 ;        BRUIKEN BUFFER:
                   29 ;                        0: BUFFER 0
                   30 ;                        1: BUFFER 1
                   31 ; VLAG : BEVAT EEN INFORMATIEBIT DAT VERANDERD
                   32 ;        KAN WORDEN DOOR HET ZENDSYSTEEM:
                   33 ;                        0: NIET BEZIG
```

```
                        34 ;                          1: BEZIG IN BUFFER
                        35 ;
                        36 ; DOOR EEN GOED PROTOCOL (M.B.T. EEN MEETSYSTEEM)
                        37 ; KAN HET MUTUAL EXCLUSION PROBLEEM IN HET GEMEEN-
                        38 ; SCHAPPELIJKE GEHEUGEN TUSSEN OPNEMERSYSTEEM EN
                        39 ; ZENDSYSTEEM WORDEN OPGELOST.
                        40 ;
                        41 ;************************************************
                        42 ;
                        43 ; PROGRAMMA EQUIVALENTEN
                        44 ;
                        45 ;************************************************
                        46 ;
5000                    47 YK12P   EQU     5000H   ; PRIVE
5002                    48 YK24P   EQU     5002H   ; PRIVE
5005                    49 INDEX   EQU     5005H   ; PRIVE,MET TESTROUTINE
5010                    50 PRT1    EQU     5010H   ; LOW ORDER INPUTADRES
5011                    51 PRT1A   EQU     5011H   ; HIGH ORDER INPUTADRES
5020                    52 YK12G   EQU     5020H   ; GEMEENSCHAPPELIJK
5024                    53 WYZER   EQU     5024H   ;        "
5025                    54 VLAG    EQU     5025H   ;        "
                        55 ;
                        56 ;************************************************
                        57 ;
4000                    58         ORG     4000H
                        59 ;
                        60 ;************************************************
                        61 ;
                        62 FILTER:
4000 2A0050             63         LHLD    YK12P
4003 EB                 64         XCHG            ; DE:=OUDE REKENWAARDE
4004 210250             65         LXI     H,YK24P ; HL:=POINTER OUDE REKENW.
4007 3A1050             66         LDA     PRT1
400A 4F                 67         MOV     C,A
400B 3A1150             68         LDA     PRT1A
400E E60F               69         ANI     0FH     ; 12 BIT MEETWAARDE
4010 47                 70         MOV     B,A     ; BC:=X(k+1)12
                        71 ;
```

```
                        72 ; BEPAAL VERSCHIL OUDE REKENWAARDE MET NIEUWE
                        73 ; MEETWAARDE
                        74 ;
4011 79                 75          MOV     A,C
4012 93                 76          SUB     E
4013 4F                 77          MOV     C,A
4014 78                 78          MOV     A,B
4015 9A                 79          SBB     D
4016 47                 80          MOV     B,A         ; BC:=Xk+1 - Yk
4017 79                 81          MOV     A,C
4018 86                 82          ADD     M
4019 77                 83          MOV     M,A
401A 23                 84          INX     H
401B 78                 85          MOV     A,B
401C 8E                 86          ADC     M
401D 77                 87          MOV     M,A
401E 3E00               88          MVI     A,0
4020 23                 89          INX     H
4021 8E                 90          ADC     M
4022 77                 91          MOV     M,A         ; YK24:=YK24 + BC
4023 2B                 92          DCX     H
4024 2B                 93          DCX     H           ; HL POINTER NAAR YK24P
                        94 ;
                        95 ; NEEM JUISTE 12 BIT VOOR OUTPUT EN YK12P
                        96 ;
4025 7E                 97          MOV     A,M
4026 1F                 98          RAR
4027 1F                 99          RAR
4028 1F                100          RAR
4029 1F                101          RAR
402A 1F                102          RAR
402B E607              103          ANI     07H
402D 47                104          MOV     B,A
402E 23                105          INX     H
402F 7E                106          MOV     A,M
4030 17                107          RAL
4031 17                108          RAL
4032 17                109          RAL
```

```
4033 E6F8        110          ANI     0F8H
4035 B0          111          ORA     B
4036 5F          112          MOV     E,A
4037 7E          113          MOV     A,M
4038 1F          114          RAR
4039 1F          115          RAR
403A 1F          116          RAR
403B 1F          117          RAR
403C 1F          118          RAR
403D E607        119          ANI     07H
403F 47          120          MOV     B,A
4040 23          121          INX     H
4041 7E          122          MOV     A,M
4042 17          123          RAL
4043 17          124          RAL
4044 17          125          RAL
4045 E608        126          ANI     08H
4047 B0          127          ORA     B
4048 57          128          MOV     D,A     ; DE:=YK+1 12
4049 EB          129          XCHG
404A 220050      130          SHLD    YK12P   ; BEWAAR 12 BIT REKENWRDE
404D EB          131          XCHG
                 132 ;
                 133 ; VOEG AAN 12 BIT VOOR OUTPUT NOG 4 BIT VAN
                 134 ; EVENTUELE TESTROUTINE TOE
                 135 ;
404E 3A0550      136          LDA     INDEX
4051 B2          137          ORA     D       ; DE NU 16 BIT INFORMATIE
4052 57          138          MOV     D,A
                 139 ;
                 140 ; VERGELIJK MSBYTE IN GEMEENSCHAPPELIJK BUFFER
                 141 ; MET NIEUWE MSBYTE. ONGELIJK DAN BEIDE BYTES
                 142 ; NAAR BUFFER (MET PROTOCOL), GELIJK DAN ALLEEN
                 143 ; LSBYTE NAAR BUFFER, GEEN MUTUAL EXCLUSION PRO-
                 144 ; BLEEM.
                 145 ;
4053 3A2450      146          LDA     WYZER   ; WIJST NAAR ONVRIJ BUFFER
4056 4F          147          MOV     C,A     ; BEWAAR COPY
```

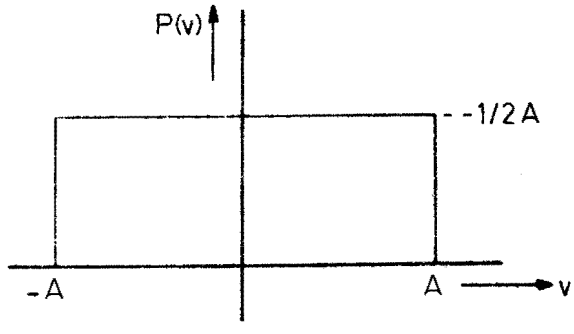
```
4057 87          148          ADD     A
4058 212150      149          LXI     H,YK12G+1
405B 85          150          ADD     L
405C 6F          151          MOV     L,A     ; HL POINT NAAR BEZET BUFF

405D 7E          152          MOV     A,M
405E BA          153          CMP     D       ; MSBYTES GELIJK?
405F C26540      154          JNZ     PROTOC  ; NEE: UITSLUITING
4062 2B          155          DCX     H
4063 73          156          MOV     M,E     ; LSBYTE NAAR ONVRIJ BUFF
4064 C9          157          RET
                 158 ;
                 159 ; PROTOCOL ZET MSB&LSB IN VRIJ BUFFER EN VERAN-
                 160 ; DERT DE WYZER INDIEN VLAG = 0.
                 161 ;
                 162 PROTOC:
4065 3E01        163          MVI     A,01H
4067 A9          164          XRA     C       ; VERANDERT WIJZER
4068 4F          165          MOV     C,A
4069 87          166          ADD     A
406A 212050      167          LXI     H,YK12G
406D 85          168          ADD     L
406E 6F          169          MOV     L,A     ; HL POINT VRIJ BUFFER
406F 73          170          MOV     M,E
4070 23          171          INX     H
4071 72          172          MOV     M,D     ; DE NAAR GEM.GEHEUGEN
4072 3A2550      173          LDA     VLAG
4075 B7          174          ORA     A       ; TEST WAARDE VLAG
4076 C0          175          RNZ             ; ZENDSYSTEEM BEZIG
4077 79          176          MOV     A,C     ; VERANDERDE WIJZER
4078 322450      177          STA     WYZER
407B C9          178          RET
4000             179          END     4000H
```

ASSEMBLY COMPLETE, NO ERRORS

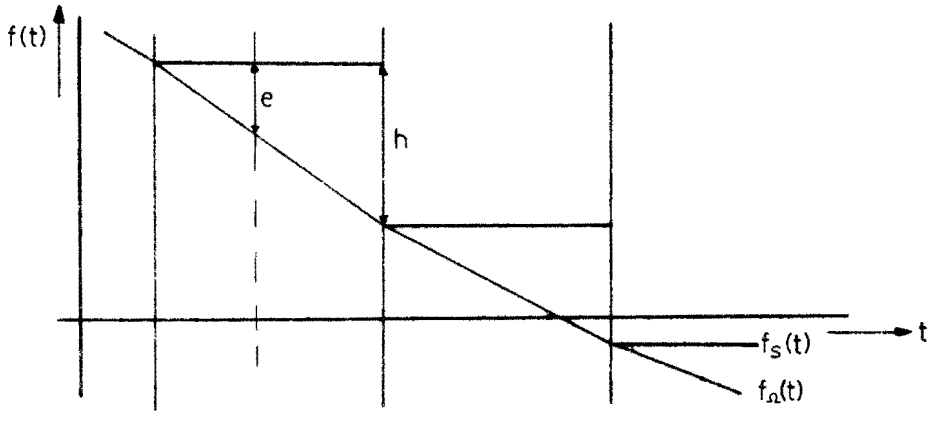
B.0 SIGNAAL/RUISVERHOUDING

We beschouwen een analoog signaal $f_\Omega(t)$, dat in de frequentie begrensd is tot Ω en in de amplitude van -A tot +A volt. We nemen een uniforme kansdichtheidsfunctie P(v) voor de spanning van het signaal aan.



Figuur B.1: Kansdichtheidsfunctie van de spanning.

Dit signaal wordt bemonsterd met frequentie $\omega_o = 2\Omega$, juist voldoende volgens het bemonsteringstheorema. Nu geldt dat de waarden van de monsters, op twee opeenvolgende sampletijdstippen, onafhankelijk zijn: ze hebben dezelfde kansdichtheidsfunctie P(v).

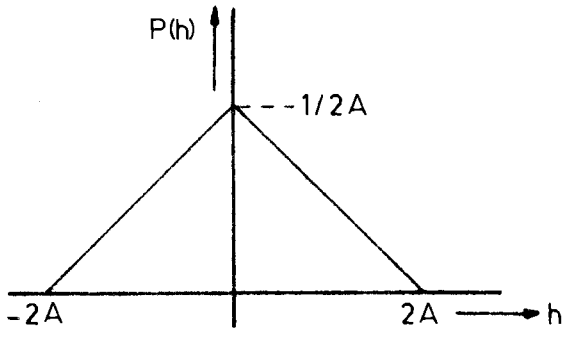


Figuur B.2: Bemonstering.

De kans op een verschil h tussen twee opeenvolgende monsters is de kans op spanning x op tijdstip t, maal de kans op spanning x-h,op tijdstip t+T, voor elke toegestane waarde van x. In formule:

B.0.1 $P(h) = \int_{-\infty}^{+\infty} P_t(x)\, P_{t+T}(x-h)\, dx$, de convolutieintegraal.

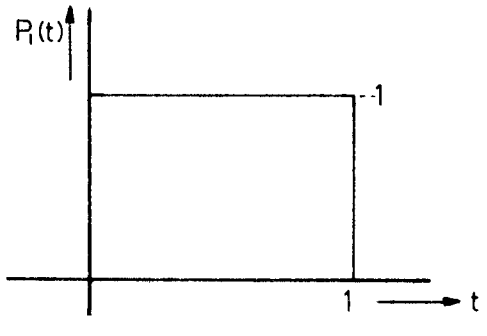
We vinden, daar $P_t(x) = P_{t+T}(x) = P(x)$, de volgende kansdichtheidsfunctie voor het verschil h:



Figuur B.3: Kansdichtheidsfunctie van het verschil h.

Wanneer we nu van het bemonsterde signaal $f_s(t)$, op willekeurig tijdstip, een waarde overnemen (opnieuw bemonsteren), hebben we niet de momentane waarde van $f_a(t)$, maar de laatste samplewaarde $f_s(t)$. Het verschil e hiertussen is afhankelijk van het moment van bemonsteren en de momentane helling van het analoge signaal $f_a(t)$. We namen in het vorige aan dat $f_a(t)$ tussen de bemonsteringsmomenten van $f_s(t)$ door een rechte lijn benaderd mag worden.

Doordat willekeurig "geprikt" wordt in $f_s(t)$, hebben we een uniforme verdeling voor de kans op een bepaald tijdstip tussen twee monsters:

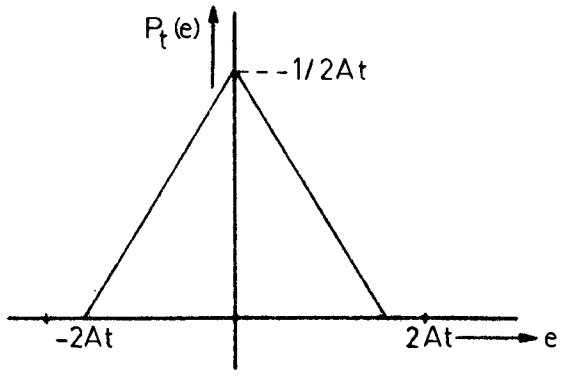


Figuur B.4: Kansdichtheidsfunctie van t.

De fout e volgt uit de formule:

B.0.2 $e = h*t$

n.l. de momentane helling maal de tijd na de laatste verandering in $f_s(t)$. We kunnen nu de kansdichtheidsfunctie $P_t(e)$ als functie van het tijdstip t vaststellen:



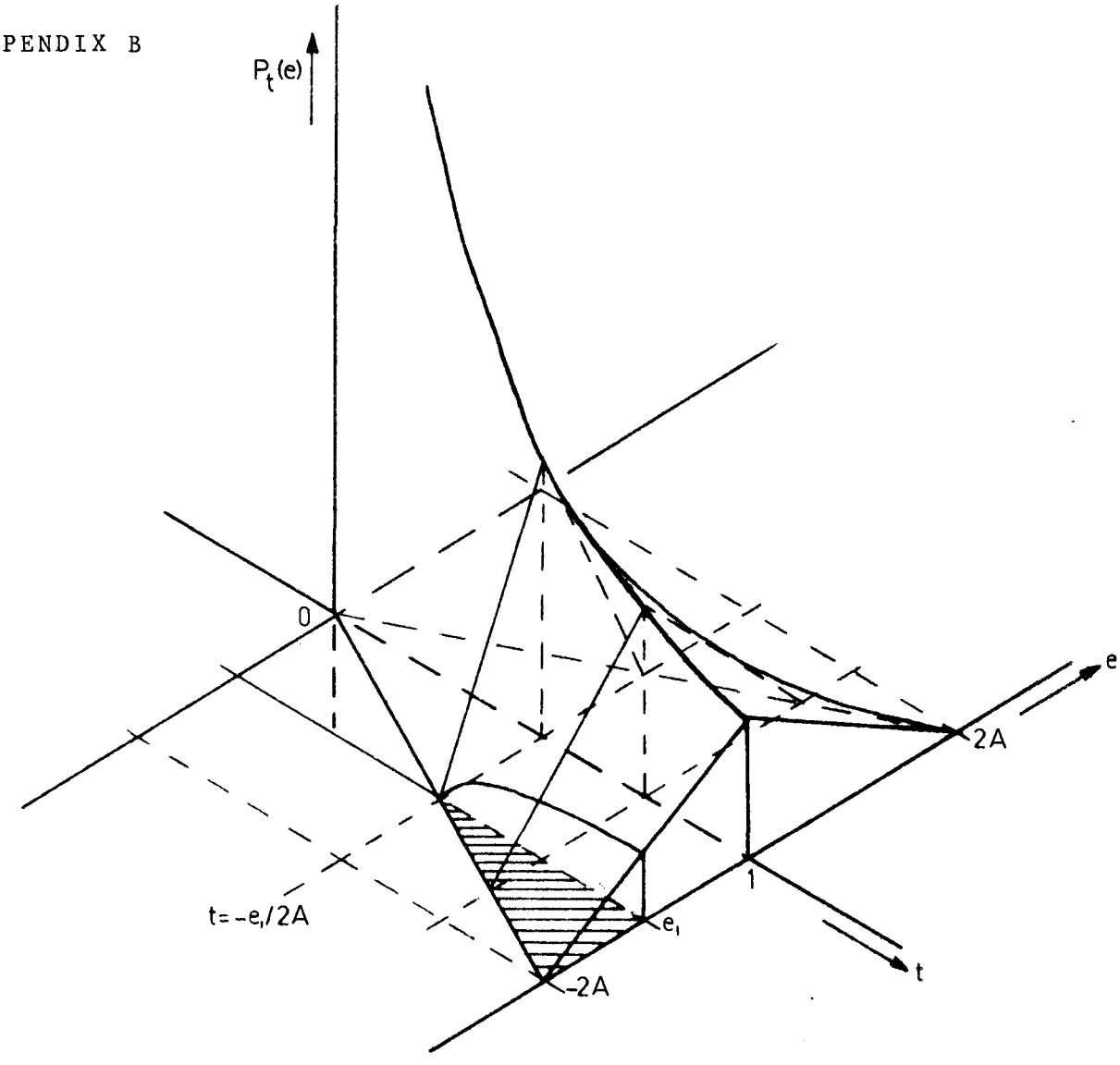
Figuur B.5: $P_t(e)$ afhankelijk van t.

De kans op een fout e1 bij willekeurige bemonstering van $f_s(t)$ is nu de kans op een waarde t1 maal de kans dat dan e1 kan optreden, voor alle e1 en alle t1:

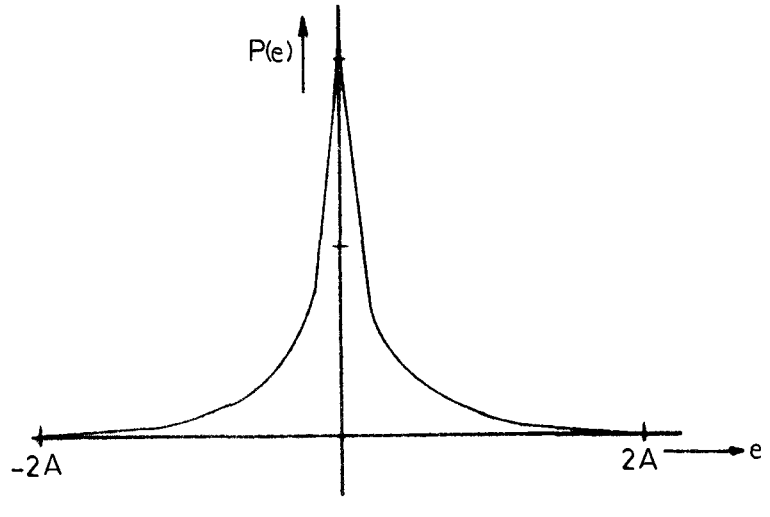
De kansdichtheidsfunctie voor e, P(e), wordt nu bepaald door:

$$\text{B.0.3} \quad P(e) = d/de \int_{-e/2A}^{1} \int_{-2At}^{e} P_1(t) * P_t(e)\, de\, dt =$$

$$= \begin{cases} -(1 + e/2A + \ln(-e/2A))/2A, & \text{voor } e<0 \\ -(1 - e/2A + \ln(+e/2A))/2A, & \text{voor } e>0. \end{cases}$$



Figuur B.6: "voor alle e1 en alle t1".



Figuur B.7: Kansdichtheidsfunctie van foutspanning e.

Voor het ruisvermogen vinden we:

B.0.4 $Pr = \sigma^2(e) = \int_{-\infty}^{+\infty} (p(e)* (e-E(e))**2)\ de =$

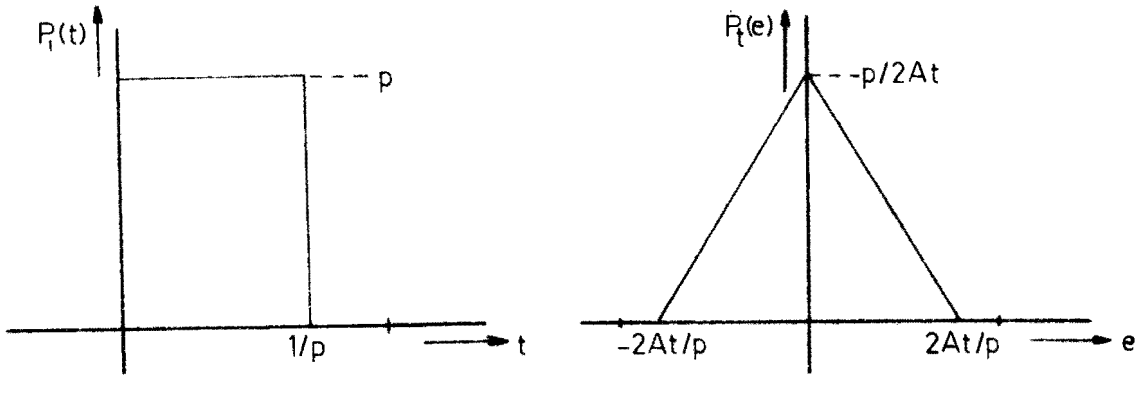
$= 2 \int_{-\infty}^{0} [-(1+e/2A+1n(-e/2A))/2A]*e**2\ de =$

$= (A**2)*2/9$

Voor het vermogen van het signaal met een verdeling als in figuur B.1, geldt (1/3)*A**2. De signaal/ruisverhouding bedraagt:

B.0.5 $s/r = 10*\log\ (1,5*(A/A)**2) = +\ 1,6$ dB.

B.1 SNELLER BEMONSTEREN. We zullen nu eerst het effect berekenen wanneer ω_0 groter gemaakt wordt. Stel $\omega_0 = \omega_x \gg \Omega_x$ en $\omega_x/\Omega_x = p$. Voor de verdeling van het tijdstip van bemonstering van $f_s(t)$ en voor $P_t(e)$ volgt nu:

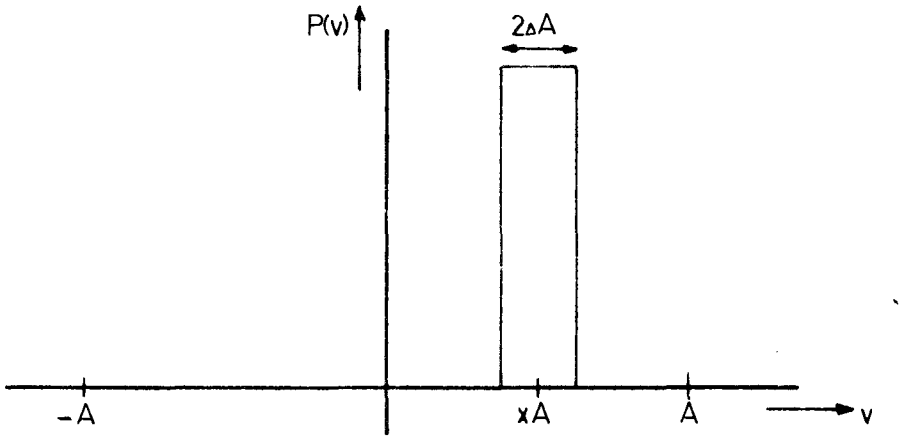


Figuur B.8: Nieuwe $P_1(t)$ en $P_t(e)$.

In vergelijking B.0.3 wordt nu elke A vervangen door A/p, resultaat:

B.1.1 $s/r = 10*\log\ (1,5*p**2)$

Op dezelfde wijze kunnen we het effect van een meer fysische aanname met betrekking tot de verdeling van de spanning van een meetsignaal berekenen:



Figuur B.9: Nieuwe P(v): fluctuatie rond een gemiddelde.

Deze verdeling verkrijgt men i.p.v. figuur B.1 wanneer het signaal over kortere tijd beschouwd wordt (in figuur B.1 was het beschouwingsinterval oneindig groot). De tijdfunctie kan nu worden voorgesteld als $f_{\Omega}(t) = x*A + f_{\Omega}'(t)*\Delta A$, met $q = A/\Delta A$.

We kunnen samenvattend stellen dat voor de, door asynchroon bemonsteren van een bemonsterd signaal, geïntroduceerde signaal/ruisverhouding geldt:

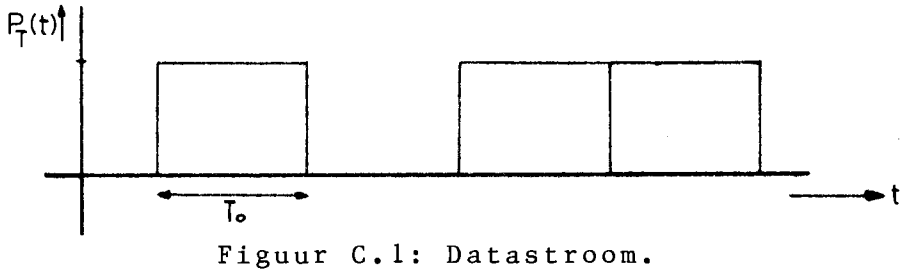
B.1.2 | s/r = 10*log (1,5*(p*q)**2) (vergelijk 5.1.2)

met voor p = 25 en q = 40 : s/r = 61.5 dB.

APPENDIX C

C.0 KLOKREGENERATIE. Voor synchroon datatransport is het noodzakelijk, dat aan ontvangkant aan de USART een klok wordt aangeboden, welke in fase is met de datastroom. Bij het zoeken naar een goedkope modem, in verband met het aantal benodigde exemplaren, die de gewenste baudrate kan verwerken (19.2 kbaud),bleek de meest geschikte niet over deze faciliteit te beschikken. De modem, de Tillegraaf van de firma Enraf Nonius, werkt volgens de asynchrone methode; deze is voor het signaal "doorzichtig". Elk signaal mag worden aangeboden met een maximale bitfrequentie van 19.2 kbaud.

Wanneer we toch synchroon datatransport tussen de USARTs willen bedrijven, moet de zendklok (fo) uit het signaal van de modem afgeleid kunnen worden. Hiertoe onderzoeken we het frequentiespectrum van een datastroom met bitfrequntie fo = 1/To:

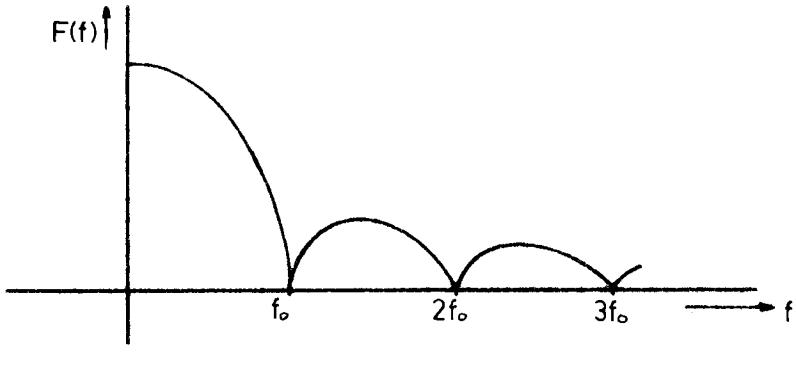


Figuur C.1: Datastroom.

Fourierreeksontwikkeling voor een puls:

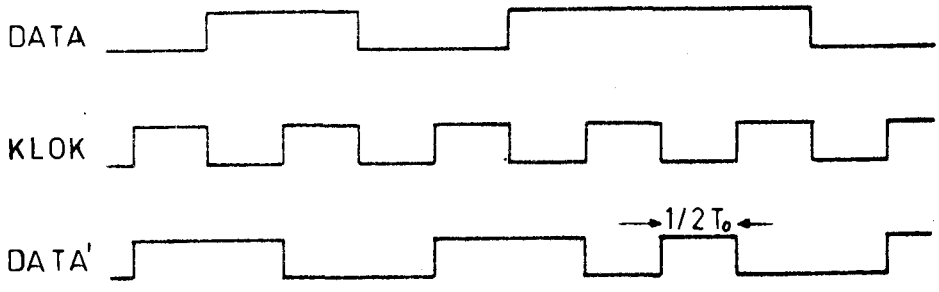
$$C.0.1 \quad P_{T_o}(t) = \sum_{n=0}^{\infty} (2/\pi n)*\sin(\pi n/16)*\sin(2\pi nt/16T_o) + 1/16$$

hieruit volgt het frequentiespectrum voor een reeks pulsen:



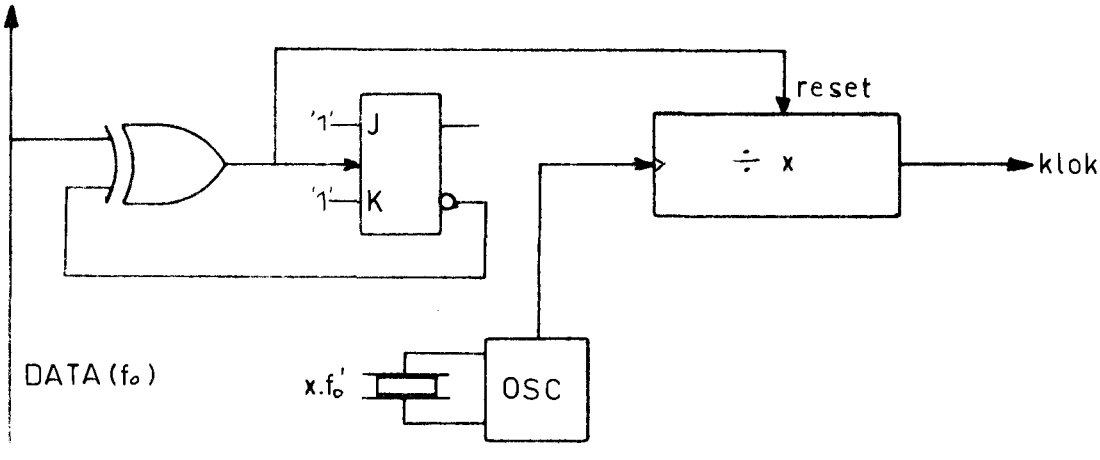
Figuur C.2: Spectrum van de bitstroom.

In het spectrum ontbreekt de grondfrequentie fo (de gewenste zendklok). Om toch een klokregeneratie met behulp van een filter (b.v. een PLL) mogelijk te maken, kunnen we de datastroom moduleren met de zendklok:



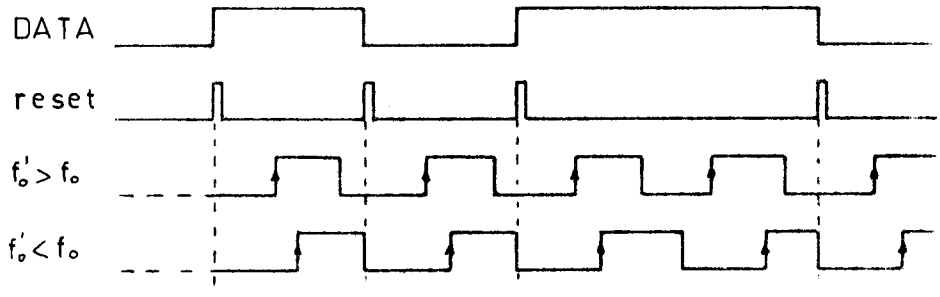
Figuur C.3: Modulatie.

Het nadeel is hier dat slechts met de halve seinsnelheid gewerkt kan worden, wegens de beperking van de modems.
De benodigde klok kan binnen zekere grenzen uit de datastroom gedestilleerd worden met behulp van de volgende schakeling:



Figuur C.4: Klokregeneratie.

Elke overgang in de datastroom reset de x-deler en synchroniseert zo de ontvangklok:



Figuur C.5: Synchronisatie van de klok f'o.

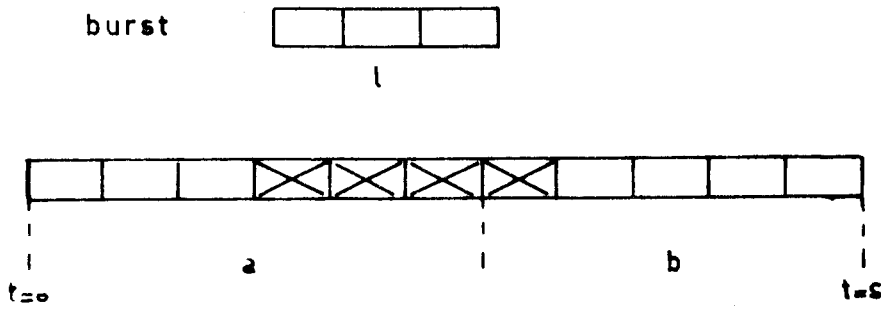
De deler (b.v.1/16) is nodig om de goede fase tussen f'_o en de databits te verkrijgen. Bij een $\Delta f = (f'o - fo)/fo$ van 5% zou minstens één dataovergang per 20T nodig zijn om nog tot goede beslissingen te komen. Door geschikte codering te kiezen is hieraan eenvoudig te voldoen, voorts is een veel kleinere Δf te verkrijgen door aan zend en ontvangkant kristallen te gebruiken.

APPENDIX D

D.0 OPTIMALISEREN BLOKGROOTTE. Wanneer we besluiten bloktransport met retransmissie toe te passen, moeten we een keuze maken ten aanzien van de blokgrootte. In de meeste on-line verbindingen wordt de grootte zodanig gekozen dat de verwachting voor de tijdsduur tot een blok goed is ontvangen minimaal is op een door ruis en overspraak gestoord kanaal. We gaan ervan uit dat door forward correction het effect van deze ruis geen rol meer speelt voor het retransmissiemechanisme, de bitfouten zijn reeds gecorrigeerd. We kunnen ons nu beperken tot het kiezen van die blokgrootte, die, bij het optreden van bursts (statische ontladingen schakelmanipulaties), de foutenkans per tijdseenheid het kleinst maakt.

Het uitgangspunt bij het bepalen van deze blokgrootte is een constante hoeveelheid te transporteren informatie per tijdseenheid en een vaste hoeveelheid overhead per blok. Deze overhead is noodzakelijk om de blokken van elkaar te kunnen onderscheiden en synchronisatie van de USARTs te bewerkstelligen. We zullen een tijdsinterval van s seconde beschouwen, waarin s*m meetwaarden moeten worden verstuurd. Hiervoor gebruiken we a blokken met een overhead van v woorden per blok. Een blok bevat nu s*m/a + v woorden. Stel voorts dat er s*w woorden per tijdsinterval verstuurd kunnen worden. Het aantal blokken b dat overblijft voor retransmissie is nu bekend:

$$b = (s*w-s*m-a*v)/(s*m/a+v).$$



Figuur D.1: Blokken in tijdsinterval s.

Laat het detectiemechanisme perfect zijn ,d.w.z. elke fout in een blok wordt opgemerkt. Een burst die zich uitstrekt over een duur van 1 blokken vernietigt nu 1+1 datablokken. Het retransmissiemechanisme faalt wanneer minder dan a blokkken (hoe dan ook verdeeld over s) correct over komen. Het optimaliseren van de blokgrootte geschiedt nu door een maximaal toelaatbare burstduur 1 te vinden, lmax, met als variabele a. lmax Is het maximum aantal toegestane foute blokken maal de blokduur (uitgedrukt in woorden, aangezien een woord een vaste hoeveelheid tijd representeert).

$$lmax=(b-l)*(s*m/a+v)= s*w - s*m -a*v -s*m/a - v$$

$$d/da\ lmax= - v + s*m/a**2$$

Het maximum van 1 treedt op bij $d/da\ l=0$. Hieruit volgt $a=\sqrt{s*m/v}$, en lmax is dan:

$$lmax= s*w - s*m - v - 2\sqrt{s*m*v}.$$

De burstlengte, die geen aanleiding geeft tot fouten, is, na optimalisering van de bloklengte, voornamelijk afhankelijk van de grootte van het beschouwde tijdsinterval s. De mogelijke grootte van het tijdsinterval wordt bepaald door de geheugencapaciteit van het zendsysteem. De te verzenden informatie moet worden bewaard tot deze, binnen het interval, door het ontvangstsysteem goed ontvangen is. Dit kan het gehele tijdsinterval duren.

Wanneer we uitgaan van de maximale seinsnelheid van de modem (zie appendix C) van 19,2 kbaud en woorden van 14 bit breedte (i.v.m. forward correction, zie 6.5) met een overhead van 6 woorden per blok in een interval van 1 seconde, krijgen we:

aantal datablokken (a) = 10

retrblokken (b) = 9

woorden (w) =1370

maximale burst =0,38 sec

meetwaarden (m) = 600 /sec.

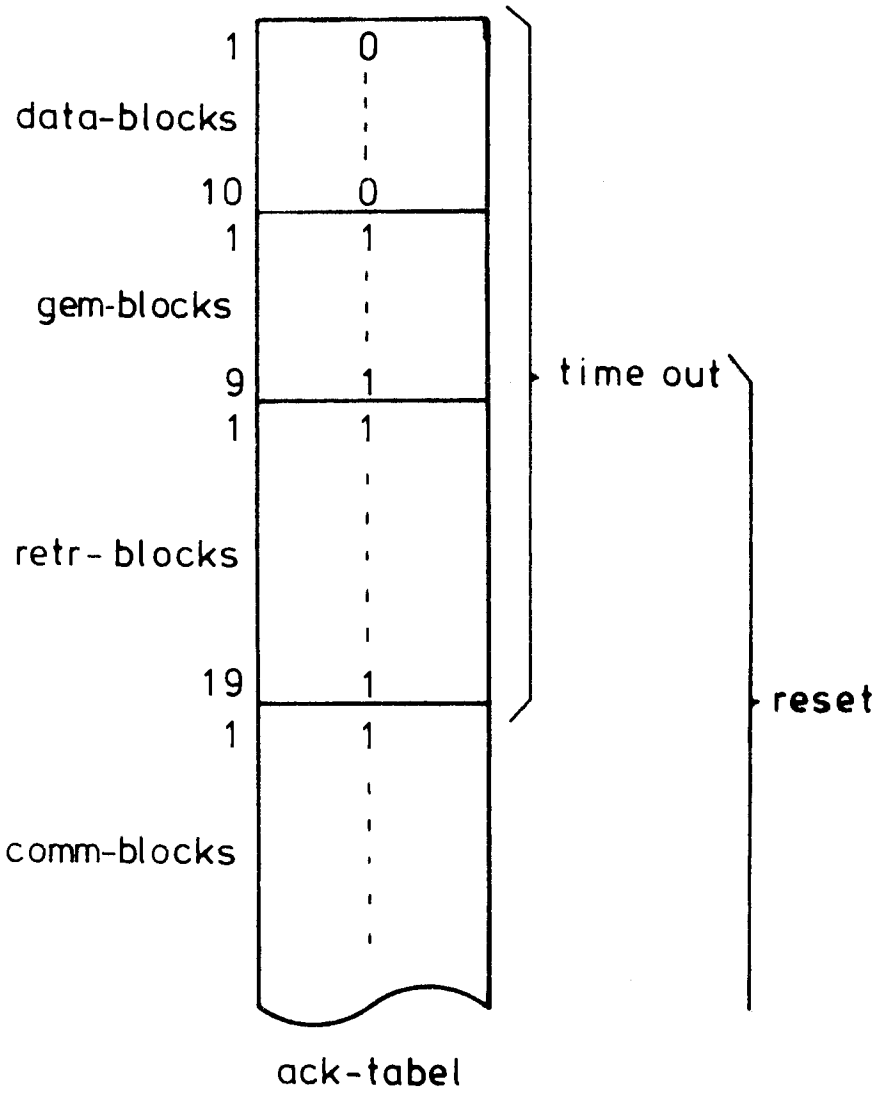
E.0 TRANSMISSIEPROTOCOL

In het volgende zal de programmatuur beschreven worden die nodig is voor het correct overbrengen van de datablokken en het synchroniseren van zender en ontvanger, opdat aan de eis $\Delta T=c$ wordt voldaan.

E.1 ZENDSYSTEEM,#1. Het zendsysteem heeft 4 belangrijke interrupts (figuur 6.3) als startpunt van bepaalde routines. We zullen nu een overzicht maken van de activiteiten die op de verschillende interrupts moeten volgen:

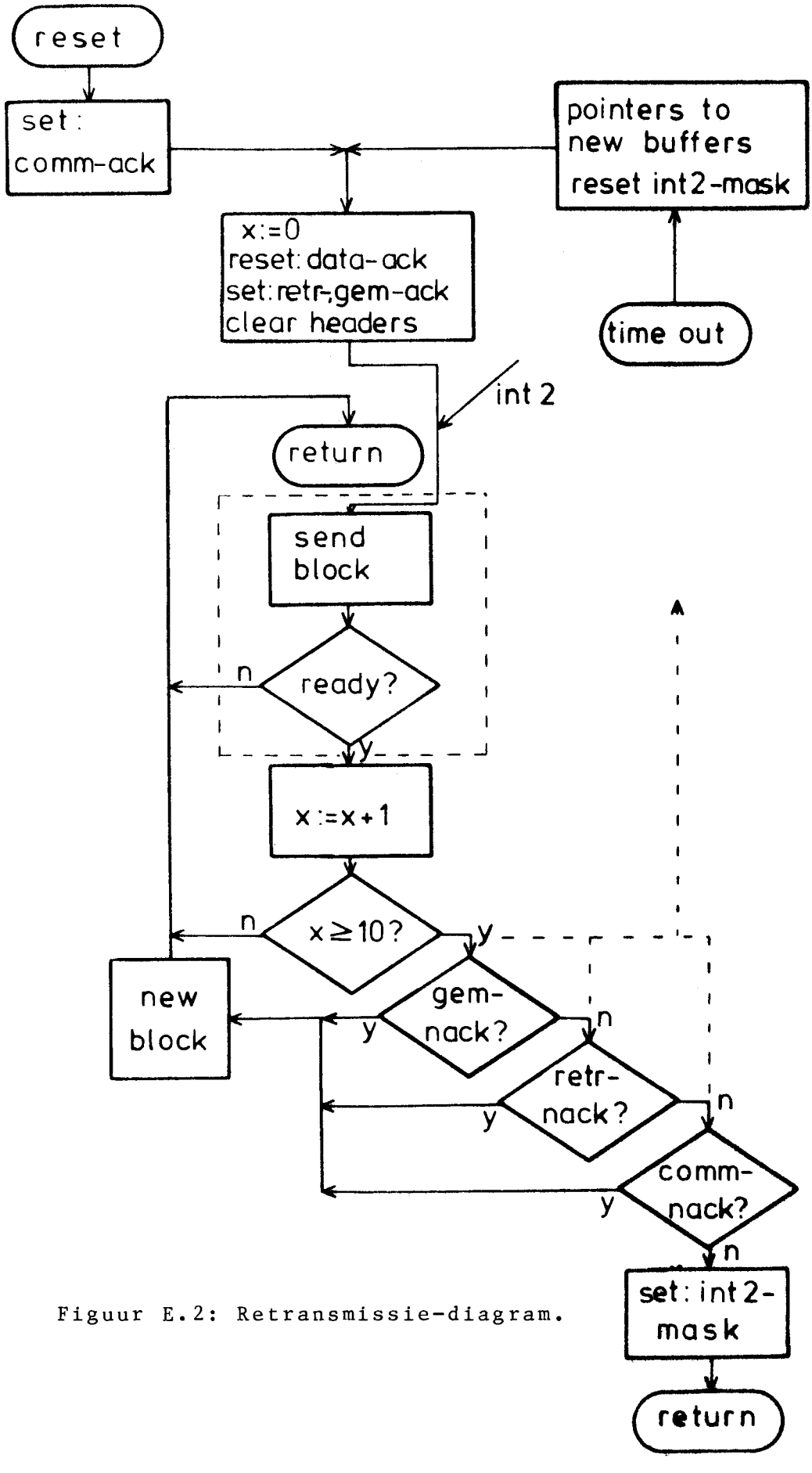
INT0: x1, 4Hz
INT1: x2, 32Hz
INT2: transmitter empty
INT3: receiver ready.

E.1.1 TRANSMITTER. Het zendmechanisme, INT2, maakt gebruik van een buffer, een tabel en een pointer. Het buffer, buffer B (figuur E.8), bevat alle in een seconde te versturen data- en gemiddelde-blokken. Onder data-blokken verstaan we die blokken van 64 bytes, die de 8 minst significante bits van een meetwaarde bevatten. In een eventueel gemiddelde-blok worden de 8 meest significante bits verstuurd wanneer deze veranderd zijn. In de acknowledge-tabel staan alle goedmeldingen gestuurd door het ontvangstsysteem, #2. Door goedmeldingen i.p.v. foutmeldingen te laten versturen, worden heen en retourkanaal op fouten gecontroleerd.

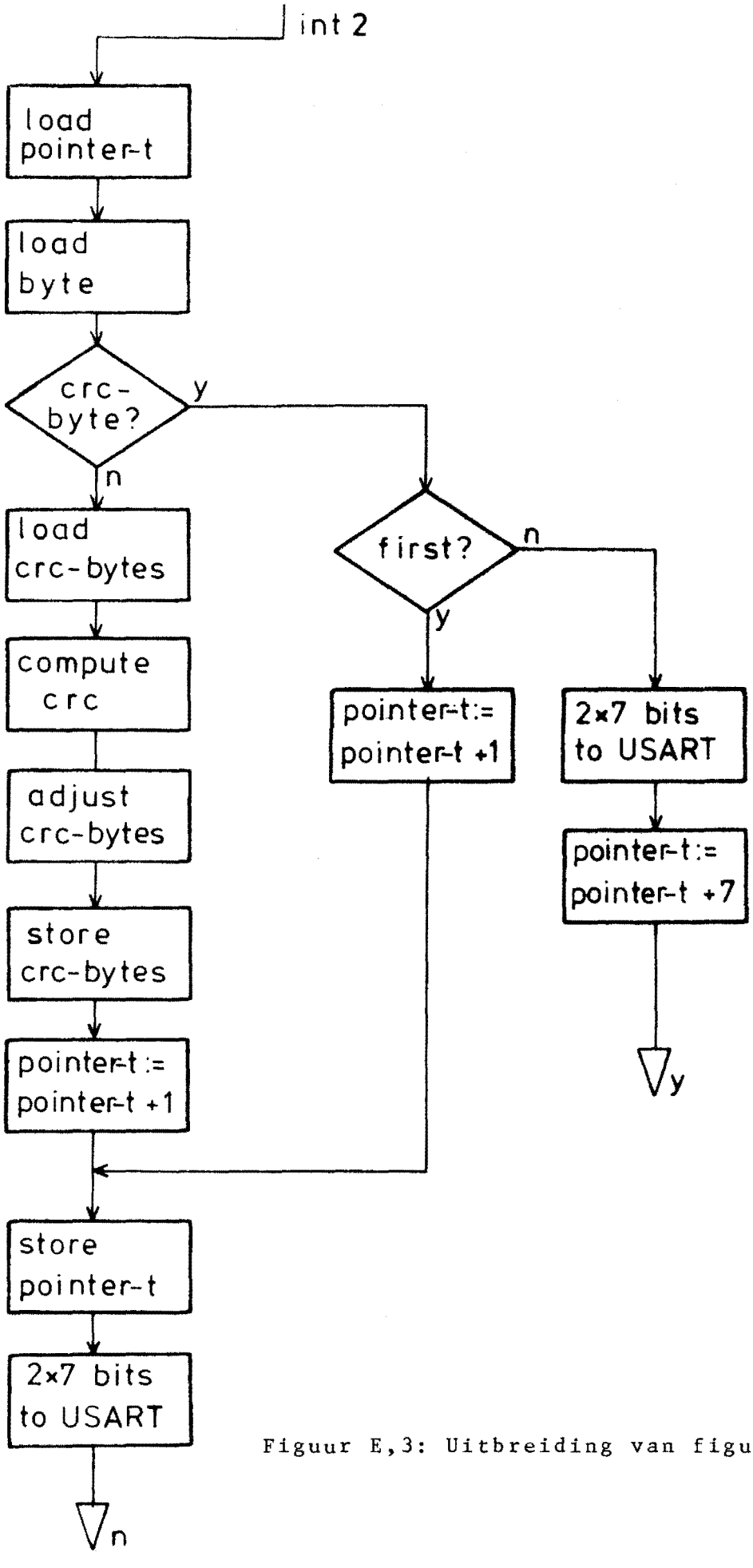


Figuur E.1: Acknowledge-tabel.

De plaatsen in de tabel corresponderen rechtstreeks met de blokken in Buffer B. Voor andere gegevens dan meetwaarden zijn nog blokken in een commando-buffer gereserveerd, in het gemeenschappelijk geheugen tussen opnemersysteem en zendsysteem.
Het genereren van een blok gebeurt nu door het plaatsen van een ack=0 op de betreffende plaats in de tabel. Het zendmechanisme weet nu welke blokken verstuurd moeten worden. Na goedmelding verdwijnt automatisch de aanvraag (ack:=1).
Voor het zendmechanisme is een pointer nodig, pointer-t, om de momentane plaats in buffer B te onthouden. Eerst volgt nu het mechanisme van retransmissie, met op de volgende pagina een uitwerking van het gestippelde blok:



Figuur E.2: Retransmissie-diagram.

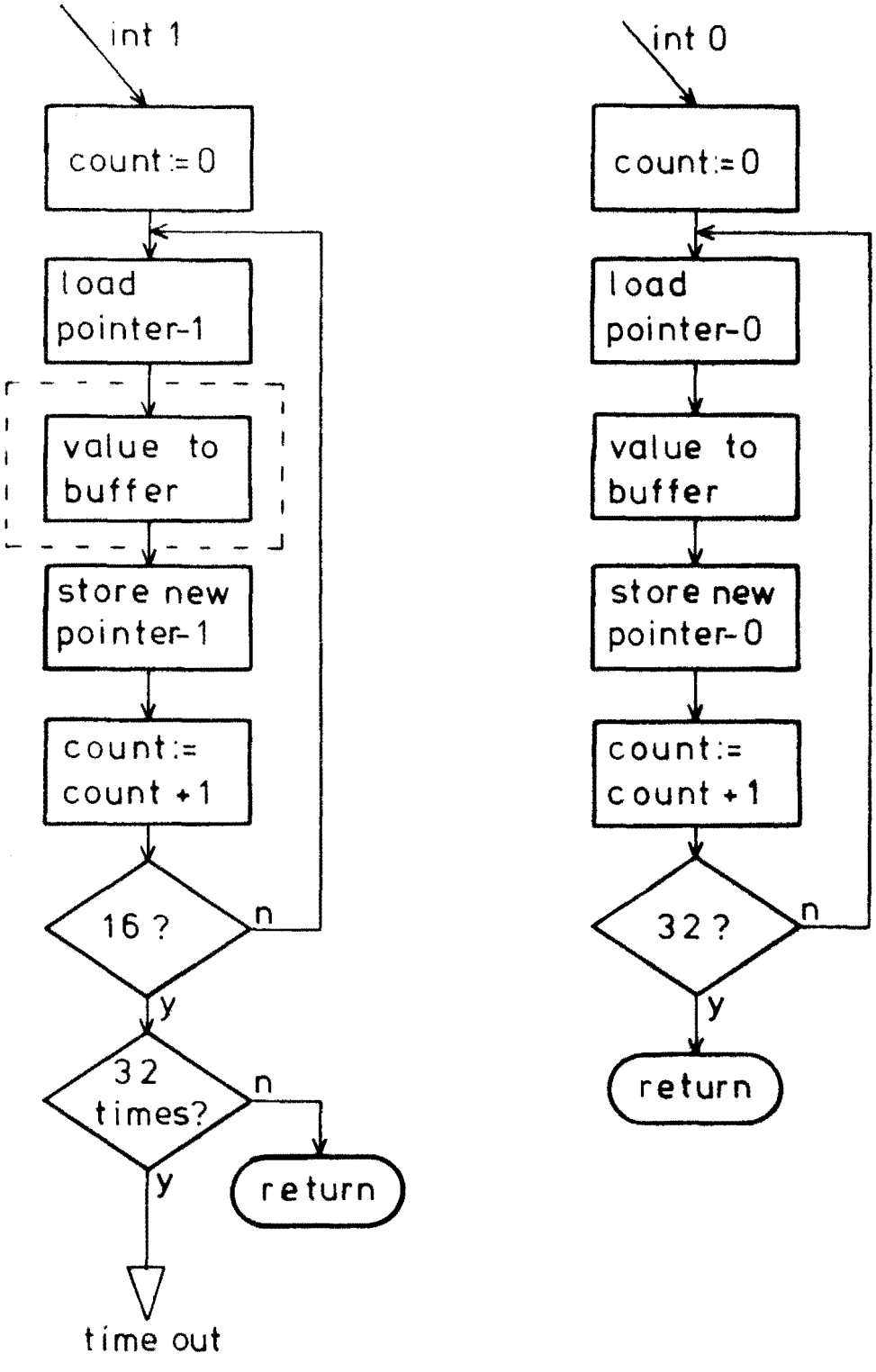


Figuur E,3: Uitbreiding van figuur E.2.

E.1.2 BEMONSTEREN. De interrupts 0 en 1 starten het bloksgewijs overnemen van meetwaarden van het gemeenschappelijke geheugen naar het prive-geheugen:

INT0: 4* per seconde worden van 32 kanalen de 16 bit meetwaarden opgehaald,
INT1: 32* per seconde worden van 16 kanalen de 16 bit waarden overgenomen.

Deze meetwaarden worden in buffer A geplaatst, dit heeft dezelfde vorm als buffer B (zie figuur E.8). Ook hier wijzen pointers (-0 en -1) naar de volgende vrije plaats in het buffer.
Na de 32-ste INT1 wordt de time-out routine gestart: het zendmechanisme wordt onderbroken of opnieuw geactiveerd (figuur E.2) maar daarvoor worden de pointers (-0, -1 en -t) naar het andere buffer verplaatst. Het zendsysteem krijgt een nieuwe seconde informatie te versturen en het bemonstermechanisme weer een vrij buffer.



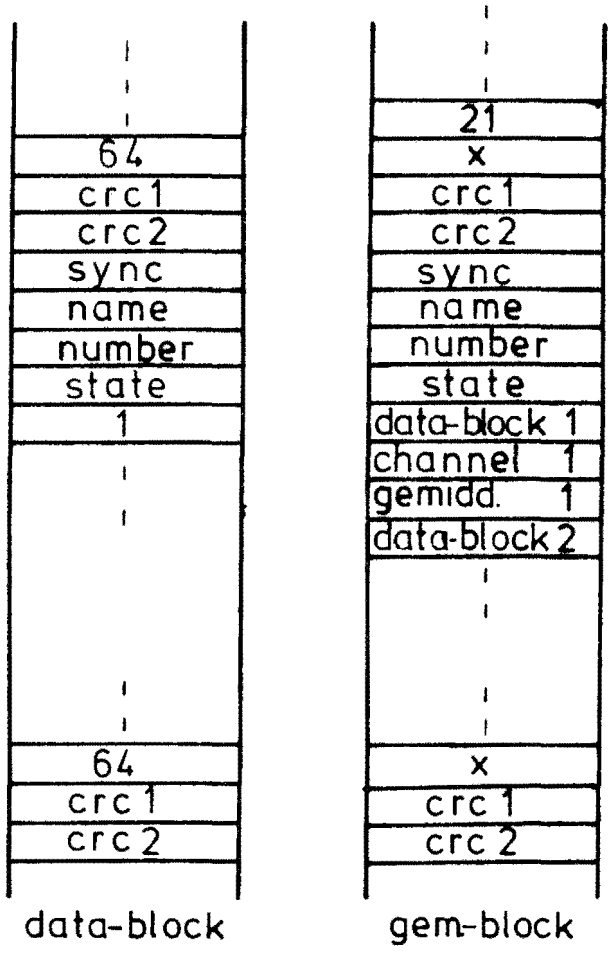
Figuur E.4: Bemonsterings-diagram.

Bij het plaatsen van de meetwaarde in het buffer wordt gebruik-

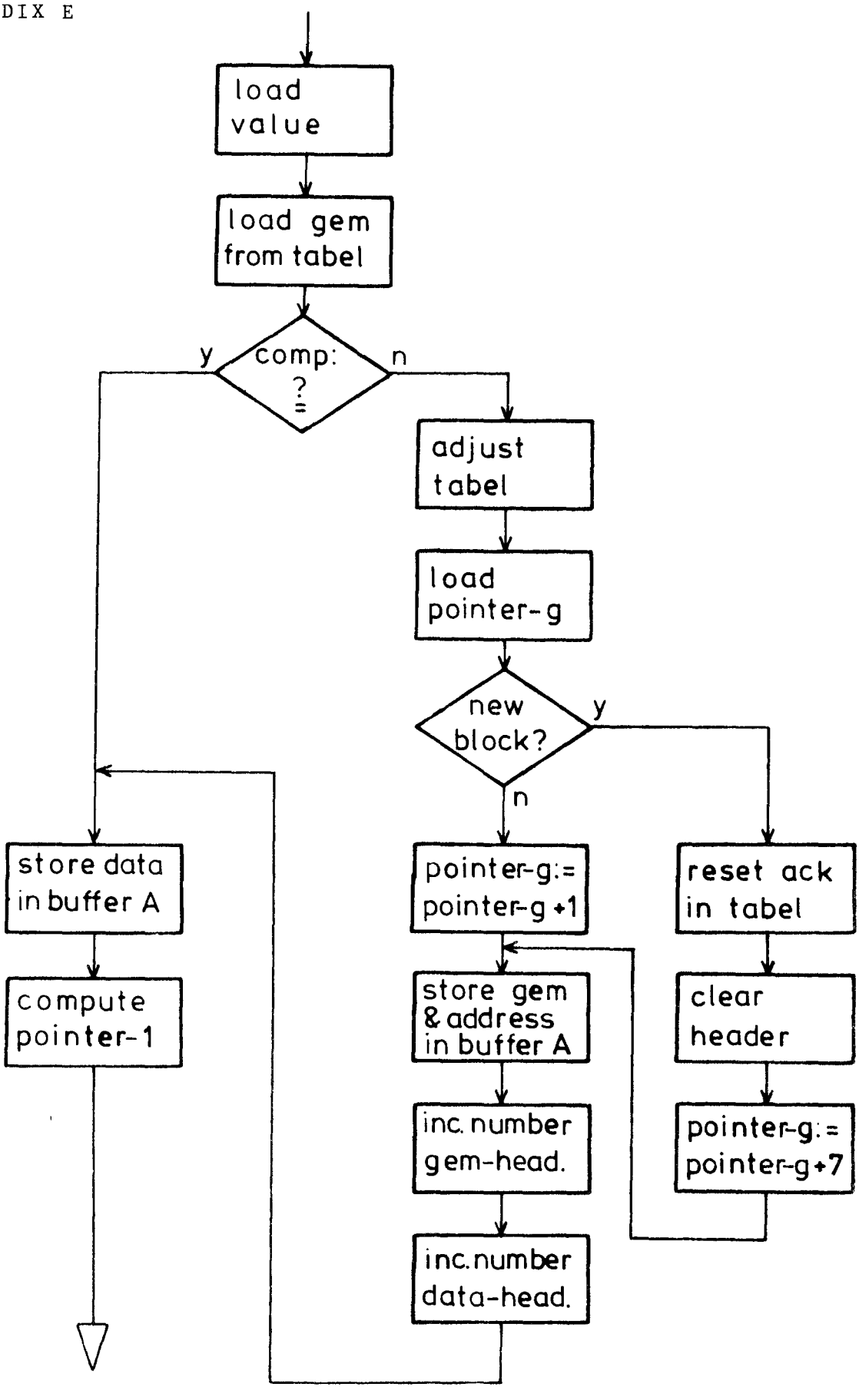
gemaakt van een gemiddelde-tabel. Hierin zijn van alle 48 kanalen de laatste meest significante byte bewaard.
Van elke meetwaarde (value) uit het gemeenschappelijk geheugen wordt nu de meest significante byte (gem) vergeleken met de tabel-waarde. Bij verschil wordt de tabelwaarde vernieuwd. In de header van elk data-blok is een teller (number) opgenomen, die aangeeft hoeveel gemiddelden behorend bij het betreffende blok zijn veranderd. (Dit is noodzakelijk om bij ontvangst van het data-blok maar bij verlies van het gemiddelde-blok, dit data-blok te kunnen verwerpen.)

Deze teller wordt nu met één verhoogd. Voorts wordt de gemiddelde-pointer geladen, pointer-g, en wordt het nieuwe gemiddelde, voorzien van de naam van het betreffende data-blok en het nummer in dit blok, in het gemiddelde-blok geplaatst. De teller, number, in dit blok wordt ook verhoogd, omdat dit een blok met variabele inhoud is.

Wanneer een nieuw gem-blok moet worden gegenereerd, gebeurt dit door op de overeenkomende plaats in de tabel ack=0 te maken.

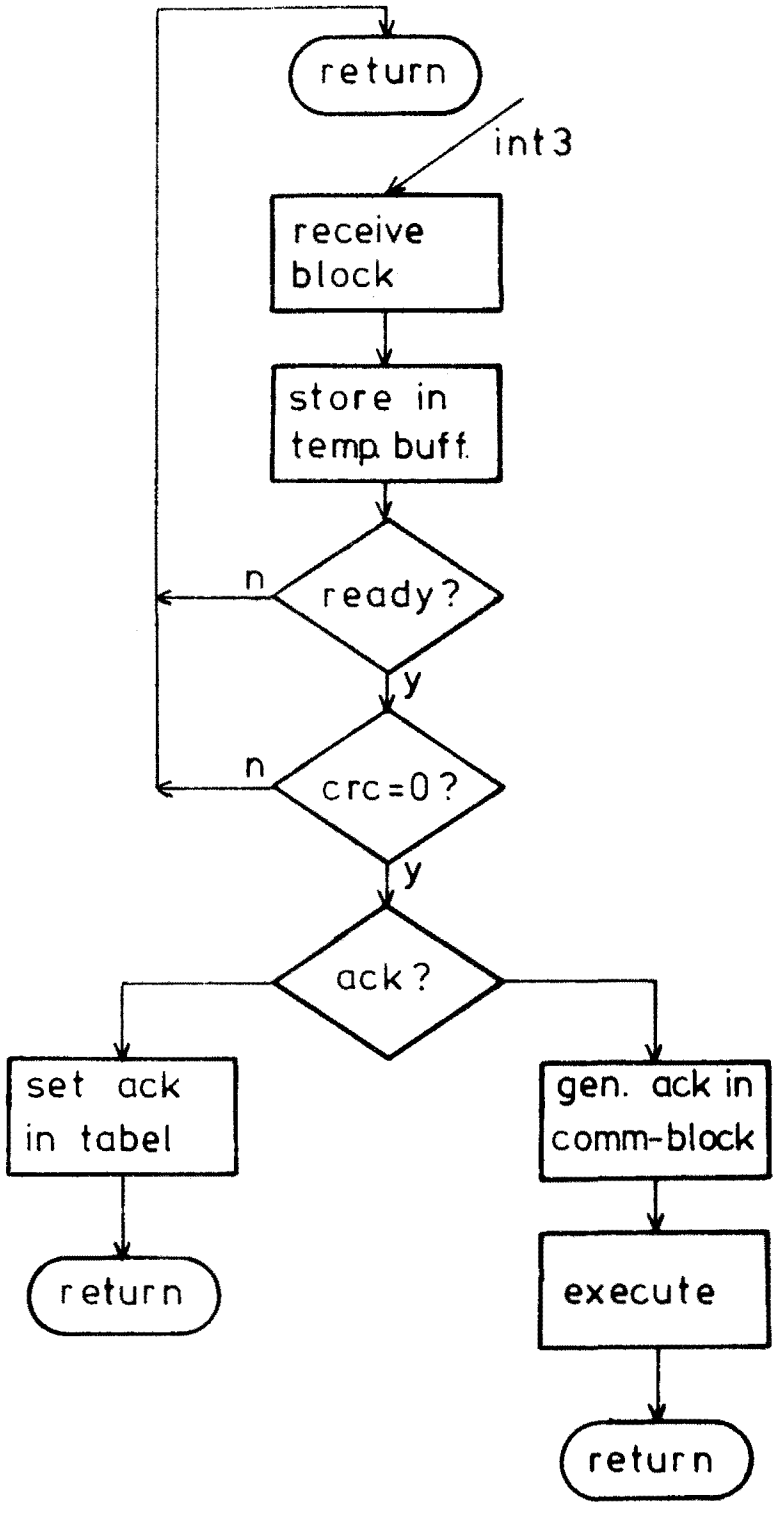


Figuur E.5: De blokken in buffer A.



Figuur E.6: Uitbreiding van figuur E.4.

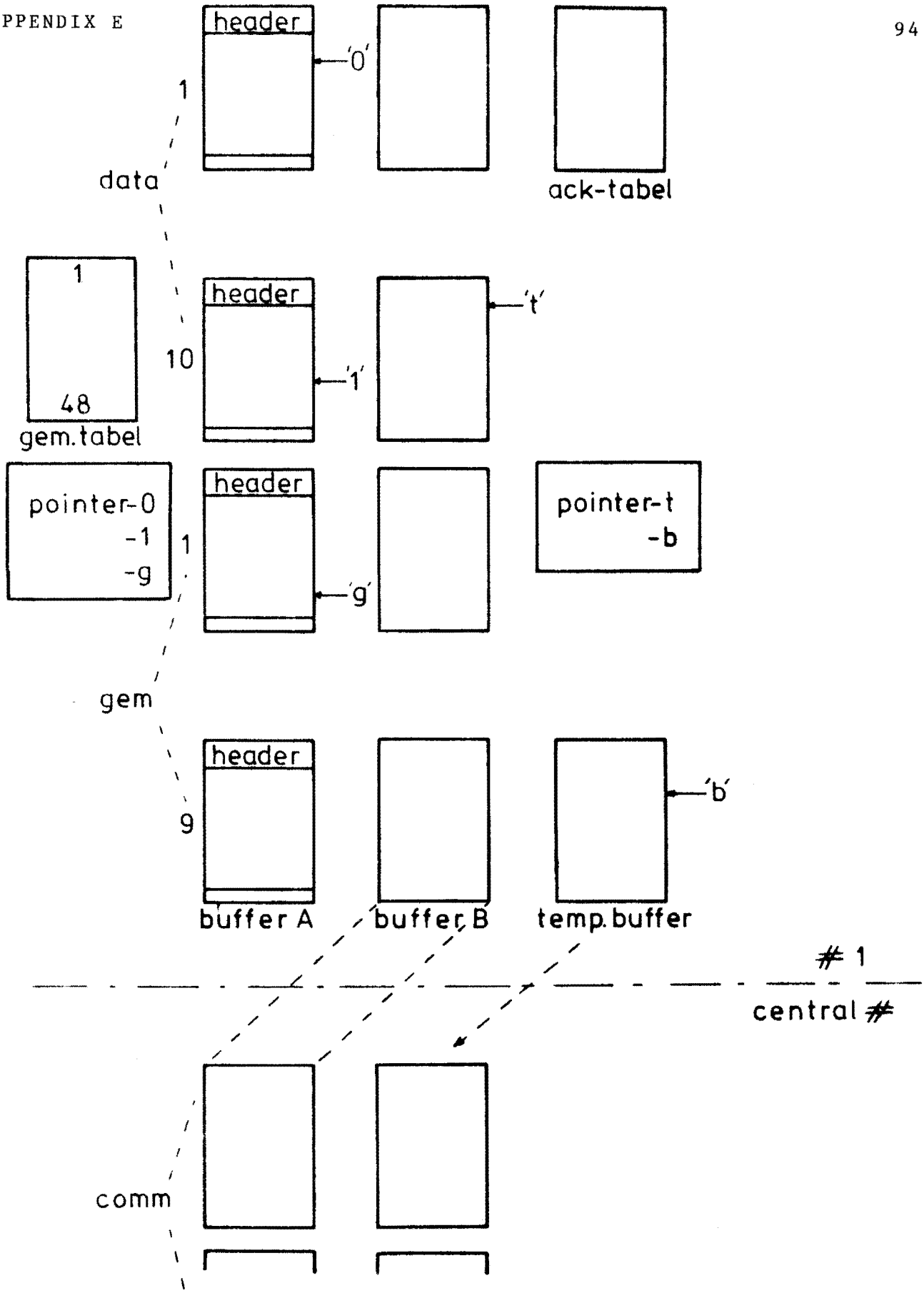
E.1.3 RECEIVER. Tenslotte nog de receiver-ready-interrupt, die het retransmissiemechanisme voltooit:



Figuur E.7: Receiver-diagram.

Wanneer het geen goedmelding betreft moet uit het ontvangen blok kunnen blijken wat de bedoeling is (b.v. nieuwe opstartgegevens, het transmissiesysteem kan eventueel het opnemersysteem middels interrupts hierop attenderen).

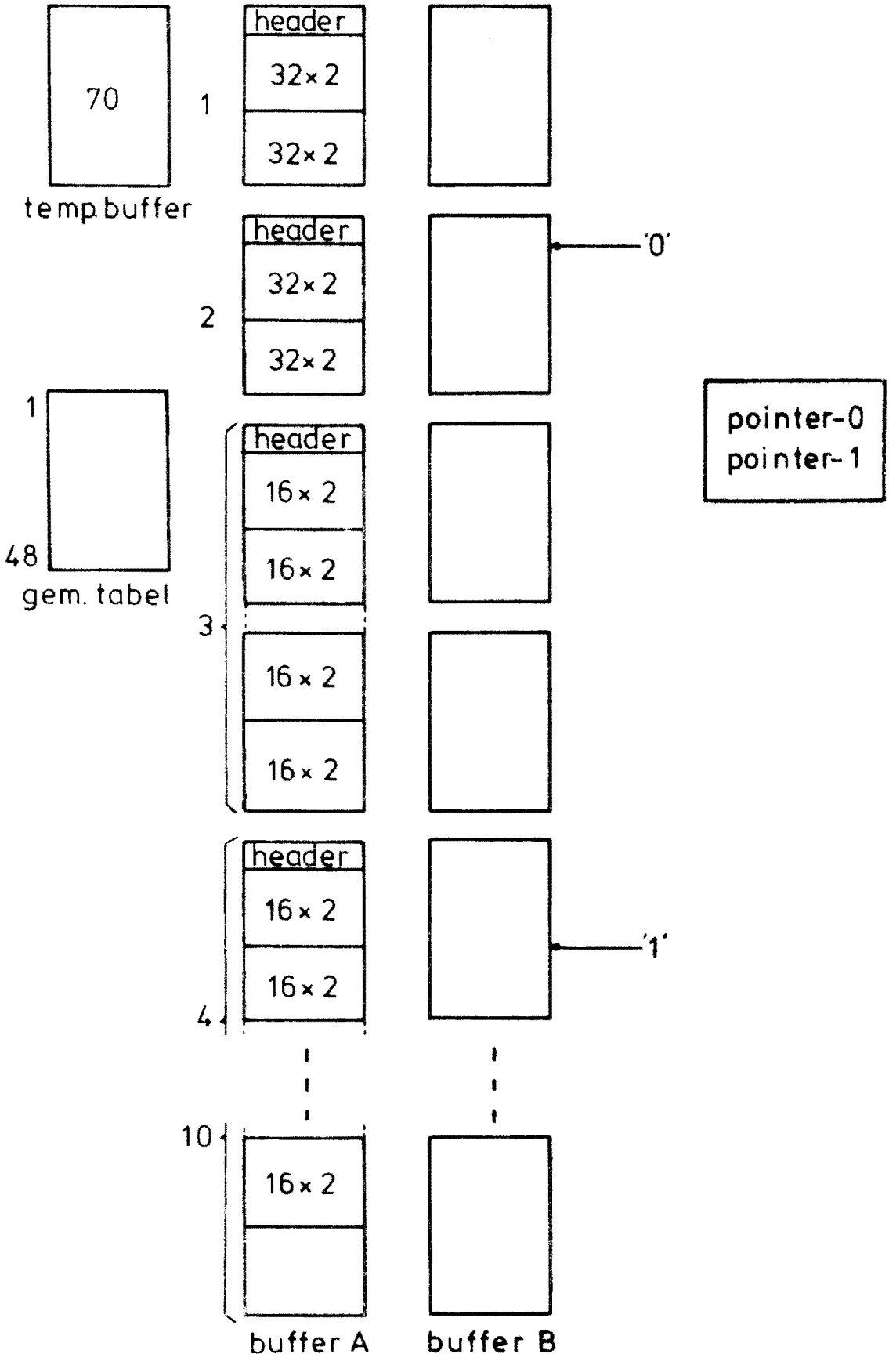
E.1.4 GENEREREN van BLOKKEN. De transmitter-routine (figuur E.2) controleert voortdurend de ack-tabel op nog te versturen blokken. Het genereren van de verzoeken voor de 10 data-blokken wordt door de time-out routine verzorgd. De bemonster-routines, INT0 en INT1, starten eventuele gemiddelde-blokken op.
De transmissie-routine genereert zelf de retransmissie-blokken, het blok krijgt een nieuwe naam en er wordt een ack=0 in de tabel geplaatst.
De commando-blokkken in het gemeenschappelijke geheugen kunnen zowel door het opnemersysteem worden aangevraagd als door de receiver-interrupt. Zendsysteem, #1, kan een commando-blok genereren, voor goedmeldingen van een ontvangen commando-blok, van ontvangsysteem #2 (figuur E.7). Tevens kan het opnemersysteem commando-blokken aanvragen, b.v. voor verzoeken om opstartgegevens. Om problemen te voorkomen kunnen hiervoor aparte buffers worden gereserveerd, of er moet een protocol voor worden gemaakt.
Tenslotte volgt een overzicht van de geheugenindeling van het zendsysteem, met een gedeelte van het gemeenschappelijke geheugen:



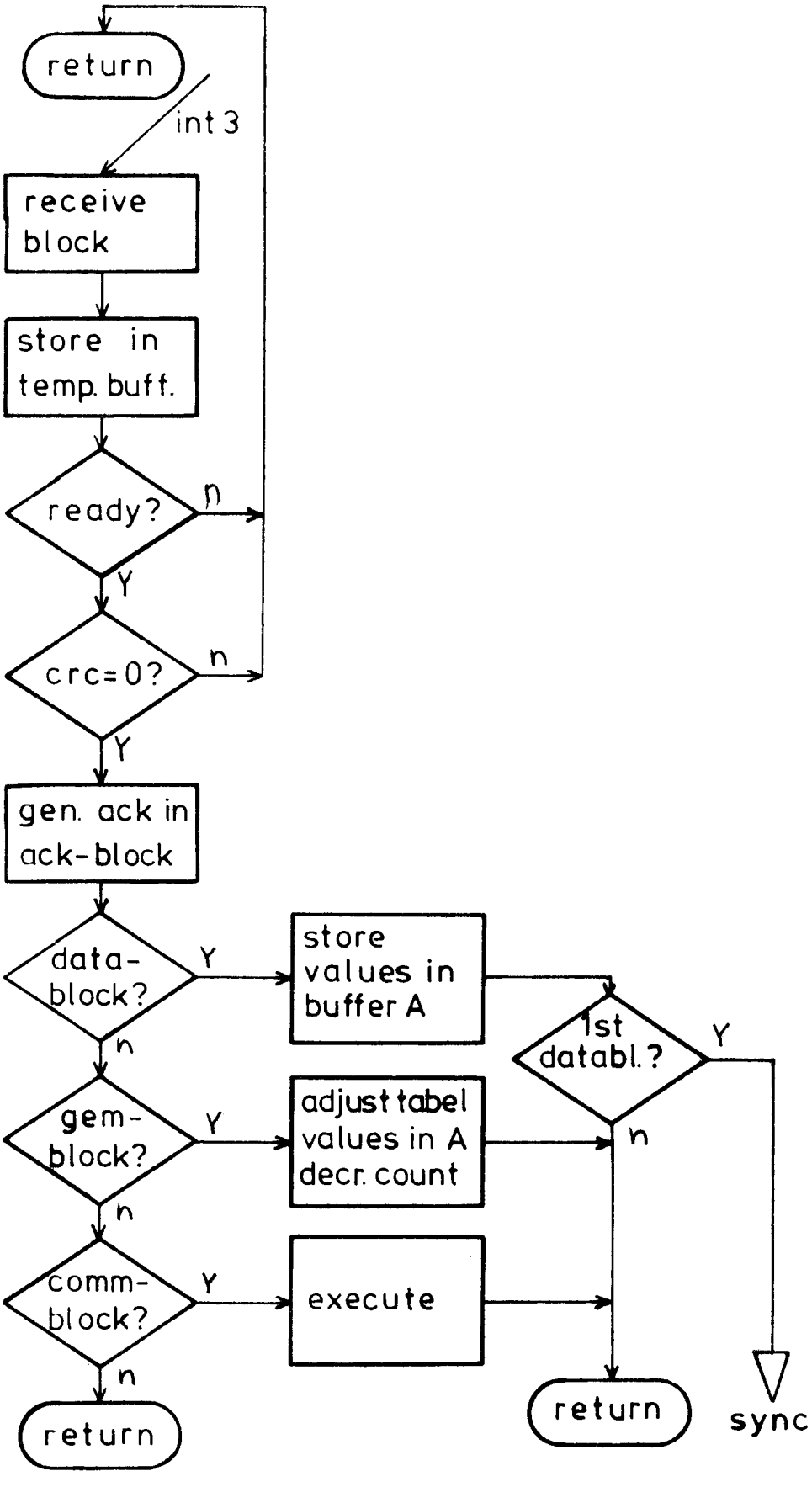
Figuur E.8: Geheugenindeling van het zendsysteem.

E.2 ONTVANGSTSYSTEEM. Aan de andere kant van het transmissiekanaal heeft systeem #2 nu te maken met het reeds vastgestelde zendschema. We kunnen hier volstaan met een overzicht van de belangrijkste buffers en pointers (figuur E.9) en de behandeling van de receiver-ready-interrupt, INT3.

E.2.1 RECEIVER. De receiver-interrupt regelt het vullen van buffer A. In buffer A worden de minst significante bytes (data) weer samengevoegd met de meest significante (gem) uit de tabel. Bij ontvangst van gem-blokken wordt de gem-waarde in buffer A veranderd en de teller (number) in de header verlaagd. Is de stand 0 geworden dan is het betreffende blok geheel aangepast. Uiteraard wordt ook de waarde in de gem-tabel aangepast (zie figuur E.10).



Figuur E.9: Geheugenindeling van het ontvangstsysteem.



Figuur E.10: Receiver-diagram.

E.2.2 SYNCHRONISATIE. De synchronisatie van #1 en #2 om de constante tijdvertraging ΔT te creeren vindt plaats na het herkennen van het eerste data-blok (het origineel). Nu worden de buffers A en B omgezet (door de pointers te veranderen) en de pointers -0 en -1 in de beginstand gezet.

De interrupts 0 en 1 verplaatsen nu de respectievelijke pointers naar de volgende gereedstaande data-blokken. Pointer-0 wijst dan naar de nieuwe meetwaarden van de 32 kanalen van 4 Hz, en pointer-1 naar de nieuwe waarden van de 16 kanalen van 32 Hz.

[Deze pointers kunnen worden opgevat als basisadressen voor de DMA-controller (Intel catalogus, 1977: 8257). Een request produceert dan direct het dan geldige blok meetwaarden. Voor de 32 Hz kanalen kan dan de "auto load"-ingang gebruikt worden, terwijl een verzoek voor de 4 Hz kanalen vergezeld gaat van een interrupt, om de processor het oude basisadres te laten herstellen.]

Wanneer een pointer door INT0 of INT1 naar een blok geplaatst zou moeten worden, dat fouten bevat, kunnen foutmeldingen naar een gebruiker van het distributiesysteem gestuurd worden. Dat het blok fouten bevat is op te maken uit de state- en numberbytes in de header. De state van een blok wordt in buffer A initieel op "fout" gezet, pas wanneer het blok ook inderdaad goed is ontvangen wordt de state "goed" gemaakt, slechts een number $\neq 0$ kan nu nog tot een fout blok leiden.

F.0 MATRIX <-> BUS

Een maat voor de kwaliteit van een verbindingsnetwerk, binnen een real time meetsysteem, is de verwachte doorlooptijd (DLT). Dit is de tijd die verstrijkt tussen het aanvragen van een blok-transport en het einde van het transport.

We beschouwen twee systemen, een parallelle busstructuur, met breedte a, en een enkelvoudige matrix. Om de twee systemen met elkaar te kunnen vergelijken, zullen we van beide de DLT berekenen.

F.1 UITGANGSPUNTEN. We stellen:

* x producenten en y consumenten

* geen voorkeur voor een combinatie (x,y)

* DLT enige vertraging in het systeem

* aanvragen van consumenten (gebruikers) Poisson verdeeld, gemiddelde λ.

Uit de theorie (lit. 7) is bekend, dat voor de bezettingsgraad van een loket geldt:

$$r = \lambda * \mu / c$$ = verkeersdrukte/aantal loketten

met: μ = E(verwerkingstijd) = E(v)

λ = E(aankomstsnelheid)

E(x)= verwachting van x

Zodat voor de doorlooptijd geldt:

DLT=E(wachttijd) + E(v)

=E(n)*E(v) + E(v)= (E(n) + 1)*E(v)

met: n= aantal in de wachtrij, en E(n)= r/(r-1).

F.2 MODELLEN. Om de DLT te kunnen berekenen maken we voor de beide systemen een loketmodel:

BUS: breedte a, 1 loket, $\lambda_b = \lambda$, $E_b(v)=T$.

MARIX: (x,y), x loketten, $\lambda_m = \lambda/x$, $E_m(v)=aT$.

We hebben daarmee aangenomen:

* transporttijd matrix is a maal groter dan transporttijd bus

[Dit is geoorloofd, aangezien elke seisnelheid door een matrix gerealiseerd, ook op een bus (en wel a-voudig) bereikt kan worden.]

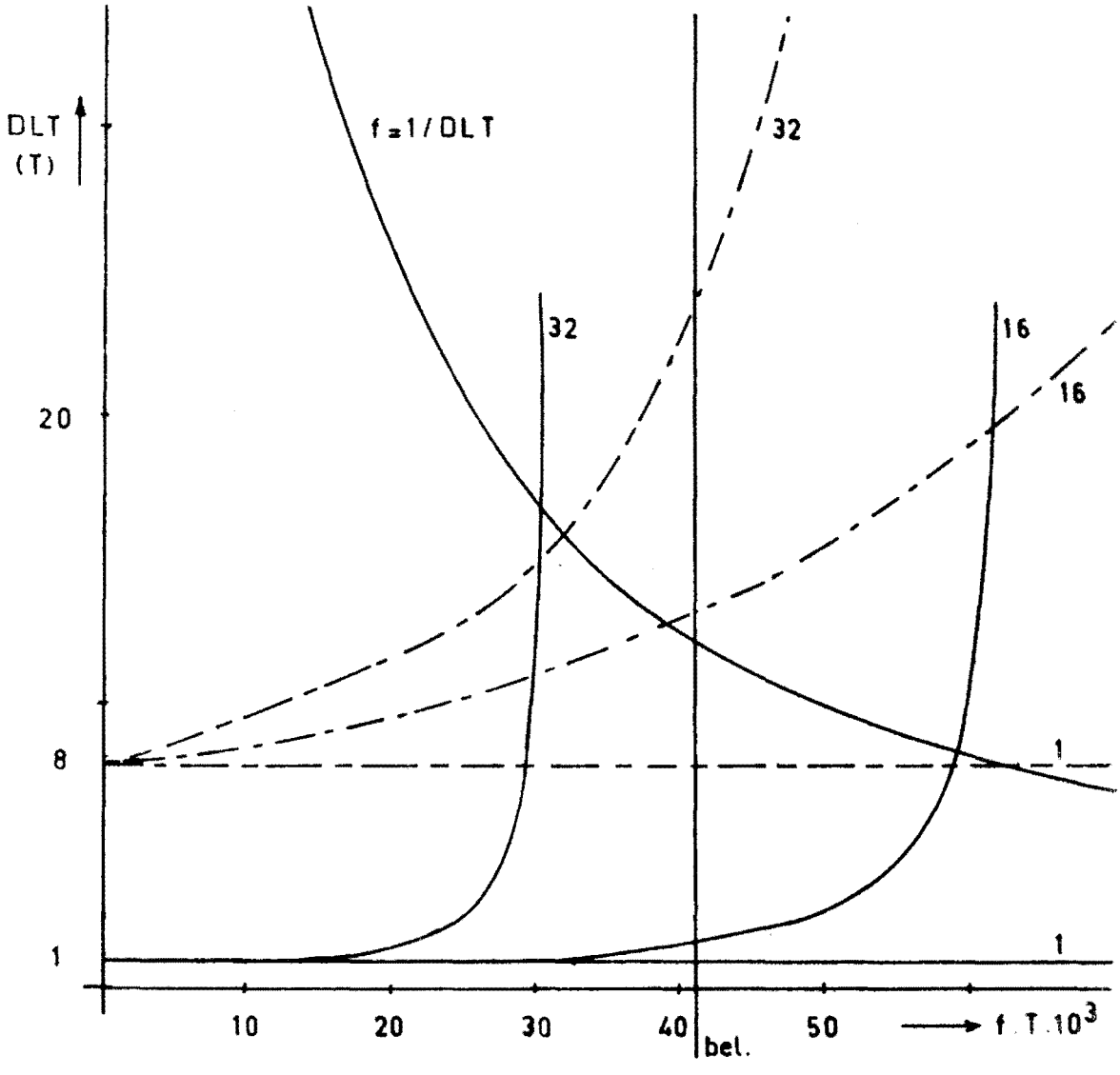
We vinden als oplossing voor de doorlooptijden van de systemen:

DLTb = [(f*T*y)/(1-f*T*y)]*T

DLTm = [(f*T*a*y/x)/1-f*T*a*y/x)]*aT

met: f= aanvraagfrequentie van een cunsument y.

Aan deze formules is direct de kracht van een matrixsysteem te zien. Onder de gestelde condities is de verwachte doorlooptijd niet afhankelijk van het totale aantal consumenten en producenten, maar slechts van de verhouding y/x. Het nadeel van de matrixstructuur is de factor a, hetgeen ook uit de grafiek blijkt.



Figuur F.1: Doorlooptijd van de twee systemen.

met: -- - -- : matrix
------- : bus
x = 32 (producenten)
y = 1, 16, 32 (consumenten)

In de grafiek is de lijn f=1/DLT aangegeven: in het meetsysteem heeft het geen zin langer op een aanvraag te wachten dan de geldigheidsduur van het aangevraagde datablok. De belastingslijn in

de figuur geldt voor een aanvraagfrequentie van de gebruikers van 1000Hz (f=32Hz*32 groepen) en een transporttijd T=~40μsec. (hoofdstuk 8.3).

F.3 CONCLUSIE. Onder de gestelde condities is het bussysteem sneller dan het matrixsysteem en beter geschikt om aan de analoge eis (geringe faseverschuiving van blokken 5.5.8) te kunnen voldoen.

G.0 INTERFACE.

In figuur G.1 is een interface afgebeeld tussen een 8080-microcomputersysteem en de distributie-bussen. In de volgende tabel worden de signalen van het combinatorische netwerk beschreven, dat uitgevoerd kan worden als PROM, FPLA of met TTL-circuits:

comb.:

reset = Ad0 . $\overline{Ad1}$. $\overline{R}/W$. besel . exec

int4 = Ad0 . $R/\overline{W}$. besel . exec

ack a = $\overline{Ad0}$. $\overline{Ad1}$. $\overline{R}/W$. besel . exec

ack b = scsel . exec

req = Ad1 .(scsel+besel). exec

stb =($\overline{Ad0}$. $\overline{Ad1}$. $\overline{R}/W$. scsel +
$\overline{Ad0}$. $\overline{Ad1}$. $R/\overline{W}$. besel). exec

en 3 = desel + scsel

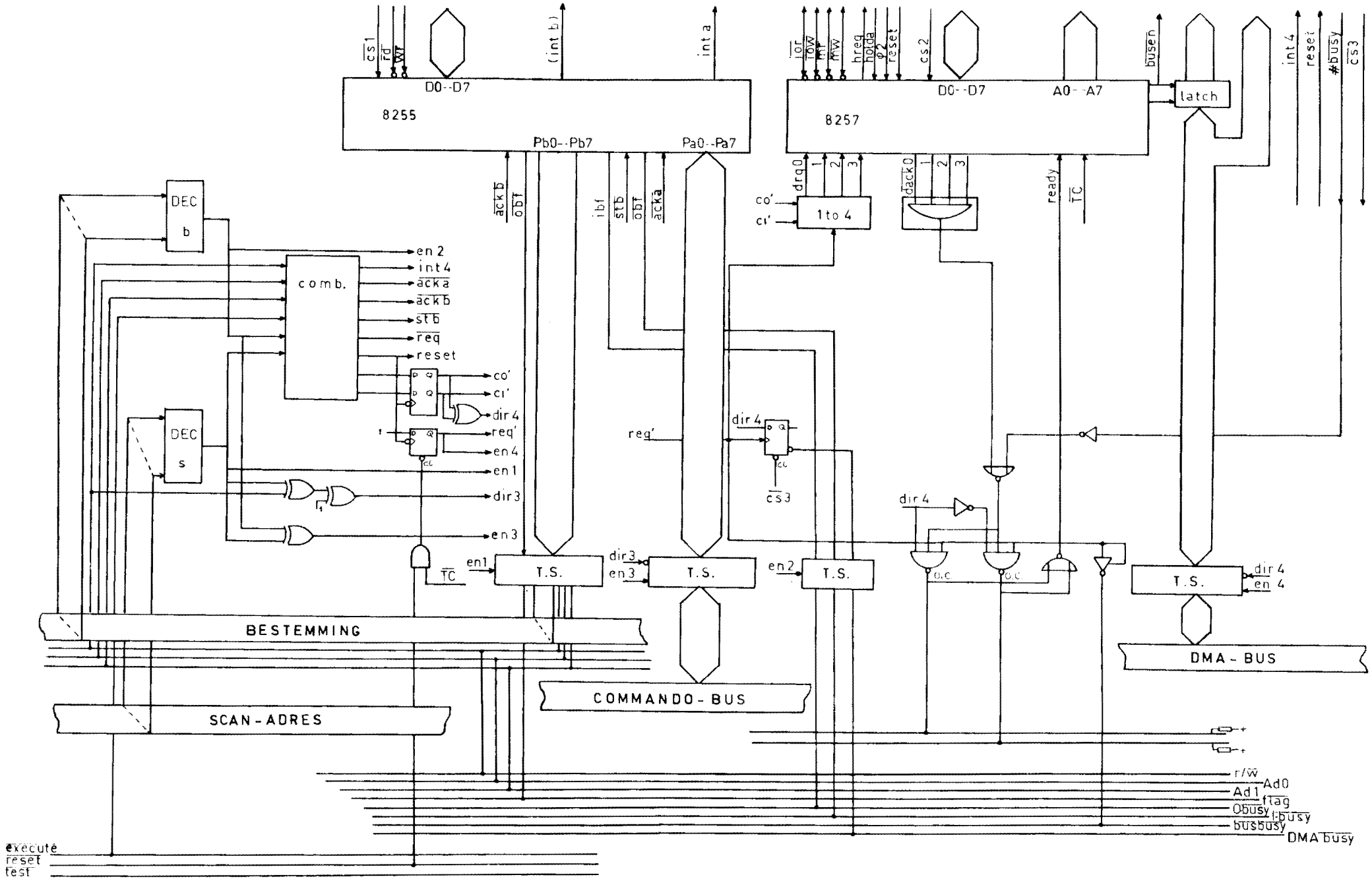
dir 3 = $\overline{R}/W$ + scsel

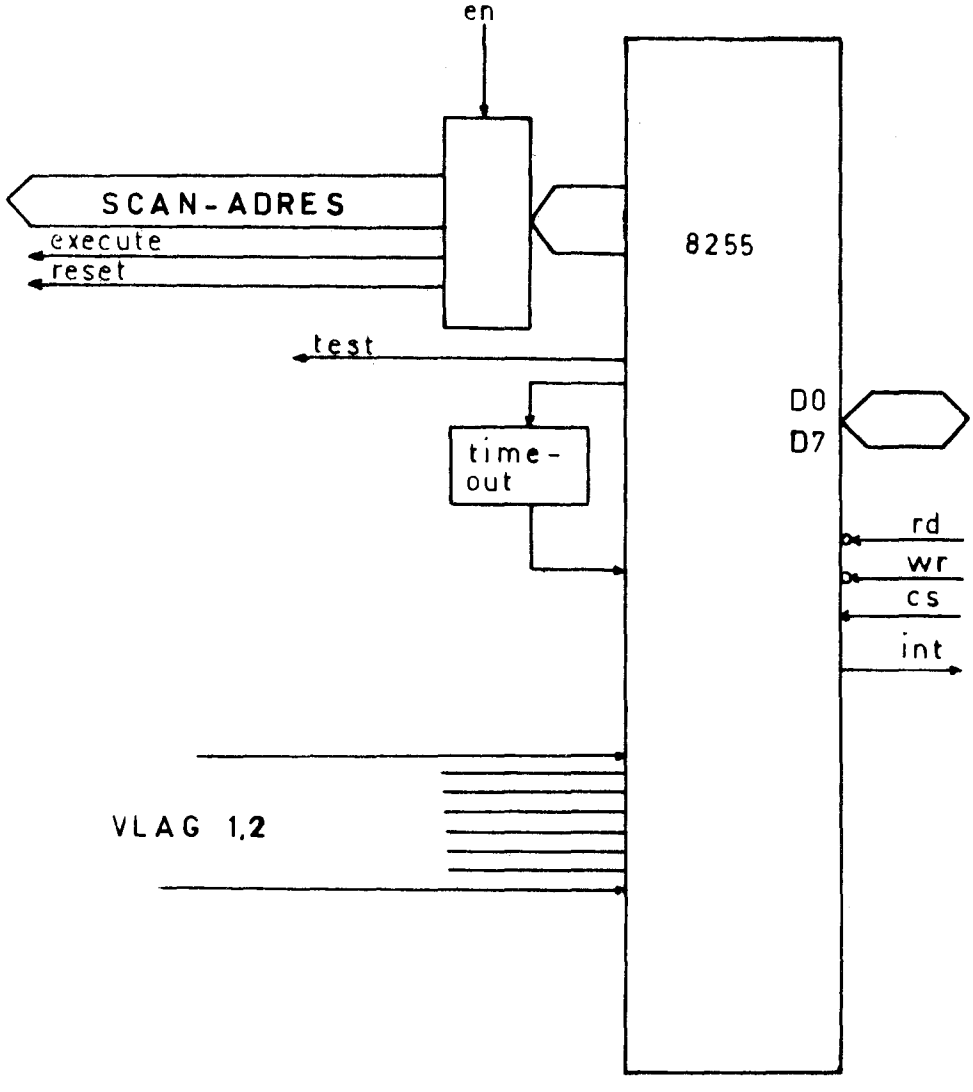
en 4 = req'

dir 4 = c0' $\oplus$ c1'

c0 =(Ad0 + $R/\overline{W}$).scel + $\overline{Ad0}$. $\overline{R}/W$. $\overline{scsel}$

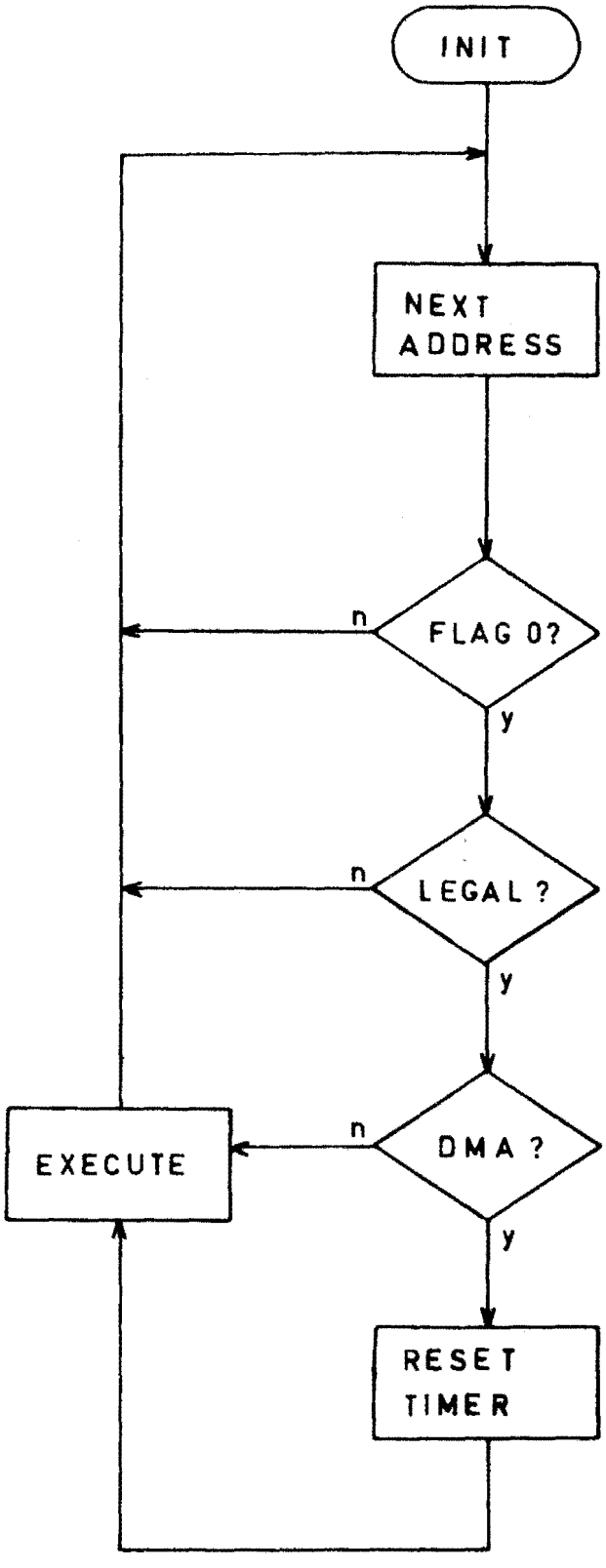
c1 = Ad0 . $\overline{R}/W$


8255
D0--D7
Pb0--Pb7
Pa0--Pa7
8257
A0--A7
latch
1 to 4
comb.
DEC b
DEC s
T.S.
BESTEMMING
SCAN-ADRES
COMMANDO-BUS
DMA-BUS
execute
reset
test
r/w
Ad0
Ad1
flag
Obusy
ibusy
busbusy
DMA busy



DISTRIBUTIE BUSSEN INTERFACE PROCESSORBUS

Figuur G.2: Arbiter-interface.



Figuur G.3: Arbiter flow-diagram.