Hardware van en software voor het meetsysteem PCM2

Hardware van en software voor het meetsysteem PCM2

A. Dortmans, K. Koekkoek,
G. Teurlinx

WFW 86.032

Hardware van en software voor het meetsysteem PCM2

A. Dortmans
K. Koekkoek
G. Teurlinx

WFW juni 1986
WFW 86.032

Inhoudsopgave

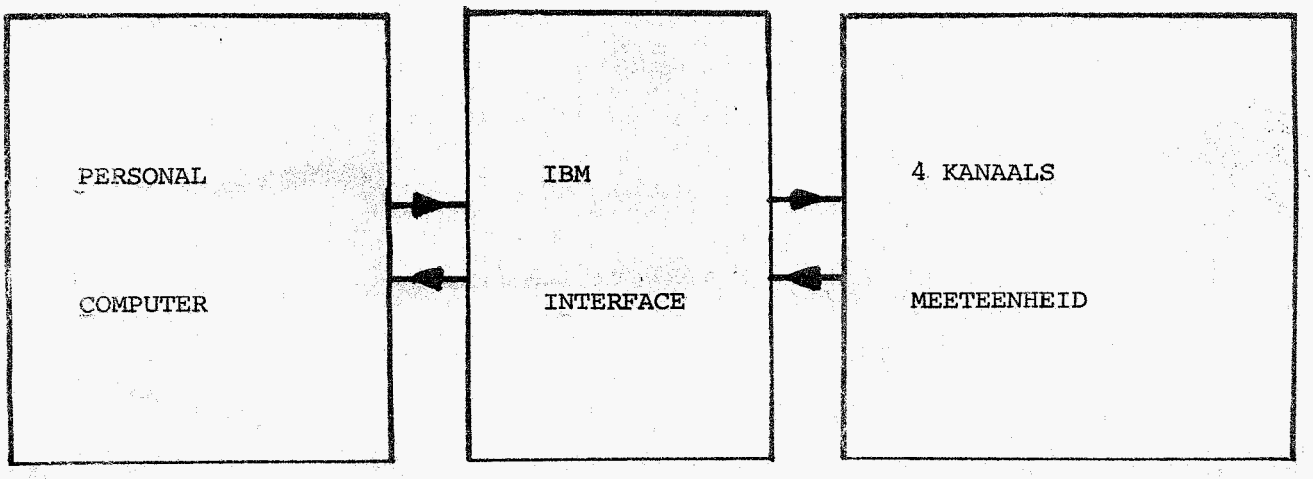
Inleiding

Het analyseren van het (dynamisch) gedrag van een systeem vereist in vele gevallen een complex data-acquisitie systeem. Vaak is een eerste eis dat zo'n systeem in staat is een aantal analoge signalen simultaan te meten, de gemeten waarden om te zetten naar digitale vorm en op te slaan. Daarnaast moet het systeem ook enige vorm van on-line postprocessing toelaten, hetgeen duidelijke eisen stelt aan de intelligentie van het systeem.
Een voorbeeld hiervoor kan zijn het analyseren van het dynamisch gedrag van een systeem dmv. transfer-functie-analyse. Hierbij worden 2 signalen op equidistante tijdstippen gemeten en vervolgens gebruikt voor het bepalen van de overdrachtsfunctie tussen de 2 signalen dmv Fast Fourier Transform technieken. De overdrachtsfunctie legt dan een verband tussen de signalen in het frequentiedomein. Natuurlijk zijn deze analyses ook uit te voeren mbv. een zgn. wave-analyser. Deze systemen zijn echter vrij kostbaar en zodanig star dat ingrijpen in het feitelijke meet-en verwerkingsproces nauwelijks mogelijk is.
Het hier besproken data-acquisitiesysteem is ontwikkeld met het doel een vrij flexibel meet-en verwerkingssysteem te krijgen tegen een redelijke prijs. Uitgangspunten hierbij zijn:

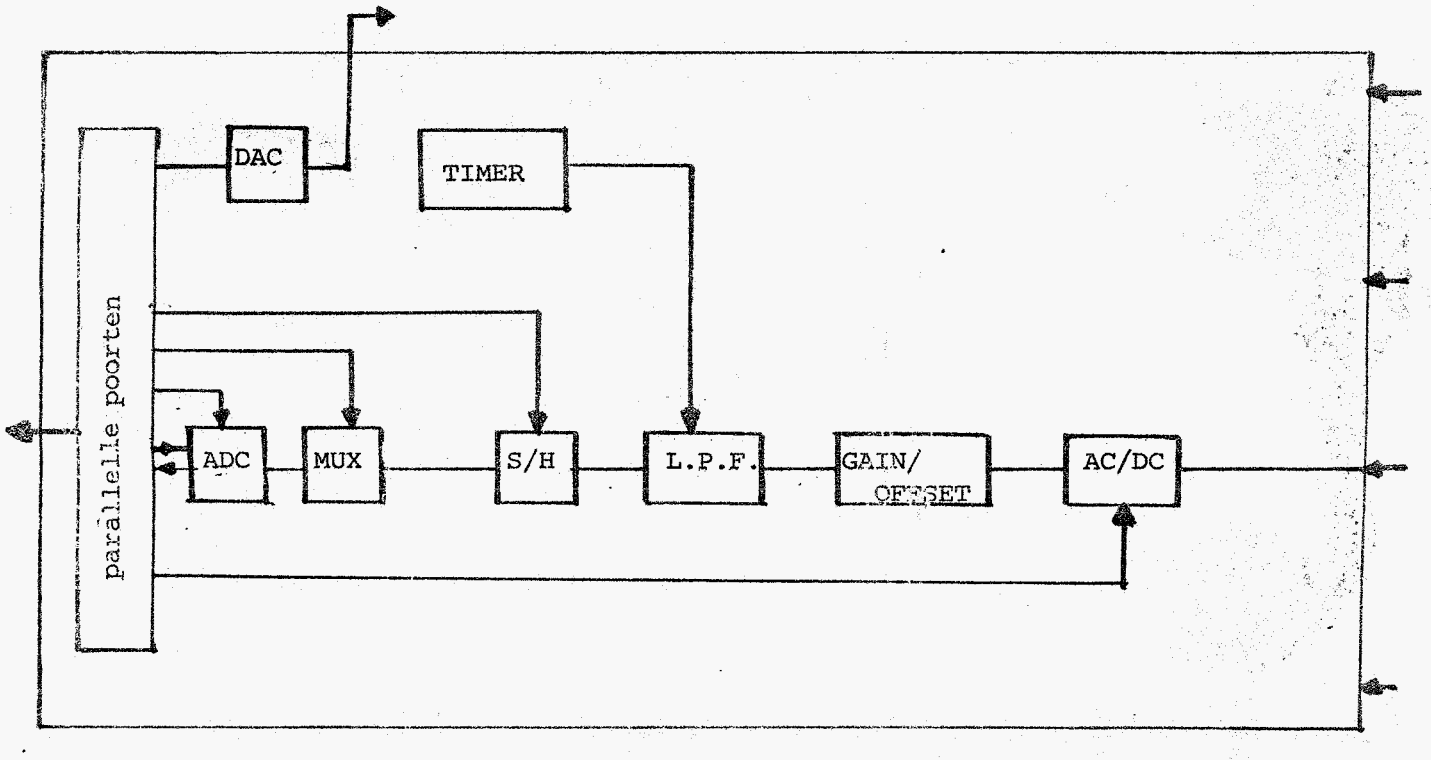
- het systeem moet een flexibel,software gecontroleerd, meetproces mogelijk maken
- het systeem moet voorzien in een aantal mogelijkheden wat betreft signaal-conditionering (filteren, versterken etc.)
- het systeem moet het aansturen van externe apparatuur (bijv. stappenmotoren) mogelijk maken

Op basis van de gestelde eisen is gekozen voor een systeem bestaande uit een standaard Personal Computer en een separate 4 kanaals meet-eenheid (totale kosten excl. man-uren ong. f 10000,-). De hardware is uitgebreid met een hoeveelheid software die kan dienen als uitgangspunt bij het ontwikkelen van een specifiek meet- en verwerkingsprogramma, dwz zij verschaft een aantal basisroutines voor het aansturen van de meeteenheid en het doen van metingen die gecombineerd kunnen worden naar gelang de gebruiker dit wenselijk acht.

Om een globaal idee van het meetsyteem te geven bespreken we nu in het kort de opzet van de hardware. Een gedetailleerde bespreking van hard- en software vindt plaats in de rest van dit verslag.
Globaal kunnen we de hardware indelen in 3 onderdelen nl.:



De PC dient hierbij als centraal stuurmiddel, als opslagmedium voor de gemeten waarden en voor de eventuele postprocessing. Om communicatie met de feitelijke meeteenheid mogelijk te maken wordt gebruik gemaakt van een standaard IBM uitbreidingskaart.
De meeteenheid kunnen we als volgt indelen:



Elk van de 4 analoge ingangen is identiek uitgevoerd en omvat:

- een omschakelbare AC/DC ingang

- een instelbare gain en offset om het ingangssignaal optimaal aan te passen aan de range van de Analoog Digitaal Converter (ADC)

- een analoog, low-pass, anti-aliasing filter met een variabele kantelfrequentie die ingesteld m.b.v. de programmeerbare timer-chip. Deze anti-aliasing filters zijn meestal nodig om de frequentie inhoud van het analoge signal te beperken, omdat we immers moeten voldoen aan het bemonsteringstheorema van Shannon, dat zegt dat de maximale frequentie in een signaal, dat met een gegeven sample frequentie wordt bemonsterd, hoogstens de helft van deze samplefrequentie mag zijn.

- een Sample-and-Hold schakeling die wordt gestuurd dmv de parallelle poorten. Deze S/H schakelingen zijn ingebouwd om het simultaan meten van de 4 kanalen mogelijk te maken, door bij de start van het meetproces alle S/H gelijktijdig te activeren.

De parallelle poorten dienen enerzijds ter sturing van andere componenten van de meeteenheid, anderzijds verschaffen zij een aantal digitale in- en uitgangen. Zo is een triggeringang aanwezig waarmee het starten van een meetproces kan worden gestuurd. Tevens zijn digitale uitgangen aanwezig die het mogelijk maken een 3-tal stappenmotoren te sturen. Voor het aansturen van externe apparatuur (bijv. een elektromechanische shaker) is een Digitaal Analoog Converter (DAC) aanwezig, waarmee een gewenste analoge spanning in de range van -10 tot +10 Volt gegenereerd kan worden. De timer-chip dient behalve voor het controleren van de kantelfrequentie van de anti-aliasing filters, ook als klok voor het controleren van de sample-snelheid.

Hoofdstuk 1 Opzet van het meetsysteem

Bij het opzetten van het meetsyteem is uitgegaan van een aantal randvoorwaarden:

- het meetsysteem moet zo flexibel mogelijk zijn met een aantal mogelijkheden wat betreft signaalregeling en sturing van randapparatuur
- het meetsysteem moet in staat zijn met een redelijke snelheid een hoeveelheid data in te nemen (max. sample frequentie ong. 8 KHz., max aantal samples 32767)
- de aansturing van het systeem moet direct mogelijk zijn vanuit een hogere programmeertaal zdd dat de gebruiker zich niet hoeft te verdiepen in de aansturing van het systeem op machinetaal-nivo (tenzij niet-voorziene eisen worden gesteld)
- het meetsysteem moet tegen een redelijke prijs te bouwen en zonodig in veelvoud leverbaar zijn.

Op basis van deze randvoorwaarden is daarom gekozen voor een systeem bestaande uit:

- een IBM (compatibele) Personal Computer (prijs ong. f 8000,= anno 1986)
- een standaard IBM interface kaart (prijs ong. f 80,=)
- een zelf te ontwikkelen meeteenheid (geschatte materiaalkosten f 2000,=)
- een zelf te ontwikkelen ondersteunend softwarepakket dat goed gestructureerd en gedocumenteerd is.

In het navolgende zullen we de precieze invulling van een en ander bekijken.

Hoofdstuk 2 Gewenste mogelijkheden van de hardware van het meetsyteeem

Op voorhand zijn een aantal gewenste mogelijkheden van de hardware van het meetsyteem aangegeven:

- maximaal 4 analoge ingangen die simultaan gemeten moeten kunnen worden
- ingebouwde laag-doorlaat anti-aliasing filters met een variabele kantel-frequentie
- 12 bits Analoog-Digitaal conversie in de range van -5 tot +5 Volt (oplossend vermogen = 10/4096 = 2.5 millivolt)
- aanpasbare versterking/verzwakking (gain) v/h ingangssignaal en mogelijkheden voor het verschuiven van het DC-nivo van het ingangssignaal (offset)
- inschakelbare filters voor het wegfilteren van het DC-nivo van het ingangssignaal
- mogelijkheden voor het extern triggeren van gebeurtenissen
- voor het genereren van analoge signalen in de range van -10 to +10 Volt een 8 bits Digitaal Analoog Converter (DAC)
- mogelijkheden voor het genereren van digitale signalen (signalen met of het nivo 0 Volt of het nivo +5 Volt) voor het aansturen van randapparatuur (bijv. stappenmotoren)
- mogelijkheden voor het volledig automatisch calibreren van ingestelde gain/offset van een bepaalde ingang

Als eis hierbij is verder gesteld dat de instelling/aansturing van deze mogelijkheden zoveel mogelijk vanuit een gebruikersprogramma geregeld moeten kunnen worden. In hoofdstuk 4 zullen we zien hoe een en ander in werkelijkheid uitgevoerd is. Eerst geven we echter in hoofdstuk 3 enige begrippen die verduidelijkend kunnen zijn voor de lezer die niet direct met de in latere hoofdstukken gebruikte termen bekend is.

Hoofdstuk 3 Enige elementaire begrippen: bits,bytes,I/O lijnen en interrupts

In de volgende hoofdstukken komen begrippen als bytes,I/O lijnen en interrupts regelmatig aan de orde. We zullen trachten deze enigszins te verduidelijken voor de ermee onbekende lezer.

Zoals wel bekend zal zijn is de fundamentele werkeenheid van een PC het bit, dat een waarde 0 of 1 kan hebben. Een bij elkaar behorende verzameling van 8 bits noemen we een byte. Als we aannemen dat een byte een bepaalde (gehele) getalwaarde vertegenwoordigt dan is de afspraak de volgende:

byte = [bit7 bit6 ..... bit1 bit0]

waarde byte = 2^7* bit7 + 2^6* bit6 + ... + 2^0* bit0

(met bit7 als het most significant bit (MSB) en bit0 als het least significant bit (LSB)).
Het byte 01000110 vertegenwoordigt dan de decimale waarde 70.
Met deze binaire schrijfwijze kunnen we dan getallen representeren die liggen in de range 0 (byte 00000000) - 255 (byte 11111111).
Wanneer we nu 2 bytes (16 bits) combineren tot een word dan kunnen we getallen representeren die liggen in de range 0 (word 0000000000000000) - 65535 (word 1111111111111111). Het zal duidelijk zijn dat de notatie met bits niet erg praktisch is. Daarom wordt vaak een compactere notatie gebruikt, de zgn. hexadecimale notatie. Een verzameling van 4 bits , een zgn. nibble die een waarde aan kan nemen tussen 0 (nibble 0000) en 15 (nibble 1111), kan hiermee verkort worden genoteerd mbv 16 unieke symbolen (te weten 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F die respectievelijk de waarde 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14 en 15 vertegenwoordigen). Het byte 01000110 kunnen we dan compact noteren als 46 (immers 4 = 4 = 0100 en 6 = 6 = 0110). Om aan te geven dat 46 een hexadecimaal getal is noteren we dit met 46H. De decimale waarde van dit hexadecimale getal is eenvoudig te bepalen (immers 46H = 4*16^1 + 6*16^0 = 70). Een byte kan dan altijd gerepresenteerd worden met een hexadecimaal getal in de range 00H (= 0) -

FFH (= 255). Words kunnen we zo representeren met hexadecimale getallen in de range 0000H (= 0) - FFFFH (= 65535).
Tot dusver kunnen we zo alleen positieve getallen representeren. Een methode voor het representeren van negatieve getallen wordt gegeven door de zgn two's complement notatie. Het two's complement van een getal wordt gevormd door elk bit van het getal te veranderen in 0 als het 1 is en in 1 als het 0 is en er vervolgens 1 bij op te tellen (Het two's complement van 01000110 is dus 10111001 + 1 = 10111010 = BAH = 186). De afspraak bij het gebruik van de two's complement notatie is nu dat het getal negatief is als het MSB 1 is en positief als het MSB 0 is. De waarde van het negatieve getal volgt uit de two's complement van het getal. BAH is dan dus een negatief getal met als waarde -(two's complement van BAH) = -(01000101 + 1) = -(01000110) = -46H = -70 (ofwel BAH als negatief getal = 256 - BAH als positief getal).
Indien we een byte dus zien als een mogelijk negatief getal dan ligt de waarde tussen -128 (byte 80H) en +127 (byte 7FH). Op eendere wijze liggen words dan tussen -32768 (word 8000H) en +32767 (word 7FFFH).
In het vervolg nemen we steeds aan dat we hexadecimale getallen zien als positieve getallen tenzij we aanduiden dat we de two's complement notatie willen gebruiken.

Voor het aansturen van randapparatuur etc zijn binnen de PC 65535 zgn I/O lijnen beschikbaar (fig 3.1) die genummerd zijn van 0000H tot FFFFH.

RANGE	DEVICE
	System Board
0—1F	8237 4-channel DMA controller
20—3F	8259 8-channel interrupt controller
40—5F	8253 3-channel counter/timer circuit
60—7F	8255 24-line parallel I/O interface
80—9F	DMA 64K page register
0A0—0BF	NMI mask bit latch
0C0—0C7	PC*jr* sound generator
0C8—0EF	Reserved
0F0—0FF	PC*jr* floppy diskette interface
100—1FF	Not usable
	I/O Channel
200—20F	Game I/O adapter
210—217	Expansion unit
220—24F	Reserved
250—277	Not used
278—27F	Second parallel printer interface (LPT2)
280—2EF	Not used
2F0—2F7	Reserved
2F8—2FF	Second 8250 serial UART interface (COM2) ←
300—31F	Prototype card
320—32F	Hard disk
330—377	Not used
378—37F	First parallel printer interface (LPT1)
380—38C	SDLC or secondary binary synchronous interface
390—39F	Not used
3A0—3A9	Primary binary synchronous
3B0—3BF	Monochrome display and first parallel printer
3C0—3CF	Reserved
3D0—3DF	Color/graphics display adaptor
3E0—3EF	Reserved
3F0—3F7	5-1/4" floppy disk drive controller
3F8—3FF	First 8250 serial UART interface (COM1)

Fig. 3.1 Overzicht van de I/O lijnen in de PC

Een I/O lijn is gekoppeld aan hardware en kan gebruikt worden voor het binnenhalen (Input) of versturen (Output) van bytes naar de hardware. De processor van de PC kent hiervoor 2 basisinstructies nl IN en OUT, die als volgt gebruikt worden:

```
IN AL,DX
OUT DX,AL
```

Hierin is AL een 8 bits register (een eenheid in de processor die 8 bits kan bevatten) en DX een 16 bits register. DX bevat het nummer van de gewenste I/O lijn en AL het binnengehaalde (bij IN) of het te versturen (bij OUT) byte. Bij het aansturen van chips en schakelingen in de meeteenheid worden deze instructies intensief gebruikt.

Vaak komt het in een (meet)programma voor dat we moeten wachten met verdere acties in het programma totdat een bepaald randapparaat zijn taak heeft vervuld (bijv. we moeten wachten totdat een Analoog Digitaal converter klaar is met het converteren van de waarde van het analoge signaal naar een binaire representatie ervan alvorens we deze binaire waarde kunnen uitlezen). Een vaak gebruikte methode is dan om te kijken (via een I/O lijn) naar een bepaald signaal dat informatie bevat over de status van het randapparaat en te wachten totdat dit signaal aangeeft dat het randapparaat klaar is ("polling"). Schematisch gezien kan dat als volgt lopen:

```
      { start actie randapparaat }
WACHT:IN AL,IOLYN                    [ haal status binnen via IOLYN ]
      { heeft AL de goede waarde ? }[ bijv 0 = niet klaar; 1 = wel klaar ]
      NEE: spring terug naar WACHT
      JA : ga verder met programma
```

Een nadeel bij het gebruik van deze methode bij processen die snel doorlopen moeten worden is dat het pollen teveel tijd kan kosten (het doorlopen van de loop kost tijd !). Daarom wordt bij tijdkritische processen gebruik gemaakt van zgn. interrupts. Essentieel hierin is dat bijv. het status-signaal van het randapparaat via een zgn. interrupt lijn verbonden wordt met

een speciale chip in de PC , de zgn (INTEL 8259A) interrupt controller. Deze chip biedt mogelijkheden tot het aansluiten van 8 van dergelijke interrupt lijnen die genummerd worden met IRQ0 ... IRQ7. De interrupt controller komt in actie op het moment dat de spanning op de interrupt lijn omslaat van 0 naar +5 Volt. Op dat moment signaleert de interrupt controller dit aan de processor in de PC (interrupt request) die vervolgens het proces dat hij op dat moment uitvoert onderbreekt (interrupt) en verder gaat met het uitvoeren van een stuk programma (interrupt service routine) dat zich in het geheugen van de PC bevindt. Na beeindiging van dit programma hervat de processor het onderbroken proces. Het zal duidelijk zijn dat we met deze methode veel sneller kunnen reageren op externe gebeurtenissen. Om dit proces mogelijk te maken moet er natuurlijk wel het een en ander geregeld worden. Ten eerste kunnen er bij meerdere aangesloten interrupt-lijnen interrupts tegelijk gevraagd worden door randapparatuur. Daarom hebben de interrupt-lijnen IRQ0 .. IRQ7 ieder een eigen prioriteit (IRQ0 de grootste, IRQ7 de laagste). Indien IRQ0 en IRQ4 dus simultaan gevraagd worden dan zal de interrupt controller eerst IRQ0 doorlaten en dan pas IRQ4. Verder is het natuurlijk niet gewenst dat gedurende de interruptafhandeling een andere interrupt doorgelaten wordt. De interrupt controller blokkeert derhalve interrupts totdat de actuele interrupt afgehandeld is (hetgeen we in de interrupt service routine aan de interrupt controller moeten laten weten). Omdat er meerdere interrupt-lijnen aanwezig zijn moet ergens vastgelegd worden wat de bij een bepaalde interrupt behorende interrupt service routine is. Dit gebeurt door middel van de zgn. interrupt vector. Dit is een op een speciale plaats in het geheugen van de PC opgeborgen adres dat aangeeft waar de bij een bepaalde interrupt horende interrupt service routine te vinden is. Het operating system van de PC biedt mogelijkheden om deze interrupt vectoren naar wens in te stellen, waardoor we in staat zijn een eigen routine te benutten voor het afhandelen van een bepaalde interrupt. Voorts is er de mogelijkheid om aan de interrupt controller op te geven welke interrupts wel en welke niet doorgelaten mogen worden. Dit gebeurt door het sturen van een byte naar de interrupt controller command port die gekoppeld is aan een gereserveerde I/O lijn (I/O lijn 0020H). Dit byte (het zgn. interrupt mask) heeft in de standaard PC configuratie de volgende vorm:

10111100 = BCH

Daarmee wordt vastgelegd dat IRQ6, IRQ1 en IRQ0 doorgelaten mogen worden (de bits 6, 1 en 0 zijn 0 gemaakt). Deze interrupts zijn gekoppeld aan:

- de timer interrupt die ervoor zorgt dat de real-time klok van de PC 18.2 keer per seconde geupdate wordt (IRQ0)
- de keyboard interrupt die ervoor zorgt dat ingetoetste karakters doorgespeeld worden naar de PC (IRQ1)
- de floppy disk interrupt die gebruikt wordt bij schrijven naar en lezen van de floppy disk (IRQ6)

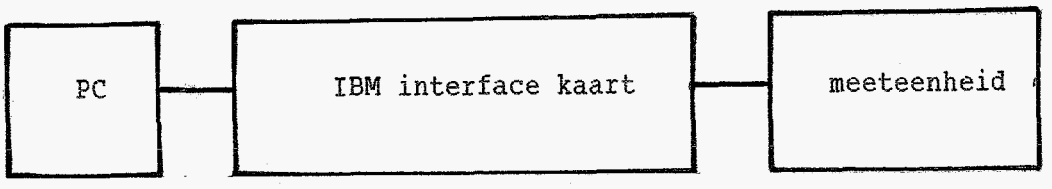
Speciaal voor gebruikerstoepassingen is nu IRQ2 gereserveerd. Bij tijdkritische toepassingen kan dit problemen opleveren omdat immers IRQ0 en IRQ1 een grotere prioriteit hebben. Door echter het interrupt mask aan te passen kunnen we bereiken dat

- IRQ2 geaccepteerd wordt
- overige interrupts niet geaccepteerd worden

Dit is te bereiken door het interrupt mask 11111011 = FBH te gebruiken (waarmee de real-time klok wordt stilgezet, het toetsenbord inactief wordt en de floppy disk niet meer te gebruiken is !).
In de ontwikkelde software voor het meten van signalen wordt IRQ2 gebruikt. We komen hier later op terug.

Hoofdstuk 4 Hardware van het meetsysteem

De hardware van het meetsyteem kunnen we grofweg indelen in 3 eenheden:



Als centraal sturings en verwerkingsstation gebruiken we de PC (512Kb RAM,20 Mb Winchester hard disk, 360Kb floppy disk, 8087 numerieke coprocessor, monochrooom grafisch beeldscherm, seriele en parallelle interface, 4.7 MHz clock)

De IBM interface kaart is aangesloten via een van de vrije slots in de PC in de zgn I/O mapped mode (volgens IBM specificaties worden hiervoor de I/O lijnen 0300H-0319H gebruikt). Deze interface kaart wordt alleen gebruikt voor het aanspreekbaar maken (chip select) van componenten op de feitelijke meeteenheid en het data-transport tussen meeteenheid en PC.

De meeteenheid bevat alle relevante chips en schakelingen voor het feitelijke meet- en stuurproces (zie fig 4.1). De meest essentiele onderdelen hierin zijn 2 INTEL 8255A parallelle poorten (PP0 en PP1) en de INTEL 8253 timerchip. Hieraan gekoppeld zijn onder meer de AD-converter (ADC), DA-converters (DAC's) en digitale in/uitgangen.
We zullen de precieze opbouw van dit systeem nu verder bespreken.

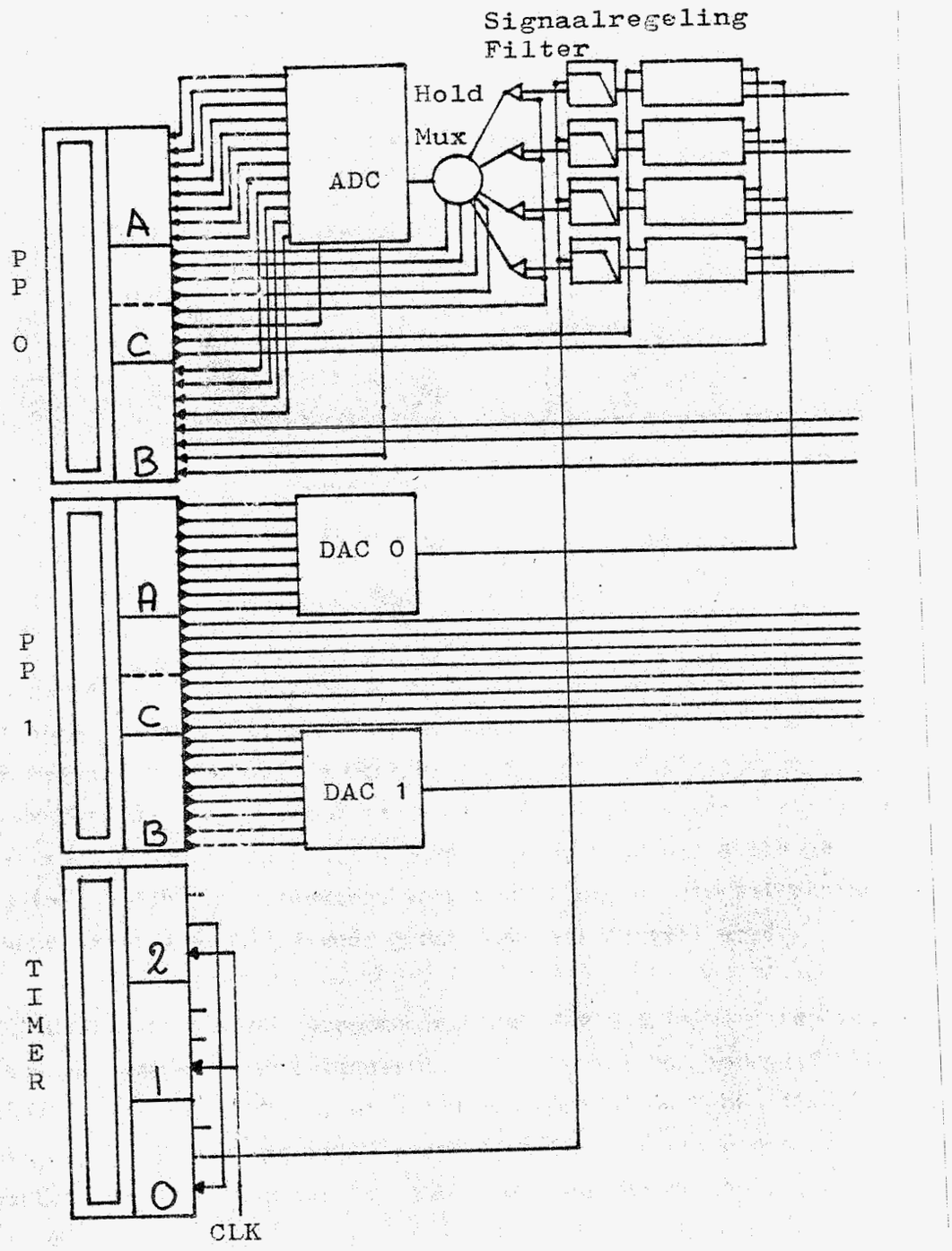
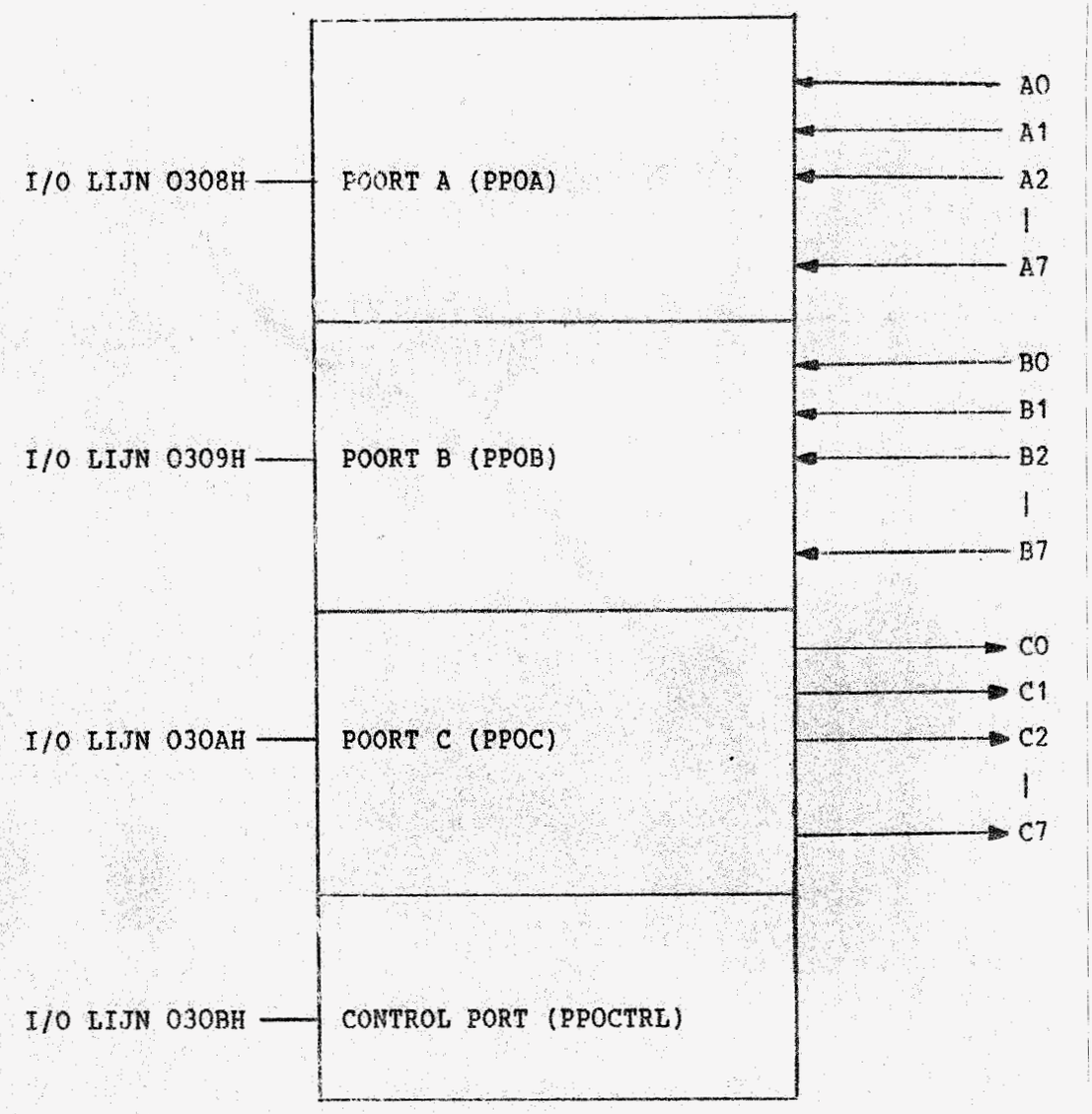


Fig. 4.1. Overzicht hardware meeteenheid

par. 4.1 parallelle poort PPO

Parallelle poort PPO (zie [2]) dient als centraal stuurorgaan voor het feitelijke meetproces. In het huidige meetsysteem moet de poort in de zgn mode 0 gezet worden (met control word #10, zie [2]) hetgeen vanuit een programma dient te gebeuren.
In deze mode functioneert de poort als volgt:



De poort is ingedeeld in 3 groepen (registers of ports) van 8 digitale in/uitgangen (met ieder als mogelijke waarde 0 of 1) nl: PPOA met 8 ingangen, PPOB met 8 ingangen en PPOC met 8 uitgangen. Wat betekent dit nu ?

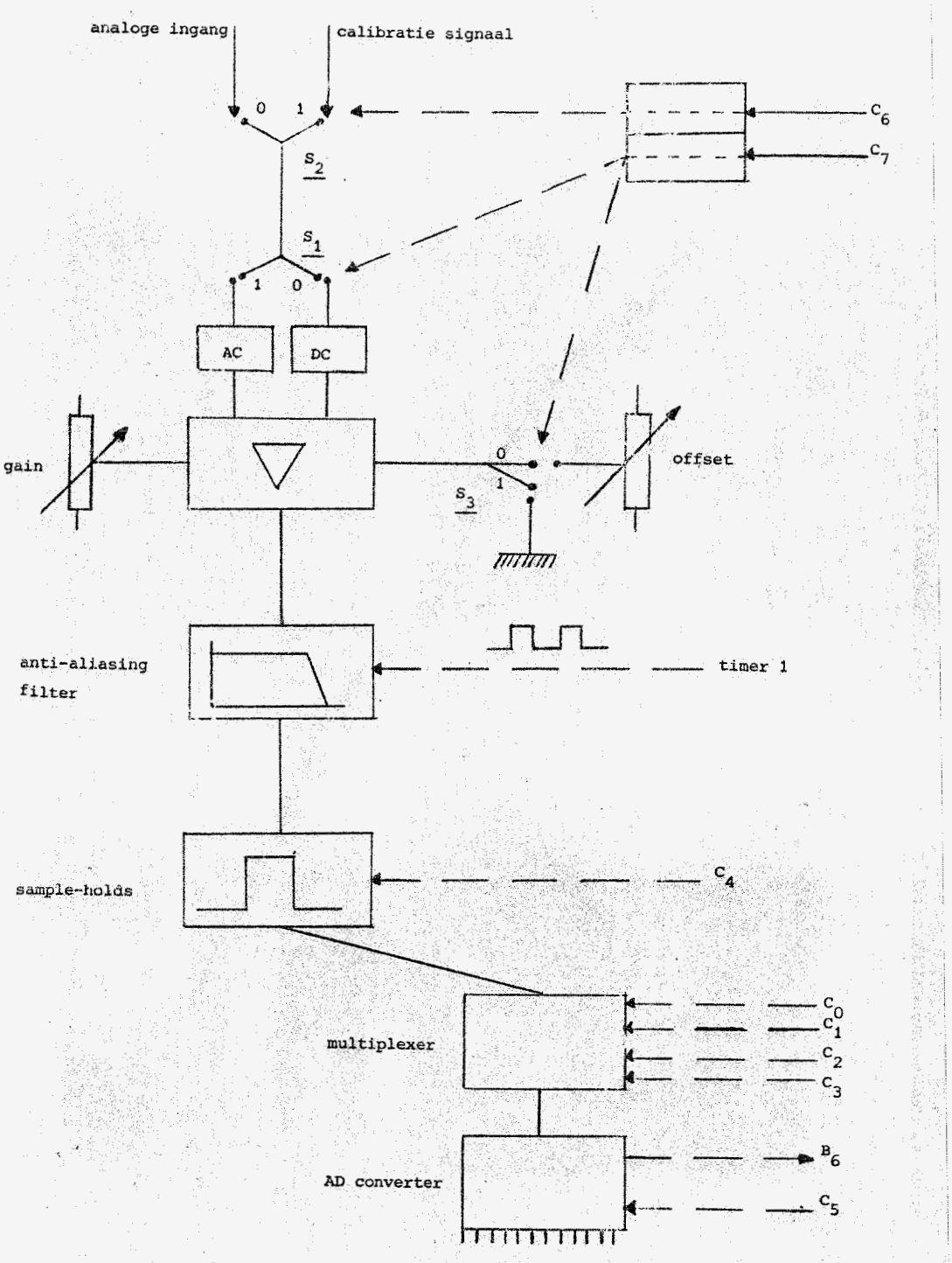
PPOA is aangesloten op I/O lijn 0308H. Mbv de basisinstructie IN kunnen we nu in register AL een byte ophalen waarvan de bits precies overeen komen met de waardes A7 ... A0. Daarmee is dan precies te achterhalen wat de waarde was van de signalen A7 ... A0. PPOC is geschakeld als output port. Mbv de basisinstructie OUT kunnen we dan via I/O lijn 030AH een byte sturen zodat de waarden C7 ... C0 gezet worden op de waarden van de bits van het byte, m.a.w. het sturen van het byte 01000111 naar PPOC zorgt ervoor dat de lijnen C7 ... C0 op de respectieve waarden 0,1,0,0,0,1,1 en 1 gezet worden waarbij 0 overeenkomt met 0 Volt en 1 met +5 Volt. Hiermee ontstaat dus de mogelijkheid tot het aansturen van randapparatuur. Het op deze wijze aansturen van randapparatuur via I/O lijnen heet wel het werken in de I/O-mapped mode (in tegenstelling tot de zgn. memory mapped mode waarbij de randapparatuur gekoppeld is aan geheugenplaatsen. Het schrijven naar of het lezen van een geheugen plaats komt dan overeen met resp OUT en IN instructies). Door de keuze van de mode van de parallelle poort leiden IN's op PPOC en OUT's op PPOA of PPOB tot onzinnige resultaten.
De control port PPOCTRL van PPO (I/O lijn 030BH) kunnen we gebruiken voor :

- het initialiseren van PPO. Voor de door ons gebruikte mode komt dit neer op OUT 030BH,92H
- het individueel setten (1 maken) of resetten (0 maken) van de lijnen aangesloten op PPOC. Willen we bijvoorbeeld lijn C3 op 1 zetten dan kan dit door naar PPOCTRL het volgende byte te sturen:

  0 0 0 0 1 0 0 1 = 09H

  waarbij 0 0 0 0 default is, 1 0 0 (= 4) het nummer van de te setten lijn en 1 de gewenste waarde, ofwel OUT 030BH,09H.

Aangesloten op PPO is de elektronica die het feitelijke meetproces verzorgt en zorg draagt voor de signaalregeling. Voor elk van de maximaal 4 analoge ingangen (ofwel kanalen met nummers 0,1,2,3) ziet dit er als volgt uit:



In de volgende paragrafen zullen we de werking en aansturing van de diverse elementen bespreken.

par. 4.2. calibratie-switch

Schakelaar S2 (de calibratie switch) kent 2 standen (0 of 1). In stand 0 wordt een extern (te meten) signaal doorgeleid, in stand 1 wordt dit signaal echter afgekoppeld en wordt het zgn calibratie-signaal doorgeleid. Dit signaal wordt gegenereerd door DAC0 die gekoppeld is aan parallelle poort PP1 (zie par. 4.14.). Dit calibratie-signaal dient ervoor om automatische calibratie van het meetsysteem mogelijk te maken. De stand van switch S2 kan nu bestuurd worden door het setten of resetten van lijn C6 van PPOC. Hierbij geldt de volgende afspraak:

lijn C6 = 0 = extern signaal als ingaand signaal
1 = calibratiesignaal van DAC0 als ingaand signaal

par. 4.3. AC/DC switch

Voor het wegfilteren van het DC-nivo van het doorgeleide signaal is een highpass filter (kantelfrequentie 0.03 Hz) aanwezig. Dit filter kan in of uitgeschakeld worden mbv. schakelaar S1. In stand 0 is dit filter uitgeschakeld (het DC-nivo wordt niet weggefilterd) in stand 1 gebeurt dit wel. De stand van switch S1 kan nu bestuurd worden door het setten of resetten van lijn C7 van PPOC, waarbij de afspraak is

lijn C7 = 0 = AC/DC filter uitgeschakeld
1 = AC/DC filter ingeschakeld

par. 4.4. gain en offset

T.b.v. het versterken/verzwakken of het verschuiven van het DC nivo van het ingangssignaal zijn een tweetal potentiometers aangebracht. Een potentiometer dient als gain (versterking 0 ... 5), de ander als offset (verschuiving -10 ... + 10 Volt van DC nivo). Deze potentiometers worden met de hand bediend.

par. 4.5. aangelegde koppelingen tussen calibratie-switch, AC/DC-switch en offset

Wanneer het DC-filter ingeschakeld is, dan is het niet zinvol dat de offset-schakeling nog functioneert. Daarom is een switch S3 aangebracht die in stand 1 de offset uitschakelt en deze in stand 0 van C7 actief laat. Verder zal bij een ingeschakelde calibratiespanning het DC-filter uitgeschakeld moeten zijn, omdat een constante calibratiespanning anders weggefilterd wordt tot 0 Volt. Daarom zijn de standen van de switches S1,S2 en S3 gekoppeld aan de mogelijke waarden van de lijnen C6 en C7 van PPOC:

lijn C6	lijn C7	S1	S2	S3
0	0	0	0	0
1	0	0	1	0
0	1	1	0	1
1	1	0	1	1

par 4.6. anti-aliasing filters

Ter voorkoming van aliasing bij het meten van een signaal is voor elk kanaal een laagdoorlaat (lineaire fase) anti-aliasing filter ingebouwd (48 dB/octaaf) (zie [3] en appendix E). Om oversturing van het filter te voorkomen moet het ingaande signaal van het filter altijd tussen -5 en +5 Volt liggen. De kantelfrequentie (-3dB punt) van het filter kan worden ingesteld mbv. de frequentie van een symmetrische blokgolf die we genereren mbv de INTEL 8253 timerchip (zie par. 4.18.). Om een kantelfrequentie van f_c Hz te krijgen moet een blokgolf met een frequentie van $100*f_c$ Hz aangeboden worden.
Voor kantelfrequenties beneden 1 Hz is een goede werking van de filters niet gegarandeerd (instabiel gedrag).

par. 4.7. sample-hold schakelingen

Het gefilterde signaal wordt vervolgens aangeboden aan zgn. sample-hold schakelingen (zie [4]) die aangebracht zijn om het simultaan meten van de 4 kanalen mogelijk te maken. De stand van deze sample-holds wordt gestuurd door lijn C4 van PPOC:

lijn C4 = 0 = sample hold inactief --> signaal wordt doorgeleid
1 = sample hold actief --> signaal wordt bevroren op de waarde die het had bij het setten van lijn C4

par. 4.8. multiplexer

De signalen van de sample-holds worden doorgeleid naar de multiplexer (zie [5]) ,die ervoor zorgt dat een van de signalen wordt doorgeleid naar de Analoog Digitaal converter (ADC). De keuze van dit signaal wordt bepaald door de waarden van lijnen C0 ... C3 van PPOC:

C0,C1 en C2 : kanaalkeuze
C3 : multiplexer enable/disable

met

kanaal	C0	C1	C2
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0

en

C3 = 0 = multiplexer disable
1 = multiplexer enable

Het kiezen van een kanaal dient als volgt te gebeuren:

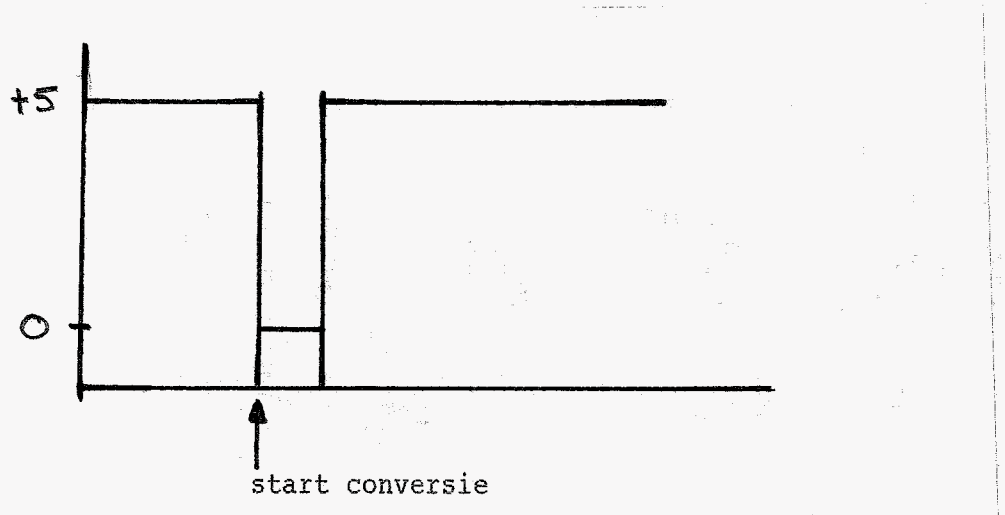
- zet C3 op 0 (multiplexer disable)
- zet C0 ... C2 op de juiste waarde (kies kanaal)
- zet C3 op 1 (multiplexer enable)

par. 4.9. Analoog Digitaal converter

Het door de multiplexer doorgeleide signaal kan nu mbv de ADC (zie [6]) gedigitaliseerd worden (ingangssignaal -5 ... +5 Volt; 12 bits conversie) N.B. de ADC is dus ook overstuurd als de anti-aliasing filters overstuurd zijn. Het converteren van de ADC wordt gestuurd door lijn C5 van PPOC waarbij de conversie als volgt gestart dient te worden:

- zet C5 op 0
- zet C5 op 1

In de tijd gezien krijgen we dan het volgende verloop van het signaal op C5:



Na ong. 35 microseconden is de ADC klaar met de conversie. We kunnen dit controleren aan de hand van de status van de ADC zoals gegeven door lijn B6 van PPOB:

lijn B6 = 0 = ADC klaar met conversie
1 = ADC bezig met conversie

Op het moment dat de ADC klaar is met de conversie worden de geconverteerde 12 bits-waarde (in de range van 0 ... 4095 ofwel 000H ... FFFH) doorgeleid naar PPO en wel als volgt:

bit	11	10	9	8	7	6	5	4	3	2	1	0
lijn	B3	B2	B1	B0	A7	A6	A5	A4	A3	A2	A1	A0

met bit 11 als het MSB en bit 0 als het LSB. Hierbij geldt bijv. dat -5 Volt het bitpatroon 000000000000 = 000H (= 0) en 0 Volt het bitpatroon 100000000000 = 800H (=2048) oplevert. Voor het omrekenen van deze waarden naar Volts kunnen we de volgende uitdrukking gebruiken:

$$\text{input in Volt} = \frac{10}{4096} * \text{gedigitaliseerde waarde ADC} - 5 \text{ Volt}$$

De geconverteerde waarde kunnen we als volgt ophalen via PPO:

```
IN LOWBYTE,0308H      { haal laagste 8 bits op PPOA }
IN HIGHBYTE,0309H     { haal hoogste 4 bits en 4 dummy bits op PPOB }
```

Omdat de lijnen B4 ... B7 in dit geval een onzinnige betekenis hebben moeten de 4 hoogste bits van HIGHBYTE 0 gemaakt worden (bijv met een logische AND instructie) De waarde van de ADC kunnen we dan in een word opbergen (bijv de variabele ADCVAL) volgens:

```
ADCVAL = LOWBYTE + 256*HIGHBYTE ( 0 <= ADCVAL <= 4095 )
```

par. 4.10. externe trigger

Lijn B7 van PPOB is gereserveerd voor de externe trigger. Indien we van buitenaf iets willen laten sturen (bijv start van een meting) dan kan de waarde van B7 (0 of 1) hiertoe gebruikt worden. De betekenis van de waarde van B7 is afhankelijk van de toepassing. Als lijn B7 0 is dan is de spanning op de triggeringang +5 Volt en als B7 1 is dan is de spanning 0 Volt.

par. 4.11. reserve input-lijnen

De lijnen B4 en B5 van PPOB zijn gereserveerd voor mogelijke uitbreidingen indien deze extra input-lijnen vereisen.

par. 4.12. Resume van de functies van PPO

Samengevat heeft de parallelle poort PPO dus de volgende functies:

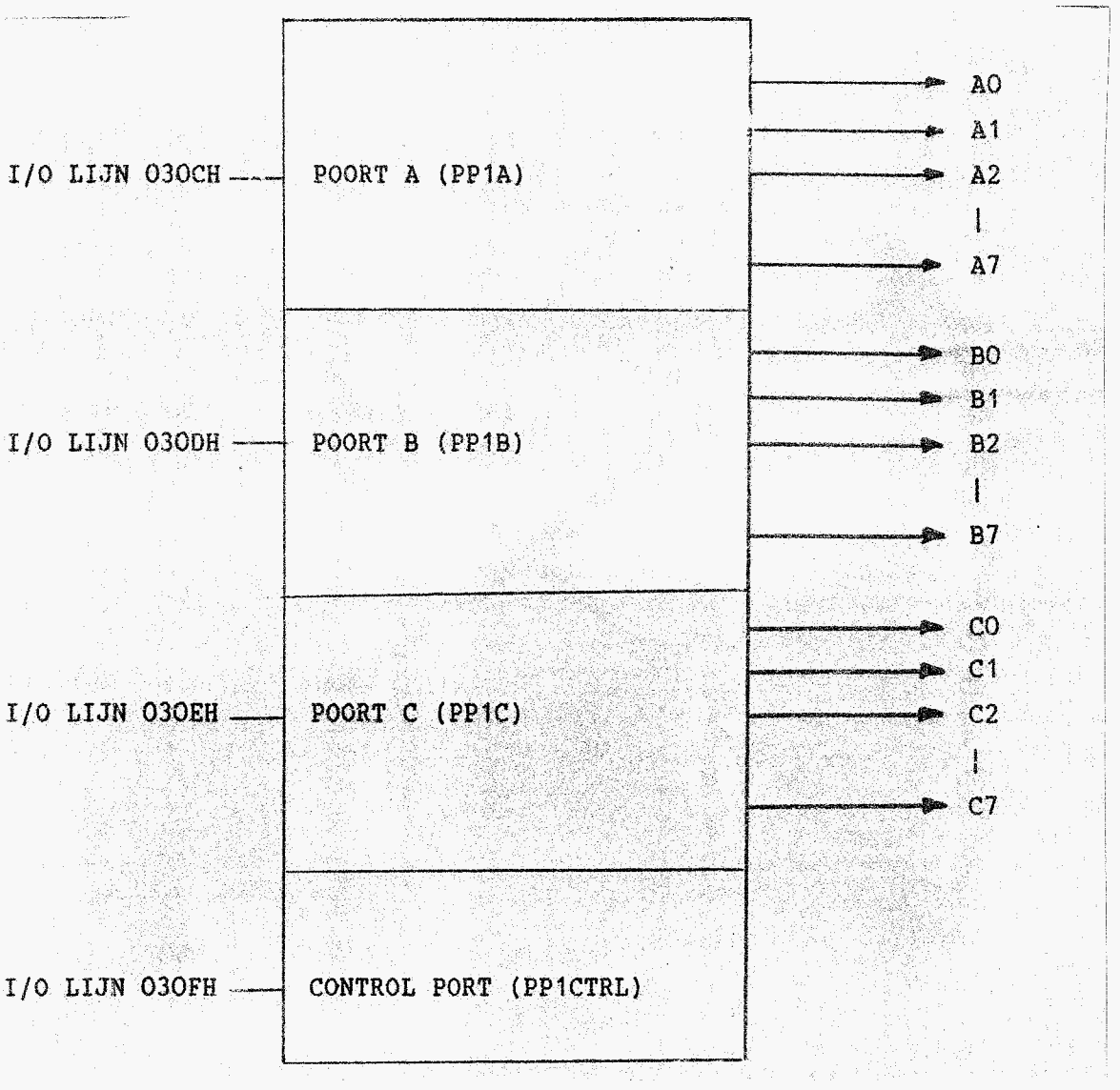
- PPOA
  * bevat de 8 laagste bits van de ADC na conversie (lijnen A0 ... A7)

- PPOB
  * bevat de 4 hoogste bits van de ADC na conversie (lijnen B0 ... B3)
  * bevat status van de ADC (lijn B6)
  * bevat externe trigger (lijn B7)
  * bevat reserve input bits (lijnen B4 en B5)

- PPOC
  * bevat aansturing multiplexer (lijnen C0 ... C3)
  * bevat sturing sample-holds (lijn C4)
  * bevat start AD conversie (lijn C5)
  * bevat sturing calibratie-switch (lijn C6)
  * bevat sturing AC/DC filter (lijn C7)

par. 4.13. parallelle poort PP1

Parallelle poort PP1 is ingericht als stuurmiddel voor externe apparatuur en het mogelijk maken van de automatische calibratie. Ook deze poort wordt in mode 0 geplaatst (echter met control word #0) zdd de 24 lijnen als volgt ingedeeld zijn:



(Het initialiseren in deze mode kan gebeuren door OUT 030FH,80H).

De aan de verschillende lijnen gekoppelde hardware wordt in de volgende paragrafen besproken.

par. 4.14. DAC0

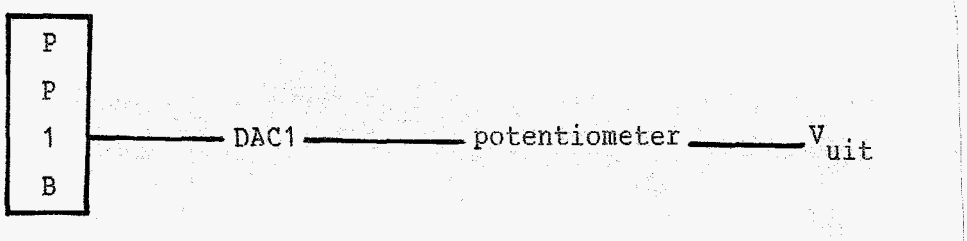
De 8 bits Digitaal Analoog converter DAC0 (zie [7]) dient voor het genereren van een mogelijkheid tot automatische calibratie. Het setten/resetten van de lijnen A0 ... A7 van PP1A komt overeen met het genereren van een bepaalde spanning afgeleverd door DAC0 die indien de calibratie-switch S2 (zie par. 4.2) corect is ingesteld dient als ingang voor het meetsysteem (in andere gevallen heeft DAC0 geen functie). Het sturen van een byte met waarde DAVAL (0 <= DAVAL <= 255) via I/O lijn 030CH komt overeen met het genereren van een spanning VCAL ter grootte van

$$VCAL = -\frac{10}{256} * DAVAL - 5 \text{ Volt} \quad (-5 <= VCAL <= +5 \text{ Volt })$$

(bijv OUT 030CH,80H zorgt voor een calibratiespanning van 0 Volt)

par. 4.15. DAC1

De 8 bits DA converter DAC1 (zie [7]) is bedoeld voor het genereren van stuurspanningen van externe apparaten en werkt analoog aan DAC0, zij het dat de geleverde spanning nu extern beschikbaar is.



V_{uit} kan nu nog geregeld worden m.b.v de potentiometer (versterking 0 ... 2) zodanig dat $-10 <= V_{uit} <= +10$ Volt).

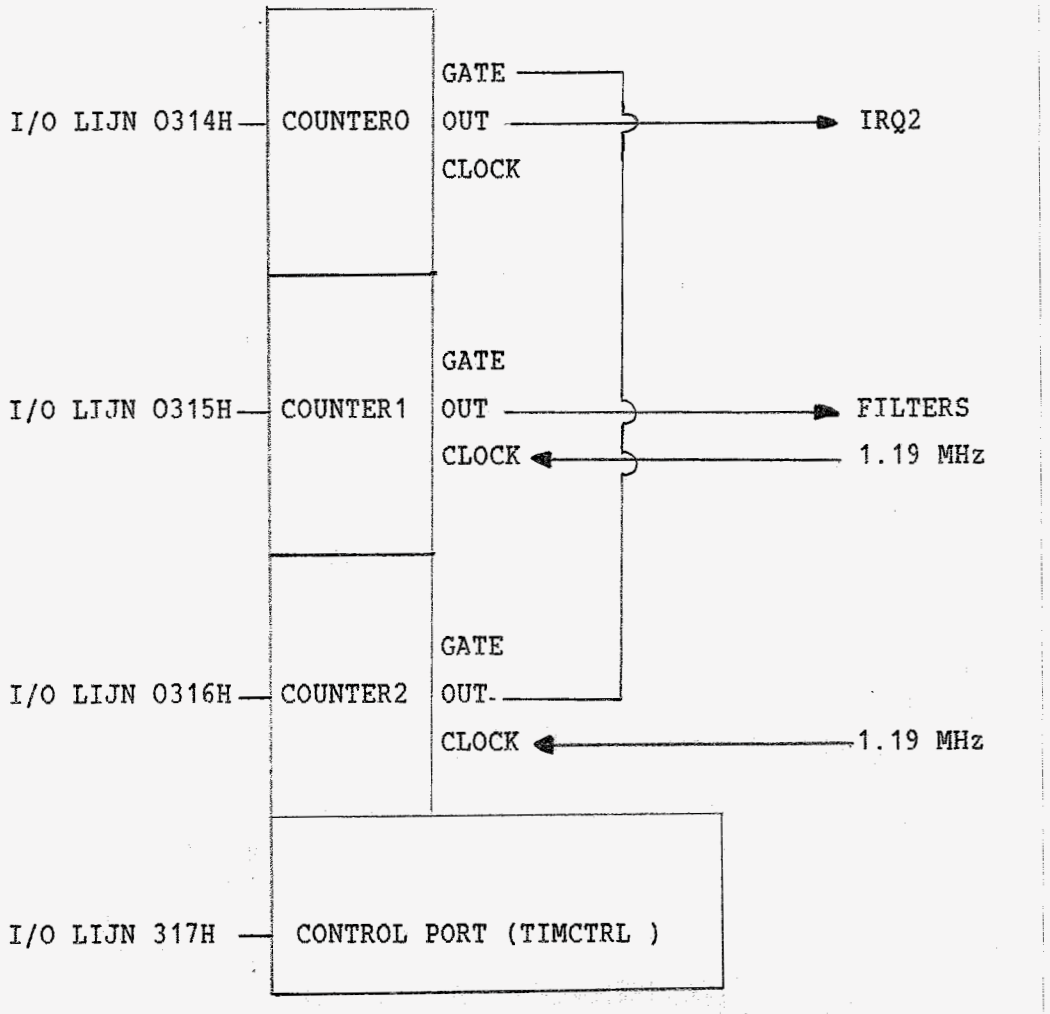
par. 4.16. digitale stuurlijnen

De 8 output-lijnen van PP1C zijn bedoeld voor het aansturen van externe apparatuur mbv. digitale lijnen (lijnen met een signaal van 0 of +5 Volt).

Deze zijn gelocaliseerd op PP1C (lijnen C0 ... C7) en kunnen bijvoorbeeld gebruikt worden voor het sturen van stappenmotoren. De functie van deze lijnen is vrij te kiezen.

par. 4.17. de 8253 timer/counter

De Intel 8253 timer/counter chip ([8]) bevat 3 timers/counters die als volgt aangesloten zijn:



De 3 counters worden in mode 3 (square wave generator) gezet door het sturen van een geschikt byte naar de de control port.
Wat is nu de functie van deze counters ? Daartoe kijken we eerst wat de counters doen in mode 3. We nemen counter 1 als voorbeeld. Via de I/O lijn 0315H kunnen we (in 2 stappen) een word naar deze counter sturen (die in feite een 16 bits register is en derhalve waarden aan kan nemen tussen 0 en 65535). Stel we laden zo de counter initieel met de waarde 20000 (ofwel we laden de counter met het deeltal 20000). Dan zal de counter vervolgens zijn stand telkens met 2 verlagen op het moment dat de clock-ingang hoog (+5 Volt) wordt. Als clock-ingang voor counter 1 gebruiken we een blokgolf van

1.19 MHz, waardoor de counter dus ook met een frequentie van 1.19 MHz aftelt. Indien de tellerstand dan op 10000 gekomen is dan wordt de uitgang "out" van de counter hoog gemaakt. Vervolgens gaat het aftelproces door totdat de tellerstand 0 is bereikt. Op dat moment wordt de uitgang "out" laag (0 Volt) gemaakt en wordt automatisch het oorspronkelijke deeltal geladen en begint het proces opnieuw. Daardoor ontstaat op de uitgang "out" een blokgolf met een frequentie 1.19 MHz/20000. I.h.a. zal dan de uitgang "out" een blokgolf zijn met frequentie 1.19 MHz/deeltal. Het deeltal moet hierbij liggen tussen 4 en 65535, waardoor blokgolven met frequenties gemaakt kunnen worden die op discrete nivos liggen tussen 18.1 en 297500 Hz.

par. 4.18. counter 1

Counter 1 wordt gebruikt als omschreven in par 4.17. voor het genereren van een blokgolf met frequentie $100*f_c$ waarbij f_c de gewenste waarde is van de kantelfrequentie van de anti-aliasing filters. Daartoe wordt counter 1 geladen met het deeltal dat volgt uit

$$\text{deeltal counter 1} = 1.19*10^{**}6/(100*f_c)$$

Omdat dit deeltal tussen 4 en 65535 moet liggen en omdat de filters een instabiel gedrag vertonen voor kantelfrequenties beneden 1 Hz wordt daarmee de bruikbare range van kantelfrequenties beperkt tot de range 1 ... 2975 Hz.

par. 4.19. counters 0 en 2

De counters 0 en 2 worden samen gebruikt voor het genereren van een bepaalde interrupt frequentie (de frequentie waarmee interrupts worden gegenereerd) t.b.v het meetproces en het creeren van wachtlussen. Zij zijn daartoe op een speciale manier geschakeld. De "out" van counter 2 dient namelijk als clock-ingang van counter 0 (aangesloten via de "gate" van counter 0). Door nu counter 0 te laden met het deeltal I0 en counter 2 met het deeltal I2 zal de "out" van counter 0 een deeltal zijn met frequentie f_i = 1.19 MHz/(I0*I2), ofwel (omdat 4 <= I0 <= 65535 en 4 <= I2 <= 65535) de interrupt frequentie kan ingesteld worden tussen de grenzen 0.0003 Hz en 74000 Hz. De "out" van counter 0 is nu aangesloten op de interrupt lijn IRQ2 van de INTEL 8259A

interrupt controller. Indien gewenst kunnen we de actuele stand van elke counter opvragen zonder dat het telproces gestoord wordt. Hiertoe moet de betreffende counter gelatched worden door het geven van een geschikt commando aan de timer control port. Daarna kunnen we mbv 2 IN's de betreffende counter uitlezen (eerst laagste byte dan het hoogste)

par. 4.20. resume van de hardware

In het voorafgaande hebben we gezien hoe de hardware van het systeem globaal is opgebouwd en hoe het in principe bestuurd kan worden. De mogelijkheden van het meetsysteem liggen daarmee vast:

- 4 analoge ingangen (kanalen 0 t/m 3)
- 1 analoge uitgang (DAC1)
- 10 digitale stuursignalen (lijnen B4 en B5 van PPO, lijnen C0 ... C7 van PP1)
- 1 trigger signaal (lijn C7 van PPO)
- signaal regeling mbv gain, offset en AC/DC filter
- automatische calibreermogelijkheid (DAC0)
- programmeerbare kantelfrequentie anti-aliasing filters (counter 1)
- timingsmogelijkheden m.b.v. interrupts (counter 0, counter 2 en IRQ2)

Verder zijn de analoge signalen die aan de ADC toegevoerd worden (na de signaalregelingseenheid van de meeteenheid doorlopen te zijn) beschikbaar d.m.v. een BNC aansluiting op de meeteenheid (hetgeen van nut kan zijn bij mogelijke controles).
In het hoofdstuk 5 zullen we zien wat de software inhoudt die ontwikkeld is voor het besturen en meten.

Hoofdstuk 5 Gewenste mogelijkheden en opzet van de software

Bij het ontwikkelen van de software voor het sturen van de meeteenheid is uitgegaan van een aantal overwegingen:

- de software moet te gebruiken zijn door personen die redelijk thuis zijn in Fortran zonder specifieke kennis van machinetaal (hij/zij hoeft zelf niet te kunnen programmeren in assembler tenzij een speciale toepassing dit vereist).

- een groot aantal mogelijkheden van het meetsysteem moeten vanuit Fortran kunnen worden benut

- de software moet duidelijk gestructureerd en gedocumenteerd zijn

- de software moet een redelijk snel meetproces mogelijk maken

Juist de laatste eis leidt er toe dat bepaalde tijdkritische zaken in assembler geprogrammeerd moeten worden omdat deze anders niet gerealiseerd kunnen worden. Een en ander heeft er toe geleid dat de volgende opzet gekozen is:

Fortran programma

Fortran routine bibliotheek XX.LIB

Assembler routine bibliotheek YY.LIB

Assembler macro bibliotheek ZZ.LIB

De assembler routine bibliotheek YY.LIB bevat een aantal routines die opgebouwd zijn uit elementaire handelingen die ondergebracht zijn in de assembler macro-bibliotheek ZZ.LIB. Deze macros zijn duidelijk gedefinieerde

stukjes assembler code die een specifieke taak vervullen (zij verzorgen de feitelijke aansturing van de onderdelen van de meeteenheid). Door deze macros systematisch te gebruiken kunnen de routines in YY.LIB overzichtelijk en snel opgebouwd worden. Op basis van de routine bibliotheek YY.LIB is een Fortran routine bibliotheek XX.LIB ontwikkeld waarmee de aansturing van de meeteenheid vanuit Fortran geregeld kan worden. In hoofdstukken 6,7 en 8 zullen we nader ingaan op de specifieke inhoud van de verschillende bibliotheken.

Hoofdstuk 6 De macro bibliotheek ZZ.LIB

De macro-bibliotheek ZZ.LIB (zie appendix A) bestaat uit zgn macros. Dit zijn elementaire stukjes assembler die tijdens het compileren van de routine bibliotheek YY.LIB ingevoegd worden daar waar in een routine uit YY.LIB verwezen wordt naar zo'n macro.
De bibliotheek ZZ.LIB kunnen we op dit moment indelen in 3 gedeelten nl:

- macros voor de feitelijke aansturing van de meeteenheid
- macros voor de interface Fortran-assembler
- macros voor het initialiseren, afhandelen en beeindigen van interrupts van counter 0 (IRQ2)

par. 6.1. macros voor het aansturen van de meeteenheid

Hieronder vallen de volgende macros:

- ZZSTIM initialiseren van de timers/counters in mode 3
- ZZWTM0 laden van counter 0 met een deeltal
- ZZWTM1 laden van counter 1 met een deeltal
- ZZWTM2 laden van counter 2 met een deeltal
- ZZRTM0 uitlezen van de stand van counter 0
- ZZSPPT initialiseren van de parallelle poorten PP0 en PP1
- ZZCHA2 selecteren van een kanaal mbv. de multiplexer
- ZZADST starten AD conversie
- ZZ35MU checken van status ADC totdat conversie beeindigd is
- ZZADIN uitlezen van de ADC
- ZZDA0 sturen van een byte naar DAC0
- ZZDA1 sturen van een byte naar DAC1
- ZZSCAL set switch S2 op calibratie (stand 1)
- ZZRCAL reset switch S2 (stand 0)
- ZZSHLD set sample-holds op actief
- ZZRHLD reset sample-holds
- ZZSDCF set DC-filter op actief
- ZZRDCF reset DC-filter

- ZZSTRG haal status trigger
- ZZSTRT start volgende proces of free-run of als externe trigger ok of als gebruiker dit bepaalt (mbv keyboard)
- ZZPCHAN vult variabelen met de goede waarde t.b.v. aansturing multiplexer tijdens samplen

par. 6.2. macros voor de interface Fortran-assembler

De routine bibliotheek YY.LIB bestaat uit routines die of parameters aangeleverd krijgen ter verwerking of parameters door geven naar het Fortran hoofdprogramma. Om deze parameteroverdracht te regelen zijn een aantal macros opgenomen in ZZ.LIB:

- ZZHEAD initialisatie van segmentregisters bij binnenkomst assembler routine
- ZZFOOT resetten segmentregisters bij verlaten assembler routine
- ZZLPAR haal een parameter op die overgedragen wordt vanuit Fortran
- ZZSPAR berg een parameter op die overgedragen moet worden naar Fortran
- ZZINAR initialiseer segmentregisters voor lezen uit/laden in arrays (bedoeld voor het opbergen en ophalen van resp DA en AD waarden)
- ZZSTAD berg een AD waarde op in daarvoor bedoeld array
- ZZGTDA haal een DA waarde op uit een daarvoor bedoeld array
- ZZLDSP haal een aantal getalwaarden op uit een array (bedoeld voor het binnenhalen van parameters die het meetproces definieren)

par. 6.3. macros voor interrupt afhandeling

T.b.v het interruptmechanisme zijn de volgende macros opgenomen in de ZZ.LIB:

- ZZGIRQ2 haal bestaande interrupt vector behorende bij IRQ2
- ZZRIRQ2 reset oude interrupt vector bij IRQ2
- ZZSINT set interrupt vector IRQ2 t.b.v. samplen
- ZZEINT enable interrupts IRQ2

- ZZDINT disable interrupts IRQ2
- ZZWAIT set interrupt vector t.b.v. wachtlussen

Hoofdstuk 7 De assembler routine bibliotheek YY.LIB

De routine bibliotheek YY.LIB (zie appendix B) bestaat uit assembler routines die aanroepbaar zijn vanuit Fortran en opgebouwd zijn uit macros uit de macro-bibliotheek ZZ.LIB.
We zullen deze routines kort toelichten:

- YYINIT vanuit fortran: CALL YYINIT
  YYINIT dient als initialisatie routine voor de meeteenheid. Er gebeurt hetvolgende:
  * initialisatie van counters en parallelle poorten
  * initialisatie van anti-aliasing filters op kantelfrequentie 100 Hz
  * DAC0 krijgt als uitgang 0 Volt
  * DAC1 krijgt als uitgang 0 Volt
  * sample-holds worden gereset
  * switch S2 komt op stand 0 (niet-calibreren)
  * de DC-filters worden uitgeschakeld
  * de interrupt vector van IRQ2 wordt geinitialiseerd t.b.v. het samplen
  * counter 0 en counter 2 worden geinitialiseerd voor een interrupt frequentie (sample frequentie) van 300 Hz

- YYFINI vanuit fortran: CALL YYFINI
  YYFINI dient als reset routine voor de meeteenheid. Er gebeurt bijna hetzelfde als in de routine YYINIT zij het dat de interrupt vector van IRQ2 op de normale PC waarde wordt teruggezet.

- YYSFRE vanuit fortran: CALL YYSFRE(IDIV0,IDIV2)
  YYSFRE laadt counter 0 met deeltal IDIV0 en counter 2 met deeltal IDIV2 (IDIV0 en IDIV2 zijn integer*2 waarden) zodat de interrupt frequentie wordt gezet op de waarde 1.19MHz/(IDIV0*IDIV2). Hiermee kan de sample-frequentie ingesteld worden.

- YYFFRE vanuit fortran: CALL YYFFRE(IDIV1)
  YYFFRE laadt counter 1 met deeltal IDIV1 (integer*2 waarde), zodat de kantelfrequentie van de anti-aliasing filters wordt gezet op de waarde 1.19MHz/(100*IDIV1)

- YYSDCF vanuit fortran: CALL YYSDCF
  YYSDCF activeert de DC-filters

- YYRDCF vanuit fortran: CALL YYRDCF
  YYRDCF schakelt de DC-filters uit

- YYSCAL vanuit fortran: CALL YYSCAL
  YYSCAL schakelt switch 2 in stand 1 (calibratie aan)

- YYRCAL vanuit fortran: CALL YYRCAL
  YYRCAL schakelt de calibratie-stand uit

- YYDA0 vanuit fortran: CALL YYDA0(IDA0)
  YYDA0 stuurt de laagste 8 bits van de integer*2 waarde IDA0 naar DA converter DAC0

- YYDA1 vanuit fortran: CALL YYDA1(IDA1)
  YYDA1 stuurt de laagste 8 bits van de integer*2 waarde IDA1 naar DA converter DAC1

- YYSTRG vanuit fortran: CALL YYSTRG(ISTRG)
  YYSTRG levert de status van de externe trigger in de integer*2 waarde ISTRG. Als triggerlijn op 0 dan wordt ISTRG 0 en als triggerlijn op 1 dan wordt ISTRG 128

- YYMEAS vanuit fortran: CALL YYMEAS(ISPC,IDABUF,IADBUF,IERR)
  De routine YYMEAS is bedoeld als meetroutine. Voor de functie van de routine YYMEAS zie par. 7.1.

- YYWAIT vanuit fortran: CALL YYWAIT(IWAIT)
  De routine YYWAIT is bedoeld voor het genereren van wachtlussen met een lengte van IWAIT*0.01 sec. Voor de werking van de routine YYWAIT zie par. 7.2.

par. 7.1. de routine YYMEAS

De assembler routine YYMEAS is bedoeld als feitelijke meetroutine en wordt vanuit Fortran aangeroepen met CALL YYMEAS(ISPC,IADBUF,IDABUF,IERR).
De parameters van deze routine zijn als volgt gedefinieerd:

ISPC

ISPC is een integer*2 array met 25 waarden die als volgt in te vullen zijn (sommige zijn default waarden die verder echter niet gebruikt worden tijdens het meetproces. Zij zijn ingevoerd om compatibiliteit te krijgen met de software ontwikkeld in [9]):

```
ISPC(1) = -1          : (default)
ISPC(2) = 0           : geen DA-output naar DAC1 tijdens de meting
          1           : wel DA-output naar DAC1 tijdens meting ( en wel
                        achtereenvolgens de waarden in het array IDABUF )
ISPC(3) = 1 ... 4     : aantal te meten kanalen
ISPC(4) = 0 ... 3     : nummer 1e te meten kanaal
ISPC(5) = 0 ... 3     : nummer 2e te meten kanaal ( indien gewenst )
ISPC(6) = 0 ... 3     : nummer 3e te meten kanaal ( indien gewenst )
ISPC(7) = 0 ... 3     : nummer 4e te meten kanaal ( indien gewenst )
ISPC(8) = 0           : start meting free-run
          1           : start meting mbv externe trigger
          2           : start meting door gebruiker ( keyboard )

ISPC(9) = 0           : indien externe trigger gewenst dan start meting als
                        externe trigger hoog ( +5 volt )
          128         : indien externe trigger gewenst dan start meting als
                        externe trigger laag ( 0 volt )
```

```
ISPC(10)= 0          : DC-filter uit ( wordt niet gebruikt )
          1          : DC-filter aan ( wordt niet gebruikt )
ISPC(11)= 0          : calibratie-mode uitgeschakeld ( wordt niet gebruikt )
          1          : calibratie-mode ingeschakeld ( wordt niet gebruikt )
ISPC(12)             : niet gebruikt
:
:
ISPC(19)             : niet gebruikt
ISPC(20)             : mantissa en exponent van tijdsinterval tussen twee
ISPC(21)             : samples in microseconden ( niet gebruikt )
                       ( f_s = 1.19*10**6/(ISPC(20)*10**ISPC(21)) Hz.)
ISPC(22)             : mantissa en exponent van tijdsinterval tussen 2 pulsen
ISPC(23)             : voor anti-aliasing filters in micro-seconden ( niet
                       gebruikt )
                       ( f_c =  1.19*10**4/(ISPC(22)*10**ISPC(23))
ISPC(24) = 1         : default ( wordt niet gebruikt )
ISPC(25) = 1 . 32766: aantal in te nemen samples per kanaal ( aantal in te
                       nemen samples per kanaal * aantal kanalen <= 32766 )
```

IDABUF

IDABUF is een integer*2 array met maximaal 32766 waarden die indien ISPC(2)=1 successievelijk naar DAC1 gestuurd worden tijdens de metingen met een frequentie die gelijk is aan de samplefrequentie. Elk van de DA-waarden moet liggen in de range 0 ... 255 (zie par. 4.14.).

IADBUF

IADBUF is een integer*2 array met maximaal 32766 waarden waarin de gemeten waarden van de ADC komen te staan (zie verder in deze paragraaf).

IERR

IERR is een integer*2 variabele die dient voor mogelijke foutmeldingen.

Na afloop van het meetproces zal IERR 0 zijn als er geen fouten gevonden zijn. IERR wordt 1 bij fouten in de specificaties (ISPC(3) <= 0 of ISPC(25) <= 0).

Hoe werkt de routine nu ? Stel dat we 3 kanalen willen meten n.l. de kanalen 0, 1 en 3 (In andere gevallen geldt een analoge uitleg). We gaan er vanuit dat eerder in het programma de interrupt (sample) frequentie is ingesteld (bijv. mbv de routine YYSFRE) en dat de drie kanalen simultaan gemeten moeten worden met deze frequentie. In de routine YYMEAS wordt er dan voor gezorgd dat de interrupts IRQ2 door de 8259A interrupt controller doorgelaten worden. In de routine YYINIT is er dan reeds voor gezorgd dat de interrupt service routine behorende bij het meetproces ingevuld is. Deze interrupt service routine is te vinden in de macro ZZSINT. Na binnenkomst in de interrupt service routine gebeurt er dan hetvolgende (zie figuur 7.1):

1 - zet de sample-holds op actief (bevries alle ingangen om ervoor te zorgen dat we alle signalen simultaan meten)
2 - selecteer kanaal 0 mbv. de multiplexer
3 - start de ADC
4 - stuur de volgende waarde uit IDABUF naar DAC1 indien ISPC(2) = 1
5 - berg het voorgaand ingenomen sample op in array IADBUF (een sample van kanaal 3)
6 - wacht tot de ADC klaar is
7 - lees de ADC uit (sample van kanaal 0)
8 - selecteer kanaal 1 mbv. de multiplexer
9 - start ADC
10 - berg het voorgaand ingenomen sample op in array IADBUF (een sample van kanaal 0)
11 - wacht tot de ADC klaar is
12 - lees ADC uit (sample van kanaal 1)
13 - selecteer kanaal 3 mbv. de multiplexer
14 - start ADC
15 - berg het voorgaand ingenomen sample op in array IADBUF (sample van kanaal 1)
16 - wacht tot de ADC klaar is
17 - lees ADC uit (sample van kanaal 3)
(Op dit moment zijn alle te meten kanalen eenmaal bemonsterd)

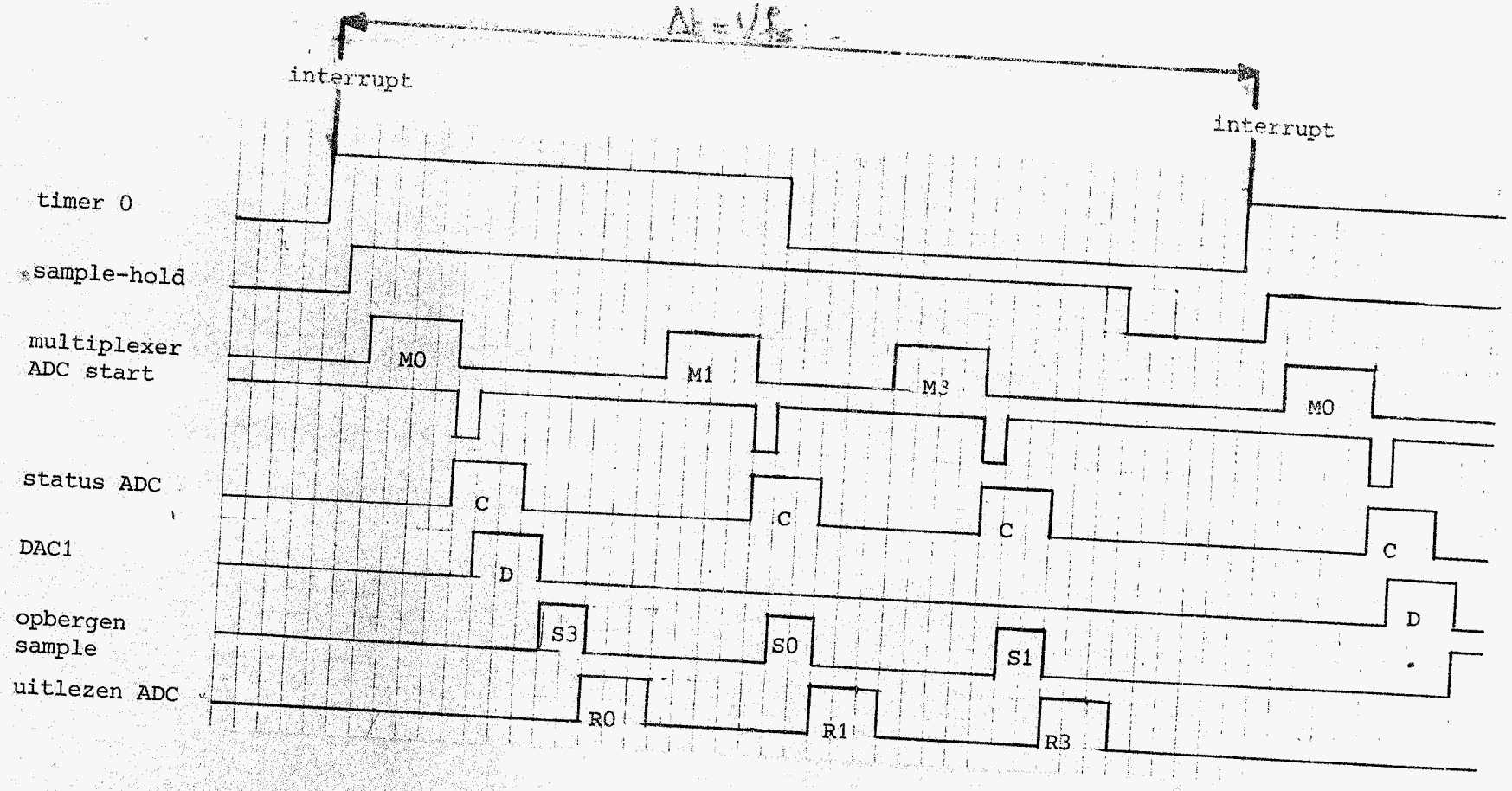


Fig. 7.1 Timingsdiagram voor de routine YYMEAS

M0 : selecteren kanaal 0 mbv multiplexer (M1 idem voor kanaal 1 etc.)

C : ADC bezig met conversie (status bit B6 hoog)

D : aansturen DAC1 indien gewenst

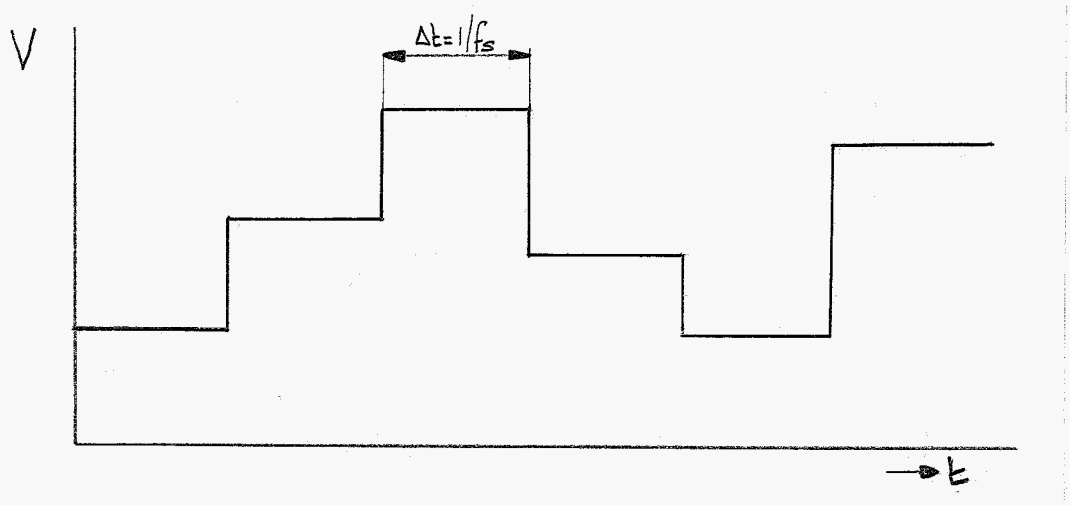
S3 : opbergen van sample van kanaal 3 (S0 idem van kanaal 0 etc.)

R0 : uitlezen ADC voor sample van kanaal 0 (R1 idem voor kanaal 1 etc.)

18 - controleer of elk kanaal zo vaak gemeten is als gegeven is in ISPC(25)
zo nee dan stap 21
zo ja dan:
19 - berg het voorgaand ingenomen sample op (sample van kanaal 3)
20 - schakel dan de interrupts IRQ2 uit door sturen van commando naar de interrupt controller
21 - reset sample holds
22 - deel de interrupt controller mee dat de interrupt afgehandeld is

We merken het volgende op:

- de tijd gedurende de conversie door de ADC wordt zoveel mogelijk benut voor het opbergen van voorgaand ingenomen samples (hetgeen natuurlijk de allereerste keer niet goed gaat omdat er dan nog niet gemeten was) en het aansturen van DAC1 indien dit gewenst is. Omdat de snelheid van het sturen naar DAC1 gelijk is aan de samplefrequentie kunnen we daarmee een signaal genereren dat een maximale frequentie $0.5*f_s$ bevat. In de tijd gezien zal het signaal van de DAC1 er i.h.a als volgt uitzien:



- de maximale sample-frequentie wordt bepaald door:
  * de tijd die de interrupt controller en de processor nodig hebben om te komen tot de feitelijke interrupt afhandeling (circa 20 micro-seconden)
  * de conversietijd van de ADC (35 micro-seconden)
  * de tijd die nodig is voor het instellen van de multiplexer
  * de tijd die nodig is voor het aansturen van DAC1

* de tijd die nodig is voor het opbergen van een sample in het array IADBUF

De maximale samplefrequentie is experimenteel bepaald (door het controleren van een aantal signalen op een oscilloscoop) en is afhankelijk van het aantal te meten signalen en de mogelijke aansturing van DAC1:

maximale sample frequentie [Hz]	DAC1 niet gestuurd	DAC1 wel gestuurd
1 kanaal meten	6750	6450
2 kanalen	4200	4150
3 kanalen	3050	3000
4 kanalen	2500	2440

Omdat er maximaal 32766 samples ingenomen kunnen worden gelden dan daarbij de volgende maximale meettijden bij gebruik van de maximale sample frequentie:

maximale meettijd [s] (maximum snelheid)	DAC1 niet gestuurd	DAC1 wel gestuurd
1 kanaal meten	4.8	5.0
2 kanalen	3.9	3.9
3 kanalen	3.6	3.6
4 kanalen	3.3	3.3

(overigens is het wel mogelijk het maximale aantal samples te vergroten door een aantal kleine aanpassingen in de routines. Daarmee kan het aantal in te nemen samples vergroot worden tot bijv. 128000. Wel zal dan de maximaal mogelijke samplefrequentie iets gereduceerd worden ivm. problemen bij het ophalen van DA en opbergen van AD waardes.)

Na afloop van het meetproces zal het array IADBUF dus de waarden van de ADC bevatten. De oorspronkelijke waardes van de ADC liggen in de range 0 ... 4095 maar in het array IADBUF liggen die in de range -2048 ... 2047 (om

aansluiting te krijgen met de software ontwikkeld in [9]). Het ingangssignaal van de ADC kan dan bepaald worden uit:

$$\text{input} = -\frac{10}{4096} * \text{ADC waarde.}$$

De samples van de kanalen zijn successievelijk opgeborgen. De allereerste waarde is echter niet een sample maar het aantal ingenomen samples per kanaal, dus IADBUF(1)=aantal samples/kanaal. Bij het meten aan N kanalen zullen de samples van kanaal K zoals gegeven in ISPC(3+J) (1 <= J <= 4) staan op de plaatsen IADBUF(1+(I-1)*N+J), I=1...ISPC(25).

par. 7.2. de routine YYWAIT

De assembler routine YYWAIT is bedoeld voor het genereren van wachtlussen en wordt aangeroepen met CALL YYWAIT(IWAIT). De wachttijd in sec volgt uit IWAIT*0.01 waarbij IWAIT een integer*2 variabele >=0 is zodat de wachttijd kan liggen tussen 0.00 en 327.0 sec.
Het wachten gebeurt dmv. de interrupts IRQ2 die hiervoor tijdelijk op een frequentie van 100 Hz gezet worden. Voor het wachten is een speciale interrupt service routine (zie macro ZZWAIT) die het aantal interrupts telt totdat de waarde IWAIT bereikt is. Ivm allerlei omschakelprocedures is de wachttijd nauwkeurig tot op 0.5 milliseconde.

Hoofdstuk 8 De fortran routine bibliotheek XX.LIB

De routines in de bibliotheek YY.LIB kunnen weliswaar direct vanuit fortran aangeroepen worden in een (meet)programma maar zij zijn daarvoor niet alle direct geschikt ivm controle van de parameters etc. Daarom is op basis van de bibliotheek YY.LIB een bibliotheek XX.LIB (zie appendix C) ontwikkeld die als uitgangspunt kan dienen bij het ontwikkelen van (meet)programmatuur. Tevens wordt in deze bibliotheek voorzien in een aantal arrays (zie de common-blocks XX.CMN in appendix C) die door de gebruiker gebruikt kunnen worden voor het opbergen van gegevens (bijv. AD en DA waarden, specificaties etc.). Bij het opzetten van de routines in XX.LIB is er min of meer vanuit gegaan dat een gebruikers programma gebruik maakt van de volgende arrays uit XX.CMN:

- XXAD voor het opbergen van de AD waarden
- XXDA voor het opbergen van de DA waaarden

Verder zijn de common blocks een aantal arrays opgenomen die door de gebruiker gebruikt kunnen worden indien hij de gemeten AD waarden op wil bergen in een file die compatibel is met die die gegenereerd worden in [9] (de arrays XXISP, XXCSP, XXRSP, XXDSP). Hoe deze arrays precies gevuld dienen te worden staat vermeld in appendix F.

Om de gebruiker in staat te stellen de gemeten data in een plotje op de PC weer te geven zijn een tweetal eenvoudige plotroutines toegevoegd die gebruik maken van de grafische bibliotheek HALO (zie [10]). Een aantal parameters die de vorm van het plaatje sturen zijn opgenomen in het common block XX3 en kunnen indien gewenst door de gebruiker veranderd worden.

We bespreken nu in het kort de routines uit XX.LIB omdat in de listing in appendix 3 voldoende commentaar opgenomen is om de werking van de routines te begrijpen.

- XXINIT vanuit fortran: CALL XXINIT
  XXINIT dient als initialisatieroutine voor de meeteenheid en een aantal parameters (zie ook de routine YYINIT)

- XXFINI vanuit fortran: CALL XXFINI

  XXFINI dient als beeindigingsroutine voor de meeteenheid (zie ook de routine YYFINI)

- XXSAFR vanuit fortran: CALL XXSAFR(FREQ,NCHAN,IDAYN,IMANT,IEXP)
  real*4 FREQ
  integer*2 NCHAN,IDAYN,IMANT,IEXP

  XXSAFR dient voor het insteleen van de samplefrequentie waarbij rekening gehouden wordt met het aantal kanalen, de mogelijke wens voor DA-output tijdens het meten omdat deze limieten stellen aan de maximale sample frequentie (zie par. 7.1.). De parameters van de routine kunnen we interpreteren in termen van de specificaties voor de metingen. Stel we een sample frequentie van 300 Hz. instellen bij een gegeven aantal kanalen en DA-behoefte zoals ingevuld in de specificaties (array XXISP) volgens par. 7.1. Dan kunnen we XXSAFR aanroepen met:

```
FREQ=300.0
CALL XXSAFR(FREQ,XXISP(3),XXISP(2),XXISP(20),XXISP(21))
```

  Na afloop heeft FREQ dan de exact ingestelde waarde voor de sample frequentie die van de verlangde kan afwijken omdat we de counters 0 en 2 moeten laden met integer deeltallen en omdat er limieten zijn aan de sample frequentie.

- XXFS vanuit fortran: FREQ=XXFS(IMANT,IEXP)
  real*4 function XXFS
  integer*2 IMANT,IEXP

  XXFS levert als waarde de actuele sample frequentie zoals volgt uit

```
XXFS = 1.19E6/(IMANT*10**IEXP)
```

  ofwel indien we de specificaties in array XXISP gebruiken:

```
FREQ = XXFS(XXISP(20),XXISP(21))
```

- XXFCOF vanuit fortran: CALL XXFCOF(FREQ,IMANT,IEXP)
  real*4 FREQ
  integer*2 IMANT,IEXP

  XXFCOF stelt de cut-off frequentie van de anti-aliasing filters in op de waarde FREQ (of althans zo dicht mogelijk erbij). In combinatie met de specificaties kan de routine gebruikt worden als

```
FREQ=100.0
CALL XXFCOF(FREQ,XXISP(22),XXISP(23))
```

  om een kantel frequentie van 100 Hz. in te stellen

- XXFC vanuit fortran: FREQ=XXFC(IMANT,IEXP)
  real*4 fucntion XXFC
  integer*2 IMANT,IEXP

  XXFC levert de kantelfrequentie van de filters zoals bepaald door IMANT en IEXP:

```
XXFC = 1.19E4/(IMANT*10**IEXP)
```

  ofwel bij gebruikmaking van de specificaties in XXISP

```
FREQ = XXFC(XXISP(22),XXISP(23))
```

- XXDAO vanuit fortran: CALL XXDAO(IVOLT)
  integer*2 IVOLT

  XXDAO stuurt het laagste byte van IVOLT naar DACO

- XXBDAO vanuit fortran: IVOLT=XXBDAO(VOLT)
  integer*2 function XXBDAO
  real*4 VOLT

  XXBDAO levert de integer*2 waarde op waarvan het laagste byte naar DACO gestuurd moet worden om een uitgangsspanning VOLT te krijgen

- XXVDA0 vanuit fortran: VOLT=XXVDA0(IVOLT)
  real*4 function XXVDA0
  integer*2 IVOLT

  XXVDA0 levert real*4 waarde (voltage) dat op DAC0 zou staan indien we er het laagste byte van IVOLT naar toe zouden sturen.

- XXDA1 vanuit fortran: CALL XXDA1(IVOLT)
  integer*2 IVOLT

  XXDA1 stuurt het laagste byte van IVOLT naar DAC1

- XXBDA1 vanuit fortran: IVOLT=XXBDA1(VOLT)
  integer*2 function XXBDA1
  real*4 VOLT

  XXBDA1 levert de integer*2 waarde op waarvan het laagste byte naar DAC1 gestuurd moet worden om een uitgangsspanning VOLT te krijgen

- XXVDA1 vanuit fortran: VOLT=XXVDA1(IVOLT)
  real*4 function XXVDA1
  integer*2 IVOLT

  XXVDA1 levert real*4 waarde (voltage) dat op DAC1 zou staan indien we er het laagste byte van IVOLT naar toe zouden sturen.

- XXVAD vanuit fortran: ADCVAL=XXVAD(IVALUE)
  real*4 function XXVAD
  integer*2 IVALUE

  XXVAD levert de ingangsspanning van de ADC bij een gemeten integer waarde IVALUE (-2048 <= IVALUE <= 2047).

- XXSDCF vanuit fortran: CALL XXSDCF(ISTAT)
  integer*2 ISTAT

  XXSDCF set (ISTAT=1) of reset (ISTAT = 0) de DC-filters

- XXSCAL vanuit fortran: CALL XXSCAL(ISTAT)
  integer*2 ISTAT

  XXSCAL set (ISTAT = 1) of reset (ISTAT = 0) de calibratie-switch

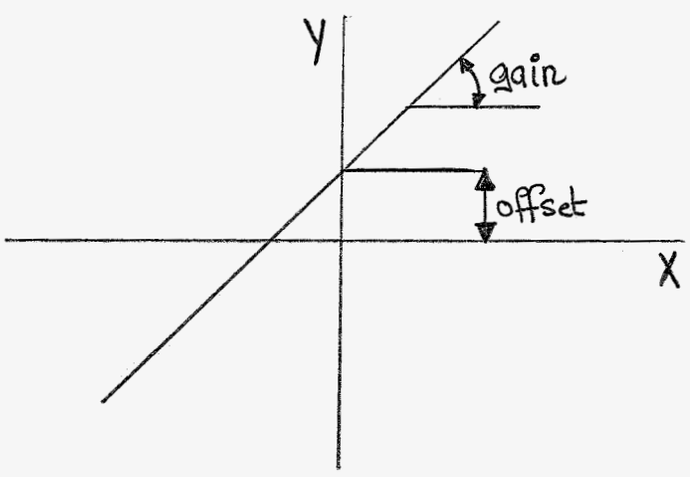
- XXWAIT vanuit fortran: CALL XXWAIT(TWAIT)
  real*4 TWAIT

  XXWAIT genereert een wachtlus van TWAIT seconden.

- XXCALI vanuit fortran:
  CALL XXCALI(ICHAN,GAIN,OFFSET,NCAL,NSAMP,IERR)
  integer*2 ICHAN,NCAL,NSAMP,IERR
  real*4 GAIN,OFFSET

  XXCALI is bedoeld voor het automatisch calibreren van de meeteenheid. Uitgangspunt is dat voor een gelijkspanning X aan de ingang van kanaal ICHAN de uitgangsspanning Y wordt gegeven door

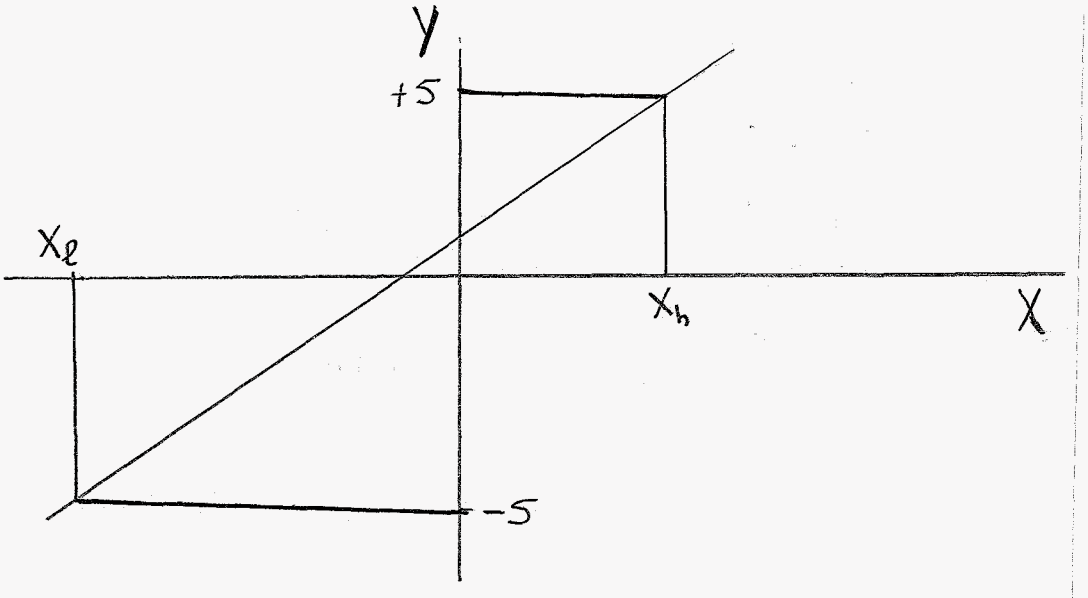
  Y = offset + gain * X



In de routine XXCALI worden gain en offset als volgt bepaald:

* de calibratie switch S_2 wordt ingeschakeld
* een calibratiespanning van -5 Volt wordt op DAC0 gezet
* er worden NSAMP samples genomen. Indien hierbij overflow van de ADC opgetreden is dan wordt de calibratie spanning met 0.0096 Volt verhoogd en herhaalt zich dit proces totdat geen overflow optreedt. Indien geen overflow optreedt dan is de dan ingestelde calibratie spanning de laagst mogelijk calibratie spanning X_1.

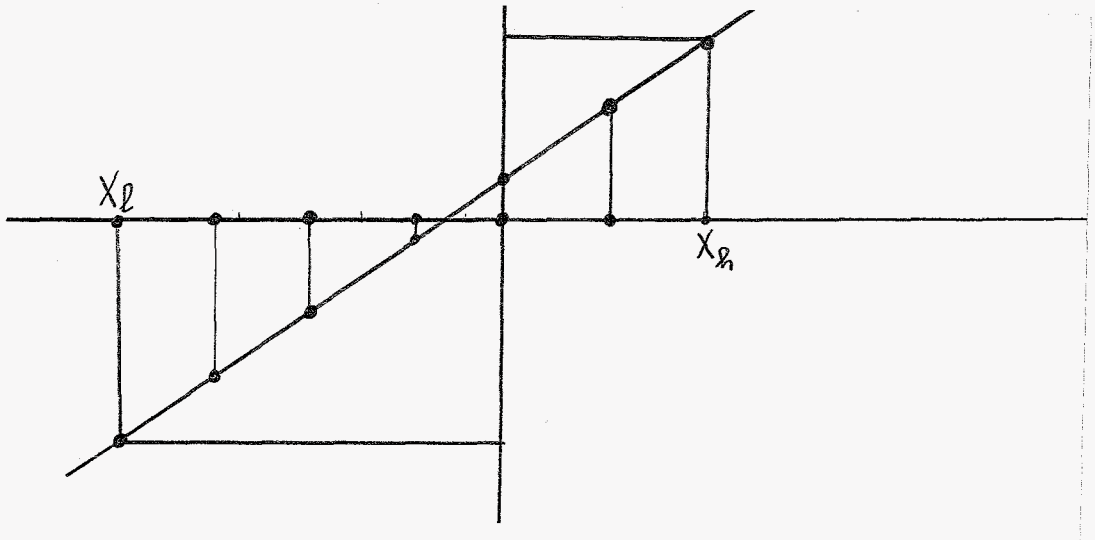
* dit wordt ook gedaan door te beginnen bij een calibratiespanning van +5 Volt (en deze bij overflow te verminderen met 0.096 Volt) tot de grootst mogelijke calibratiespanning X_h bepaald is.



* het traject X_l - X_h wordt ingedeeld in NCAL-1 gelijke stukken zdd dat we NCAL calibratiepunten krijgen. Bij ieder calibratiepunt hoort een calibratiespanning X_j

$$X_j = X_l + (X_h - X_l)* j / (NCAL - 1) \qquad \{ j = 0 \ldots NCAL - 1 \}$$

* voor ieder X_j worden NSAMP samples genomen en gemiddeld tot een waarde Y_j :



* op basis van de gemeten Y_j en de ingestelde X_j wordt mbv een lineaire regressie de gain en offset bepaald uit Y_j = offset + gain * X_j
  (j = 1 ... NCAL)
  (uiteraard moet hierbij gelden: NCAL >= 2, NSAMP >= 1, X_l < + 5 Volt en X_h > X_l)

- XXMEAS vanuit fortran: CALL XXMEAS(ISPC,IERR)
  integer*2 ISPC(25),IERR

XXMEAS is bedoeld als feitelijke meetroutine en werkt mbv. de routine YYMEAS. ISPC bevat de specificaties voor de meting (zie par. 7.1.) waarvan alleen de elementen ISPC(2) ... ISPC(9) en ISPC(25) van belang zijn. Na afloop van de meting zijn de AD waarden geplaatst in het array XXAD. Indien overflow opgetreden is dan wordt dit gemeld in de parameter IERR. Indien ISPC(2) = 1 (DA gewenst tijdens meting) dan moeten de DA waarden staan in het array XXDA.

- XXPLAD vanuit fortran: CALL XXPLAD(IFIRST,NP)
  integer*2 IFIRST,NP

XXPLAD plot een verzameling AD waarden nl de waarden XXAD(IFIRST) ... XXAD(IFIRST+NP). De waardes worden omgerekend naar Volts en getekend als in onderstaande figuur voor een testmeting:

- XXPLDA vanuit fortran: CALL XXPLDA(IFIRST,NP)
  integer*2 IFIRST,NP

  XXPLDA doet het zelfde als XXPLAD echter nu werkend op het array XXDA
  (plot DA waarden)

Literatuurlijst

[1] Software ontwikkeling voor een universeel P.C. gestuurd meetsysteem, G. Teurlinx, WFW 86.025, juni 1986.

[2] INTEL 8255A-5 programmable interface, INTEL Peripheral Design Handbook

[3] MF10CN Universal Dual Switched Capacitor Filter, National Semiconductor

[4] AD582K sample-holds, Analog Devices Databook

[5] AD7501 multiplexer, Analog Devices Databook

[6] AD574 analog-digital converter, Analog Devices Databook

[7] AD DAC08 digital-analog converter, Analog Devices Databook

[8] INTEL 8253 programmable interval timer, INTEL Peripheral Design Handbook

[9] Een P.C. gestuurd meerkanaals meetsyteem voor dynamische metingen: opbouw, werking en gebruik. L. Dortmans, K. Koekkoek, WFW 85.030, mei 1985

[10] HALO graphics primitives functional description manual, Media Cybernetics Inc.,1982

Appendix A ZZ.LIB

```
;#############################################################################
;
;  THE FOLLOWING MACROS SERVE FOR INTERFACE BETWEEN FORTRAN AND
;  ASSEMBLER
;
;
;
;  WE ASSUME A STACK CONSISTING OF:
;
;  BP--->       SAVEBP  DW              ;SAVE OF BASE POINTER
;               SAVERET DD              ;SAVE RETURN ADRESS
;               PARN    DD              ;ADRESS OF PARAMETER N
;               PARN-1  DD
;               :
;               PAR1    DD
;#############################################################################
;
PAR1    EQU     1                       ;NUMBER OF PARAMETER IN ROUTINE
PAR2    EQU     2
PAR3    EQU     3
PAR4    EQU     4
PAR5    EQU     5
PAR6    EQU     6
;
;
ZZLPAR  MACRO   VAR,PAR,NPAR            ;GET PARAMETER PAR OUT OF NPAR
        PUSH    DS                      ;ON STACK
        PUSH    BX
        LDS     BX,DWORD PTR [BP + (NPAR-PAR)*4 + 6]
        MOV     AX,[BX]
        POP     BX
        POP     DS
```

```
        MOV     VAR,AX
        ENDM
;
ZZLDSP  MACRO   PAR,NPAR                ;LOAD SPECIFICATIONS
        PUSH    SI
        PUSH    DI
        PUSH    DS
        PUSH    BX
        LDS     SI,DWORD PTR [BP+(NPAR-PAR)*4 + 6]
        MOV     DI,OFFSET SPCBUF
        CLD
        MOV     CX,LENSPC               ;COPY LENSPC WORDS
        REP     MOVSW
        POP     BX
        POP     DS
        POP     DI
        POP     SI
        ENDM
;
ZZSPAR  MACRO   VAR,PAR,NPAR            ;STORE PARAMETER PAR OUT OF
        MOV     AX,VAR                  ;NPAR ON STACK
        PUSH    DS
        PUSH    BX
        LDS     BX,DWORD PTR [BP + (NPAR-PAR)*4 + 6 ]
        MOV     [BX],AX
        POP     BX
        POP     DS
        ENDM
;
ZZINAR  MACRO   PARDA,PARAD,NPAR        ;INITIALIZE BUFFER-POINTERS
        PUSH    ES
        LES     SI,DWORD PTR [BP + (NPAR-PARDA)*4 + 6 ]
        MOV     AX,ES
        POP     ES
;
;       DABUF:[SI] POINTS TO VALUE IN DA BUFFER
```

```
;       DS MUST BE SWITCHED WHEN GETTING ONE DA VALUE ( SEE ZZGTDA )
;       ES:[DI] POINTS TO AD BUFFER
;       $$$ ES,DS,SI AND DI MAY NEVER BE CHANGED IF THE MACROS ZZGTDA
;           AND ZZSTAD MUST WORK PROPERLY AFTER USING ZZINAR $$$
;
        MOV     DABUF,AX
        LES     DI,DWORD PTR [BP + (NPAR-PARAD)*4 + 6 ]
        CLD
        ENDM
;
ZZGTDA  MACRO                           ;GET VALUE FROM DA-BUFFER
        PUSH    DS
        MOV     AX,DABUF
        MOV     DS,AX
        LODSW
        POP     DS
        ENDM
;
ZZSTAD  MACRO   VAR                     ;STORE SAMPLE IN AD-BUFFER
        MOV     AX,VAR
        AND     AH,0FH                  ;MASK 4 HIGHEST BITS
        SUB     AX,2048                 ;MAKE SURE THAT SAMPLE VALUE
                                        ;IS IN RANGE -2048..2047
        STOSW
        ENDM
;
ZZHEAD  MACRO   CSEG,DSEG,ESEG          ;SET DEFAULT SEGMENT REGISTERS
        ASSUME  CS:&CSEG,DS:&DSEG
        PUSH    BP
        MOV     BP,SP
        PUSH    DS
        MOV     AX,DSEG
        MOV     DS,AX
        PUSH    ES
        MOV     AX,ESEG
        MOV     ES,AX
```

```
        ENDM
;
ZZFOOT  MACRO                           ;PREPARE RETURN TO DOS
        POP     ES
        POP     DS
        POP     BP
        ENDM
;
;
;#########################################################################
;
;  THE FOLLOWING MACROS ARE USED FOR COMMUNICATION WITH THE INTERFACE
;
;  FIRST WE DEFINE SOME GLOBALS
;
CNTR0   EQU     0314H                   ;SAMPLE-COUNTER
CNTR1   EQU     0315H                   ;FILTER-CLOCK
CNTR2   EQU     0316H                   ;SAMPLE-CLOCK
TIMCTRL EQU     0317H                   ;TIMER-CONTROL
;
PP0A    EQU     0308H                   ;PORT A OF PP 0
PP0B    EQU     0309H                   ;PORT B OF PP 0
PP0C    EQU     030AH                   ;PORT C OF PP 0
PP0CTRL EQU     030BH                   ;CONTROLWORD PP 0
;
PP1A    EQU     030CH                   ;PORT A OF PP 1
PP1B    EQU     030DH                   ;PORT B OF PP 1
PP1C    EQU     030EH                   ;PORT C OF PP 1
PP1CTRL EQU     030FH                   ;CONTROLWORD PP 1
;
SCAL    EQU     0DH                     ;SET CALIBRATION ON
RCAL    EQU     0CH                     ;SET CALIBRATION OFF
SHOLD   EQU     09H                     ;SET HOLD ON
RHOLD   EQU     08H                     ;SET HOLD OFF
SDCON   EQU     0FH                     ;SET DC FILTER ON
SDCOFF  EQU     0EH                     ;SET DC FILTER OFF
```

```
ADCSTL  EQU     0AH             ;ADC START LOW
ADCSTH  EQU     0BH             ;ADC START HIGH
DOS     EQU     21H             ;DOS FUNCTION CALL
PRINTM  EQU     09H             ;DOS FUNCTION TO PRINT STRING
CLRKEY  EQU     0CH             ;DOS FUNCTION TO CLEAR KEYBOARD
                                ;BUFFER
INPKEY  EQU     01H             ;DOS FUNCTION TO GET AND ECHO
                                ;ANYTHING FROM KEYBOARD
;
ZZSTIM  MACRO                   ;SET TIMER MODE
        MOV     DX,TIMCTRL
        MOV     AL,36H          ;ALL COUNTERS MODE 3
        OUT     DX,AL           ;16 BIT BINARY COUNTERS
        MOV     AL,76H          ;LSB FIRST, THEN MSB FOR READ
        OUT     DX,AL           ;AND WRITE
        MOV     AL,0B6H
        OUT     DX,AL
        ENDM
;
ZZWTM0  MACRO   COUNT           ;WRITE TO TIMER 0
        MOV     DX,CNTR0
        MOV     AX,COUNT
        OUT     DX,AL
        MOV     AL,AH
        OUT     DX,AL
        ENDM
;
ZZWTM1  MACRO   COUNT           ;WRITE TO TIMER 1
        MOV     DX,CNTR1
        MOV     AX,COUNT
        OUT     DX,AL
        MOV     AL,AH
        OUT     DX,AL
        ENDM
;
ZZWTM2  MACRO   COUNT           ;WRITE TO TIMER 2
```

```
        MOV     DX,CNTR2
        MOV     AX,COUNT
        OUT     DX,AL
        MOV     AL,AH
        OUT     DX,AL
        ENDM
;
ZZRTMO  MACRO                       ;READ TIMER 0 IN AX
        MOV     DX,TIMCTRL          ;LATCH COUNTER 0
        MOV     AL,00H
        OUT     DX,AL
        MOV     DX,CNTR0            ;THEN READ
        IN      AL,DX               ;GET LSB
        MOV     AH,AL
        IN      AL,DX               ;GET MSB
        XCHG    AH,AL
        ENDM
;
ZZSPPT  MACRO                       ;SET PP MODE
        MOV     DX,PP0CTRL
        MOV     AL,92H
        OUT     DX,AL
        MOV     DX,PP1CTRL
        MOV     AL,80H
        OUT     DX,AL
        ENDM
;
ZZCHA2  MACRO                       ;SET CHANNEL FOR MUX
        MOV     DX,PP0C
        OUT     DX,AL               ;AL CONTAINS BYTE FOR MUX
                                    ;TO DISABLE MUX
        OUT     DX,AL               ;SET CHANNEL
        OR      AL,08H              ;ENABLE MUX
        OUT     DX,AL
        ENDM
;
```

```
ZZPCHAN MACRO                                ;PREPARE SET-UPS FOR MUX
        LOCAL   LMUX1,LMUX2,LMUX3,LMUX3,LMUX4,EPCHAN
LMUX1:  MOV     DUM1,OFFSET EPCHAN           ;JUMP ADRESS FOR QUIT
        MOV     AX,CHAN1                     ;GET FIRST CHANNEL
        AND     AX,003H                      ;MASK 14 HIGHEST BITS
        OR      AL,BITSTAT                   ;SET BITS FOR AC/DC,CALIBRATION
                                             ;HOLD,ADC DISABLE,MUX DISABLE
        OR      AL,10H
        MOV     MUX1,AL                      ;STORE FOR USE IN SAMPLING
        CMP     NCHAN,1                      ;MORE CHANNELS
        JG      LMUX2                        ;YES
        JMP     DUM1                         ;NO ( MUX4 IS NOT USED HERE ! )
LMUX2:  MOV     AX,CHAN2                     ;GET SECOND CHANNEL
        AND     AX,003H                      ;MASK 14 HIGHEST BITS
        OR      AL,BITSTAT                   ;SET BITS FOR AC/DC,CALIBRATION
                                             ;HOLD,ADC DISABLE,MUX DISABLE
        OR      AL,10H
        MOV     MUX2,AL                      ;STORE FOR USE IN SAMPLING
        CMP     NCHAN,2
        JG      LMUX3
        JMP     DUM1
LMUX3:  MOV     AX,CHAN3                     ;GET THIRD CHANNEL
        AND     AX,003H                      ;MASK 14 HIGHEST BITS
        OR      AL,BITSTAT                   ;SET BITS FOR AC/DC,CALIBRATION
                                             ;HOLD,ADC DISABLE,MUX DISABLE
        OR      AL,10H
        MOV     MUX3,AL                      ;STORE FOR USE IN SAMPLING
        CMP     NCHAN,3
        JG      LMUX4
        JMP     DUM1
LMUX4:  MOV     AX,CHAN4                     ;GET FOURTH CHANNEL
        AND     AX,003H                      ;MASK 14 HIGHEST BITS
        OR      AL,BITSTAT                   ;SET BITS FOR AC/DC,CALIBRATION
                                             ;HOLD,ADC DISABLE,MUX DISABLE
        OR      AL,10H
EPCHAN: MOV     MUX4,AL                      ;STORE FOR USE IN SAMPLING
```

```
        ENDM
;
ZZADST  MACRO                           ;START ADC
        MOV     DX,PPOCTRL
        MOV     AL,ADCSTL
        OUT     DX,AL
        MOV     AL,ADCSTH
        OUT     DX,AL
        ENDM
;
ZZ35MU  MACRO                           ;WAIT TILL ADC IS READY
        LOCAL HANG
HANG:   MOV     DX,PPOB
        IN      AL,DX
        TEST    AL,40H
        JNZ     HANG
        ENDM
;
ZZADIN  MACRO   HBYTE,LBYTE             ;GET HIGH BYTE ADC IN HBYTE
        MOV     DX,PPOB                 ;AND LOW BYTE IN LBYTE
        IN      AL,DX
        MOV     HBYTE,AL
        MOV     DX,PPOA
        IN      AL,DX
        MOV     LBYTE,AL
        ENDM
;
ZZDAO   MACRO                           ;OUTPUT BYTE IN AL TO DAC0
        MOV     DX,PP1A
        OUT     DX,AL
        ENDM
;
ZZDA1   MACRO                           ;OUTPUT BYTE IN AL TO DAC1
        MOV     DX,PP1B
        OUT     DX,AL
        ENDM
```

```
;
ZZSCAL  MACRO                   ;SET CALIBRATION ON
        MOV     AL,SCAL
        MOV     DX,PPOCTRL
        OUT     DX,AL
        OR      BITSTAT,40H     ;WE MUST REMEMBER THIS
        ENDM
;
ZZRCAL  MACRO                   ;SET CALIBRATION OFF
        MOV     AL,RCAL
        MOV     DX,PPOCTRL
        OUT     DX,AL
        AND     BITSTAT,0BFH    ;WE MUST REMEMBER THIS
        ENDM
;
ZZSHLD  MACRO                   ;SET HOLD
        MOV     AL,SHOLD
        MOV     DX,PPOCTRL
        OUT     DX,AL
        ENDM
;
ZZRHLD  MACRO                   ;RESET HOLD
        MOV     AL,RHOLD
        MOV     DX,PPOCTRL
        OUT     DX,AL
        ENDM
;
ZZSDCF  MACRO                   ;SET DC FILTER ON
        MOV     AL,SDCON
        MOV     DX,PPOCTRL
        OUT     DX,AL
        OR      BITSTAT,80H     ;WE MUST REMEMBER THIS
        ENDM
;
ZZRDCF  MACRO                   ;SET DC FILTER OFF
        MOV     AL,SDCOFF
```

```
        MOV     DX,PPOCTRL
        OUT     DX,AL
        AND     BITSTAT,07FH    ;WE MUST REMEMBER THIS
        ENDM
;
ZZSTRG  MACRO                   ;GET STATUS TRIGGER IN AX
        MOV     DX,PPOB         ;( AX = 0 OR 128 )
        IN      AL,DX
        AND     AL,80H
        MOV     AH,0
        ENDM
;
ZZSTRT  MACRO                   ;START MEASUREMENTS BY WAITING FOR
        LOCAL FREERUN,TRIG,EXTERN,KEYB
                                ;FOR TRIGGER IF WANTED
        MOV     DUM1,OFFSET FREERUN
        CMP     TRIGTYP,0       ;FREERUN ?
        JNE     TRIG            ;NO
        JMP     DUM1            ;YES
TRIG:   CMP     TRIGTYP,1       ;EXTERNAL ?
        JNE     KEYB
EXTERN: ZZSTRG                  ;GET TRIGGER VALUE IN AX
        CMP     AX,TRIGOK       ;OK WITH ACCEPT LEVEL ?
        JNE     EXTERN
        JMP     DUM1            ;YES
KEYB:   MOV     DX,OFFSET MES1  ;PRINT MESSAGE FOR USER
        MOV     AH,PRINTM
        INT     DOS
        MOV     AH,CLRKEY       ;CLEAR KEYBOARD AND WAIT FOR ANYTHING
        MOV     AL,INPKEY
        INT     DOS
FREERUN:NOP
        ENDM
;
;#################################################################
;
```

```
;       THESE MACROS ARE SOMEWHAT SPECIAL
;
;FIRST SOME GLOBALS
;
GETVEC  EQU     350AH           ;GET EXISTING INTERRUPT VECTOR FOR IRQ2
SETVEC  EQU     250AH           ;SET INTERRUPT VECTOR FOR IRQ2
PICMSK  EQU     21H             ;I/O PORT OF INTERRUPT CONTROLLER
ICCP            EQU     20H             ;COMMAND PORT FOR INT. CONTROLLER
EOI             EQU     20H             ;END OF INTERRUPT COMMAND
INTYES  EQU     0FBH            ;ALLOWS ONLY IRQ2 TO BE ACCEPTED
INTNO   EQU     0BCH            ;ALLOWS INTERRUPTS EXCEPT IRQ2
;
;
ZZSINT  MACRO                   ;SET INTERRUPT MECHANISM (FOR SAMPLING)
LOCAL   ESINT,INTROU,MORE,LCHAN1,LCHAN2,LCHAN3,LCHAN4,DA,NODA,ENDCHAN
        MOV     DUM1,OFFSET ESINT
        MOV     ADRINT,OFFSET ENDCHAN   ;USED AS A POINTER FOR JUMPS
;                                       ;WHEN INTERRUPT IS HANDLED
        LEA     DX,INTROU       ;SET NEW INTERRUPT VECTOR TO
        MOV     AX,SETVEC       ;INTERRUPT SERVICE ROUTINE
        INT     DOS
        JMP     DUM1
;
;################################################################
;
; INTERRUPT SERVICE ROUTINE
;
;################################################################
;
INTROU: ZZSHLD                  ;SET HOLD
LCHAN1: MOV     AL,MUX1         ;GET FIRST CHANNEL
        ZZCHA2                  ;SET MULTIPLEXER
        ZZADST                  ;START ADC
        ZZSTAD  BX              ;STORE SAMPLE WE GOT LAST
                                ;SO VERY FIRST SAMPLE IS NOT A SAMPLE
                                ;BUT (MAYBE USELESS) VALUE OF BX
```

```
        CMP     IDAYN,0         ;WE MUST DO SOME DA ?
        JZ      NODA
DA:     ZZGTDA                  ;GET NEXT DA VALUE
        ZZDA1                   ;AND OUTPUT IT
NODA:   ZZ35MU                  ;WAIT FOR ADC READY
        ZZADIN  BH,BL           ;GET SAMPLE IN BX
        CMP     NCHAN,1         ;MORE CHANNELS TO BE DONE ?
        JG      LCHAN2
        JMP     ADRINT          ;ELSE FINISH
LCHAN2: MOV     AL,MUX2
        ZZCHA2
        ZZADST
        ZZSTAD  BX              ;STORE SAMPLE WE GOT LAST ( CHANNEL 1 )
        ZZ35MU
        ZZADIN  BH,BL           ;GET SAMPLE FOR THIS CHANNEL
        CMP     NCHAN,2
        JG      LCHAN3
        JMP     ADRINT
LCHAN3: MOV     AL,MUX3         ;AND SO ON
        ZZCHA2
        ZZADST
        ZZSTAD  BX
        ZZ35MU
        ZZADIN  BH,BL
        CMP     NCHAN,3
        JG      LCHAN4
        JMP     ADRINT
LCHAN4: MOV     AL,MUX4
        ZZCHA2
        ZZADST
        ZZSTAD  BX
        ZZ35MU
        ZZADIN  BH,BL
ENDCHAN:DEC     CX              ;WE DID ONE SAMPLE / CHANNEL
                                ;CX IS SAMPLE COUNTER
        JNZ     MORE            ;ARE WE READY ?
```

```
        ZZDINT                  ;DISABLE FURTHER INTERRUPTS IRQ2
        ZZSTAD  BX              ;STORE LAST SAMPLE WE TOOK
        MOV     READY,0         ;REPORT TO MAIN PROGRAM
MORE:   ZZRHLD                  ;RESET HOLD
        MOV     AL,EOI          ;TELL ICCP INTERRUPT IS OVER
        OUT     ICCP,AL
        IRET                    ;RETURN TO INTERRUPTED PROCESS
;
;
;#########################################################################
;
;  END OF INTERRUPT ROUTINE
;
;#########################################################################
;
ESINT:  NOP
        ENDM
;
ZZGIRQ2 MACRO                   ;GET ORIGINAL INTERRUPT VECTOR IRQ2
        MOV     AX,GETVEC       ;AND SAVE IT
        PUSH    ES              ;FOR INTERRUPT IRQ2
        PUSH    BX
        INT     DOS
        MOV     AX,ES           ;AND SAVE IT
        MOV     OLDCS,AX
        MOV     OLDIP,BX
        POP     BX
        POP     ES
        IN      AL,PICMSK       ;GET EXISTING INTERRUPT MASK
        MOV     OLDMSK,AL       ;AND SAVE IT
        ENDM
;
;
ZZRIRQ2 MACRO                   ;RESET OLD INTERRUPT VECTOR IRQ2
        PUSH    BX
        MOV     BX,OLDCS
```

```
        MOV     DX,OLDIP
        MOV     AX,SETVEC
        PUSH    DS
        MOV     DS,BX
        INT     DOS
        POP     DS
        POP     BX
        MOV     AL,OLDMSK
        OUT     PICMSK,AL
        ENDM
;
ZZEINT  MACRO                   ;ALLOW IRQ2 (ONLY)
        MOV     AL,INTYES
        OUT     PICMSK,AL
        ENDM
;
ZZDINT  MACRO                   ;DISABLE IRQ2 ( BUT ALLOW OTHERS )
        MOV     AL,INTNO
        OUT     PICMSK,AL
        ENDM
;
ZZWAIT  MACRO                   ;WAIT NWAIT ( SEE CSEG ) * 0.01 SEC
        LOCAL EWAIT,INTROU,W3
        MOV     DUM1,OFFSET EWAIT
        LEA     DX,INTROU       ;SET INTERRUPT VECTOR
        MOV     AX,SETVEC
        INT     DOS
        JMP     DUM1
;
INTROU: DEC     NWAIT           ;WE HAD 0.01 SEC
        JNZ     W3
        ZZDINT                  ;WE ARE READY: STOP INTERRUPTS IRQ2
W3:     MOV     AL,EOI          ;TELL ICCP INTERRUPT IS OVER
        OUT     ICCP,AL
        IRET
;
```

```
EWAIT:  NOP
        ENDM
```

Appendix B YY.LIB

```
;
        INCLUDE ZZ.LIB
;
;
LENSPC  EQU     25              ;25 WORDS FOR MEASUREMENT PARAMETERS
;
CSEG    SEGMENT PUBLIC  'CODE'
;
MES1    DB      'Press any key to start measurement',ODH,OAH,'$'
;
BITSTAT DB      ?               ;STATUS BYTE FOR INTERFACE
;                               ;BIT 0 - 2 : CHANNEL NUMBER
;                               ;BIT 3     : MUX ENABLE ( 1=ON )
;                               ;BIT 4     : HOLD ( 1=HIGH )
;                               ;BIT 5     : ADC ENABLE ( 0=START )
;                               ;BIT 6     : CALIBRATION ( 1=ON )
;                               ;BIT 7     : AC/DC SWITCH ( 1=ON )
                                ;BITS 3-7 MAY NEVER BE CHANGED
OLDMSK  DB      ?               ;OLD INTERRUPT MASK
ADRINT  DW      ?               ;USED AS AN ADDRESS FOR JUMPS
IERR    DW      ?               ;USED AS AN ERROR RETURN
READY   DW      ?               ;USED TO SEE IF SAMPLING FINISHED
OLDCS   DW      ?               ;OLD CS FOR IRQ2
OLDIP   DW      ?               ;OLD IP FOR IRQ2
DUM1    DW      ?               ;DUMMY WORDS USED FOR JUMPS ETC.
DUM2    DW      ?
DUM3    DW      ?
IDIV0   DW      ?               ;SAVE FULL COUNT FOR COUNTER 0
IDIV1   DW      ?               ;SAVE FULL COUNT FOR COUNTER 1
IDIV2   DW      ?               ;SAVE FULL COUNT FOR COUNTER 2
NWAIT   DW      ?               ;NUMBER OF WAITS FOR ROUTINE YYWAIT
DABUF   DW      ?               ;SEGMENT FOR DA BUFFER
;
```

```
SPCBUF  EQU     THIS    WORD    ;BUFFER FOR SPECIFICATIONS
IBREAK  DW      ?               ;DUMMY
IDAYN   DW      ?               ;0 = NO DA 1 = DA DURING MEAS
NCHAN   DW      ?               ;NUMBER OF CHANNELS
CHAN1   DW      ?
CHAN2   DW      ?
CHAN3   DW      ?
CHAN4   DW      ?
TRIGTYP DW      ?               ;TRIGGERTYPE
TRIGOK  DW      ?               ;TRIGGER ACCEPT LEVEL FOR EXTERNAL TRIG
DCFILT  DW      ?               ;DUMMY
CALIB   DW      ?               ;DUMMY
SAVESPC DW      8 DUP(?)        ;DUMMIES
IARGST  DW      ?               ;DUMMY
IEXPST  DW      ?               ;DUMMY
IARGFT  DW      ?               ;DUMMY
IEXPFT  DW      ?               ;DUMMY
NREC    DW      ?               ;DUMMY
NSAMP   DW      ?               ;NUMBER OF SAMPLES
;
MUX1    DB      ?               ;CHANNEL 1 SELECT FOR MUX
MUX2    DB      ?               ;CHANNEL 2 SELECT FOR MUX
MUX3    DB      ?               ;CHANNEL 3 SELECT FOR MUX
MUX4    DB      ?               ;CHANNEL 4 SELECT FOR MUX
;
;
;#########################################################################
;
YYINIT  PROC    FAR             ;INITIALIZE INTERFACE
        PUBLIC  YYINIT
;
;       CALL YYINIT
;
        ZZHEAD  CSEG,CSEG,CSEG  ;SET SEGMENT REGISTERS
        ZZSTIM                  ;INITIALIZE TIMERCHIP SET-UP
        ZZSPPT                  ;INITIALIZE PP PORTS SET-UP
```

```
        MOV     BITSTAT,20H     ;INITIALIZE BITSTAT (DEFAULT)
        MOV     IDIV1,0077H     ;SET FILTER CUTOFF TO 100 HZ
        ZZWTM1  IDIV1
        MOV     AL,80H
        ZZDA0                   ;SET DAC0 TO 0 VOLT
        ZZDA1                   ;INITIALIZE DAC1 TO 0 VOLT
        ZZRHLD                  ;RESET SAMPLE-HOLDS
        ZZRCAL                  ;SET CALIBRATION OFF
        ZZRDCF                  ;SET DC-FILTER OFF
        ZZGIRQ2                 ;GET EXISTING INTERRUPT VECTOR IRQ2
        MOV     IDIV0,6         ;DEFAULT SAMPLING FREQUENCY = 300.05 HZ
        MOV     IDIV2,661
        ZZWTM0  IDIV0
        ZZWTM2  IDIV2
        ZZSINT                  ;SET DEFAULT IRQ2 VECTOR (SAMPLING)
        ZZFOOT                  ;RESTORE SEGMENT REGISTERS
        RET     0
YYINIT  ENDP
;
;#############################################################################
;
YYFINI  PROC    FAR             ;RESET INTERFACE
        PUBLIC  YYFINI
;
;       CALL YYFINI
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZSTIM                  ;INITIALIZE TIMERCHIP SET-UP
        ZZSPPT                  ;INITIALIZE PP PORTS SET-UP
        MOV     BITSTAT,20H     ;INITIALIZE BITSTAT (DEFAULT)
        MOV     IDIV1,0077H     ;INITIALIZE FILTERS TO 100 HZ
        ZZWTM1  IDIV1
        MOV     AL,80H
        ZZDA0                   ;SET DAC0 TO 0 VOLT
        ZZDA1                   ;INITIALIZE DAC1 TO 0 VOLT
        ZZRHLD                  ;RESET SAMPLE-HOLDS
```

```
        ZZRCAL                  ;SET CALIBRATION OFF
        ZZRDCF                  ;SET DC-FILTER OFF
        ZZRIRQ2                 ;RESET OLD IRQ2
        ZZFOOT
        RET     0
YYFINI  ENDP
;
;#############################################################################
;
YYSFRE  PROC    FAR             ;SET FREQUENCY OF INTERRUPTS
        PUBLIC  YYSFRE
                                ;CONTROLLER. INTERRUPT FREQUENCY IS
                                ;GIVEN BY 1.19E6/(IDIV0*IDIV2)
;
;       CALL YYSFRE(IDIV0,IDIV2)
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZLPAR  AX,PAR1,2
        CMP     AX,4            ;IDIV0 SHOULD BE GREATER THAN 4
        JGE     TIM0OK          ;OTHERWISE TIMERCHIP FAILS
        MOV     AX,4
TIM0OK: MOV     IDIV0,AX        ;SAVE IDIV0
        ZZWTM0  AX
        ZZLPAR  AX,PAR2,2
        CMP     AX,4            ;IDIV2 SHOULD BE GREATER THAN 4
        JGE     TIM2OK          ;OTHERWISE TIMERCHIP FAILS
        MOV     AX,4
TIM2OK: MOV     IDIV2,AX        ;SAVE IDIV2
        ZZWTM2  AX
        ZZFOOT
        RET     8
YYSFRE  ENDP
;
;#############################################################################
;
YYFFRE  PROC    FAR             ;SET SPEED TIMER 1 (FILTER CLOCK)
```

```
        PUBLIC  YYFFRE          ;CUT-OFF FREQUENCY = 1.19E6/IDIV1
;
;       CALL YYFFRE(IDIV1)
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZLPAR  AX,PAR1,1
        CMP     AX,4            ;IDIV1 SHOULD BE GREATER THAN 4
        JGE     TIM1OK          ;OTHERWISE TIMERCHIP FAILS
        MOV     AX,4
TIM1OK: MOV     IDIV1,AX        ;SAVE IDIV1
        ZZWTM1  AX
        ZZFOOT
        RET     4
YYFFRE  ENDP
;
;#############################################################################
;
YYRTMO  PROC    FAR             ;READ TIMER 0 ( SAMPLE COUNTER )
        PUBLIC  YYRTMO
;
;       CALL YYRTMO(ISAMPLE)
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZRTMO
        ZZSPAR  AX,PAR1,1
        ZZFOOT
        RET     4
YYRTMO  ENDP
;
;#############################################################################
;
YYSDCF  PROC    FAR             ;SET DC-FILTER ON
        PUBLIC  YYSDCF
;
;       CALL YYSDCF
;
```

```
        ZZHEAD  CSEG,CSEG,CSEG
        ZZSDCF
        ZZFOOT
        RET     0
YYSDCF  ENDP
;
;#########################################################################
;
YYRDCF  PROC    FAR             ;SET DC-FILTER OFF
        PUBLIC  YYRDCF
;
;       CALL YYRDCF
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZRDCF
        ZZFOOT
        RET     0
YYRDCF  ENDP
;
;#########################################################################
;
YYSCAL  PROC    FAR             ;SET CALIBRATION ON
        PUBLIC  YYSCAL
;
;       CALL YYSCAL
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZSCAL
        ZZFOOT
        RET     0
YYSCAL  ENDP
;
;#########################################################################
;
YYRCAL  PROC    FAR             ;SET CALIBRATION OFF
        PUBLIC  YYRCAL
```

```
;
;       CALL YYRCAL
;
        ZZHEAD  CSEG,CSEG
        ZZRCAL
        ZZFOOT
        RET     0
YYRCAL  ENDP
;
;###############################################################################
;
YYDAO   PROC    FAR             ;OUTPUT TO DACO
        PUBLIC  YYDAO
;
;       CALL YYDAO(IVDACO)
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZLPAR  AX,PAR1,1
        ZZDAO
        ZZFOOT
        RET     4
YYDAO   ENDP
;
;###############################################################################
;
YYDA1   PROC    FAR             ;OUTPUT TO DAC1
        PUBLIC  YYDA1
;
;       CALL YYDA1(IVDAC1)
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZLPAR  AX,PAR1,1
        ZZDA1
        ZZFOOT
        RET     4
YYDA1   ENDP
```

```
;
;#############################################################################
;
YYSTRG  PROC    FAR             ;GET STATUS TRIGGER
        PUBLIC  YYSTRG          ;RESULT WILL BE 0 OR 128
;
;       CALL YYSTRG(ISTTRG)
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZSTRG
        MOV     AH,0
        ZZSPAR  AX,PAR1,1
        ZZFOOT
        RET     4
YYSTRG  ENDP
;
;#############################################################################
;
YYMEAS  PROC    FAR             ;MEASURE SOME CHANNELS WITH(OUT) DA
        PUBLIC  YYMEAS
;
;       CALL YYMEAS(ISPEC,IDABUF,IADBUF,IERR)
;
                                ;ISPEC=I*2 ARRAY WITH SPECIFICATIONS
                                ;       AS DEFINED IN SPCBUF
                                ;IDABUF=I*2 ARRAY FOR DA VALUES
                                ;IADBUF=I*2 ARRAY FOR AD VALUES
                                ;       IADBUF(1) WILL CONTAIN NUMBER
                                ;       OF SAMPLES ON EXIT
                                ;       SAMPLES ARE STORED FROM THERE
                                ;IERR=ERROR RETURN (0=NO ERROR)
;
        ZZHEAD  CSEG,CSEG,CSEG
        MOV     IERR,0          ;INITIALIZE IERR TO 0
        MOV     READY,1         ;SET READY TO 1 : NOT READY
        MOV     DUM2,OFFSET L1  ;GET JUMP ADDRESS
```

```
        ZZLDSP  PAR1,4          ;GET SPECIFICATIONS
        CMP     NSAMP,0
        JG      L4
        JMP     DUM2
L4:     CMP     NCHAN,0
        JG      L5
        JMP     DUM2
L5:     ZZPCHAN                 ;PREPARE CHANNELS FOR MUX
        ZZINAR  PAR2,PAR3,4     ;SET BUFFERS FOR AD AND DA
        ZZSTRT                  ;START DEPENDING ON TRIGGERTYPE
        MOV     CX,NSAMP        ;CX COUNTS SAMPLES TO BE DONE IN INTER-
                                ;RUPT ROUTINE
        MOV     BX,CX           ;THIS LETS FIRST ELEMENT IN IADBUF TO
                                ;BE NUMBER OF SAMPLES TAKEN
                                ;SO ACTUAL SAMPLES START AT IADBUF(2)
                                ;(SEE MACRO ZZSINT AND INTERRUPT ROUT.)
        ZZEINT                  ;ENABLE IRQ2 ( MEASUREMENT STARTS )
LO:     CMP     READY,0         ;ARE WE READY ?
        JNZ     LO
        JZ      L2
L1:     MOV     IERR,1          ;ERROR RETURN : ERROR IN SPECIFICATIONS
L2:     ZZSPAR  IERR,PAR4,4
        ZZFOOT
        RET     16
YYMEAS  ENDP
;
;#############################################################################
;
YYWAIT  PROC    FAR                     ;WAIT NWAIT * 0.01 SEC
        PUBLIC  YYWAIT                  ;(WITH APPROX 0.3 MS DEVIATION)
;
;       CALL YYWAIT(NWAIT)
;
        ZZHEAD  CSEG,CSEG,CSEG
        ZZLPAR  NWAIT,PAR1,1            ;GET NWAIT
        CMP     NWAIT,0
```

```
        JG      W0
        JMP     W2
W0:     ZZWTM0  20                      ;SET INTERRUPT FREQUENCY AT
        ZZWTM2  595                     ;100 HZ
        ZZWAIT                          ;SET INTERRUPT VECTOR
        ZZEINT                          ;ENABLE INTERRUPTS ( WAITING )
W1:     CMP     NWAIT,0                 ;ARE WE READY ?
        JNZ     W1
        ZZWTM0  IDIV0                   ;RESET OLD SAMPLING FEQUENCY
        ZZWTM2  IDIV2
        ZZSINT                          ;RESET INTERRUPT VECTOR FOR
W2:     ZZFOOT                          ;SAMPLING
        RET     4
YYWAIT  ENDP
;
;
CSEG    ENDS
        END
```

Appendix C XX.CMN en XX.LIB

```
C
C     COMMON-BLOCKS FOR XX.LIB
C
      INTEGER*2 LENSPC
      INTEGER*4 MAXAD,MAXDA
C
C
      PARAMETER (LENSPC=25)
      PARAMETER (MAXAD=32800)
      PARAMETER (MAXDA=32800)
C
C     LENSPC=NUMBER OF PARAMETERS IN SPECIFICATIONS
C     MAXAD=LENGTH OF ARRAYS XXAD AND XXDBUF
C     MAXDA=LENGTH OF ARRAY XXDA
C
C
      PARAMETER (MAXCAL=100)
C
C     MAXCAL=MAXIMUM NUMBER OF CALIBRATION VOLTAGES
C
C
      COMMON/XX1/XXAD(MAXAD)
      COMMON/XX5/XXDBUF(MAXAD)
      INTEGER*2 XXDBUF
C
C     XXAD AND XXDBUF ARE USED FOR STORAGE OF AD-VALUES
C
      COMMON/XX2/XXDA(MAXDA)
      INTEGER*2 XXAD,XXDA
C
C     XXDA IS USED FOR STORAGE OF DA-VALUES
C
C
```

```
      COMMON/XXO/XXSPC(LENSPC),XXLOWC,XXUPPC,XXOK,XXDACO,
     & XXOUTC(MAXCAL),XXINPC(MAXCAL),XXIDVO,XXIDV1,XXIDV2,
     & XXDAC1,XXDCS,XXCALS,XXTRIG,XXTROK
      INTEGER*2 XXSPC,XXLOWC,XXUPPC,XXOK,XXDACO,XXIDVO,XXIDV1,XXIDV2
      INTEGER*2 XXDAC1,XXTRIG,XXTROK,XXDCS,XXCALS
      REAL*4 XXOUTC,XXINPC
C
C     XXSPC =ARRAY FOR SPECIFICATIONS
C     XXLOWC=LOWER LIMIT FOR CALIBRATION VOLTAGE
C     XXUPPC=UPPER LIMIT FOR CALIBRATION VOLTAGE
C     XXOK  =ERROR RETURN FOR MEASUREMENTS
C     XXDACO=VALUE SEND TO DACO
C     XXOUTC=ARRAY FOR CALIBRATION OUTPUT VOLTAGES
C     XXINPC=ARRAY FOR CALIBRATION INPUT VOLTAGES ( SEND TO DACO )
C     XXIDVO=VALUE SEND TO TIMER 0
C     XXIDV1=VALUE SEND TO TIMER 1
C     XXIDV2=VALUE SEND TO TIMER 2
C     XXDAC1=VALUE SEND TO DAC1
C     XXDCS =STATUS OF DC-FILTER SWITCH
C     XXCALS=STATUS OF CALIBRATION SWITCH
C     XXTRIG=TRIGGER TYPE
C     XXTROK=TRIGGER ACCEPT LEVEL
C
C
C
      COMMON/XX3/XXPMOD,XXIEEE,XXBACL,XXBOCL,XXPAL,XXAXCL,XXLNCL
      INTEGER*2 XXPMOD,XXIEEE,XXBACL,XXBOCL,XXPAL,XXAXCL,XXLNCL
C
C     XXPMOD=GRAPHICS MODE FOR HALO LIBRARY
C     XXIEEE=PRESENCE OF 8087 FOR HALO LIBRARY
C     XXBACL=BACKGROUND COLOR FOR HALO LIBRARY
C     XXBOCL=FOREGROUND COLOR FOR HALO LIBRARY
C     XXPAL =PALETTE NUMBER FOR HALO LIBRARY
C     XXAXCL=COLOR VALUE FOR DRAWING FOR HALO LIBRARY
C     XXLNCL=COLOR VALUE FOR DRAWING FOR HALO LIBRARY
C
```

```
C
      COMMON/XX4/XXISP(25),XXCSP(20,16),XXRSP(64),XXDSP(224)
      INTEGER*2 XXISP, XXDSP
      CHARACTER*1 XXCSP
      REAL*4 XXRSP
C
C     XXISP =ARRAY FOR SPECIFICATIONS
C     XXCSP =ARRAY FOR CHARACTER DATA
C     XXRSP =ARRAY FOR REAL DATA
C     XXDSP =ARRAY FOR DUMMY DATA
C     LENGTH OF COMMON BLOCK XX4 MUST BE 1074 BYTES IF IT IS USED
C     FOR GENERATION OF FILES WITH HEADER MATCHING FILES GENERATED WITH
C     16-CHANNEL MEASURING SYSTEM
C
```

```
$STORAGE:2
C
C##############################################################################
C
      SUBROUTINE XXINIT
$INCLUDE:'XX.CMN'
C
C     ROUTINE TO INITIALIZE SOME DEFAULT PARAMETERS AND TO INITIALIZE
C     INTERFACE
C
C     INITIALIZE INTERFACE
C
      CALL YYINIT
C
C     INITIALIZE PARAMETERS FOR PLOTTING WITH HALO LIBRARY
C     (SEE ROUTINES XXPLAD AND XXPLDA)
C
C
      XXPMOD=1
      XXBACL=0
      XXBOCL=14
      XXPAL=1
      XXLNCL=3
      XXAXCL=3
      XXIEEE=1
C
C     INITIALIZE MEASUREMENT PARAMETERS
C
      XXISP(1)=-1
      XXISP(2)=0
      XXISP(3)=4
      XXISP(4)=0
      XXISP(5)=1
      XXISP(6)=2
      XXISP(7)=3
      XXISP(8)=0
```

```
      XXISP(9)=0
      XXISP(10)=0
      XXISP(11)=0
      DO 10 I=12,19
      XXISP(I)=0
10    CONTINUE
      XXISP(24)=1
      XXISP(25)=200
      DO 20 I=1,20
      DO 15 J=1,16
      XXCSP(I,J)=' '
15    CONTINUE
20    CONTINUE
      DO 30 J=1,4
      XXCSP(1,J)='C'
      XXCSP(2,J)='H'
      XXCSP(3,J)='A'
      XXCSP(4,J)='N'
      XXCSP(5,J)='N'
      XXCSP(6,J)='E'
      XXCSP(7,J)='L'
      XXCSP(8,J)='.'
      XXCSP(9,J)='C'
      XXCSP(10,J)='O'
30    CONTINUE
      XXCSP(11,1)='0'
      XXCSP(11,2)='1'
      XXCSP(11,3)='2'
      XXCSP(11,4)='3'
      DO 40 I=1,16
      XXRSP(I)=0.0
      XXRSP(16+I)=1.0
      XXRSP(32+I)=9.99
      XXRSP(48+I)=9.99
40    CONTINUE
C
```

```
C
C     INITIALIZE INTERRUPT AND FILTER CUT-OFF FREQUENCIES
C
      CALL XXSAFR(300.0,XXISP(3),XXISP(2),XXISP(20),XXISP(21))
      CALL XXFCOF(100.0,XXISP(22),XXISP(23))
C
C
      XXDAC0=128
      XXDAC1=128
      XXIDV0=6
      XXIDV2=661
      XXIDV1=119
      XXDCS=0
      XXCALS=0
      XXTRIG=0
      XXTROK=0
      RETURN
      END
C
C##############################################################################
C
      SUBROUTINE XXCALI(ICHAN,GAIN,OFFSET,NCAL,NSAMP,IERR)
$INCLUDE:'XX.CMN'
      INTEGER*2 ICHAN,NCAL,NSAMP,IERR
      REAL*4 GAIN,OFFSET
C
C     SUBROUTINE TO DETERMINE GAIN AND OFFSET OF CHANNEL ICHAN
C
C     PARAMETERS:
C     -----------
C
C     ICHAN (INPUT  I*2 ) = CHANNEL NUMBER TO CALIBRATE ( 0,1,2 OR 3 )
C     GAIN  (OUTPUT R*4 ) = GAIN OF CHANNEL ICHAN
C     OFFSET(OUTPUT R*4 ) = OFFSET OF CHANNEL ICHAN IN VOLTS
C     NCAL  (INPUT  I*2 ) = NUMBER OF CALIBRATION VOLTAGES >= 2
C     NSAMP (INPUT  I*2 ) = NUMBER OF SAMPLES TO BE AVERAGED AT
```

```
C                           ONE CALIBRATION VOLTAGE <= 32766
C     IERR  (INPUT  I*2 ) = ERROR RETURN
C                           0 = NO ERROR
C                           1 = ERROR IN INPUT PARAMETERS
C                           2 = ERROR IN ROUTINE YYMEAS
C                           3 = CAN'T CALIBRATE
C                           4 = CAN'T CALIBRATE
C                           5 = RANGE FOR CALIBRATION/NCAL TOO SMALL
C                           6 = LINEAR REGRESSION SINGULAR
C
C
C     METHOD:
C     -------
C     FIRST RANGE IS SELECTED FOR CALIBRATION VOLTAGES SET WITH DACO
C     BY CHECKING ON OVERFLOW. NEXT NCAL EQUIDISTANT CALIBRATION
C     VOLTAGES ARE SELECTED, SET AT DACO AND OUTPUT OF CHANNEL ICHAN
C     IS SAMPLED NSAMP TIMES. THESE SAMPLES ARE AVERAGED TO GET ONE
C     DATA POINT FOR CALIBRATION. THE SET OF NCAL DATAPOINTS THUS
C     OBTAINED IS USED IN A LINEAR REGRESSION TO OBTAIN THE GAIN AND
C     OFFSET GIVEN BY : OUTPUT = OFFSET + GAIN*INPUT
C     SAMPLING FREQUENCY AND FILTER CUT-OFF FREQUENCY MUST BE SET
C     SOMEWHERE ELSE
C
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     YYSCAL,YYMEAS,YYWAIT,YYDAO,XXVDAO,XXVAD,YYRCAL
C
C     LOCAL DATA
C     ----------
      INTEGER*2 DVCAL,I,J,NWAIT,DUMSPC(LENSPC),DACINC
      REAL*4 DUM1,SIN,SINS,SOU,SOUIN,DETERM
      PARAMETER ( NWAIT = 200 )
      PARAMETER ( DACINC = 25 )
C
C     CHECK ON PARAMETERS
```

```
C
      IF ( NCAL .LE. 1 .OR. NSAMP .LT. 1)THEN
         IERR = 1
         RETURN
      ENDIF
      IF ( ICHAN .LT. 0 .OR. ICHAN .GT. 3)THEN
         IERR=1
         RETURN
      ENDIF
C
C     CREATE DUMMY SPECIFICATIONS FOR YYMEAS
C
      DUMSPC(25)=NSAMP
C  = NUMBER OF SAMPLES TO TAKE
      DUMSPC(2)=0
C  = NO DA TO DAC1 DURING MEAUREMENTS
      DUMSPC(3)=1
C  = 1 CHANNEL TO MEASURE
      DUMSPC(4)=ICHAN
C  = THIS CHANNEL
      DUMSPC(8)=0
C  = TRIGGER FREE-RUN
C
C     SET INTERFACE AT CALIBRATION
C
      CALL YYSCAL
C
C     SELECT RANGE FOR CALIBRATION
C     FIRST GET LOWER LIMIT
C     WE START AT -5 VOLT AT DAC0 AND INCREMENT THIS VALUE TILL
C     NO OVERFLOW OF ADC OCCURS
C
      XXLOWC=0
10    CALL YYDAO(XXLOWC)
C
C     WAIT FOR DAC0 TO STABILIZE
```

```
C
      CALL YYWAIT(NWAIT)
C
C     THEN MEASURE
C
      CALL YYMEAS(DUMSPC,XXDA,XXDBUF,XXOK)
      IF ( XXOK .NE. 0 )THEN
         IERR=2
         RETURN
      ENDIF
C
C     CHECK ON OVERFLOW
C
      DO 20 I=1,NSAMP
      IF ( XXDBUF(I+1) .EQ. 2047)THEN
         IERR=3
         RETURN
      ENDIF
      IF ( XXDBUF(I+1) .EQ. -2048 )THEN
         XXLOWC=XXLOWC+DACINC
         GOTO 10
      ENDIF
20    CONTINUE
C
C     LOWEST LIMIT IS NOW FOUND
C     GET HIGHEST LIMIT
C     WE START AT +5 VOLT AT DACO AND DECREMENT TILL NO OVERFLOW OCCURS
C
      XXUPPC=255
30    CALL YYDAO(XXUPPC)
      CALL YYWAIT(NWAIT)
      CALL YYMEAS(DUMSPC,XXDA,XXDBUF,XXOK)
      DO 40 I=1,NSAMP
      IF ( XXDBUF(I+1) .EQ. -2048 )THEN
         IERR=4
         RETURN
```

```
      ENDIF
      IF ( XXDBUF(I+1) .EQ. 2047)THEN
         XXUPPC=XXUPPC-DACINC
         GOTO 30
      ENDIF
40    CONTINUE
C
C     HIGHEST LIMIT IS FOUND
C     GET INCREMENTAL INTERVALS FOR CALIBRATION
C
      DUM1=FLOAT(XXUPPC-XXLOWC)/FLOAT(NCAL-1)
      DVCAL=IFIX(DUM1)
C
C     ADJUST IF NECESSARY DUE TO ROUNDING ERRORS
C
      IF ( (XXLOWC+NCAL*DVCAL) .GT. XXUPPC)DVCAL=DVCAL-1
      IF ( DVCAL .LE. 0 )THEN
         IERR=5
         RETURN
      ENDIF
C
C     START LOOP OVER CALIBRATION VOLTAGES
C
      DO 100 I=1,NCAL
      XXDACO=XXLOWC+(I-1)*DVCAL
      CALL YYDAO(XXDACO)
      CALL YYWAIT(NWAIT)
      CALL YYMEAS(DUMSPC,XXDA,XXDBUF,XXOK)
C
C     AVERAGE SAMPLES TO GET ONE DATAPOINT FOR CALIBRATION
C
      XXOUTC(I)=0.0
      DO 60 J=1,NSAMP
      XXOUTC(I)=XXOUTC(I)+XXVAD(XXDBUF(J+1))
60    CONTINUE
      XXOUTC(I)=XXOUTC(I)/FLOAT(NSAMP)
```

```
C
C     STORE INPUT CALIBRATION VOLTAGE
C
      XXINPC(I)=XXVDAO(XXDACO)
100   CONTINUE
C
C     NCAL DATAPOINTS FOR LINEAR REGRESSION ARE FOUND
C
C     GET GAIN AND OFFSET WITH LINEAR REGRESSION ON NCAL DATAPOINTS
C
      SIN=0.0
      SINS=0.0
      SOU=0.0
      SOUIN=0.0
      DO 200 I=1,NCAL
      SIN=SIN+XXINPC(I)
      SINS=SINS+XXINPC(I)**2
      SOU=SOU+XXOUTC(I)
      SOUIN=SOUIN+XXOUTC(I)*XXINPC(I)
200   CONTINUE
      SIN=SIN/FLOAT(NCAL)
      SINS=SINS/FLOAT(NCAL)
      SOU=SOU/FLOAT(NCAL)
      SOUIN=SOUIN/FLOAT(NCAL)
      DETERM=SIN*SIN-SINS
      IF ( ABS(DETERM) .LE. 1.E-10 )THEN
         IERR=6
         RETURN
      ENDIF
      GAIN=(SIN*SOU-SOUIN)/DETERM
      OFFSET=(-SINS*SOU+SIN*SOUIN)/DETERM
      IERR=0
C
C     RESET INTERFACE FOR CALIBRATION
C
      CALL YYRCAL
```

```
      RETURN
      END
C
C#########################################################################
C
      REAL*4 FUNCTION XXVDAO(VALUE)
      INTEGER*2 VALUE
C
C     XXVDAO RETURNS VOLTS AT DACO WHEN VALUE IS SEND TO IT
C     VALUE (INPUT  I*2) IS IN RANGE FROM 0 ... 255
C     XXVDAO(OUTPUT R*4) IS IN RANGE -5.0 TO +5.0 VOLT
C
      XXVDAO=(10.0/256.0)*FLOAT(VALUE)-5.0
      END
C
C#########################################################################
C
      REAL*4 FUNCTION XXVDA1(VALUE)
      INTEGER*2 VALUE
C
C     XXVDA1 RETURNS VOLTS AT DACO WHEN VALUE IS SEND TO IT
C     VALUE (INPUT  I*2) IS IN RANGE FROM 0 ... 255
C     XXVDA1(OUTPUT R*4) IS IN RANGE -5.0 TO +5.0 VOLT
C
      XXVDA1=(10.0/256.0)*FLOAT(VALUE)-5.0
      END
C
C#########################################################################
C
      INTEGER*2 FUNCTION XXBDAO(VOLT)
      REAL*4 VOLT
C
C     XXBDAO RETURNS BYTE (PACKED AS LOW BYTE) TO BE SEND AT DACO TO
C     OBTAIN VOLTAGE VOLT AT OUTPUT
C     VOLT  (INPUT  R*4) IS IN RANGE FROM -5 TO + 5
C     XXBDAO(OUTPUT I*2) IS IN RANGE 0 ... 255
```

```
C
      XXBDA0=IFIX( (VOLT+5.0)*256.0/10.0 )
      IF ( XXBDA0 .LT. 0 )XXBDA0=0
      IF ( XXBDA0 .GT. 255)XXBDA0=255
      END
C
C#############################################################################
C
      INTEGER*2 FUNCTION XXBDA1(VOLT)
      REAL*4 VOLT
C
C     XXBDA1 RETURNS BYTE (PACKED AS LOW BYTE) TO BE SEND AT DAC1 TO
C     OBTAIN VOLTAGE VOLT AT OUTPUT
C     VOLT  (INPUT  R*4) IS IN RANGE FROM -5 TO + 5
C     XXBDA1(OUTPUT I*2) IS IN RANGE 0 ... 255
C
      XXBDA1=IFIX( (VOLT+5.0)*256.0/10.0 )
      IF ( XXBDA1 .LT. 0 )XXBDA1=0
      IF ( XXBDA1 .GT. 255)XXBDA1=255
      END
C
C#############################################################################
C
      REAL*4 FUNCTION XXVAD(VALUE)
      INTEGER*2 VALUE
C
C     XXVAD RETURNS VOLTS AT ADC AS MEASURED IN VALUE
C     VALUE (INPUT  I*2) IS IN RANGE -2048...2047
C     XXVAD (OUTPUT R*4) IS IN RANGE -5.0 TO +5.0 VOLT
C
      XXVAD=(5.0/2048.0)*FLOAT(VALUE)
      END
C
C#############################################################################
C
      SUBROUTINE XXFCOF(FREQ,IMANT,IEXP)
```

```
$INCLUDE:'XX.CMN'
      REAL*4 FREQ
      INTEGER*2 IMANT,IEXP
C
C     SUBROUTINE TO SET FILTER CUT-OFF FREQEUNCY AT VALUE AS CLOSE
C     AS POSSIBLE TO FREQ
C
C     PARAMETERS
C     ----------
C     FREQ (INPUT  R*4) DESIRED FILTER CUT-OFF FREQUENCY
C     FREQ (OUTPUT R*4) ACTUAL FILTER CUT-OFF FREQUENCY ON EXIT
C     IMANT (OUTPUT I*2) MANTISSA FOR FILTER FREQUENCY
C     IEXP  (OUTPUT I*2) EXPONENT FOR FILTER FREQUENCY
C                        FILTER FREQUENCY IS 1.19*10**4/(IMANT*10**IEXP)
C
C     METHOD
C     ------
C     DETERMINE I*2 VALUE XXIDV1 NECESSARY TO SEND TO TIMER 1 TO
C     GET FILTER CUT-OFF FREQUENCY GIVEN BY 1.19E6/(100*XXIDV1)
C     (4 <= XXIDV1 <= 11900) ( 1 <= FREQ <= 2975 HZ)
C     DUE TO CONVERSION TO I*2 NOT EVERY DESIRED FREQUENCY CAN BE SET
C     NEXT IMANT AND IEXP ARE DETERMINED AND RETURNED
C
C     ROUTINES AND FUNCTIONS USED
C     ---------------------------
C     YYFFRE,XXEXMA
C
C     LOCAL DATA
C     ----------
C
      XXIDV1=IFIX(1.19E6/(100.0*FREQ))
      IF ( XXIDV1 .LT. 4 )XXIDV1=4
      IF ( XXIDV1 .GT. 11900)XXIDV1=11900
C
C     SET FILTER CUT-OFF FREQUENCY
C
```

```
      CALL YYFFRE(XXIDV1)
C
C     RETURN FILTER CUT-OFF FREQUENCY
C
      FREQ=1.19E6/(100.0*FLOAT(XXIDV1))
C
C     DETERMINE IMANT AND IEXP
C
      CALL XXEXMA(1.19E6/(100.0*FREQ),IMANT,IEXP)
C
      RETURN
      END
C
C##############################################################################
C
      SUBROUTINE XXSAFR(FREQ,NCHAN,IDAYN,IMANT,IEXP)
$INCLUDE:'XX.CMN'
      REAL*4 FREQ
      INTEGER*2 NCHAN,IMANT,IEXP,IDAYN
C
C     SUBROUTINE TO SET SAMPLING FREQUENCY TO VALUE AS CLOSE AS
C     POSSIBLE TO FREQ WITHIN THE LIMITS ALLOWABLE AND DETERMINED
C     BY NUMBER OF CHANNELS TO BE MEASURED AND NEED FOR DA-OUTPUT
C     TO DAC1 DURING MEASUREMENTS
C
C     PARAMETERS
C     ----------
C     FREQ   (INPUT  R*4)DESIRED SAMPLING FREQUENCY
C     FREQ   (OUTPUT R*4)ACTUAL SAMPLING FREQUENCY ON EXIT
C     NCHAN  (INPUT  I*2)NUMBER OF CHANNELS TO BE MEASURED
C                        ( 1<=NCHAN<=4)
C     IDAYN  (INPUT  I*2)DA-OUTPUT REQUIRED (0 = NO , 1 = YES )
C     IMANT  (OUTPUT I*2)MANTISSA FOR TIME-INTERVAL BETWEEN SAMPLES
C     IEXP   (OUTPUT I*2)EXPONENT FOR TIME-INTERVAL BETWEEN SAMPLES
C                        SAMPLING FREQUENCY = 1.19*10**6/(IMANT*10**IEXP)
C
```

```
C     METHOD
C     ------
C
C     WE DETERMINE THE I*2 VALUES XXIDVO AND XXIDV2 THAT ARE TO
C     BE SEND AT TIMERO AND TIMER2 RESPECTIVELY, SUCH THAT THE
C     SAMPLING FREQUENCY WILL BE 1.19E6/(XXIDVO*XXIDV1)
C     THEN IMANT AND IEXP ARE CALCULATED
C     HOWEVER, THE SAMPLING FREQUENCY IS BOUND TO LIMITS BY THE
C     THE NUMBER OF CHANNELS TO BE MEASURED AND DA-OUTPUT REQUIREMENTS
C     THESE LIMITS ARE:
C
C     NCHAN               NO DA-OUTPUT            DA-OUTPUT
C     -----               ------------            ---------
C     1                   0.01...6750 HZ          0.01...6450 HZ
C     2                   0.01...4200 HZ          0.01...4150 HZ
C     3                   0.01...3050 HZ          0.01...3000 HZ
C     4                   0.01...2500 HZ          0.01...2440 HZ
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     YYSFRE,XXEXMA
C
C     LOCAL DATA
C     ----------
      LOGICAL LDAYN
      INTEGER*4 IM4F
C
C
C     CHECK FREQUENCY RANGE DEPENDING ON NEED FOR DA OUTPUT
C
      IF ( IDAYN .EQ. 0 )LDAYN=.FALSE.
      IF ( IDAYN .EQ. 1 )LDAYN=.TRUE.
      IF ( NCHAN .LT. 1 )NCHAN=1
      IF ( NCHAN .GT. 4 )NCHAN=4
      GOTO (10,20,30,40),NCHAN
C
```

```
C     NCHAN=1
C
10    IF ( LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 6450.0)FREQ=6450.0
      ENDIF
      IF ( .NOT. LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 6750.0 )FREQ=6750.0
      ENDIF
      GOTO 100
C
C     NCHAN=2
C
20    IF ( LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 4150.0)FREQ=4150.0
      ENDIF
      IF ( .NOT. LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 4200.0 )FREQ=4200.0
      ENDIF
      GOTO 100
C
C     NCHAN=3
C
30    IF ( LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 3000.0)FREQ=3000.0
      ENDIF
      IF ( .NOT. LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 3050.0 )FREQ=3050.0
      ENDIF
      GOTO 100
C
```

```
C     NCHAN=4
C
40    IF ( LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 2440.0)FREQ=2440.0
      ENDIF
      IF ( .NOT. LDAYN )THEN
         IF ( FREQ .LT. 0.01 )FREQ=0.01
         IF ( FREQ .GT. 2500.0 )FREQ=2500.0
      ENDIF
      GOTO 100
C
C     DETERMINE PRODUCT OF XXIDVO AND XXIDV2
C
100   IM4F=INT(1.19E6/FREQ)
C
C     IF IM4F .LE. 32767 (FREQ .GT. 36 HZ) THEN WE SET XXIDVO TO 4
C     AND CALCULATE XXIDV2. THIS GIVES BEST RESULTS FOR LARGER SAMPLING
C     FREQUENCIES
C
C     IF IM4F .GT. 32767 (FREQ .LE. 36 HZ) THEN WE SET XXIDVO=XXIDV2
C     THIS GIVES GOOD RESULTS FOR LOWER SAMPLING FREQUENCIES
C
      IF ( IM4F .LE. 32767 )THEN
         XXIDVO=4
         XXIDV2=IFIX(1.19E6/(4.0*FREQ))
      ENDIF
      IF ( IM4F .GT. 32767 )THEN
         XXIDVO=IFIX(SQRT(1.19E6/FREQ))
         XXIDV2=XXIDVO
      ENDIF
      IF ( XXIDVO .LT. 4)XXIDVO=4
      IF ( XXIDV2 .LT. 4)XXIDV2=4
C
C     SET FREQUENCY
C
```

```
      CALL YYSFRE(XXIDVO,XXIDV2)
C
C     AND RETURN SAMPLING FREQUENCY
C
      FREQ=1.19E6/(FLOAT(XXIDVO)*FLOAT(XXIDV2))
C
C     DETERMINE IMANT AND IEXP
C
      CALL XXEXMA(1.19E6/FREQ,IMANT,IEXP)
C
      RETURN
      END
C
C#############################################################################
C
      REAL*4 FUNCTION XXFS(IMANT,IEXP)
      INTEGER*2 IMANT,IEXP
C
C     XXFS RETURNS INTERRUPT FREQUENCY GIVEN BY
C     1.19E*10**6/(IMANT*10**IEXP)
C
      XXFS=1.19E6/(FLOAT(IMANT)*10**IEXP)
C
      END
C
C#############################################################################
C
      REAL*4 FUNCTION XXFC(IMANT,IEXP)
      INTEGER*2 IMANT,IEXP
C
C     XXFC RETURNS FILTER CUT-OFF FREQUENCY GIVEN BY
C     1.19*10**4/(IMANT*10**IEXP)
C
      XXFC=1.19E4/(FLOAT(IMANT)*10**IEXP)
C
      END
```

```
C
C#######################################################################
C
      SUBROUTINE XXEXMA(VALUE,IMANT,IEXP)
      REAL*4 VALUE
      INTEGER*2 IMANT,IEXP
C
C     SUBROUTINE TO DETERMINE MANTISSA IMANT END EXPONENT IEXP OF
C     REAL VALUE>=1.0 SUCH THAT VALUE = IMANT*10**IEXP ( APART FROM
C     ROUNDING ERRORS DUE TO CONVERSION TO INTEGER*2 )
C
C     PARAMETERS
C     ----------
C     VALUE  (INPUT  R*4)
C     IMANT  (OUTPUT I*2)
C     IEXP   (OUTPUT I*2)
C
C     METHOD
C     ------
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C
C     LOCAL DATA
C     ----------
      REAL*4 DUM
C
      IF ( VALUE .LT. 1.0 )VALUE=1.0
      DUM=VALUE
      IEXP=0
1     IF ( DUM .LT. 32767.0 )GOTO 2
      DUM=DUM/10.
      IEXP=IEXP+1
      GOTO 1
2     IMANT=INT(DUM)
C
```

```
      RETURN
      END
C
C
C##############################################################################
C
      SUBROUTINE XXSDCF(ISTAT)
$INCLUDE:'XX.CMN'
      INTEGER*2 ISTAT
C
C     SUBROUTINE TO SET DC-FILTER ON OR OFF DEPENDING ON VALUE OF
C     ISTAT
C
C     PARAMETERS
C     ----------
C     ISTAT (INPUT I*2) 0 = DC-FILTER OFF
C                       1 = DC-FILTER ON
C
C     METHOD
C     ------
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     YYRDCF,YYSDCF
C
C     LOCAL DATA
C     ----------
C
C
      IF ( ISTAT .EQ. 0 )CALL YYRDCF
      IF ( ISTAT .EQ. 1 )CALL YYSDCF
C
      XXDCS=ISTAT
      RETURN
      END
C
```

```
C##############################################################################
C
      SUBROUTINE XXSCAL(ISTAT)
$INCLUDE:'XX.CMN'
      INTEGER*2 ISTAT
C
C     SUBROUTINE TO SET CALIBRATION ON OR OFF DEPENDING ON VALUE OF
C     ISTAT
C
C     PARAMETERS
C     ----------
C     ISTAT  (INPUT I*2) 0 = CALIBRATION OFF
C                        1 = CALIBRATION ON
C
C     METHOD
C     ------
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     YYSCAL,YYRCAL
C
C     LOCAL DATA
C     ----------
C
C
      IF ( ISTAT .EQ. 0 )CALL YYRCAL
      IF ( ISTAT .EQ. 1 )CALL YYSCAL
C
      XXCALS=ISTAT
      RETURN
      END
C
C##############################################################################
C
      SUBROUTINE XXDAO(VOLT)
$INCLUDE:'XX.CMN'
```

```
      REAL*4 VOLT
C
C     SUBROUTINE TO SET DACO TO VALUE DEPENDING ON VOLT
C
C     PARAMETERS
C     ----------
C     VOLT   (JNPUT  R*4)DESIRED OUTPUT VOLTAGE TO DACO
C                        -5 <= VOLT <= +5
C     VOLT   (OUTPUT R*4)ACTUAL OUTPUT VOLTAGE OF DACO
C                         ( MAY BE DIFFERENT DUE TO ROUNDING TO I*2 )
C
C     METHOD
C     ------
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     XXBDAO,YYDAO,XXVDAO
C
C     LOCAL DATA
C     ----------
C
      XXDACO=XXBDAO(VOLT)
      CALL YYDAO(XXDACO)
      VOLT=XXVDAO(XXDACO)
      RETURN
      END
C
C##############################################################################
C
      SUBROUTINE XXDA1(VOLT)
$INCLUDE:'XX.CMN'
      REAL*4 VOLT
C
C     SUBROUTINE TO SET DAC1 TO VALUE DEPENDING ON VOLT
C
C     PARAMETERS
```

```
C     ----------
C     VOLT   (INPUT  R*4)DESIRED OUTPUT VOLTAGE TO DAC1
C                        -5 <= VOLT <= +5
C     VOLT   (OUTPUT R*4)ACTUAL OUTPUT VOLTAGE AT DAC1
C                         ( MAY BE DIFFERENT DUE TO ROUNDING TO I*2 )
C
C     METHOD
C     ------
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     XXBDA1,YYDA1,XXVDA1
C
C     LOCAL DATA
C     ----------
C
C
      XXDAC1=XXBDA1(VOLT)
      CALL YYDA1(XXDAC1)
      VOLT=XXVDA1(XXDAC1)
      RETURN
      END
C
C
C##############################################################################
C
      SUBROUTINE XXMEAS(ISPC,IERR)
$INCLUDE:'XX.CMN'
      INTEGER*2 ISPC(LENSPC),IERR
C
C     SUBROUTINE TO MEASURE CHANNELS ACCORDING TO SPECIFICATIONS
C     IN ARRAY ISPC. RESULTS WILL BE PLACED IN ARRAY XXAD. IF DA
C     OUTPUT IS REQUIRED THIS SHOULD BE IN ARRAY XXDA.
C
C     PARAMETERS
C     ----------
```

```
C     ISPC  ( INPUT  I*2 ) ARRAY WITH MEASUREMENT PARAMETERS
C     IERR  ( OUTPUT I*2 ) ERROR RETURN
C                          0 =  NO ERROR
C                          1 =  NCHAN <=0
C                          2 =  NSAMP <=0
C                          10+I = OVERFLOW ON CHANNEL I
C
C     METHOD
C     ------
C     SAMPLING IS DONE AS SPECIFIED IN SPECIFICATIONS IN ARRAY ISPC
C     SAMPLING FREQUENCY ETC MUST BE SET BEFORE. ON EXIT OF SAMPLING
C     ROUTINE YYMEAS SAMPLES ARE STORED IN ARRAY XXDBUF IN SOMEWHAT
C     CUMBERSOME MANNER. HENCE SEQUENCE OF SAMPLES IS REORGANIZED IN
C     ARRAY XXAD SUCH THAT ON EXIT XXAD CONTAINS SAMPLES:
C     XXAD(1)...XXAD(ISPC(25)) = SAMPLES OF CHANNEL ISPC(4)
C     XXAD(ISPC(25)+1)...XXISP(2*ISPC(25)) = SAMPLES OF CHANNEL ISPC(5)
C     ETC.
C     AFTER THIS REORGANIZATION CHECK IS MADE ON OVERFLOW
C
C
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     YYMEAS
C
C     LOCAL DATA
C     ----------
      INTEGER*2 I,J,K
C
C
C     CHECK PARAMETERS
C
      IERR=0
      IF ( ISPC(3) .LT. 1 )THEN
         IERR=1
         RETURN
```

```
      ENDIF
      IF ( ISPC(25) .LT. 1 )THEN
         IERR=2
         RETURN
      ENDIF
C
C     MEASURE WITH AD VALUES IN ARRAY XXDBUF
C
      IF ( ISPC(8) .EQ. 0 )WRITE(*,10)
      IF ( ISPC(8) .EQ. 1 )WRITE(*,11)
      IF ( ISPC(8) .EQ. 2 )WRITE(*,12)
10    FORMAT(/,20X,'Sampling ....')
11    FORMAT(/,20X,'Sampling if external trigger ok ....')
12    FORMAT(/,20X,'Sampling if you want too ....',/)
      CALL YYMEAS(ISPC,XXDA,XXDBUF,IERR)
      WRITE(*,20)
20    FORMAT(/,20X,'Ready    !!!!')
C
C     REORGANIZE SAMPLES
C
      DO 100 K=1,ISPC(25)
      DO 50 J=1,ISPC(3)
      XXAD((J-1)*ISPC(25)+K)=XXDBUF((K-1)*ISPC(3)+J + 1 )
50    CONTINUE
100   CONTINUE
C
C     CHECK ON OVERFLOW
C
      DO 200 I=1,ISPC(3)
      DO 150 J=1,ISPC(25)
      K=(I-1)*ISPC(25)+J
      IF ( (XXAD(K) .EQ. -2048) .OR. (XXAD(K) .EQ. 2047) )THEN
      IERR=10+ISPC(3+I)
      GOTO 200
      ENDIF
150   CONTINUE
```

```
200   CONTINUE
C
C
      RETURN
      END
C
C##########################################################################
C
      SUBROUTINE XXWAIT(TWAIT)
$INCLUDE:'XX.CMN'
      REAL*4 TWAIT
C
C     SUBROUTINE TO WAIT FOR TWAIT SEC'S ( 0.05 <= TWAIT <= 327.0 )
C
C     PARAMETERS
C     ----------
C     TWAIT ( INPUT  R*4 ) NUMBER OF SECONDS TO WAIT
C
C     METHOD
C     ------
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     YYWAIT
C
C     LOCAL DATA
C     ----------
      INTEGER*2 IWAIT
C
C
      IF ( TWAIT .LT. 0.05 )TWAIT=0.05
      IF ( TWAIT .GT. 327.0)TWAIT=327.0
      IWAIT=IFIX(TWAIT/0.01)
      CALL YYWAIT(IWAIT)
C
      RETURN
```

```
      END
C
C#########################################################################
C
      SUBROUTINE XXPLAD(IFIRST,NP)
$INCLUDE:'XX.CMN'
      INTEGER*2 IFIRST,NP
C
C     SUBROUTINE TO PLOT DATA IN ARRAY XXAD,STARTING FROM POSITION
C     XXAD(IFIRST)
C
C     PARAMETERS
C     ----------
C     IFIRST (INPUT  I*2)FIRST POINT IN XXAD TO PLOT
C     NP     (INPUT  I*2)NUMBER OF POINTS TO PLOT
C
C     METHOD
C     ------
C
C     A PICTURE IS PLOT IN CURRENT GRAPHICS MODE WHICH HAS PREVIOUSLY
C     BEEN SET AND IS INDICATED IN PARAMETER XXPMOD. VALUES IN XXAD
C     ARE TRANSFORMED TO VOLTS AND NEXT TO SCREEN COORDINATES SUCH THAT
C     THE RANGE OF THE Y-AXIS CORRESPONDS TO RANGE OF AD CONVERTER
C     (-5 TO +5 VOLT)
C     THE HORIZONTAL AXIS IS DIVIDED INTO NP INTERVALS
C     COLOR VALUES FOR LINES ETC MUST BE PREVIOUSLY SET AND ARE STORED
C     IN COMMON-BLOCK XX3
C     CURRENT GRAPHICS MODE CAN BE SET USING HALO ROUTINES INITGR AND
C     SETIEE:
C     CALL INITGR(XXPMOD)
C     CALL SETIEE(XXIEEE)
C     WHERE XXPMOD=0 : 320*200 MODE
C                  1 : 640*200 MODE
C           XXIEEE=1 : NO 8087 INSTALLED
C           XXIEEE=0 : 8087 INSTALLED
C     TO RETURN TO 80*25 ALPHANUMERIC MODE USE HALO ROUTINE CLOSEG:
```

```
C     CALL CLOSEG
C
C
C     SUBROUTINES AND FUNCTIONS USED ( FROM HALO LIBRARY )
C     ----------------------------
C     SETPAL,SETCOL,CLR,SETLNS,SETLNW,MOVABS,LNABS
C
C     LOCAL DATA
C     ----------
      INTEGER*2 I,J,JBEGIN,IXP,IYP,IYVAL
      REAL*4 BOTX,UPPX,BOTY,UPPY
C
C
C
C     SELECT BACKGROUND,FOREGROUND,LINESTYLE,LINEWIDTH AND COLOR
C     FOR AXES
C
      CALL SETPAL(XXBOCL,XXPAL)
      CALL SETCOL(XXBACL)
      CALL CLR
      CALL SETPAL(XXBOCL,XXPAL)
      CALL SETLNS(1)
      CALL SETLNW(1)
      CALL SETCOL(XXAXCL)
C
C     DRAW BORDER=AXES
C
      IF ( XXPMOD .EQ. 0 )BOTX=10.0
      IF ( XXPMOD .EQ. 0 )UPPX=310.0
      IF ( XXPMOD .EQ. 0 )BOTY=10.0
      IF ( XXPMOD .EQ. 0 )UPPY=190.0
      IF ( XXPMOD .EQ. 1 )BOTX=10.0
      IF ( XXPMOD .EQ. 1 )UPPX=630.0
      IF ( XXPMOD .EQ. 1 )BOTY=10.0
      IF ( XXPMOD .EQ. 1 )UPPY=190.0
      CALL MOVABS(IFIX(BOTX-1.0),IFIX(BOTY-1.0))
```

```
      CALL LNABS(IFIX(BOTX-1.0),IFIX(UPPY+1.0))
      CALL LNABS(IFIX(UPPX+1.0),IFIX(UPPY+1.0))
      CALL LNABS(IFIX(UPPX+1.0),IFIX(BOTY-1.0))
      CALL LNABS(IFIX(BOTX-1.0),IFIX(BOTY-1.0))
C
C     LABEL AXES AT 10 INTERVALS
C
      DO 2110 I=1,9
      JBEGIN=IFIX(BOTX+(UPPX-BOTX)*FLOAT(I)/10.0)
      CALL MOVABS(JBEGIN,IFIX(UPPY)-1)
      CALL LNABS(JBEGIN,IFIX(UPPY)+3)
      JBEGIN=IFIX(BOTY+(UPPY-BOTY)*FLOAT(I)/10.0)
      CALL MOVABS(IFIX(UPPX)+1,JBEGIN)
      CALL LNABS(IFIX(UPPX)+6,JBEGIN)
2110  CONTINUE
C
C     DRAW VALUES AFTER TRANSFORMATION TO SCREEN COORDINATES
C
      CALL SETCOL(XXLNCL)
C
C     DATA STARTS AT XXAD(IFIRST)
C
      IXP=IFIX(BOTX)
      IYVAL=XXAD(IFIRST)
      IYP=IFIX(BOTY+FLOAT(2047-IYVAL)*(UPPY-BOTY)/4095.0)
      CALL MOVABS(IXP,IYP)
      IF ( NP .LT. 2 )GOTO 2300
      DO 2200 I=2,NP
      IXP=IFIX(BOTX+FLOAT(I-1)*(UPPX-BOTX)/FLOAT(NP-1))
      IYVAL=XXAD(IFIRST+I-1)
      IYP=IFIX(BOTY+FLOAT(2047-IYVAL)*(UPPY-BOTY)/4095.)
      CALL LNABS(IXP,IYP)
2200  CONTINUE
2300  RETURN
      END
C
```

```
C##############################################################################
C
      SUBROUTINE XXPLDA(IFIRST,NP)
$INCLUDE:'XX.CMN'
      INTEGER*2 IFIRST,NP
C
C     SUBROUTINE TO PLOT DATA IN ARRAY XXDA,STARTING FROM POSITION
C     XXDA(IFIRST)
C
C     PARAMETERS
C     ----------
C     IFIRST (INPUT  I*2)FIRST POINT IN XXDA TO PLOT
C     NP     (INPUT  I*2)NUMBER OF POINTS TO PLOT
C
C     METHOD
C     ------
C
C     A PICTURE IS PLOT IN CURRENT GRAPHICS MODE WHICH HAS PREVIOUSLY
C     BEEN SET AND IS INDICATED IN PARAMETER XXPMOD. VALUES IN XXDA
C     ARE TRANSFORMED TO VOLTS AND NEXT TO SCREEN COORDINATES SUCH THAT
C     THE RANGE OF THE Y-AXIS CORRESPONDS TO RANGE OF DA CONVERTER
C     (-5 TO +5 VOLT)
C     THE HORIZONTAL AXIS IS DIVIDED INTO NP INTERVALS
C     COLOR VALUES FOR LINES ETC MUST BE PREVIOUSLY SET AND ARE STORED
C     IN COMMON-BLOCK XX3
C     CURRENT GRAPHICS MODE CAN BE SET USING HALO ROUTINES INITGR AND
C     SETIEE:
C     CALL INITGR(XXPMOD)
C     CALL SETIEE(XXIEEE)
C     WHERE XXPMOD=0 : 320*200 MODE
C                  1 : 640*200 MODE
C           XXIEEE=1 : NO 8087 INSTALLED
C           XXIEEE=0 : 8087 INSTALLED
C     TO RETURN TO 80*25 ALPHANUMERIC MODE USE HALO ROUTINE CLOSEG:
C     CALL CLOSEG
C
```

```
C
C     SUBROUTINES AND FUNCTIONS USED ( FROM HALO LIBRARY )
C     ----------------------------
C     SETPAL,SETCOL,CLR,SETLNS,SETLNW,MOVABS,LNABS
C
C     LOCAL DATA
C     ----------
      INTEGER*2 I,J,JBEGIN,IXP,IYP,IYVAL
      REAL*4 BOTX,UPPX,BOTY,UPPY
C
C
C
C     SELECT BACKGROUND,FOREGROUND,LINESTYLE,LINEWIDTH AND COLOR
C     FOR AXES
C
      CALL SETPAL(XXBOCL,XXPAL)
      CALL SETCOL(XXBACL)
      CALL CLR
      CALL SETPAL(XXBOCL,XXPAL)
      CALL SETLNS(1)
      CALL SETLNW(1)
      CALL SETCOL(XXAXCL)
C
C     DRAW BORDER=AXES
C
      IF ( XXPMOD .EQ. 0 )BOTX=10.0
      IF ( XXPMOD .EQ. 0 )UPPX=310.0
      IF ( XXPMOD .EQ. 0 )BOTY=10.0
      IF ( XXPMOD .EQ. 0 )UPPY=190.0
      IF ( XXPMOD .EQ. 1 )BOTX=10.0
      IF ( XXPMOD .EQ. 1 )UPPX=630.0
      IF ( XXPMOD .EQ. 1 )BOTY=10.0
      IF ( XXPMOD .EQ. 1 )UPPY=190.0
      CALL MOVABS(IFIX(BOTX-1.0),IFIX(BOTY-1.0))
      CALL LNABS(IFIX(BOTX-1.0),IFIX(UPPY+1.0))
      CALL LNABS(IFIX(UPPX+1.0),IFIX(UPPY+1.0))
```

```
      CALL LNABS(IFIX(UPPX+1.0),IFIX(BOTY-1.0))
      CALL LNABS(IFIX(BOTX-1.0),IFIX(BOTY-1.0))
C
C     LABEL AXES AT 10 INTERVALS
C
      DO 2110 I=1,9
      JBEGIN=IFIX(BOTX+(UPPX-BOTX)*FLOAT(I)/10.0)
      CALL MOVABS(JBEGIN,IFIX(UPPY)-1)
      CALL LNABS(JBEGIN,IFIX(UPPY)+3)
      JBEGIN=IFIX(BOTY+(UPPY-BOTY)*FLOAT(I)/10.0)
      CALL MOVABS(IFIX(UPPX)+1,JBEGIN)
      CALL LNABS(IFIX(UPPX)+6,JBEGIN)
2110  CONTINUE
C
C     DRAW VALUES AFTER TRANSFORMATION TO SCREEN COORDINATES
C
      CALL SETCOL(XXLNCL)
C
C     DATA STARTS AT XXDA(IFIRST)
C
      IXP=IFIX(BOTX)
      IYVAL=XXDA(IFIRST)
      IYP=IFIX(BOTY+FLOAT(255-IYVAL)*(UPPY-BOTY)/255.0)
      CALL MOVABS(IXP,IYP)
      IF ( NP .LT. 2 )GOTO 2300
      DO 2200 I=2,NP
      IXP=IFIX(BOTX+FLOAT(I-1)*(UPPX-BOTX)/FLOAT(NP-1))
      IYVAL=XXDA(IFIRST+I-1)
      IYP=IFIX(BOTY+FLOAT(255-IYVAL)*(UPPY-BOTY)/255.)
      CALL LNABS(IXP,IYP)
2200  CONTINUE
2300  RETURN
      END
C
C#######################################################################
C
```

```
      SUBROUTINE XXFINI
$INCLUDE:'XX.CMN'
C
C     ROUTINE TO RESET INTERFACE TO DEFAULT PARAMETERS
C
C     PARAMETERS
C     ----------
C
C     METHOD
C     ------
C
C     SUBROUTINES AND FUNCTIONS USED
C     ------------------------------
C     YYFINI
C
C     LOCAL DATA
C     ----------
C
C
C     RESET INTERFACE
C
      CALL YYFINI
C
      RETURN
      END
C
C#############################################################################
```

Appendix D Een eenvoudig meetprogramma

In het volgende volgt een eenvoudig meetprogramma. Stel we willen van kanaal 2 van het meetsyteem 3000 samples met een sample frequentie van 1000 Hz. en een filterfrequentie van 1000 Hz., zonder DA tijdens de meting. Tevens willen we de gemeten waarden in een plotje hebben. Dan kan dat als volgt in Fortran 77 op de PC:

```
$STORAGE:2
$INCLUDE:'XX.CMN'
      CHARACTER*1 CDUM
      REAL*4 SAMPFR,FILTFR
      INTEGER*2 IERR
      SAMPFR=3000.0
      FILTFR=1000.0
C     INITIALISEER MEETEENHEID, SAMPLE FREQUENTIE EN FILTER FREQUENTIE
      CALL XXINIT
      CALL XXSAFR(SAMPFR,XXISP(20),XXISP(21))
      CALL XXFCOF(FILTFR,XXISP(22),XXISP(23))
C     VUL XXISP MET DE GOEDE SPECIFICATIES VOOR ZOVER NODIG NA HET AANROEPEN
C     VAN XXINIT
      XXISP(3)=1
      XXISP(4)=2
      XXISP(25)=3000
C     DOE DE METING
      CALL XXMEAS(XXISP,IERR)
      IF ( IERR .NE. 0 ) WRITE(*,10)IERR
10    FORMAT(/,1X,' ERROR IN XXMEAS NO:',1X,I5)
C     EN PLOT HET PLAATJE MET AD WAARDEN
      CALL INITGR(XXPMOD)
      CALL SETIEE(XXIEEE)
      CALL XXPLAD(1,XXISP(25))
      READ(*,*)CDUM
      CALL CLOSEG
C     RESET MEETEENHEID VOOR VERLATEN PROGRAMMA
```

```
CALL XXFINI
CALL EXIT
END
```

APPENDIX E De anti-aliasing filters

De anti-aliasing filters (zie [3]) zijn in feite 8^e orde Butterworth filters (48 db/oct) met een programmeerbare kantelfrequentie (-3db punt) f_c. Mbv de HP5423A wave analyser zijn de overdrachtsfuncties van de 4 filters doorgemeten voor een kantelfrequentie van 100 Hz. (zie fig. E.1.a - E.1.c met CHANO = kanaal 0 etc.). We zien dat de filters onderling redelijk goed overeenstemmen (afwijkingen circa 1%). Theoretisch gezien moeten de filters voldoen aan de volgende uitdrukking voor de overdrachtsfunctie:

$$H(f) = \prod_{k=1}^{4} \frac{1}{1 - \lambda^2 + 2j\alpha_k\lambda}$$

met

$$\lambda = f/f_c$$

$$\alpha_k = \cos\left(\frac{2k-1}{16} \pi \right)$$

ofwel

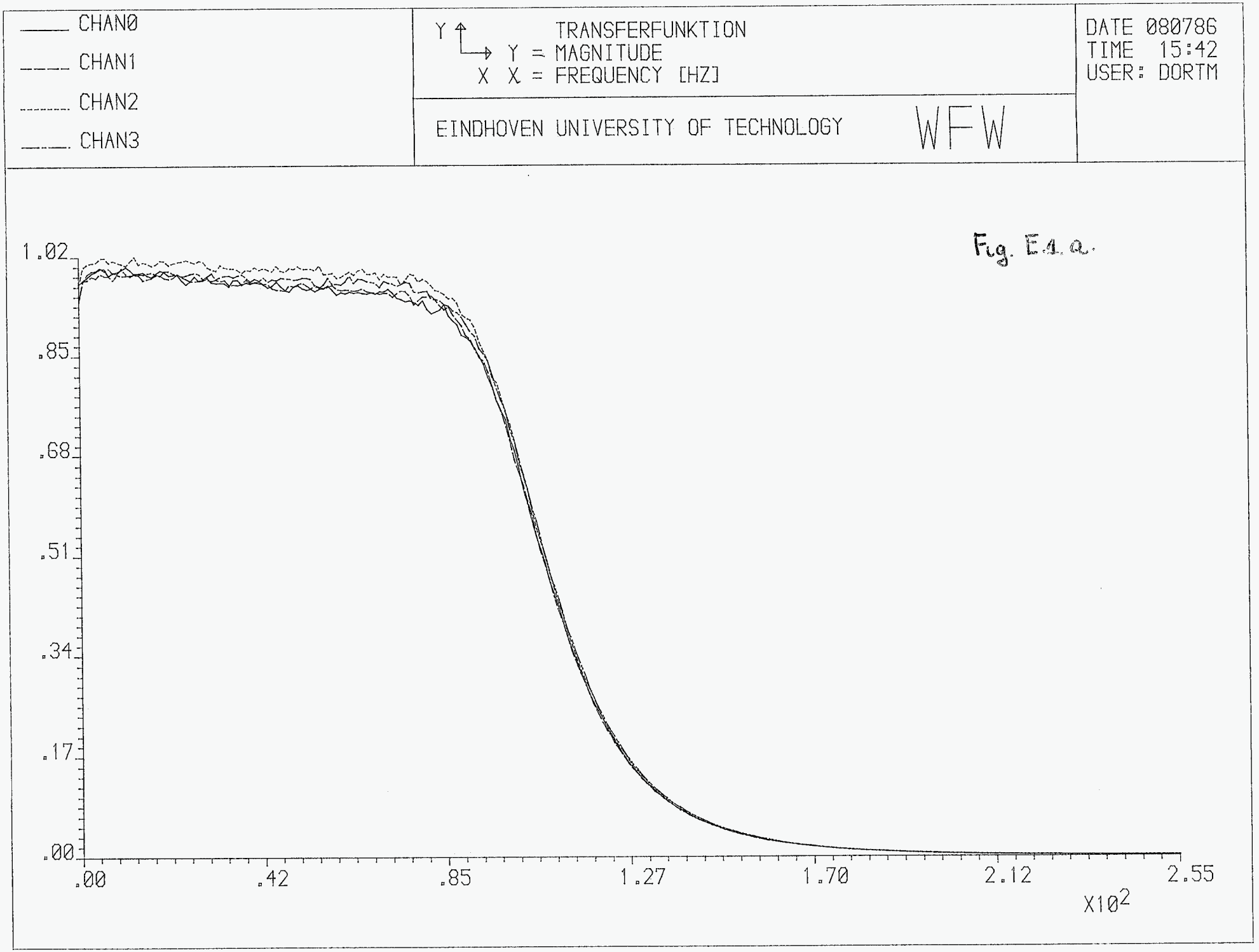
$$|H(f)| = \frac{1}{1 + \lambda^{16}}$$

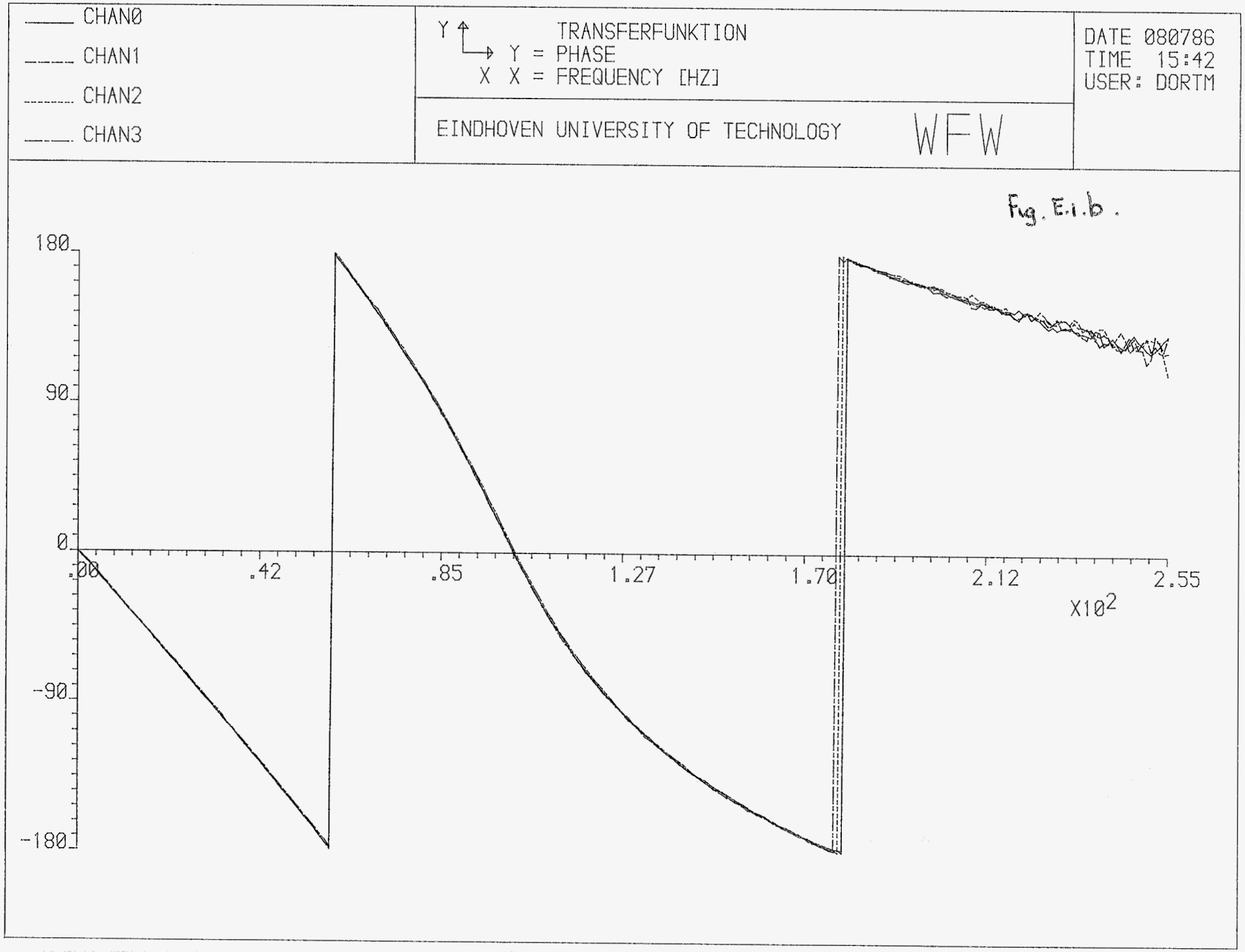
$$\varphi = \arg\left(H(f) \right) = - \sum_{k=1}^{4} \arctan\left(\frac{2\alpha_k\lambda}{1 - \lambda^2} \right)$$

Uit de figuren E.2.a - E.2.c blijkt dat althans voor kanaal 0 dir redelijk klopt.

Wanneer we de phasedraaiing van de filters bekijken dan blijkt dat deze tot aan de kantelfrequentie redelijk goed lineair verloopt (daarboven is niet interessant ivm de afzwakking). Deze lineaire phasedraaiing zorgt ervoor dat een signaal met verschillende frequentie componenten weliswaar afgezwakt wordt maar de tijdsverschuiving (tgv de phasedraaiing) voor alle

frequenties even groot is. Dat wil zeggen dat we met de phasedraaiing geen rekening hoeven te houden (hetgeen bijv. voor een eenvoudig Rc filter niet het geval is !).


CHAN0
CHAN1
CHAN2
CHAN3
Y
X
TRANSFERFUNKTION
Y = MAGNITUDE
X = FREQUENCY [HZ]
EINDHOVEN UNIVERSITY OF TECHNOLOGY
WFW
DATE 080786
TIME 15:42
USER: DORTM
Fig. E.1.a.
1.02
.85
.68
.51
.34
.17
.00
.00
.42
.85
1.27
1.70
2.12
2.55
X10^2


CHAN0
CHAN1
CHAN2
CHAN3
Y
X
TRANSFERFUNKTION
Y = PHASE
X = FREQUENCY [HZ]
EINDHOVEN UNIVERSITY OF TECHNOLOGY
WFW
DATE 080786
TIME 15:42
USER: DORTM
Fig. E.1.b.
180
90
0
-90
-180
.00
.42
.85
1.27
1.70
2.12
2.55
X10^2

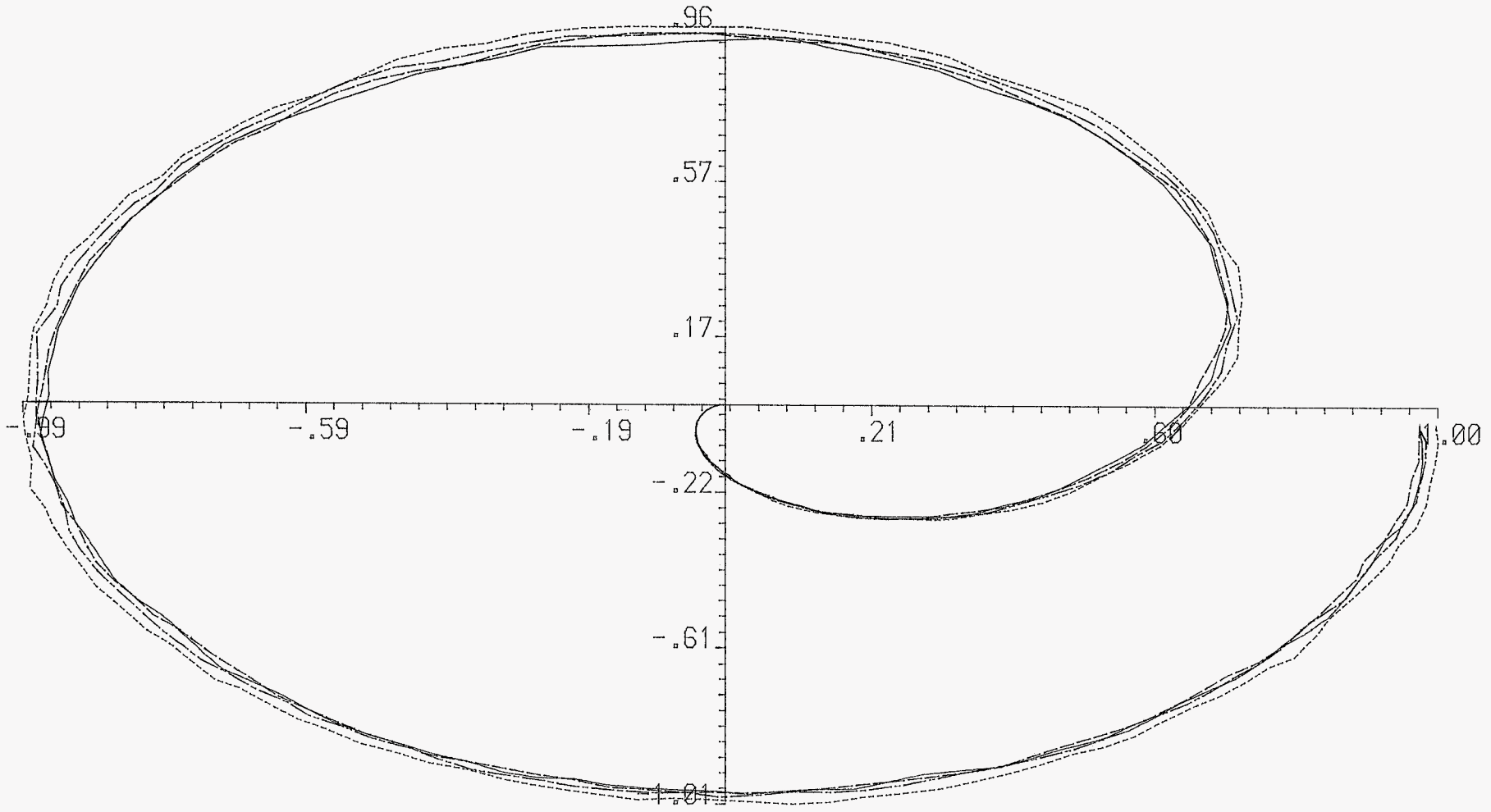
CHAN0
CHAN1
CHAN2
CHAN3

Y ↑ TRANSFERFUNKTION
Y = NYQUIST IMAG
X X = NYQUIST REAL

EINDHOVEN UNIVERSITY OF TECHNOLOGY WFW

DATE 080786
TIME 15:43
USER: DORTM

Fig. E.1.c.



CHAN0

CHAN0 FIT

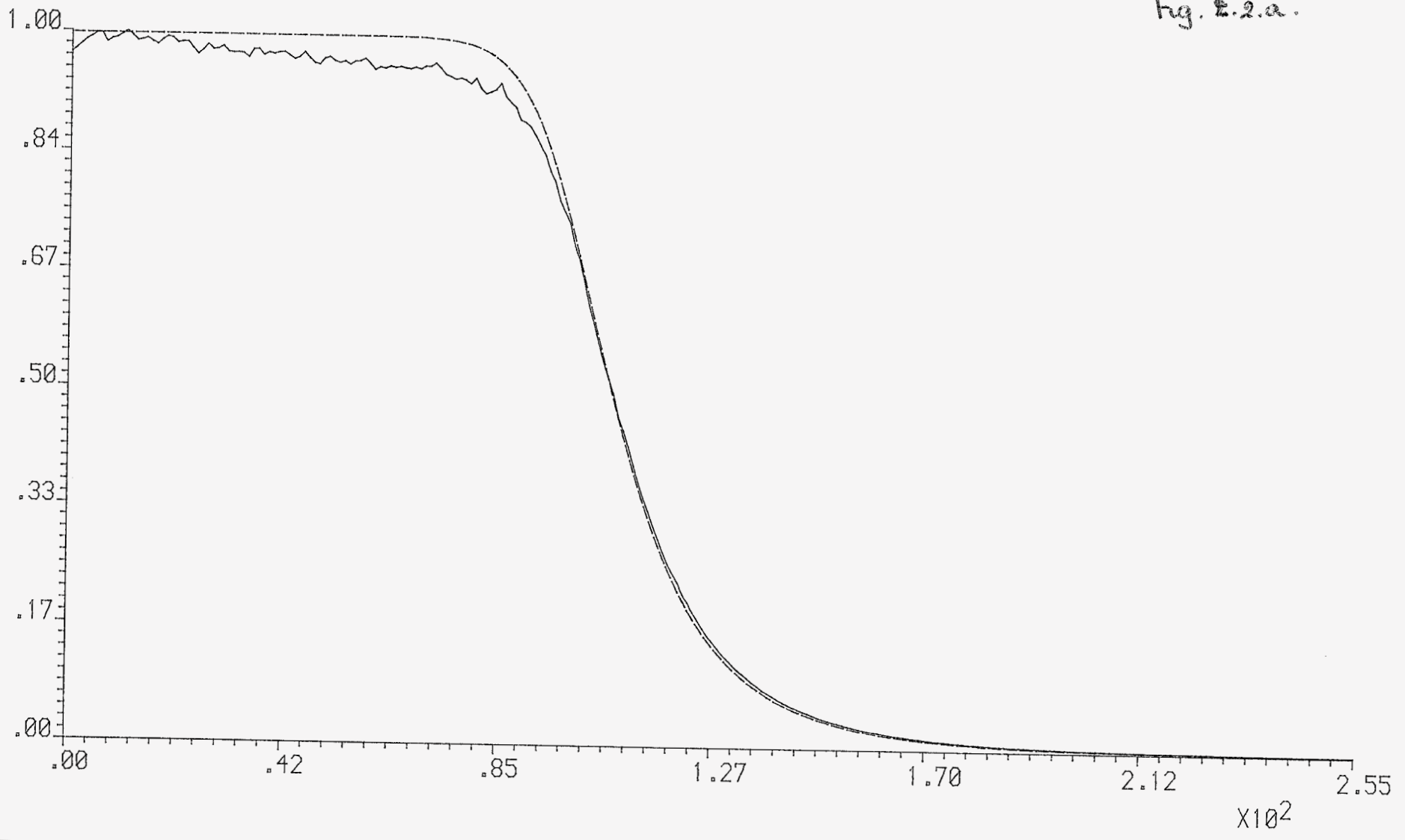
Y
X

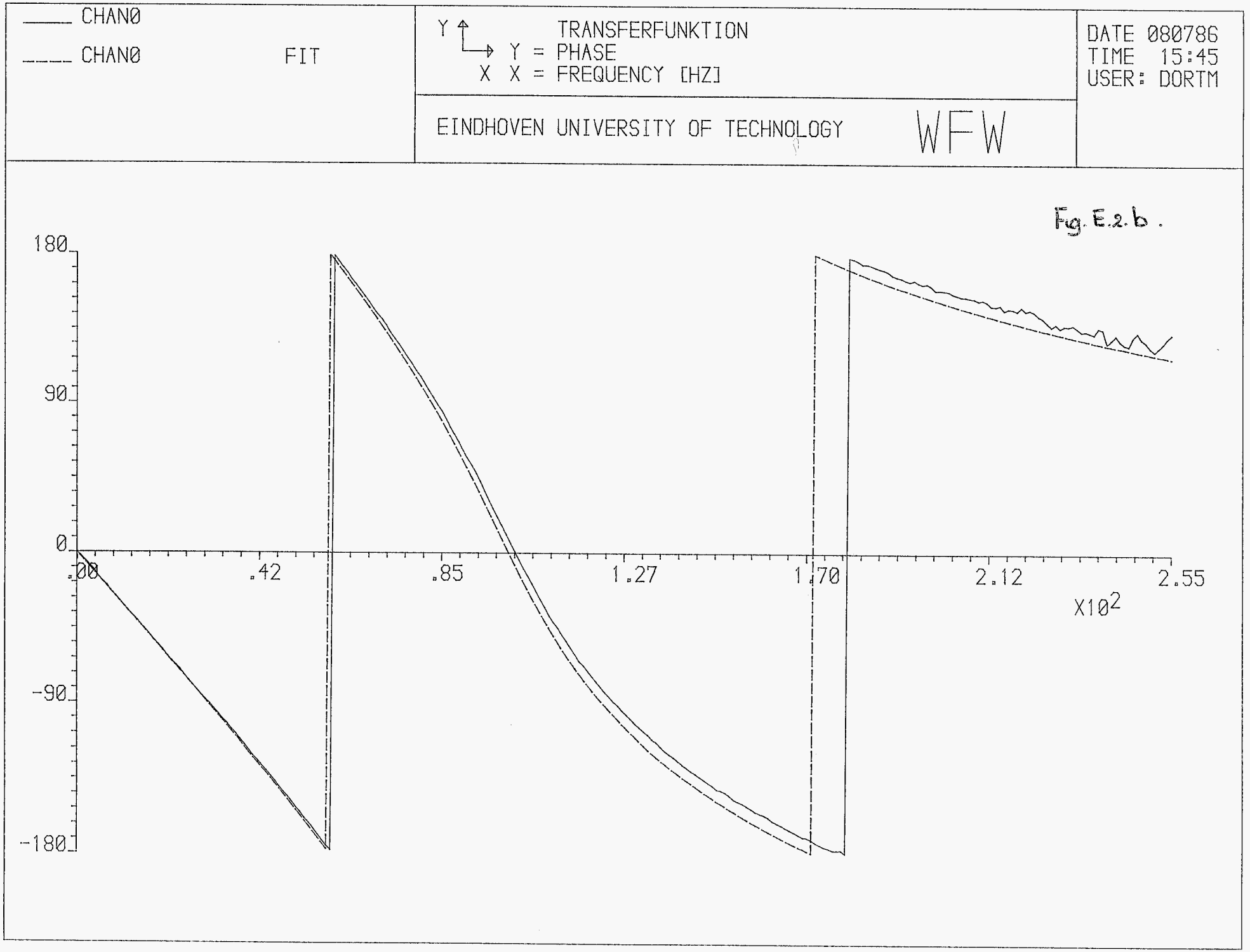
TRANSFERFUNKTION
Y = MAGNITUDE
X = FREQUENCY [HZ]

DATE 080786
TIME 15:45
USER: DORTM

EINDHOVEN UNIVERSITY OF TECHNOLOGY WFW

Fig. E.2.a.




CHAN0
CHAN0 FIT
Y
X
TRANSFERFUNKTION
Y = PHASE
X = FREQUENCY [HZ]
EINDHOVEN UNIVERSITY OF TECHNOLOGY
WFW
DATE 080786
TIME 15:45
USER: DORTM
Fig. E.2.b.
180
90
0
-90
-180
.00
.42
.85
1.27
1.70
2.12
2.55
X10^2

CHAN0

CHAN0 FIT

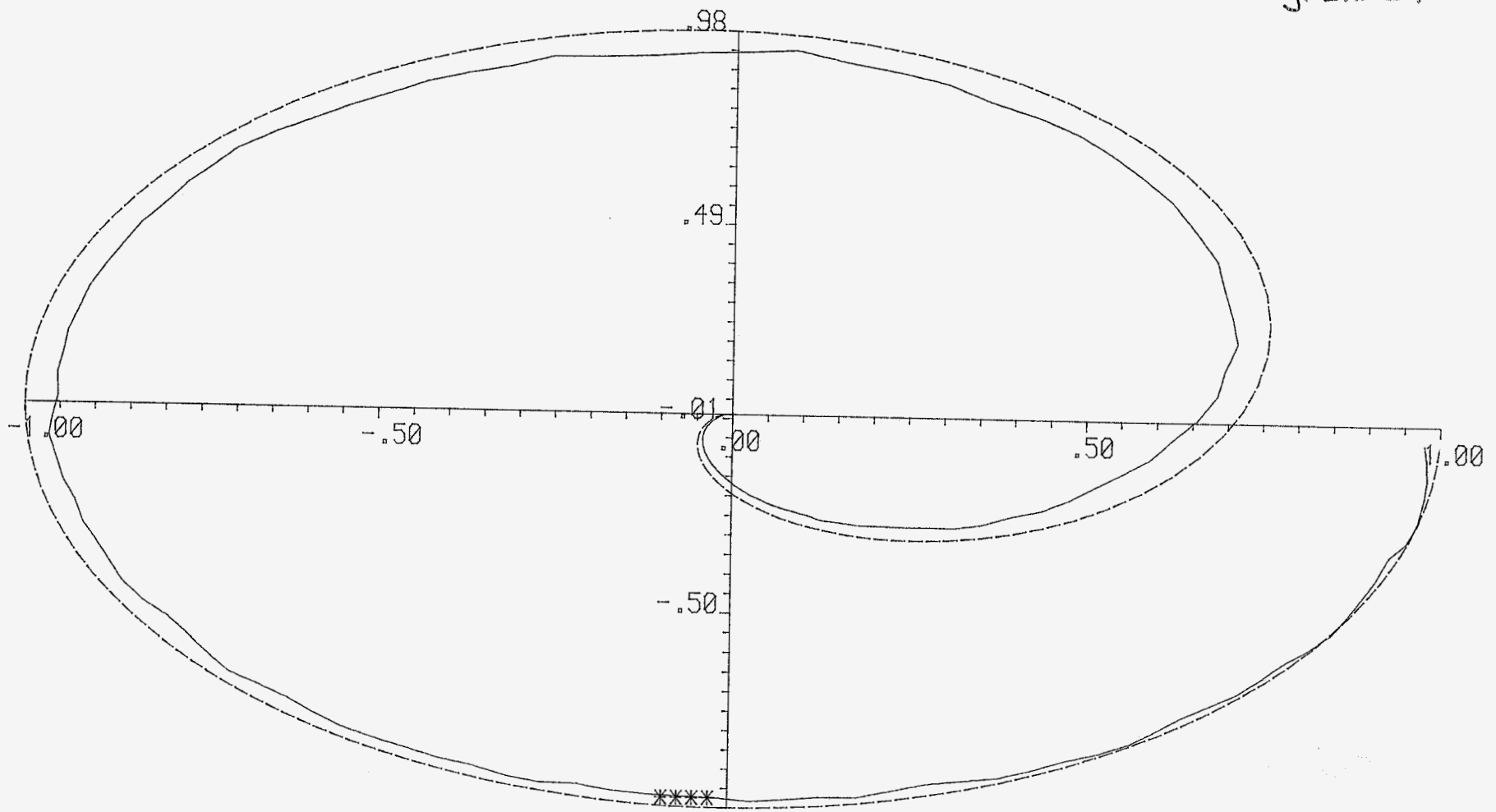
Y
X

TRANSFERFUNKTION
Y = NYQUIST IMAG
X = NYQUIST REAL

DATE 080786
TIME 15:44
USER: DORTM

EINDHOVEN UNIVERSITY OF TECHNOLOGY WFW

Fig. E.2.c.



APPENDIX F OPBOUW VAN DATAFILES GEGENEREERT MBV. HET 16 KANAALS MEETSYSTEEM

Om een datafile te maken die compatibel is met die die gemaakt worden met het 16 kanaals meetsysteem moeten we een file maken die er als volgt uitziet:

- 25 integer*2 (50 bytes) (stel XXISP(1) ... XXISP(25))
- 320 character*1 (320 bytes) (stel XXCSP(1) ... XXCSP(320))
- 64 real*4 (256 bytes) (stel XXRSP(1) ... XXRSP(64))
- 224 character*1 (224 bytes) (stel XXDSP(1) ... XXDSP(224))
- meetwaarden in integer*2 formaat (tussen -2048 en +2047)

De header van de file tot aan de meetwaarden bevat specificaties van de meting en moet 1074 bytes lang zijn. De variabelen in de header hebben de volgende betekenis:

- XXISP
  XXISP heeft dezelfde betekenis als gegeven in par. 7.1. behalve dat nu XXISP(3) = 1 en XXISP(4) = kanaalnummer van de meetwaarden in de file = K (K = 0,1,2 of 3)

- XXCSP
  XXCSP(1+K*20) ... XXCSP((K+1)*20) bevat 20 charcter*1 variabelen die kunnen dienen voor het opslaan van een "usercode" bij kanaal K
  Indien gewenst kunnen we deze 320 characters ook opvatten als 160 integer*2 variabelen die dan in A2 formaat de usercode bevatten

- XXRSP
  XXRSP(1+K) bevat de offset van de opnemer die gekoppeld is aan kanaal K (eenheid Volt)
  XXRSP(17+K) bevat de versterking van de opnemer die gekoppeld is aan kanaal K (eenheid Volt/mechanical unit)
  XXRSP(33+K) bevat de offset van de filterunit van kanaal K
  XXRSP(49+K) bevat de versterking van de filterunit van kanaal K

- XXDSP

  XXDSP is nog niet gebruikt en kan derhalve vrij ingekleed worden.

  (bijvoorbeeld 224 character*1, 112 integer*2 of 56 real*4)

Na de header van de file volgen direct de meetwaarden in integer*2 formaat.
Het aantal meetwaarden wordt gegeven door XXISP(25).