PROGRAMMERING AV PI-REGULATOR SAMT OLIKA STÄLLORGAN FÖR PROCESSDATAMASKIN

LEIF STEEN

Rapport RE - 39 okt. 1968

PROGRAMMERING AV PI-REGULATOR
SAMT OLIKA STÄLLORGAN
FÖR PROCESSDATAMASKIN

EXAMENSARBETE I ÄMNET REGLERTEKNIK

Utfört sommaren 1968 av

Leif Steen

I N N E H Å L L S F Ö R T E C K N I N G

AVDELNING A

Input/Output vid processdatamaskiner.

Kapitel I. Analog/Digital omvandling och Digital/Analog omvandling

Analog/Digitalomvandlare och Digital/Analogomvandlare tillhör numera ofta datamaskinernas primära kringutrustning. Orsakerna till att man vill använda DM är naturligtvis de nästa obegränsade möjligheter att lagra, administrera och bearbeta stora mängder data (t.ex mätresultat) som dessa erbjuder. Den höga mätnoggrannhet som digital mätning erbjuder är en annan positiv faktor.

En blick på vad man kan åstadkomma med ett AD-system ger som inledning en större förståelse för dess möjligheter. Med ett AD-system kan man exempelvis ...

- snabbt göra tiotusentals mätningar under tiden en månsarkost avfyras
- styra valsverk med hög precision
- avkoda signaler som sänds från en satelit i ständigt kretslopp runt jorden
- kontrollera och styra den elektriska distributionsapparaten
- göra biomedicinska simuleringar och därigenom rädda tusentals människoliv
- försäkra sig om att en dödsbringande interkontinental robot är redo att släcka lika många liv.

Vanligen tar AD-systemet emot reell-tidsfakta, både av analog och digital typ från en yttre källa. Dessa data används för att:

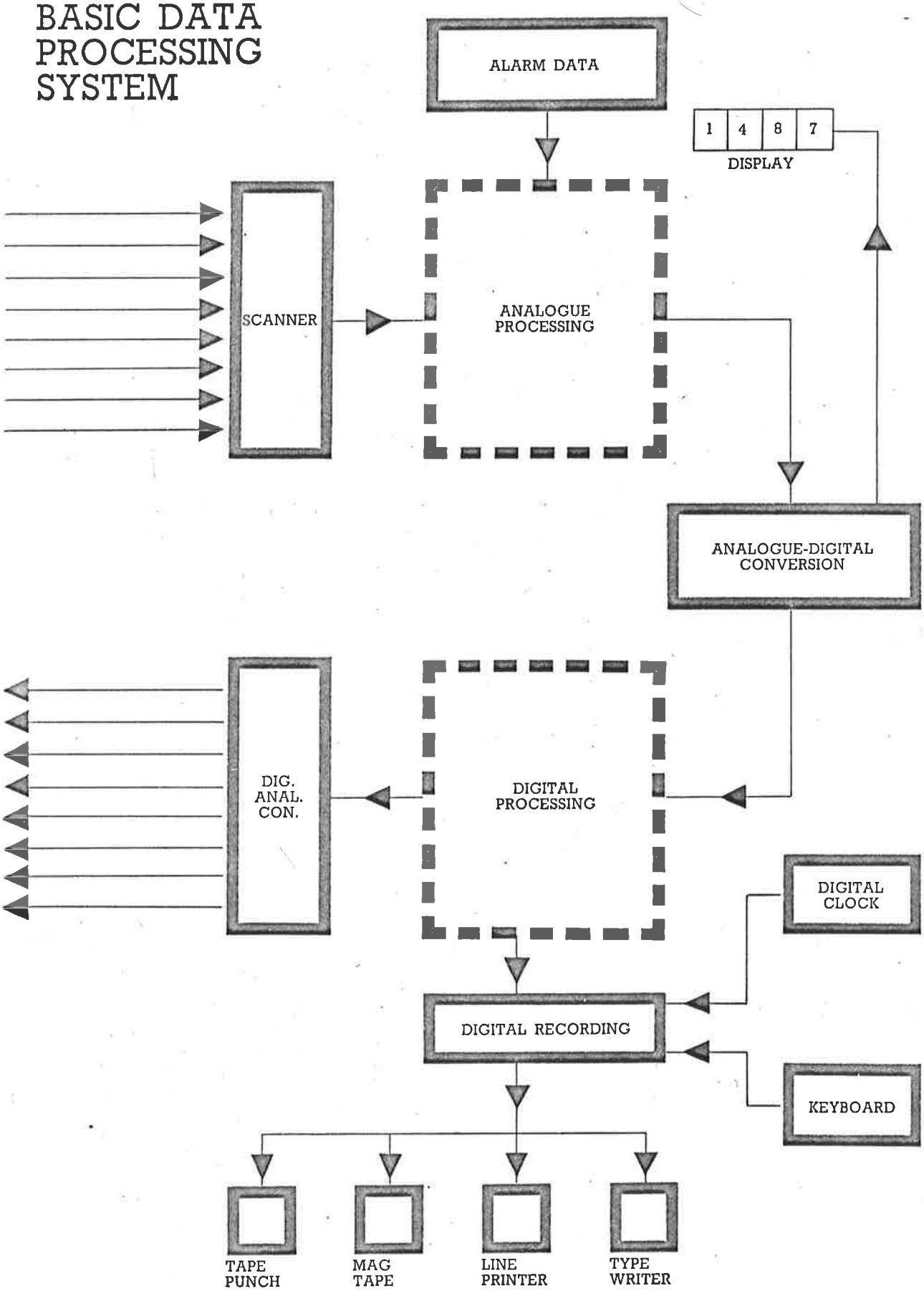
- presentera data för omedelbar användning (on-line) och/eller bevara den för kommande bruk (off-line)
- analysera data för att detektera onormala tillstånd i processer

- använda dessa data för att generera styrsignaler för den externa processen.

Tidigare AD-system användes huvudsakligen för att spela in data. Moderna AD-system kan om de körs on-line med en datamaskin utföra alla tre operationerna lika lätt.

Ett exempel på ett 'Basic Data Processing System' finns i figuren på nästa sida. Där framgår A/D och D/A omvandlingens centrala plats i systemet. Figuren är hämtad ur referens 12.

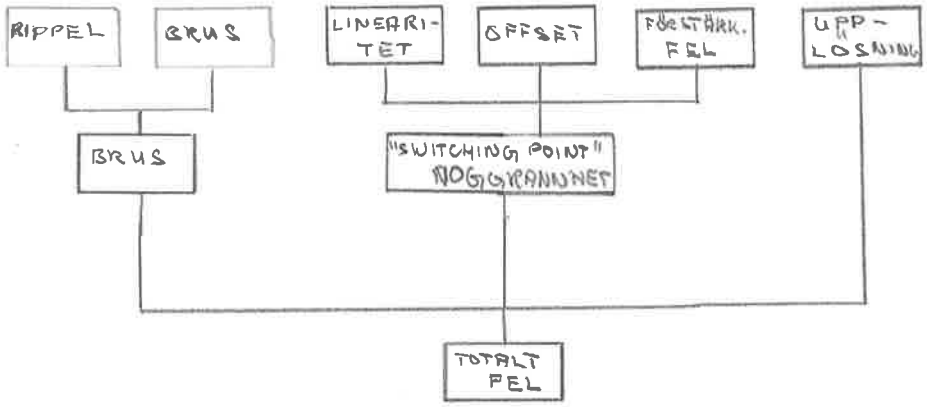
BASIC DATA PROCESSING SYSTEM



I.1 NOGGRANNHET

Insikten i hur noggrann en omvandling blir, är naturligtvis mycket väsentlig. När det gäller system där precisionskraven inte är påfallande stränga, räcker som regel en procentsiffra för hela systemet (t. ex. ≥ 1 %). Vid mätningar som kräver noggrannhet ≥ 0.1 % bör både digitala och analoga felkällor specificeras.

Speciellt i system med hög precision ingår inte isolerade felkällor (t. ex. brus) i de allmäna specifikationerna. Det är därför viktigt att känna till de olika feltyperna, deras orsaker, hur de mäts och specificeras och när de är viktiga. Ett blockschema ger en klarare bild av felens additivitet.



I.1.a Digitala felkällor

När en kontinuerlig signal upplöses i kvanta, uppstår ett fel som kan vara upp till minsta kvantats storlek. I en lineär omvandlare är det minsta kvantat lika med den minst signifikanta biten. De flesta omvandlare har upplösningsfelet centrerat, så att det är $\pm$ 1/2 LSB (dvs. $\pm$ 1/2 av den minst signifikanta biten, LSB = least signifikant bit).

I.1.b Analoga felkällor

Omvandlarens dc-precision (eller switching point accuracy) är avhängig av förstärkarens off-setström, kalibrering och linearitet. Olinearitеter beror på förändringar i förstärkningen (eller "common mode" effekter) när inspänningen förändras. Offset och förstärkning kan justeras tills dessa fel är försumbara.

Det analoga felet hos en DA-omvandlare mäter man lätt genom att stoppa in ett digitalt värde på ingången och mäta utsignalen. Upplösningsfelet gör det svårare att lokalisera det analoga felet hos en AD-omvandlare. Spänningen för vilken utgången slår över från ett digitalt värde till ett närliggande, är emellertid ganska väldefinierad. Den kan mätas och jämföras med det teoretiska värdet. Den nämda ändringen i digitalvärde sker vid "the switching point".

Rippel hos referensspännigen och annat brus mäts ofta separat. Orsaken till detta är bland annat att vissa av dessa fel ibland kan försummas i slutresultatet.

I.1.c Differential linearitet

Differentiell linearitet är variationen i storlek hos den spänning som krävs för att få en AD-omvandlare att gå från en switching point till nästa. Det är således en variation i de olika tillståndens storlek. Värdet ges vanligen i procent av tillståndsstorleken.

Differentiallinearitеten är en del av den totala linearitеten, men förtjänar speciell uppmärksamhet när AD-omvandlaren används i histogram-tillämpningar. När man t. ex. plottar antalet ingångar mot deras digitala värde och ett tillstånd är dubbelt så stort som ett närliggande tenderar den att göra dubbelt så många markeringar.

Differentiallinearitet är ettbav de få noggrannhetsfel som speciellt drabbar omvandlingstekniken. Den blir bäst när omvandlaren går igenom alla tillstånd sekventiellt. Ju kortare ordlängd, desto

bättre blir naturligtvis den differentiella linearitetens. Dålig upplösning innebär emellertid att man förlorar eller jämnar ut skarpa toppar i histogrammet.

Det finns några lösningar för att komma tillrätta med den differentiella linearitetens: man kan ändra offset eller ordlängden hos omvandlaren. För dessa ändamål monteras ofta omkastare på apparaten. Man kan också programmera så att det kontrollerande organet kan göra förändringen automatiskt.

I.1.d Felfördelning

För omvandlare med 10 eller 11 bitar består det digitala felet vanligen av 1/3 till 1/2 av totala felet. Sålunda har ett typiskt 10-bitars system ett upplösningsfel på ± 1/2 LSB och ett analogt fel på 0.1 %.

Om kravet på noggrannheten är lågt, kan ordlängden vara den största felkällan. Totala felet kan betraktas som en avrundning. Vid hårdare ställda krav på precision, är det önskvärt att man kan minimera de olika feltyperna, både de analoga och digitala var för sig. Det digitala felet är lätt att göra mindre, man ökar helt enkelt antalet bitar. Naturligtvis bör man hålla sig inom praktiska gränser, en omvandlare med ett totalt fel på 0.1 % och med ordlängden 20 bitar är t. ex knappast försvarbar.

I.1.e Monotonitet

Monotonitet är ett sätt att försäkra sig om att alla bitar är meningsfulla. Det betyder att alla tillstånd måste existera och komma i rätt ordning. När värdet på DA-omvandlarens ingång ökas, ska utgångsspänningen också öka. Den får absolut inte minska. På liknande sätt måste ökningen av ingångsspänningen på AD-omvandlaren medföra att det digitala utgångsvärdet är lika eller ökar. Det får aldrig hoppa över några tillstånd. Mest sannolikt är att monotoniteten förloras då man ändrar mellan de digitala tillstånden 0111... och 1000...

I.2 Typer av systemnoggrannhet.

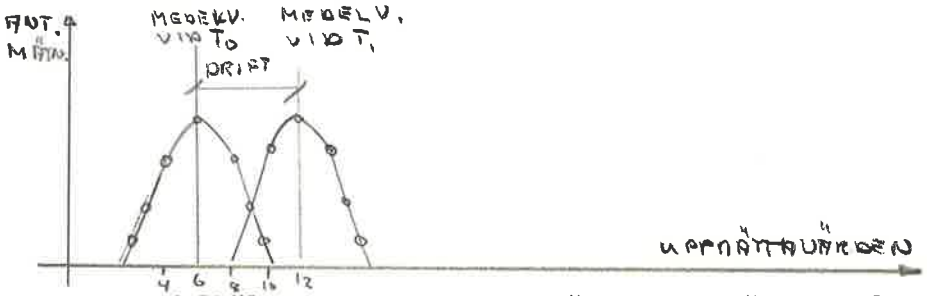
Den i datamaskinsammanhang så ofta använda termen "system accuracy" tolkas för det mesta olika av köpare, säljare och tillverkare. Någon exakt definition finns inte. "System accuracy" hänförs vanligen till tre typer av systemuppträdande:

- Reproducerbarhet
- Absolut noggrannhet
- Kalibrerad noggrannhet

Vilken av dessa tre typer som är mest framträdande beror på AD-systemet och hur det ska användas.

I.2.a Reproducerbarhet

är systemets uppträdande då man gör succesiva mätningar av fasta analoga insignaler. Teoretiskt borde resultaten vara desamma, men brus och andra störningar ger ett varierande resultat. En fördelningskurva kan plottas:



Om alla felkällor gav helt slumpmässiga tillägg, skulle vi få en normalfördelning. Så blir det hemellertid inte. Vi får istället en kurva som är något avsmalnad. Detta är typiskt för de flesta AD-system. Orsaken är systematiska eller periodiska fel och tendenser hos vissa komponenter att "tycka om" en del värden mer än andra.

Reproducerbarhet mäts vanligen över en kort tidsperiod, så kort att förstärknings och offset-instabiliteter inte spelar in. Ibland används uttrycket "long-term repeatability". Man har då mätt ~~repeterbarhet~~ över en viss tidsperiod t. ex. 24 timmar. En siffra på long-term repeatability ~~stability~~ ger naturligtvis en viss uppfattning även på systemet drift. Mera definitiva uppgifter får man om specifikationerna anger både repeatability och drift från medelvärdet.

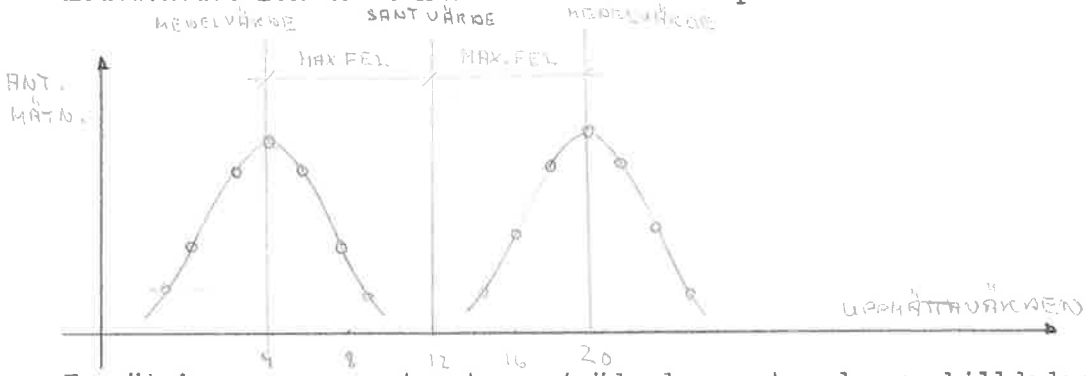
felet aldrig helt. Liksom i tidigare fall måste bland annat varaktighet och temperaturintervall specificeras.

Kalibrerad noggrannhet tillämpas i system som kräver mycket hög noggannhet och där den extra kostnad som en kalibreringsreferens medför är berättigad. Systemet måste också innehålla möjligheter att korrigera det mätta värdet med kalibreringsinformationen.

I.2.b Absolut noggrannhet

För en viss analog insignal har ett AD-system en viss teoretisk utsignal, som den bör producera. Detta är det sanna värdet. På grund av fel i förstärkning, offset-strömmar och fel i referensspänningar fördelar sig utsignalens värde istället kring ett medelvärde som kan vara skillt från det sanna värdet. De nämnda felen kan i viss mån justeras bort, men justeringen har ändlig upplösning. Därför kan de som regel inte reduceras till noll.

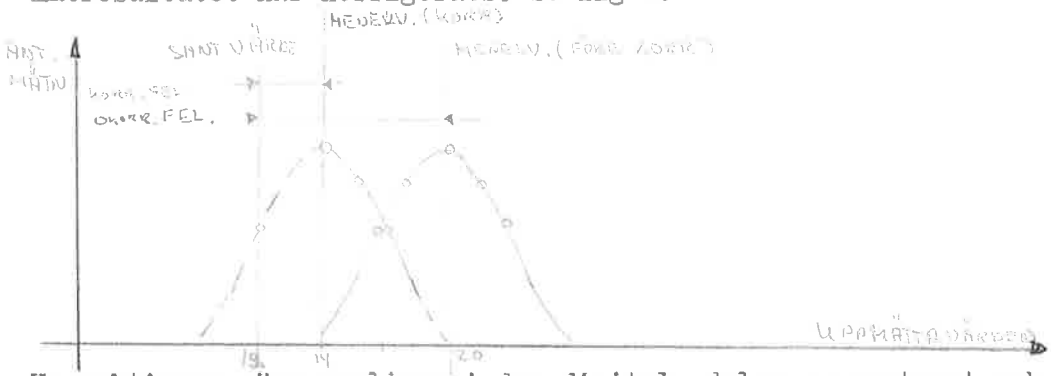
Medelvärdet varierar med tid och temperatur, varför felet mellan medelvärdet och det sanna värdet varierar på samma sätt.



En mätning av systemets uppträdande som tar dessa skillnader mellan det sanna värdet och medelvärdet med i beräkningen tar fram absolut noggrannhet.

I.2.c Kalibrerad noggrannhet

Den kalibrerade noggrannheten är systemets mätnoggrannhet när mätresultatet har korrigerats. Se figur!



Korrektionen görs vanligen i den digitala delen av systemet och är en enkel uppgift när en DM ingår. Huvudsakligen består kalibrering och korrektion av en kontinuerlig ändring av offset och förstärkning, så att skillnaden mellan medelvärde och sant värde minmeras. På grund av inexaktheter i referensspänningen och olineariteter i systemet elimineras

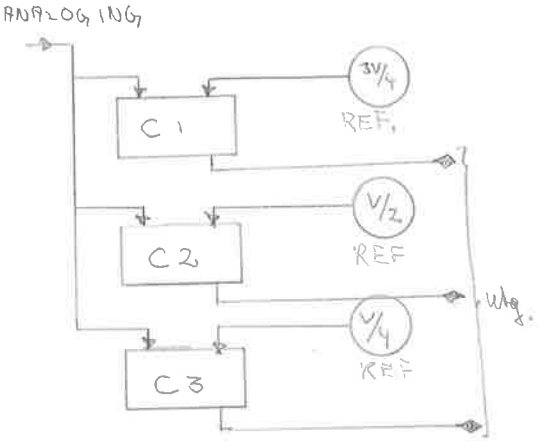
I.3 ANALOG/DIGITAL OMVANDLING - METODER

Grundkretsen i alla AD-omvandlare är komparatorn. Denna jämför två spåningar, en okänd och en referens, med varandra och indikerar vilken av de två som är störst.

De två huvudmetoderna vid AD-omvandling är den integrerande metoden och metoden med succesiva approximationer, vilka behandlas underspeciella rubriker. Ytterligare en rad metoder ska kort beröras nedan.

I.3.a Den simultana metoden

En metod som kräver ett stort antal komparatorer demonstreras i nedastående figur.



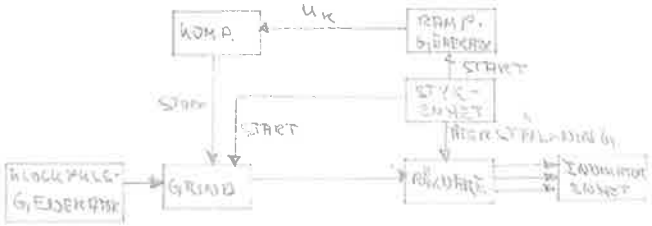
C_1	C_2	C_3	Inspänning
off	off	off	0 - V/4
on	off	off	V/4 - V/2
on	on	off	V/2 -3V/4
on	on	on	3V/4 - V

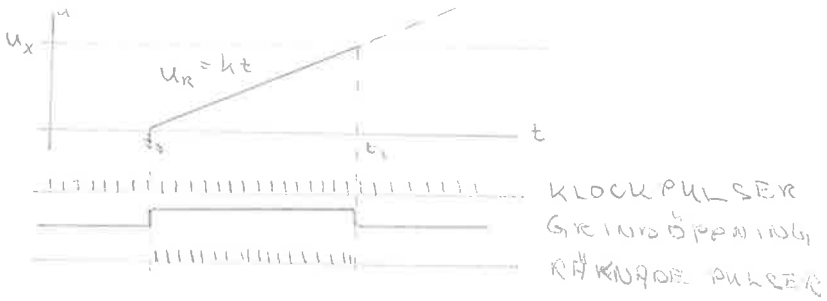
Varje komparator har en referensspänning på sin ena ingång. Funktionssättet framgår av vidstående tabell. I denna tillämpning har spänningsområdet delats i fyra intervall, vilket ger två binära bitar information. Sju komparatorer skulle ge tre bitar,osv. Allmänt ger $2^N - 1$ komparatorer N bitar information.

Den simultana metoden är extremt snabb för system som kräver liten upplösning. Behöver man många bitar blir metoden snabbt oerhört kostsam.

I.3.b Rampspänningsmetoden

Principen för en AD-omvandlare enligt rampspänningsmetoden framgår av nedanstående figur. Man mäter den tid det tar för ett linjärt svep att vära upp till den spänning vi vill mäta. Tidsintervallet mäts sedan med hjälp av en klockpulsgenerator, en av komparatorn styrd grind och en räknare.





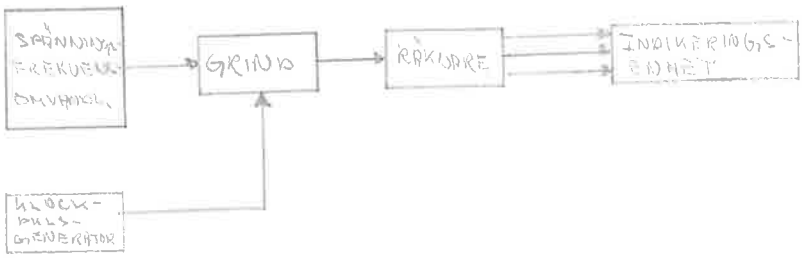
Metoden används bl.a. i system som omvandlar tryck till digitala värden. Svepet som används är en tryckramp som kontinuerligt jämförs med det okända trycket. En differential-manometer sköter jämförelsen. Under tiden som svepet går fylls räknaren med pulser av konstant frekvens. När trycken är lika utlöser en ventil ett relä, som stoppar räknaren och ger utläsningssignal.

AD-omvandlare med spännings-tidsomvandling är enkla och relativt billiga. De är emellertid förhållandevis långsamma och har sällan bättre noggrannhet än o.1 %. En annan nackdel med dessa omvandlare är att de mäter spänningen i en bestämd tidpunkt.

I.3.c Den integrerande metoden

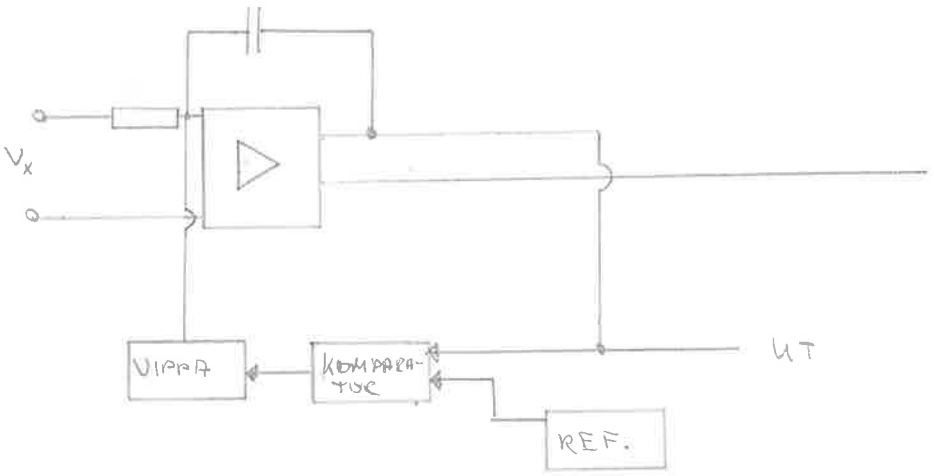
Den integrerande AD-omvandlaren mäter medelvärdet av den analoga signalen över en fast tidsperiod. Om perioden är tillräckligt lång är det uppmätta värdet lika med likströmskomponenten i ingångssignalen. Högfrekvensbrus tenderar då att utjämnas mot noll. Större mätfel resulterar ofta från 50-periodig strömförsörjning. De flesta integrerande AD- omvandlare har en omvandlingstid motsvarande en eller flera 50 Hz-perioder, för att minimera bruset från kraftkällan.

De primära kretsarna i en integrerande AD-omvandlare framgår av nedanstående blockschema.



Mätspänningen omvandlas först till en frekvens i en spännings-frekvensomvandlare, vars utsignal är ett pulståg med en mot mätspänningen proportionell pulsrepetitionsfrekvens. Ofta används en spännings-frekvens-omvandlare av följande typ i komersiella instrument.

Den centrala delen är en integrerande operationsförstärkare som genererar en rampspänning vars lutning är proportionell mot inspänningen. När rampen når referensspänningsnivån, låter vi en vippa snabbt kortsluta kondensatorn, varpå en ny rampspänning startar. Den genererade sågtandsspänningen får styra en pulsgenerator.



Den räknare vi kopplar efter pulsgeneratorn kommer inte att ange mätspänningens verkliga integral utan dess meddelst integration erhållna tidsmedelvärde under ett klockpulsintervall.

Eftersom medelvärdesbildningen medför en lågpassfiltrering, kan man använda denna typ av AD-omvandlare för mätning av spänningar med överlagrade störsignaler utan att behöva införa ett speciellt lågpassfilter.

Eftersom medelvärdet över flera perioder av insignalen till integratorn måste vara noll, gäller vid konstant mätstorhet

$$U_{xo}\ T = U_{po}\ \tau$$

dvs. pulsfrekvensen

$$f = \frac{1}{T} = \frac{1}{U_{po}\,\tau}\ U_{xo}$$

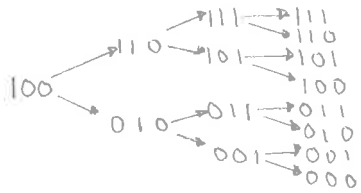
är proportionell mot mätspänningen. Proportionalitetskonstanten är inveterade värdet av ytan hos de återförda spänningspulserna. Omvandlarens statiska noggrannhet bestäms sålunda huvudsakligen endast

av hur bra man kan göra spänningspulser med konstant och känd spännings-tidsyta.

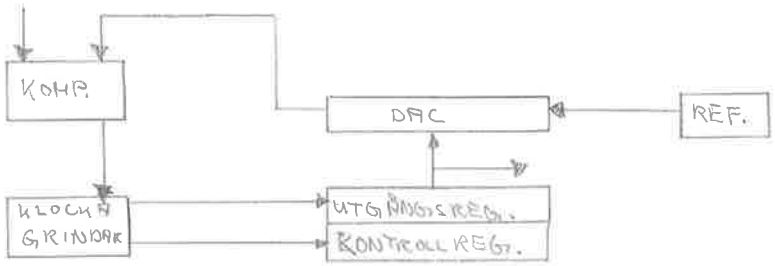
Omvandlingssnabbheten bestäms i första hand av klockpulsfrekvensen, som i sin tur bestäms av maximalt mätvärde, önskad upplösning och spänningsfrekvensomvandlarens proportionalitetskonstant. Maximala omvandlingsfrekvensen för en AD-omvandlare av integrerande typ kan vara ca 50 Hz.

I.3.d Metoden med succesiva approximationer

För omvandlare med hög hastighet och många kanaler lämpar sig metoden med succesiva approximationer bäst. Metoden kräver bara ett omvandligssteg per bit för att konvertera vilket tal som helst. Karakteristiskt för den succesivt approximerande AD-omvandlaren är att den gör en serie gissningar som bättre och bättre ansluter sig till det riktiga värdet på den analoga ingångssignalen. Se schema.



Som tidigare nämnts kan dessa AD-omvandlare göras mycket snabba. Genom att använda serie-parallellförfarande kan man nå ned till omvandlingstider mellan 0.1 och 0.2 mikrosekunder per bit.



De viktigaste elementen i denna typ av ADC är:

- en komparator som hela tiden jämför den analoga ingångsspänningen med den internt genererade spänningen.
- en kontrollenhet med klocka, grindar och register som styr omvandlingen
- en DA-omvandlare som genererar en konstant föränderlig spänning, styrd av värdet i kontrollenhetems utgångsregister.

AD-omvandlingen går till på följande sätt:

- Den analoga signal som ska omvandlas läggs på komparatorns ingång och en starpuls ges till kontrollorganet.
- Startpulsen nollställer registren och sätter en 1 i den mest signifikanta biten hos kontrollregistret (CR). Den startar också komparatorn.
- Processen startar genom att en 1 sätts i Utgångsregistret (UR) i den position som CR bestämmer. Därvid genererar DA-omvandlaren en spänning som är proportionell mot talet i UR.
- Denna spänning jämförs med den okända i komparatorn. Om den genererade spänningen har större amplitud än den analoga ingångsspänningen nollställs den tidigare satta ettan
- en etta sätts i nästa position i CR
- förfarandet upprepas tills ettan i CR hamnar utanför registret

UR indikerar nu det deigitala värde som är ekvivalent med ingångssignalen.

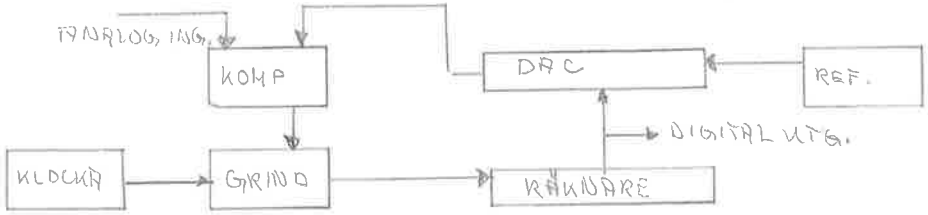
Varje approximation reducerar felet till ungefär hälften. Omvandlingstiden hos den succesivt approximerande ADC´n begränsas av komparatorns stabiliseringstid och av DA-omvandlaren i den interna spänningsgeneratorn. Dagens solid-state elektronik begränsar AD-omvandling av denna typ till ungefär en bestämning per mikrosekund. En tiobitars bimär ADC av denna typ gör cirka 100000 omvandlingar/sek. Tider omkring 1.5 mikrosek/bit är normala.

För att nå omvandlingsfrekvenser större än 100 kHz, krävs modifiering av den skisserade typen av AD-omvandlare. Den högre hastigheten uppnås ofta genom att man använder flera komparatorer och spänningsgeneratorer. Varje approximation bestämmer korrekt värde på två eller fler bitar. Förfarandet/ kallas för serie-parallellteknik.

I.3.e Återkopplingsmetoder

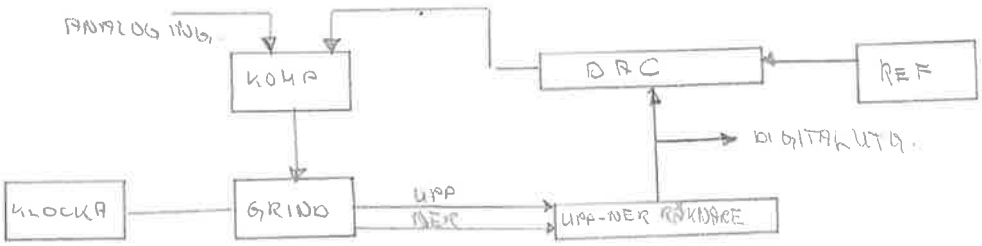
Dessa bygger på principen att man ska internt i AD-omvandlaren generera sin referensspänning. Detta sker i en DA-omvandlare som gör AD-omvandlaren till ett slutet, återkopplat system. Två typer förekommer; räknemetoden och den kontinuerliga metoden.

I räknemetoden styr ett flip-flopregister (FF) DAC'en. FF räknar från noll tills utspänningen från den interna spänningsgeneratorn är lika med eller överstiger den analoga inspänningen.



Räknemetoden är bra i system som kräver hög upplösning, därför att en ökning av antalet bitar, bara medför att ett fåtal nya kretsar behövs. Flera ingånga kan lätt omvandlas samtidigt. Omvandlingstiden ökar emellertid snabbt med antal bitar. Det beror på att räknaren måste hinna med 2^N räkk markeringar. Medelomvandlingstiden blir hälften av denna tid.

En lätt modifiering av räknemetoden får man genom att byta ut den enkla räknaren mot en som kan räkna både framåt och bakåt. När en gång den rätta digitala värdet beräknats, kan omvandlaren följa den analoga signalens variationer kontinuerligt. Metoden kallas den kontinuerliga metoden och är mycket snabbare för en kanal.



I.3.f Sektionsräknare

Räknemetoden är en enkel och bra metod. Dess stora nackdel är att den kan kräva alldeles för många räkningar när orden har många bitar. Om det digitala ordet t.ex har N bitar, kan antalet markeringar bli ända upp till 2^N.

Ett sätt att minska antalet steg är att dela upp räknaren i sektioner. Man kan exempelvis dela upp en 10-bitars räknare i två om 5 bitar vardera. När man börjar omvandlingen sätts ettor i den minst signifikanta räknarens alla positioner. Man räknar sedan upp den mest signifikanta räknaren på vanligt sätt tills komparatorn ger utslag. Den andra räknaren nollställs då och börjar från början.

Omvandlingstiden beror på hur många sektioner man har. Sektionsräknare som räknar både upp och ner är vanliga. Sektionsräknartekniken används framförallt i digitalvoltmetrar där varje utgång ska vara decimal; varje sektion representerar då en decimal siffra.

Genom att använda en smula extra logiska kretsar och ibland några redundanta, kan omvandlingstid och precision förbättras i samtliga nämnda metoder. Korrigerande redundanta steg kan också användas för att kompensera ändringar i den analoga insignalen under tidigare steg. Därigenom minskas aperturtiden.

I.3.9 Synchro AD-omvandlare.

"The shaft encoder" är en omvandlare som förvandlar vinkelläget hos en axel till en digital signal, representerande roterade grader från en fast referensvinkel. Dessa apparatetr är vanligen absolut kodade, dvs signalen för varje vinkelläge är unikt istället för att man behöver räkna in ett antal lika pulser.

Två typer av "shaft coder" utmärker sig framför de andra. Det är rymdgeometrisk kodning och synchro-metoden, av vilka den förra behandlas något senare.

Principiellt går metoden ut på att man använder en synchro tillsammans med en elektronisk sifferomvandlare för att få ett digitalt värde på den vinkel man önskar mäta. Man kan använda synchron antingen som amplitudmodulator eller som fasskiftare. Av dessa används "synchro resolvern" vanligast som fasskiftare. Resolvern arbetar i systemet som ett fasvridningsdon och är därvid speciellt utformad för att med hög noggrannhet lämna en fasvinkel, vars tidslägen är en funktion av axelvinkeln.

Synchron har många goda fördelar. Den är en väletablerad produkt, som tål temperaturer upp till 200° C och kraftiga vibrationer.

Svenska Ackumulatoraktiebolaget Jungner ger ut en AD-omvandlare typ ADO-3 som fungerar enligt denna metod. Den har 13 bitars noggrannhet med en omvandlingstid på 2 millisek. Mäthastigheten är 500 mätvärden/sek och vinkelnoggrannheten 0.04°. Mätapparaturen är administrativt enkel, tack vare en elektronisk växel utan Reed-reläer.

Problemet med Synchros har länge varit att de inte kunde användas när insignalen som skulle följas gav hastigheter som överskred 4 till 8 grader per sek. En ny metod refererad i (11) gör det möjligt att

följa hastigheter upp till 2000 grader per sekund.

Metoden innebär att man balanserar en automatisk noll-brygga med hjälp av analoga servo-principer.

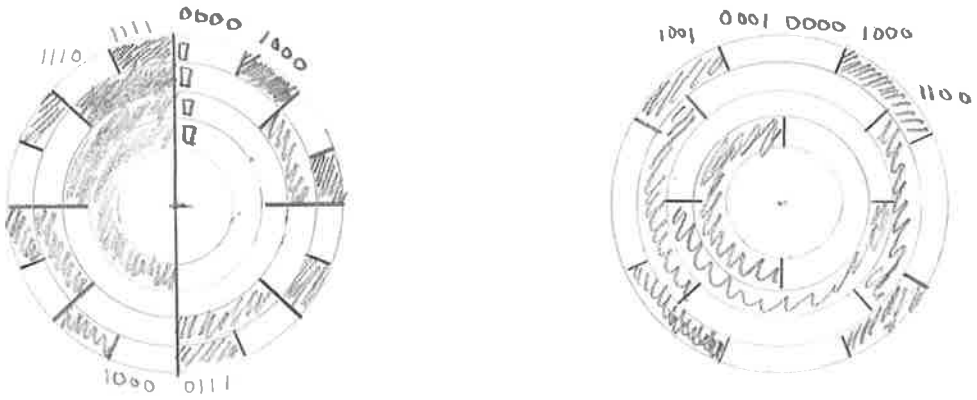
I.3.h Den cykliska enbitsmetoden

Konstruktionen av integrerade kretsår har gjort att den cykliska enbitsmetoden blivit aktuell. I denna åstadkomms omvandlingen genom att den analoga ingångsspänningen påföres en operationsförstärkare, som multiplicerar ingångsspänningen med (+2) eller (-2) beroende på spänningens polaritet. Därigenom erhålles en enkelriktad positiv spänningsfunktion. Denna spänningsfunktion jämförs med en referensspänning. Skillnadsspänningen påförs nästa steg, som arbetar på identiskt sätt. Till varje steg hör en digital utgång, som är en funktion av polariteten hos ingångsspänningen för detta steg. En komplett cyklisk en-bits-omvandlare kan erhållas genom att ett antal en-bits- system seriekopplas.

I.3.i Rymdgeometrisk kodning

Spatial Encoding

Den mest direkta omvandlingsmetoden, men också en av de som används minst för närvarande är rymdgeometrisk kodning. Den består av ett rymdgeometriskt mönster utlagt på en cirkulär skiva, över en CRT eller framför ett antal fotoceller. Ett typiskt mönster visas i nedanstående figur.



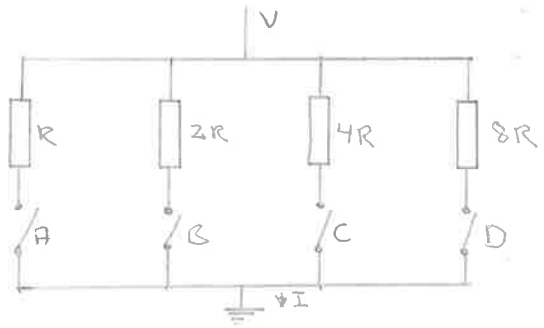
Kodningen kan avläsas t. ex genom släpborstar, mikrobrytare eller ljusstrålar. När borstar används gör man vanligen inte upplösningen starkare än 7 bitar. Med hjälp av fotoelektrisk avläsningsteknik har man kunnat placera 16 bitar på samma skiva. Vill man uppnå ännu högre upplösning går det bra om man kopplar samman skivan med en reduktionsväxel. Växeln kan emellertid ge upphov till felaktigheter genom glapp. Redundanstekniken har här erbjudit vissa hjälpmedel.

Det finns ett stort problem med kodning av det här slaget; tvetydigheter när flera bitar skiftar samtidigt kan nämligen mycket lätt uppkomma. Standardknepet för att komma ifrån dessa svårigheter är att använda en s. k. Gray- eller unit distance kod. I denna skiftar aldrig två bitar samtidigt.

I.4 DIGITAL/ANALOG - OMVANDLING

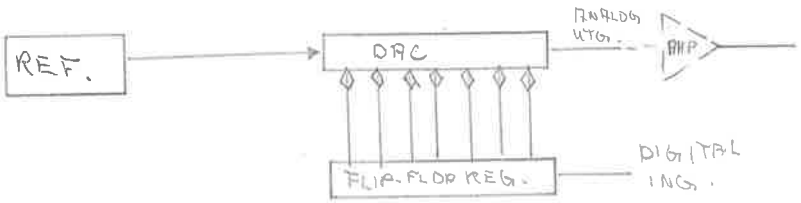
Omvandlingen från digitalt siffervärde till analog spänning görs oftast med hjälp av ett antal viktade motstånd. Både mekaniska och elektroniska omvandlare förekommer.

Principen för DA-omvandlaren kan åskådliggöras enligt nedanstående figur:



Varje brytare representerar en binär bit. Om t. ex. brytare D sluts, medan de andra är öppna, blir strömmen V/8R osv. Varje brytare fördubblar strömstyrkan, precis som varje binär bit har dubbla vikten av den föregående. Strömstyrkorna adderas om fler än en brytare sluts. På detta sätt blir den totala strömmen proportionell mot det binära tal som brytarna representerar.

Ännu ett steg närmare verkligheten kommer vi i följande blockschema:



Motstånden är viktade så att varje bit i registret bidrar till utspänningen i förhållande till sitt värde. Den digitala ingångsspänningen bestämmer den analoga utgångsspänningen, på grund av att

nätverket bara åt ett passivt element.

Oftast är inte utspänningarna från FF-registret så precisa som behövs i analoga sammanhang. Därför kopplas nivå förstärkare mellan FF-registret och motståndslänkarna. Förstärkarna förser motstånden med antingen jordpotential eller referenspotential.

I vissa fall kopplas en operationsförstärkare eller en katodföljare till utgången från DA-omvandlaren för att minska utgångsresistansen.

Det händer att man vill avläsa flera analoga ingångar nästan samtidigt. För att tidsskillnaden mellan avläsningarna ska bli försumbart liten används en DAC-konstruktion med dubbla buffertregister.

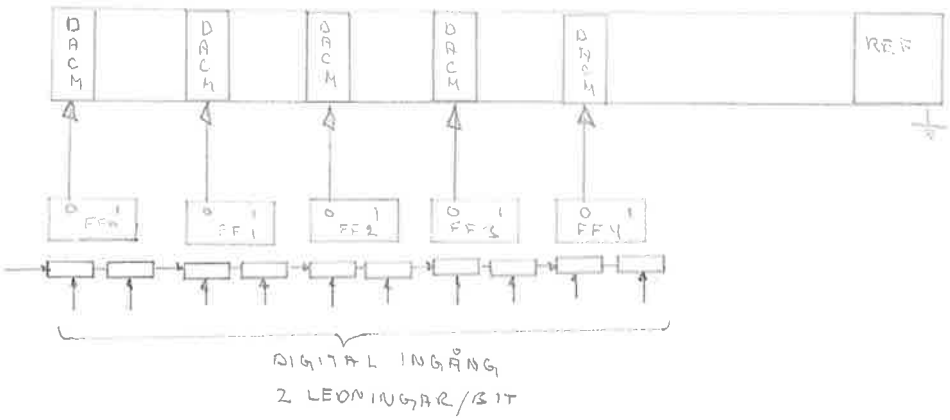
Vid långsamma omvandlingar är en elektromekanisk omvandlare med reläer som brytare fullt tillräcklig. Den elektroniska varianten är emellertid vad det gäller snabbhet mera tilltalande. Hastigheter upp till 1 miljon omvandlingar/sek kan åstadkommas. Praktiskt är gränsen 100 000 omvandlingar/sek.

Den maximala omvandlingsfrekvensen hos DA-omvandlaren bara av den tid som behövs för inläsning i FF-registret. 10 MHz kan lätt uppnås, men siffran kan vara missvisande. Stabiliseringstiden bestämmer den maximalt användbara omvandlingsfrekvensen.

Stabiliseringstiden hos en DAC mäts mellan de tidpunkter då den digitala informationen ges och då det analoga resultatet har stabiliserat sig mellan felgränserna.

Utgången från motståndsnätet har höga frekvenstransienter innan den stabiliseras. Om den förbinds med en lågfrekvens-apparat gör det ju inget. I vissa tillämpningar är det önskvärt att ha en jämn övergång mellan tillstånden istället för att ha en minimal stabiliseringstid. Under sådana förhållanden dämpar man ned svängningarna med en kondensator eller ett lågpassfilter. Om utgången DAC går via en förstärkare, bestämmer denna maximala förhållandet vid förändring av frekvensen. Därför tar ofta första värdet längre tid för omvandlaren att läsa.

För att ytterligare studera de tidsfaktorer som begränsar DA-omvandlingen, betraktar vi följande moduluppbyggda DAC.

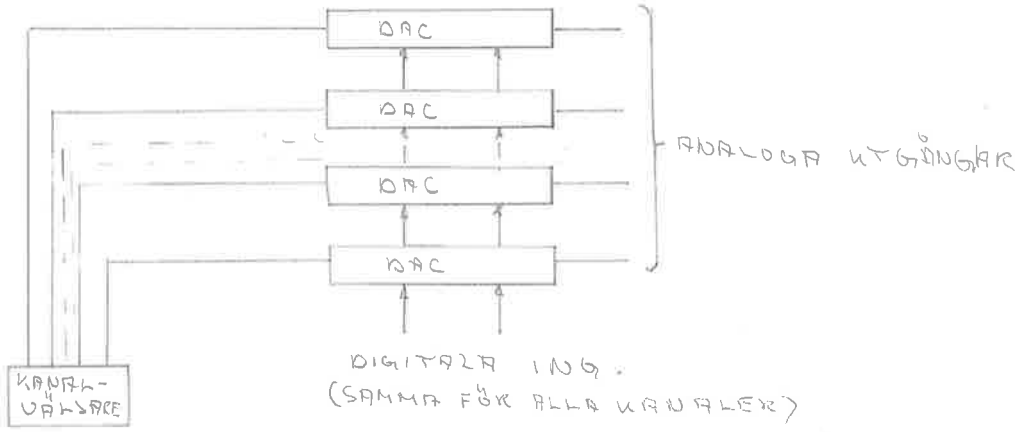


Digital information påförs samtidigt samtliga FF-element. med komplementära ingångar när en läspuls går in på sin speciella ingång. Stabiliseringstiden hos DA-omvandlaren beror på hur många FF som skiftar. Mellan 01111 och 10000 skiftar samtliga FF, trots att de analoga värdena är mycket närbelägna. Transienter med avsevärda belopp (10% av fullt utslag) uppkommer på dividerkretsens utgångar av flera olika anledninagr. Dessa transienter har emellertid kort varaktighet (1/2 LSB från slutvärdet inom 2.5 mikrosek. är ett vanligt värde)! Om man stryper utgången från FF till DAC-ingången för de mest signifikanta bitarna, kan man reducera transienternas varaktighet till mindre än 0.5 Mikrosek för precisionssystem. Ett lågpassfilter på utgången jämnar ut förloppet ytterligare.

I.5 Multiplex vid DA-omvandling

Vid DA-omvandling är problemet vanligen att behandla digital, sekventiell information från en apparat, vanligen en Datamaskin, och fördela den till ett antal olika analoga mottagarenheter. Vanligtvis måste man ge samma digitala information till samtliga kanaler, således även till dem som inte addreseras av DM.

Två typer av multiplex användes. T. ex. kan en separat DA-omvandlare kopplas till varje kanal enligt figur.



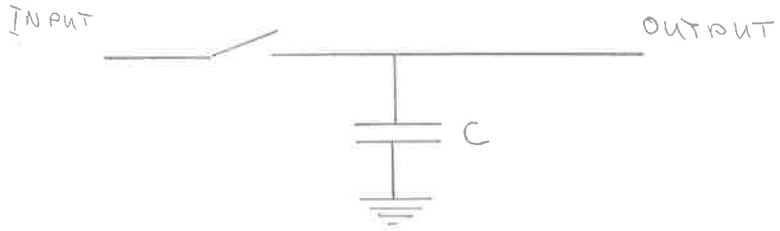
En signal från kanalväljaren lagrar här in informationen i bufferten till rätt DAC.

Ett annat sätt är att använda en DAC tillsammans med multiplexenhet och SoHkretsar på varje kanal. Den förstnämnda metoden kostar något mer än den andra, men har fördelen att information kan kvarstå på den analoga kanalen obegränsad tid utan att sjunka i värde. För den andra metoden gäller att signalen till SoH-kretsen måste förnyas med jämna mellanrum.

I.6 "Sample and Hold"-teknik

Mätning av momentanvärden hos tidvariabla spänningar i väl kända mätpunkter och med god upplösning ställer stora krav på AD-omvandlarens snabbhet. För att inte dessa krav ska bli oöverkomliga i praktiska sammanhang brukar man använda sig av "sample and hold"-teknik. Detta innebär att man inför en minneskrets, i princip en operationsförstärkare som laddar upp en kondensator, mellan mätspänningsanslutningen och omvandlaren, Under en viss tid, laddningstiden (the acquisition time) kopplas minneskretsen till mätspänningen och ställer in sig på mätspänningsvärdet. Därefter låses minnet till mätspänningsvärdet i ett visst ögonblick, då minneskretsen frikopplas från mätspänningen. Osäkerheten i denna tid kallas aperturtiden (the aperture time). AD-omvandlingen av mätspänningen kan sedan ske utan extrema krav på snabbhet hos omvandlaren.

SoH-kretsen kan representeras enligt nedanstående figur:



Hålltiden (the holding time) är den tid kretsen kan behålla en laddning utan att avklingningen är större än en bestämd procentsats av begynnelsevärdet.

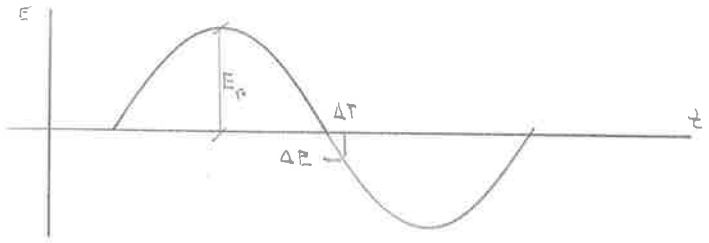
Den frekvens med vilken den ingående signalen samplas måste bestämmas noggrannt med hänsyn till frekvensen hos den ingående signalen. Olämpligt valda frekvenser kan bli ödesdigra för mätresultaten. Systemet måste exempelvis göra minst två samplingar varje period av den ingående signalens högsta frekvens.

Ett av de fenomen som kan drabba är aliasing eller frequency folding. Kort uttryckt innebär det att frekvenser högre hos mätsignalen som är högre än halva samplingsfrekvensen ($1/2\ f_s$) uppträder

som datavärden med frekvenser mindre än halva samplingsfrekvensen. Detta orsakar amplitudfel i mätningarna. Kardinalregel nummer 1 är därför: Data som ska samplas får på inga villkor innehålla frekvenskomponenter större än halva samplingsfrekvensen. Mer om detta finns att läsa i referens nummer 1, section 2.

En SoH-krets minskar den effektiva aperturtiden och ökar effektiva bandbredden hos en omvandlare. Den maximala frekvens som en AD-omvandlare eller SoH-krets kan handskas med kan beräknas genom att man bestämmer maximala spänningsförändringen m. a. p. tiden. Detta är gjort i referens nr. 3 sid. 365 ff. Räkningarnager

$$f = \frac{\Delta E}{2\pi E_p \Delta T}$$; där beteckningarna framgår av nedanstående figur.



Skillnaden mellan uppträdandet hos några AD-omvandlare med och utan SoH-kretsar framgår ur nedanstående tabell. Jämförelsen gäller Digital Equipments materiel.

ADC	Bitar	Aperture	Upplösning	Frekvens
AF01	6	9 usec	1.56 %	520 cps
A801	10	10 usec	0.1 %	30 cps
AF01	12	35 usec	0.025 %	2.2 cps
		Med A400 Sample and Hold		
AF01	6	150 nsec	1.56 %	60 000 cps
A801	10	150 nsec	0.1 %	25 000 cps
AF01	12	150 nsec	0.025 %	540 cps

Nutida explosiva utveckling inom data process området gör ofta att de erfarenheter man får under tiden man konstruerar och bygger upp en data-processanläggning är så omfattande att de tenderar att med en gång göra det uppbyggda systemet hopplöst omodernt. För vetenskapliga ändamål skapas fasta systemxxxx speciellt konstruerade för att göra något visst slag av mätningar. När uppgiften är löst, har detta dyra data-processsystem blivit onödigt, men ingalunda utslitet.

Enda utvägen ur detta dilemma är att konstruera ett system som är snabbt nog och flexibelt nog att täcka ett helt område av dessa vetenskapliga behov.

Den engelska firman EAL marknadsför s.k. "interface" som tillåter sammankoppling xxx och utbyggnad av redan existerande DPsystem. Dessa standardinterface erbjudervinte bara valfrihet i spänningsnivå utan innehåller också enkla kontrollsignaler. Systemet kallas "the MDP range" och innehåller alla grundblock som behövs för datastyrningav en process. Med standardapparatuern kan man xxxx avläsa 200 kanaler/sek. Avsökningskapaciteten kan utökas i steg om tio kanaler upp till 1000 kanaler.

I.7 Uppbyggnad med modulsystem

En AD-omvandlare blir en ganska stor och otymplig apparat om man använder uteslutande diskreta komponenter vid konstruktionen. Uppbyggnad med hjälp av integrerade kretsar kan däremot avsevärt förenkla och förbilliga arbetet. På sista tiden har man börjat använda ett modulsystem vid konstruktionen. Detta erbjuder flera goda egenskaper.

För det första blir ADC´n flexibel. Kraven på omvandlarsystemen varierar ofta vitt beroende på tillämpning. Modulsystemet möjliggör då för konstruktören att skräddarsy sin ADC förvdess exakta ändamål. Om kraven skulle ändras kan samma moduler användas för att bygga upp en annan ADC.

Moduler är ekonomiska. Pengar sparas både genom den nämnda utbytbarheten och genom en kostnadsminimal konstruktion av kretsarna.

Typiska priser för en DAC är $1000 och för en ADC $ 2000. Om flera system kan byggas samman, minskar kostnaden då kraftkällor och kopplingspaneler kan användas gemensamt. Exceptionellt höga krav på hastigheten medför högre kostnader. Priserna gäller en kanal!

En tredje fördel med att använda modulsystem är att den färdiga omvandlaren aldrig behöver kalibreras om på fabriken. Procedurer för kalibrering och justering kan levereras tillsammans med modulerna.

I.8 Tabeller

Tabell 1.

Control Data Corporation marknadsför 4 skilda typer av A/D-system. De långsamma analoga systemen gör det möjligt att med 1700-maskinens A/D-omvandlare mäta små analoga signaler. Tre sådana typer finns, vilka var och en har skilda karakteristika och uppträdanden. Se bilaga 1/1. Grundskillnaden mellan dem ligger i vilka typer av multiplex som används och vilka analog/digital-omvandlare som använts.

Referens 1.

System	Långsamt typ I	Långsamt typ II	Långsamt typ III	Snabbt
Analog ingångs-spänning	-10 - +40 mV	-10 - +40 mV	-50 - +50 mV	±5V eller ±10V
Analog källa	Externa signaler via signalbegränsare	Externa signaler via signalbegränsare	Externa signaler via signalbegränsare	
Multiplexer-typ	kvicksilverreläer	James Microscan reläer	James Microscan reläer med 0.5 Hz filter	solid-state
Antal kanaler	16 - 128 (1024)	16 - 128 (1024)	16 - 128 (1024)	8 - 64 8stegs intervall (256)
Förstärkare	behövs ej	behövs ej	diff. först, isolerad ingång, single gain	
A/D omvandlare	Integrerande typ	Integrerande typ	successiv approx.	snabb succesiv approx. med sample och håll-krets
Digital utgång	14 bitar	14 bitar	12 bitar	8, 10, 12 eller 14 bitar
Avläsnings-hastighet	30 avl/sek	40 avl/sek	200 avl/sek	250, 100, 50 eller 40 kHz
Noggrannhet	± 0,1 %	± 0,1 %	± 0,1 % (± 1/2 sista biten)	± 0,03 % (± 1/2 sista biten)

Tabell I.1

Översikt över olika typer av A/D - omvandlare för CD 1700. Sammanställd ur General Information Manual för Analog/Digital system, Control Data 1700 Computer System.

Karakteristika	AF01	AF02	AF03	AF04
Omvandlingsmetod	Succ. appr.	succ. appr.	succ. appr.	integrerande
Upplösning	6 - 12 bitar	6 - 12 bitar	6 - 12 bitar	6 områden med 6 decimala siifror i va
Antal kanaler(max)	64	1024	1024	1000 standard, 10 000 tillsats
Multiplexer	isolerad grind FET	isol. grind FET	3-poligt Reedrelä	3-poligt Reedrelä
Max common-mode spänning från linje-jord	± 10 V Totalt	± 10 V	± 100 V	± 300 V
Avläsningshastighet	100K kanaler/sek (6) 25K kanaler/sek (12)	70K kanaler/sek 20K kanaler/sek	200 kanaler/sek (utan filter)	till 50 kanaler/sek

Tabell I.2

Fyra typer hög- och lågnivå AD-omvandlare med multiplex finns till PDP-DM. AF01 och AF02 är två General-Purpose AD-omvandlare där enbart antalet kanaler skiljer. AF03 är ett skärmat (guarded) tre-tråds system med justerbar förstärkning differential förstärkare och common mode rejection lika med 120 dB. AF04 , en skärmad avsökande voltmeter, har en upplösning upp till 0.001 %. Den har mätområden mellan 10 mV och 300 V, 1000 avsökta ingångar och C.M.R. = 140 dB.

Tabell I.3

Karakteristika för AD-omvandlare

I PDP´sAD-omvandlare typ AF01, AF02 och AF03 kan ordlängden förändras mellan 6 till 12 bitar. Detta påverkar såväl maximala "switching point" felet som omvandlingstiden.

ordlängd (bitar)	Max switching point fel ± 1/2 LSB	Omvandlingstid (usek)
6	+ 1.6 %	9.0
7	+ 0.8 %	10.4
8	+ 0.4 %	12.0
9	+ 0.2 %	13.5
10	+ 0.1 %	18.0
11	± 0.05 %	25.0
12	+ 0.025 %	35.0

	Bäst för multiplex eller kont. ingångar?	Omvandlingstid 5 bitar (usec)	Omvandligstid 10 bitar (usec)	Aperturtid 5 bitar (usec)	Aperturtid 10 bitar (usec)	Konstant aperturtid?	Relativ kostnad	Anmärkningar
Simultana	Båda	Ej tillämpligt				Ja	Beror på lösningen	Exellent för system med låg upplösning - opererar på cirka 100 nanosekunder.
Räkne-	M	24 av.	1792 av.	1.5	3.5	Nej	Låg	Tillåter många omvandlingar samtidigt
Kontinuerliga	C	1.5	2	1.5	2	Ja	Låg till medium	Extremt hög hastighet för kontinuerlig ingång, men blir sämre vid stora förändringar
Succ. approximation	M	7.5	36	7,5	36	Ja	Medium	Vanligast - god hastighet i förh. till priset
Rampspänning	M	16 av.	512 av.	1	1	Nej	Beror på lösningen	God differentiell linearitet-billig för system med liten upplösning
Sektionsräknare	M	18	112-224	1.5	3.5	Nej	Låg till medium	Används tillsammans med digitalvoltmeter

Tabell. 4. Översikt över olika A/D omvandlingsmetoder

I. 9 REFERENSLISTA

1.	Control Data 1700	Analog/Digital Systems, General Information Manual
2.	Digital	Small Computer Handbook
3.	Digital	Logic Handbook 1968
4.	Digital	The Logic Handbook 66 - 67
5.	Digital	Industrial Handbook 1968
6.	Elteknik -68 s.108 ff	Brodin-Lindquist, Vilken digital voltmeter ska jag köpa?
7.	Elektronik -68 Nr 12 s 46	Analog-Digitalomvandlare med integrerade kretsar
8.	Åström, Reglerteknik, LTH -68	TLTH/KBV Samplade system Kap. II
9.	Bartee,Mc Graw-Hill -66	Digital Computer Fundamentals, sec. edition, kap 8.
10.	Martin, Prentice Hall, -65	Programming Real-Time Computer Systems
11.	Roberts, Control E. 1-1968	Servo-style Circuit Speeds Synchro-to-Digital Conversion
12.	EAL, broschyr	Modular Data Processing

KAPITEL II

Undersökning av tre olika sätt att generera styrsignaler för ställorgan.

IBM 1800 kontrollerar I/O genom två metoder:

1. Data Channel Control DCC

2. Direkt Program Control DPC

DCC (kallas ibland Direct Memory Access, DMA)

överför data mellan PDM-kärnminnet och I/O med hastighet som kontrolleras av I/O-organet. PDM-programmet avbryts enbart under den tid som behövs för att läsa in eller skriva ut ett data-ord från kärnminnet. Metoden kallas också "cycle stealing"eftersom den i normala fall bara upptar en minnescykel i programmet. Företrädesvis användes den vid massminnen (band, skiva, trumma) och till AD/DA-kanaler.

DPC

överför individuella data mellan PDM-kärnminnet och I/O med en takt som kontrolleras av en programrutin. Lokaliseringen av avbrottet kan ska hardware-styrt eller programstyrt.
PDM-programmet avbryts för den tid som behövs för

1. att hoppa till I/O-rutin

2. uppfylla avbrottsfrågan

3. hoppa tillbaka till PDM-programmet.

I de fall vi undersöker ska styrsignalen genereras på tre olika sätt. I samtliga fall är det otvetydigt bäst att använda DPC. Som grund för programmeringen ligger informationen i referens 1. Erfarenheterna från IBM-programmet bör kunna tillämpas även på andra PDM.

II. 1 Digital styrning av Stegmotor.

Styrningen av stegmotorn sker genom att vi sänder ut ett visst antal pulser proportionellt mot värdet på u(t).

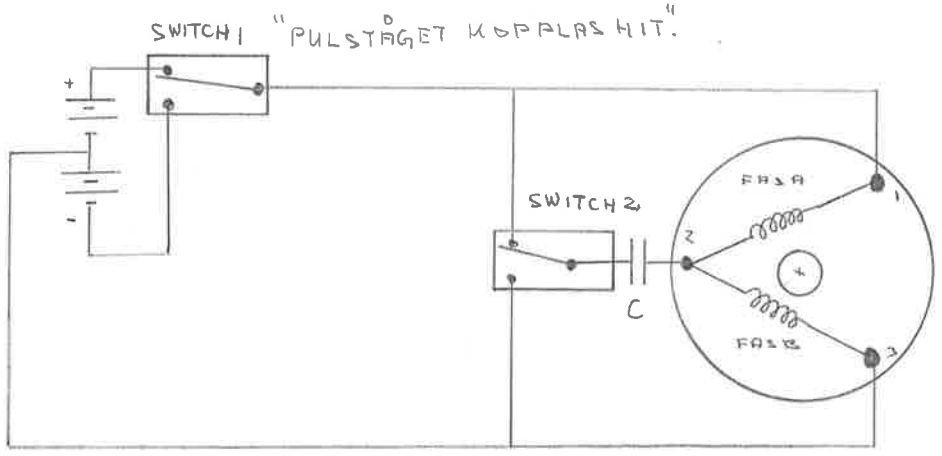
Till IBM/1800 hör en 'solid-state'-krets som lämnar 3 ms pulser med 450 mA strömstyrka och 48vdc. 16 punkter kan addresseras samtidigt genom att sätta en "1" för puls och en "0" för ingen puls.

Pulse Output (PO) ingår i schemat över digitala och analoga utgångar, bilaga 1.

Den primära användningen för PO är att ge pulståg förvatt styra olika typer av stegmotorer.

Genom en XIO write-instruktion överförs data till 16-bitars registret. Utgången stängs omedelbart när registret laddats och öppnas inte förrän det antal pulser registret anger har klingat ut. Pulsen startas genom en XIO control - instruktion. Innehållet i ackumulatorn minskas med ett varje gång en puls klingarvut. På så sätt startas ett pulståg som räknas av programmet.

Positiv eller negativ styrsignal anger åt vilket håll stegmotorn ska rotera. Rotationsriktningen ändras genom att switch 2 slås över i nedanstående bild. Vippan slås över enbart om rotationsriktningen ska vara negativ. Efter utmatningen av pulståget slås den tillbaka.



Denna switch slås över med en'Electronic Contact Operate' (ECO) som kräver 10 µs operationstid. Tiden för switch 2 att vippa över adderas till ECO-tiden. Order ges genom XIO write ECO.

De negativa talen måste 2-kompletteras för att få rätt storlek.

Ett skissartat program för styrning av en stegmotor skulle då se ut så här:

```
        LD      UT          /ladda in UT i ackumulatorn
        BSC L   REVER,Z+    /hoppa till REVER om negativ
BACK    XIO writePO         /skriv in i I/O organet register
        XIO control         /skicka ut pulsen
        S       ETT         /minska med 1
        BSC L   BACK,+-     /hoppa tillbaka om ej 0
        BSC L   START       /hoppa till START
REVER   EOR     ETTOR       /komplementera medför 1-kompl.
        A       ETT         /+1 medför 2-kompl.
        XIO writeECO        /reversera m.h.a. ECO
BAKA    XIO writePO         /Se tidigare
        XIO control
        S       ETT
        BSC L   BAKA,+-
        XIO writeECO        /vippa tillbaka relät
        BSC L   START
```

Minnesutrymme 21 celler

Summan av ordernas medelexekveringstider 85 µs

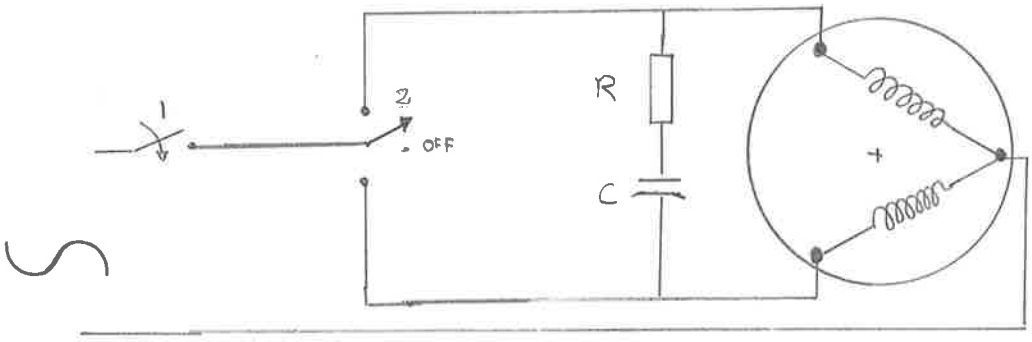
För varje stegg upptas 23 µs effektiv programtid.

Pulserna är 3 ms långa och en viss tid bör förflyta mellan varje puls. Den tid som inte åtgår för direkt styrning, kan användas av maskinen till andra sysslor.

II.2 Digital styrning av Synkronmotor.

Styrsignalen sänds ut i form av en start och en stoppuls. Tiden mellan dessa är proportionell mot amplitudem hos u(t).

ECO utnyttjas i detta fall. En ECO-utgång får sköta det relä 2 som bestämmer rotationsriktning (se figur) och en annan ger styrpulserna.



Relä 1 dras efter startpuls och ger spänning åt motorn. Under tiden räknas u(t) ned och en ny puls (stoppuls) ges till reläet, som släpper. För nedräkningen användes interval timer A i kärnminnes-cell 00004. Den kortaste tillgängliga periodtiden för 250 µs. Timern sätts till önskat värde genom programmet. Den räknar sedan uppåt tills den slår om och kommer till noll. Den begär då ett avbrott och hoppar till en order som ger stoppulsen. Eftersom Intervall timern räknar uppåt'komplementerar' vi det positiva talet.

Ett skissartat program skulle då se ut så här:

```
LD        UT
BSC L     REVER,-
XIO write ECO2
XIO write IT
XIO write ECO1
```

```
        XIO control
        .....
        XIO write  EC01

        XIO write  EC02

        BSC L START

REVER EOR        ETTOR

        A          ETT

        XIO write  IT

        XIO write  EC01

        XIO control
        .....
        XIO write  EC01

        BSC L      START
```

Minnesutrymme 19 celler

Summan av ordernas medelexekveringstider 97 µs

Varje ändring i utsignalen kostar 57 µs effektiv programtid i medeltal.

Upplösningen beror framför allt på frekvensen hos klockan. Det medför att upplösningen inte kan drivas längre än till 125 µs.

II. 3 ANalog styrning med D/A-omvandling

u(t) digital/analogomvandlas till en konstant analog spänning, som skall genereras under en viss bestämd tid.

IBM anger 4 modeller av DAC i referens 1. Tabellmässigt ser deras prestanda ut så här:

modell	bitar	omvandlings-tid	max. utspänning	ant. utgångar	utg.imp.	upplösning
1	10	4 µs	±4.995	1	10k	10 bitar
2	10	4 µs	+4.995	2	10k	10 "
3	13	4 µs	±4.9994	1	10k	13 + tecken
4	13	4 µs	±4.9994	2	10k	13 + tecken

Dessa omvandlare kräver en precisionsspänningskälla.

Ett skissartat program blir då:

```
LD          UT
XIO write DAC
XIO control
BSC L       START
```

Instruktionen XIO control triggar en monostabil multivibrator, som står tillslagen en viss enhetstid. Under denna tiden är en'gate' öppen som släpper igenom den analoga spänning som är proportionell mot u(t) till motorn.

Minnesutrymme 5 celler

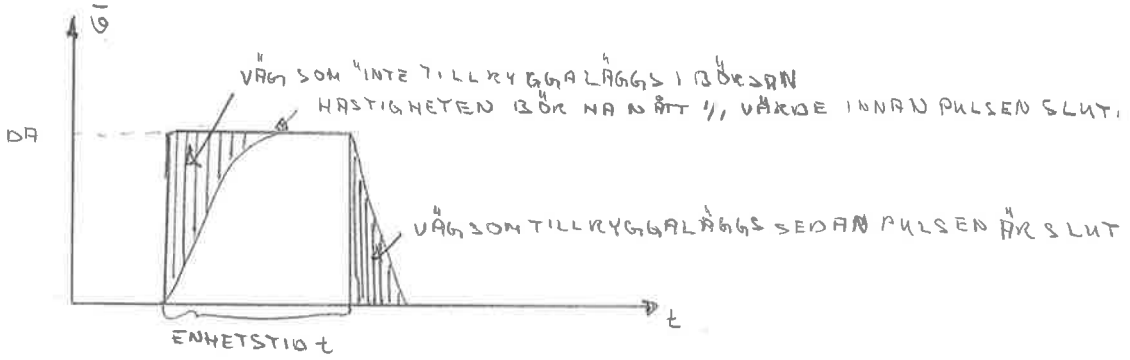
Summan av ordernas medelexekveringstiderna 23 µs.

Varje ändring i utsignalen kostar 23 µs i effektiv programtid.

Enhetstiden får inte vara större än att den förra utsignalens ändring avklingat när nästa kommer.

I denna lösning matas motorn med en konstant analog spänning under en viss bestämd tid. Denna enhetstid kan provas ut individuellt för varje motor. En viss tidsosäkerhet (Δt) förekommer i längden av denna puls. Om pulsen görs längre ökar exaktheten i styrningsmetoden, $\frac{\Delta t}{t}$ minskar nämligen.

Tröghetsmomentet hos ställorganet spelar också in, men trögheten vid igångsättningen kan matchas mot trögheten när signalen"slås av".



Lösningen ställer stora krav på motorns linearitet. Sambandet mellan spänning och rotationshastighet, liksom mellan spänning och moment bör vara linjära.

Motorn ska ha samma startmoment i båda riktningarna.

Vissa av de här problemen kan man komma ifrån genom en linjär förstärkning av utsignalen från DA-omvandlaren och , en stor utväxling mellan motoraxeln och ställorganet.

Upplösningen bestämms av DA-omvandlaren ($\pm \frac{1}{2}$ LSB), av enhetspulsens upplösning ($\frac{\Delta t}{t}$) och av de ingående komponenternas linearitet.

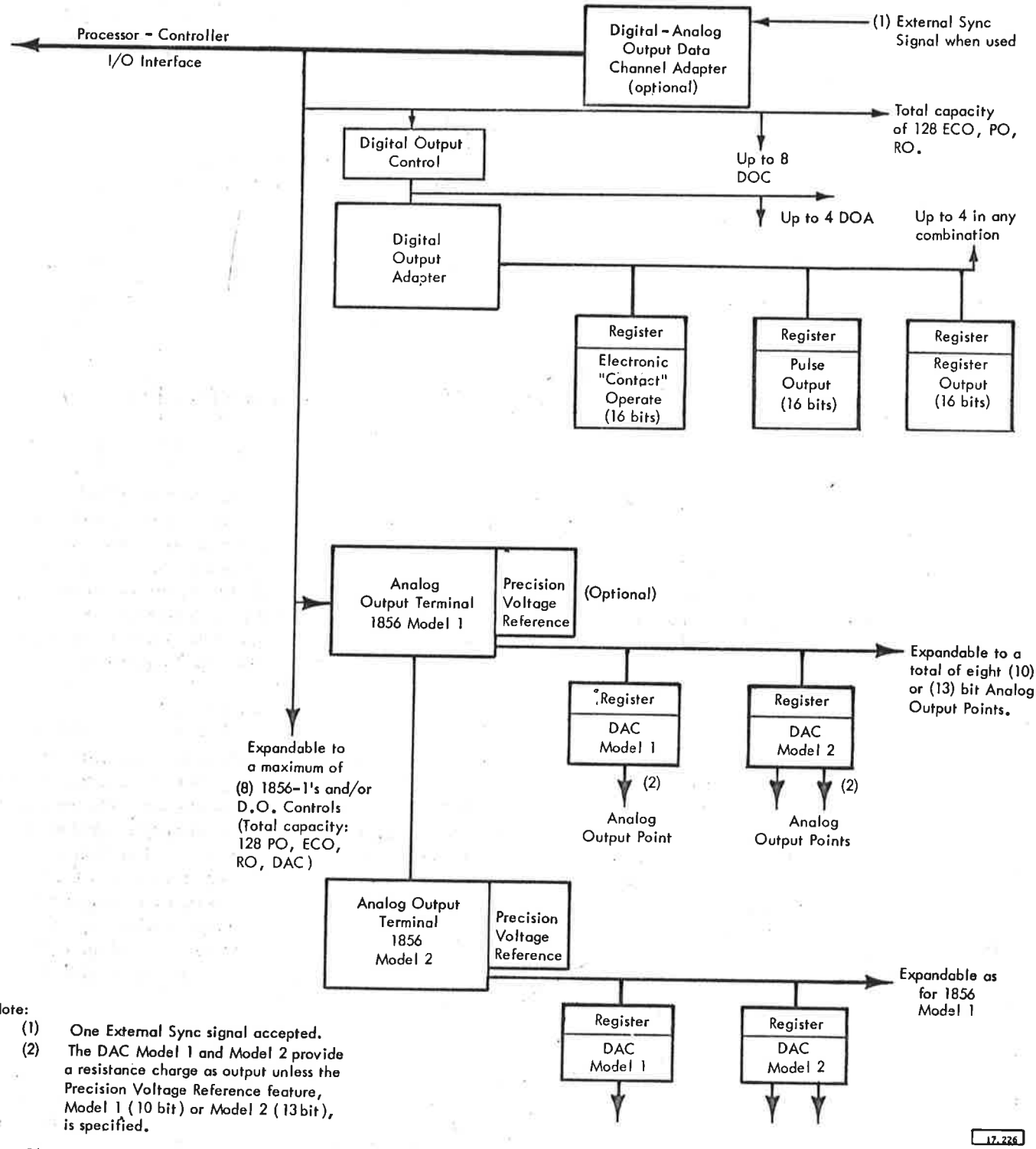
II. 4 Sammanställning över styrmetoderna

METOD	1 (stegmotor)	2 (synkronmotor)	3 (DA-omvandl.)
Minnesutrymme	21	19	5
Σ medelexek. tider	85 µs	97 µs	23 µs
Effektiv program-tid/ändring	23 µs/puls	57 µs	23 µs
Upplösning beror bl.a. av	stegets längd	frekv.hos klockan motorns lin.	DA-omv. $\pm \frac{1}{2}$ LSB enhetsp. uppl komp. lin. $\frac{\Delta t}{t}$

Det som här sagts får endast betraktas som en inledande diskussion över styrmetoderna. Systemens krav måste givetvis fälla utslaget. Detta kan bara tjäna som en pekpinne åt den enskilde som "sitter med styrlag och systemkrav i sin hand".

II. 5. Referenser

1. IBM 1800 Data Acquisition and Control System Reference Manual (Form A26-5918-1)
2. SLO-SYN Catalog SS1265E-2, The Superior Electric Company, Bristol, Connecticut, Sandblom & Stohne AB



Note:

(1) One External Sync signal accepted.

(2) The DAC Model 1 and Model 2 provide a resistance charge as output unless the Precision Voltage Reference feature, Model 1 (10 bit) or Model 2 (13 bit), is specified.

Figure 21. Digital and Analog Output Feature Schematic

KAPITEL III

DDC - PAKET - ETT EKONOMISKT SÄTT ATT PROGRAMMERA

När man programmerar styrningen av en process har man ofta en rad önskemål att uppfylla. Till exempel kan man vilja ha (ref 1):

1. Möjligheter att mata in kontrollkonstanter, ställa in värden, alarmgränser och andra data on-line.
2. Möjligheter att ändra styrlagen medan resten av systemet är on-line.
3. Inte behöva begränsa sig till en standard-styrlag.
4. Enkla möjligheter att ändra från DM-styrning till konventionell analog styrning.

Dessa krav pekar mot komplexa program, individuellt skrivna för varje process och som behöver stort minnesutrymme. Ur ekonomisk synpunkt kräver vi samtidigt:

1. Att programmet tar minimalt minnesutrymme.
2. Att programmet blir ett standardpaket för alla processer så att tiden för uppritning av flödesschema och programmering reduceras.

Lösningen till problemet har blivit det s.k. ddc-paketet, dvs standardbyggblock av program. Grundblocket är en enkel subrutin för PID-reglering. Genom att lagra parametrarna kan vi få så många styrblock vi önskar.

Via operatörens kontroll-panel kan blocken kopplas samman till mer komplexa system. På så sätt får man ett standard program-paket som kan assembleras och modifieras on- eller off-line för att passa många reglertillämpningar. Genom denna teknik reduceras programmeringstiden enligt uppgift med en faktor 5, jämförd med tidigare tekniker. (ref 1).

III.1 'FILL-IN-THE-FORM Programming' (ref 10)

Ett av de stora problemen med programmering av PDM är kommunikationen mellan processingenjören och programmeraren. Processingenjören kan för lite om programmering och programmeraren kan för lite om processtyrning för att resultatet ska bli det bästa.

Ett av de sätt som finns att klara av svårigheterna är s.k. FILL-IN-THE-FORM Programming.(På svenska borde väl termen närmast översättas med fyller-i programmering, men termen är "nersmittad".) Processingenjören definierar processens kontroll-punkter genom att han fyller i de variabler som används och de variabeljusteringar som ska göras i en speciell blankett. Varje ifylld blankett programmeras upp till ett 80-kolumns kort. den resulterande korthögen utgör grunden till on-line programmet.

III. 2 Fördelar

FILL-IN-THE-FORM Programming är ett sätt att standardisera programvaran för on-line processer. Dess största fördelar:

1. lätt att förbereda
2. enkel felsökning
3. enkel att modifiera

En viktig biprodukt är den standardiserade systematiken, som gör det lättare att förstå dokumentationen av programmet och därmed kunna bygga vidare på det. Inte minst för den som kastas mellan olika programmeringsuppgifter är en fullständig och med samma system uppbyggd dokumentation väsentlig.

III.3 PROSPRO/1800 (ref 10) och (ref 12)

Ett av de generaliserade programmeringssystemen för processkontroll är PROSPRO/1800. Det är utvecklat av IBM och baserat på ett liknande system som gjorts av Humble Oil & Refining Co.
Systemet är orienterat mot användaren. Drt kräver detaljerade kunskaper om processen som ska styras, men användaren behöver i gengäld inte vara insatt i programmering.

Sex typer av informationsblanketter fylls i av processingenjören. I dessa specificerar han processens allmäna egenskaper, de beräkningar som ska göras och kontroll-åtgärder och ingripanden som ska göras för att lösa problemet med processtyrningen. Ingenjören behöver inte bekymra sig om de olika assembly- och kompilerings-språkens begränsningar, utan kan helt koncentrera sig på att med hjälp av sina tekniska kunskaper göra riktiga bedömningar.Uppläggningen av blanketterna är sådan att de lämnade tekniska data ger tillräckligt underlag för programmeringen och dessutom i rätt ordning.

Som nämnts krävs det sex typer av blanketter för att ge den nödvändiga informationen om processen. Var och en identifieras med ett 4-siffrigt nummer föregånget av en typsiffra:

Variable information sheet	0
General action sheet	1
General equation sheet	1
Adjustment information sheet	2
Adjustment reference sheet	2
Miscellaneous data sheet	-

Då och då uppstår problem som löses effektivare med assembly-kod eller med ett kompilerings-språk (t.ex. FORTRAN); siffran 3 identifierar en sådan dokumentation.

'Variable processor' och 'variable adjustment processor'

PROSPRO består av två huvudsektioner: 'variable processor' och 'variable adjustment processor' Se figur nedan ur ref 10.

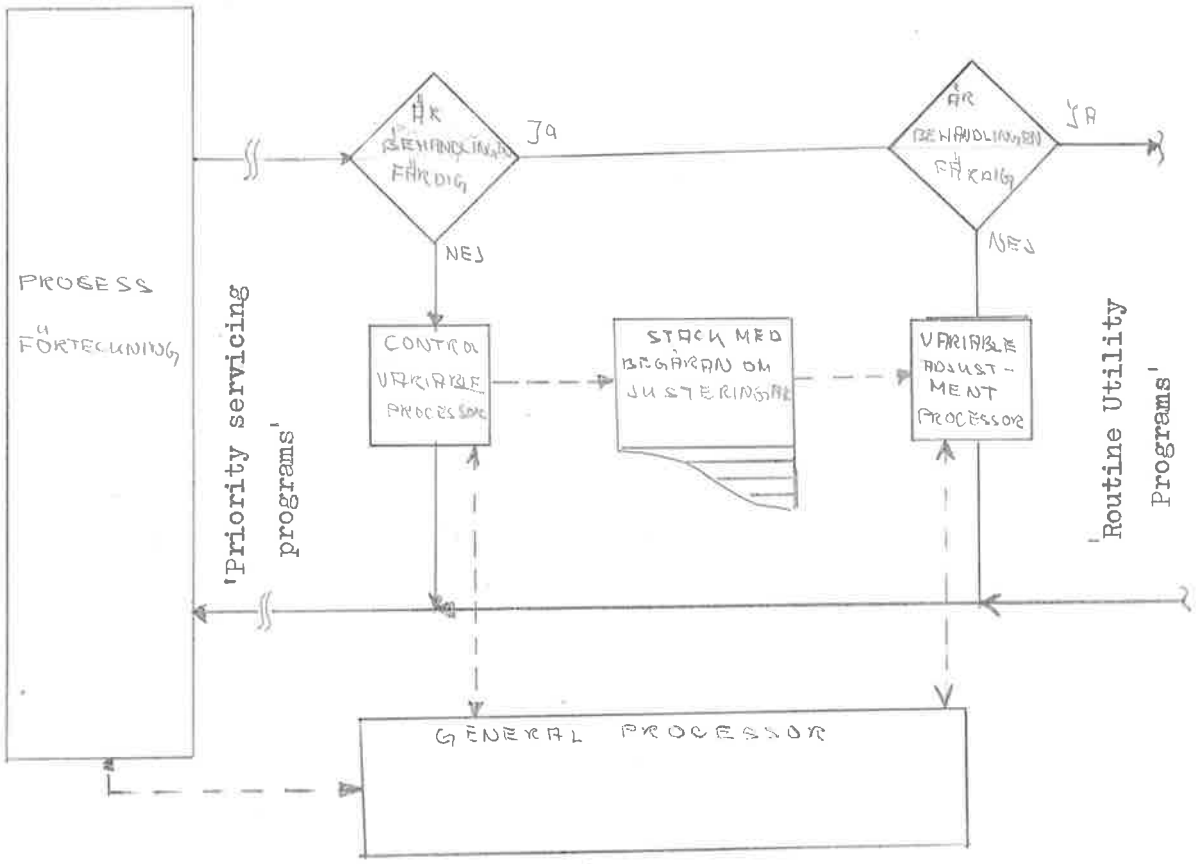


FIG.1 Diagram över PROSPRO system. Generaliserad programvara som 'variable processor' och 'variable adjustment processor' har utökats med information från blanketter från vilka man stansat en hög kort.

'Variable processor' mäter aktuellt nu-värde, kontrollerar varierande gränser, uppdaterar målvärden och begär justeringar i processen om det är nödvändigt. 'Variable adjustment processor' utför justeringarna både i 'feedforward' och'feedback' tillstånd. Informationen för detta hämtas från blanketterna.

PROSPRO har konstruerats för övervakande kontroll. Ingrepp görs bara om processen avviker från väntade eller önskade förhållanden. Systemet kan opperera 'open-loop' genom att skriva ut rekommenderade variabeljusteringar, eller 'closed-loop' genom att själv utföra ändringarna.

Ett sumariskt exempel på 'FILL-IN-THE-FORM Programming ges i referens 10.

FILL-IN-THE-FORM Programming är lämpat för ett brett område, men speciellt når det gäller långsamma processer, där svarstider under 30 sekunder ej behövs.

PROSPRO iblandf effektivare än hög-nivå-språk

Även om processingenjören är en skicklig programmerare är det inte säkert att processen vinner i effektivitet genom att den skrivs i hög-nivå-språk (t.ex FORTRAN) eftersom PROSPRO producerar ett maskinkodat program. Detta gäller speciellt för mindre DM (t.ex IBM 1800) där det program som produceras av Hög-nivå-kompilatorn kan vara långsammare än maskin-koden som produceras av PROSPRO.

I de fall där PROSPRO kan användas uppskattas kostnaden vara markant mindre än för konventionell programmering. Det är också en stor fördel att processingenjören helt kan koncentrera sig på sina primära uppgifter och slipper den rent formella och rutinbetonade programmeringen.

III.4 DDC-paket för styrning av glastillverkning (ref 11)

I artikeln "A DDC Software Package for a FLOAT GLAS PROCESS" berättar N.R.Patel, Ford Motor Co om hur man styr glas-tillverkningen i en av Fords nya glasfabriker. Processen framställer ett 100 tum brett kontinuerligt band av 1/8 tum tjockt bilglas med hastigheten 8 miles per dygn, 7 dar i veckan.

Datamaskinen, en IBM 1800, som programmerats med ett ddc-paket kontrollerar värmetillförseln för smältning, tenn-badet, härdningen och gasgenereringen. Den har cirka 500 analoga och 200 digitala ingångar, samt 80 återkopplingar.

DDC programmet arbetar under ett 'supervisory program'som är fixerat i kärnminnet. I det överordnade programmet finns kontroll för flernivåiga avbrott, intervall-tidkontroll (intervall time control) subrutiner för sekventiering av process-styrningen (process controll subroutine sequencing) och felupptäckande kontrollrutiner (error alert control routines).

Flernivåig avbrottskontroll är ett program som flyttar uppmärksamheten till processhändelser eller maskinvarumässigt upptäckta funktioner med högre nivå. Det innebär också att man måste göra alla subrutiner tillgängliga på alla prioritetsnivåer.

Intervall-tidkontroll består av programvara som använder en eller flera digitala klockor för att försäkra att processvariablerna avsöks genom 'polling'-rutiner i rätt tidsintervall. 'Polling'-rutinen definierar när en speciell punkt ska avsökas, jämför avsökningsfrekvensen med klock-referensen och initierar'scanning' av varje särskild signal under rätt cykel.

När det digitala värdeten gång har erhållits, kan det användas av DM för kontrolländamål. Om det är en kontrollvariabel, går den igenom en PID regulator eller andra algoritmer. Standard ddc-paketet tillåter användaren att lägga till kontrollalgoritmer för feedforward, adaptiv avstämning osv. Se figur 2. ur ref. 11.

DDC-paketet huvudsakligen ett tillämpat program.

Genom en uppsättning subrutiner kan man erhålla kommunikation människa-maskin. Standard ddc-paket innehåller 'utility' program som gör det möjligt för användaren att utöka datamaskinens alarm- och kontrollsystem. DDC-paketet utvecklades av IBM huvudsakligen som ett tillämpat program. Det har därför inte den utbyggda felsökning som finns i t.ex. FORTRAN-kompilatorer. Det har emellertid diagnostiska rutiner som kontinuerligt kontrollerar maskinens aritmetik, A/D och D/A rutiner osv. Paketet tillåter också temeraturkompensering.

Mångsidighet hos ddc-paket

Det är viktigt att förstå och uppskatta organisationen av data i ddc-paketet. Signaler som ska undersökas och kontrolleras "travas" i en tabell vars totala antal ord varierar beroende på komplexiteten i varje variabel. Genom detta kraftfulla data-arrangemang är det möjligt för användaren-programmeraren att direkt lägga till ord som behövs för olika tilläggs-rutiner.

Ett ord kan brytas ner i bitar, där en eller flera bitar kan användas för att karakterisera skilda egenskaper hos variabeln.

Av stor betydelse är flexibiliteten hos ddc-paket. Smärre ändringar i programmet kan ofta göras direkt via operatörens kontrollpanel. I de flesta fall kan sådana ändringar göras rättfram. Tidsvinsten blir i dessa fall betydande.

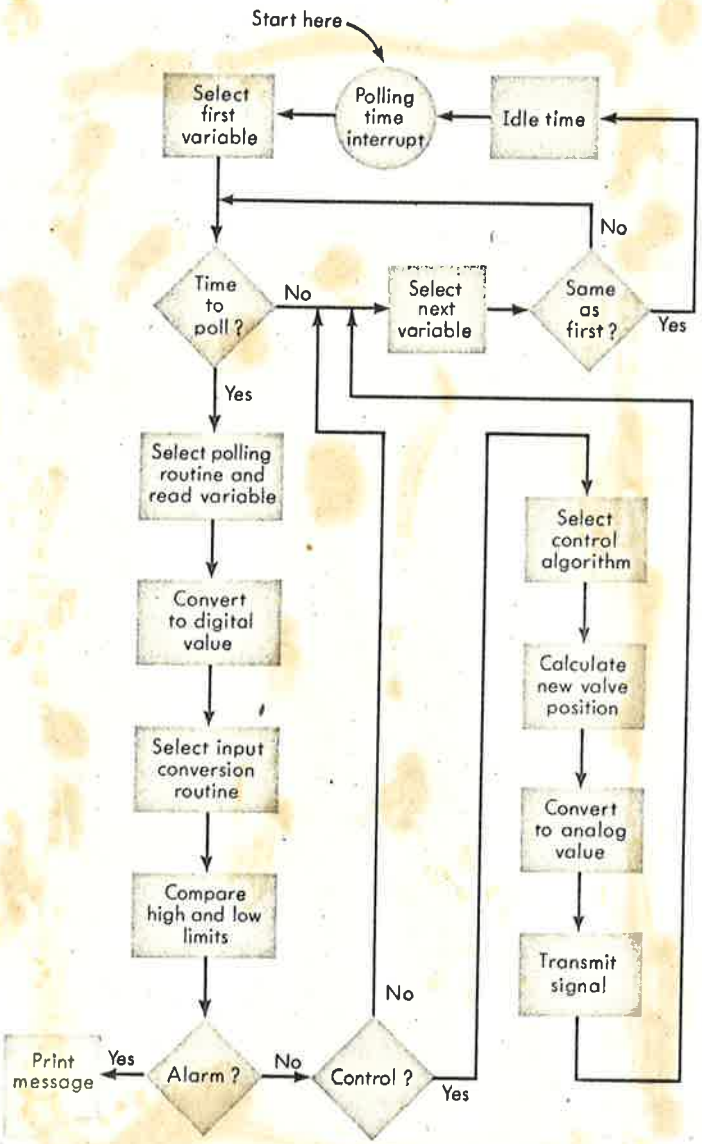


FIG. 2. Flow chart showing how the various ddc package routines are called up during the variable polling sequences.

III.5

REFERENSLISTA för DDC-paket

1. Block diagrammic programming in computer control projects.
   By W.T.Lee, R.M.Boardman and J.D.Higham. Publicering okänd.
   Uppsatsen beskriver ett programsystem för processtyrning av en fabrik med hjälp av DM on-line. Styrsystemet specificeras i blockschema och direkt i reglertekhisk terminologi. Exempel ges på syntes av enkla och komplexa styrlagar och på processtyrning med hjälp av program paket. Förhållandevis avancerat men mycket lärorikt.
2. Direkt Digital Styrning av Civilingenjör J Wickman, IBM SVENSKA AB.1966.
   Rapporten beskriver allmänt DDC och mer i detalj ett systems funktioner och programsystem. Lättläst, men ibland ovidkommande och inte helt 'up to date' .
3. (Artikelserien "Programming for better Control" ur Control Engineering med början oktober -67. Referensnummer 3 - 11. Serien är ej avslutad. Framförallt är artikel 10 och 11 intressanta ur ämnets synpunkt, men även de övriga har tagits med)
   Critical Factor in Computer Usage. Williams and Bailey. Enkel förklaring av hur DM fungerar. Spec. med a.p. Software.
4. A matter of Logic, Memory, and Timing. TJ. Harrison, IBM.
   Författaren presenterar grundblocken i en DM ur processingenjörens synvinkel. Hardware.
5. How hardware responds to software. T.J.Harrison, IBM.
   Författaren presenterar ett exempel på samarbete mellan hardware-software i en typisk process kontroll situation.
6. Importance of Manufacturer Software. S.Weaver, IBM
   Belyser samverkan mellan tillverkarens programvara och användarens.
7. Assembly vs. Compiler Languages. W.Kipiniak och P. Ouint.
   Innehåll se rubrik.
8. FORTRAN for On-line Control. J.E. Clough.
   Framsteg i kompilator-tekniken gör det möjligt för ingenjören att skriva i FORTRAN utan att behöva veckla in sig i Assembly-kod.
9. Software for a Manufacturing Line. E.C. McIRVINE, Ford.
   Författaren presenterar i flödesscheman de steg som måste tas innan man kan skriva program för On-line kontroll.

10. FILL-IN-THE-FORM Programming. Gordon W. Markham, IBM. CE 5-68
Ett sätt att programmera utan att kunna programmera.

11. A DDC Software Package for a Float Glass Process. N.R.Patel, Ford, 68 6
Författaren beskriver erfarenheterna av ddc-paket för IBM 1800 vid Fords nya glas fabrik.

12. 1800 Supervisory Program (PROSPRO/1800) Form H20-0261-0

AVDELNING B

Programmering av PI-regulator

för två processdatamaskiner.

Algoritmen skrivs i assembly-kod i fix räkning, enkel och dubbel precision samt flytande räkning. I fix räkning, enkel precision skrivs algoritmen i två versioner en skalad och en modifierad version. Den modifierade versionen innebär att insignal och referensvärde redan från början tilldelas en skalfaktor fyra, medan den skalade versionen redan från början använder full noggrannhet.

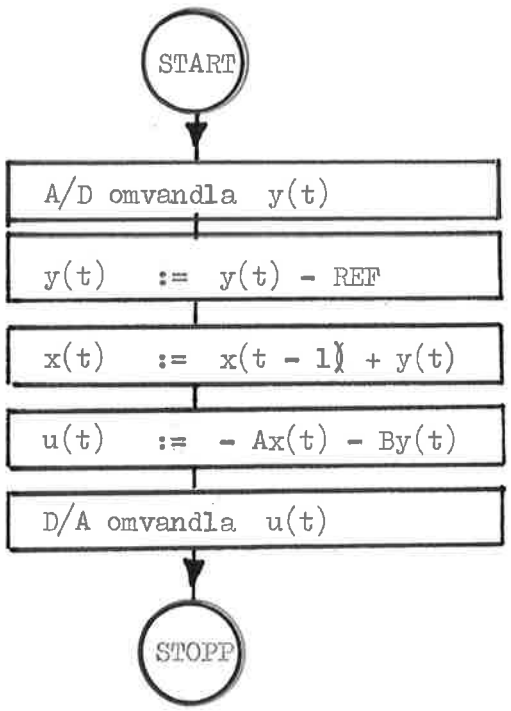
Algoritmen lyder:

A/D omvandla mätsignalen y(t)

y(t) := y(t) - REF
x(t) := x(t - 1) + y(t)
u(t) := - Ax(t) - By(t)

D/A omvandla u(t)

I flödesschema tar det sig ut så här:



KAPITEL IV

HP 2115. PROGRAMMERING AV PI-REG.

HP 2115 är en synnerligen kompakt 'general purpose digital computer', dvs en datamaskin lämpad inte bara för ren processreglering utan även för körning av aritmetiska beräkningar som ej kräver exceptionellt mycket programutrymme. Ordlängden är 16 bitar + parity check som 'option'.Minnesutrymme 4 K (8K option) och cykel-tid 2 us. Minnet är uppdelat i "sidor" om 1024 ord och första sidan + den sida man befinner sig på kan nås direkt; de andra via indirekt adressering. HP har 70 grundinstruktioner.

Från de övriga jämförda PDM skiljer sig HP speciellt beträffande möjligheterna till mikroprogrammering. Upp till 8 mikroinstruktioner kan således slås samman i ett ord.

HP saknar indexregister; istället har man två ackumulatorer i likhet med SEL 810A. Ackumulatorerna kan ordermässigt utföra samma saker, vilket inte är fallet med SEL 810A.

Användbart är också Extend-registret, inte minst för att ta hand om carry-siffror vid dubbel precision.

HP har en väl utbyggd instrumentpark av I/O - organ. På grund av att firman länge sysslat med just instrumentering, verkar det som HP vad det gäller dessa saker skulle vara överlägsen.

SOFT-ware sidan är också väl utbyggd med både FORTRAN-, ALGOL- och ASSEMBLER-kompilatorer. En styrka är maskinens BASIC-program som gör att operatören kan skriva sitt program direkt på Tele-type, starta exekveringen från den och få svar omgående via den.

AD-omvandlingshastigheten för en 14 bitars (13 bitar + tecken) omvandlare är 19 kHz. Aperturtiden 50 µs'basic'och 50 ns med SoH-krets. (ref 6/3-44)En annan typ omvandlar 12 bitar med 51 kHz och 15 bitar med 28 kHz. DA-omvandlingstiden antages vara 10µs.

IV.1 KÖRNING AV PI-REGULATOR I ASSEMBLY-KOD, FIX RÄKNING PÅ HP 2115.

Assembly-koden stansades enligt lista i Bilaga 1. Begynnelsevärden på konstanter och variabler matades in via kontrollpanelen. Härigenom kunde olika begynnelsevärden väljas så att programmet tvingades löpa igenom alla subrutiner och loopar. Subrutinerna kopplades direkt till huvudprogrammet. En räknare i programmet ställdes in på -50 före varje körning så att programmets genomlöptes 50 gånger.

Programmet kompilerades med hjälp av ASSEMBLER, EXTENDED 8 K. Sedan triviala fel hade ändrats och programmet justerats p.g.a. att maskinen saknade Extended Arithmetic Unit, kom utskrift:

```
PAGE 0001

0001                    ASMB,L,B,R
**   NO ERRORS*
```

Programmet listades enligt bilaga 2.

Den erhållna relokerbara tapen matades in tillsammans med Basic Binary Loader. Teletypen skrev ut nedanstående:

```
FIX

02000 02230

LOAD
```

FORTRAN SUBROUTINE LIBRARY matades in och programmet tog hand om .MPY. Följande utskrift erhölls:

```
MPY

02231 02341

*LØ8
```

LØ8 innebär att startadress saknas. Absolut värde för START, Ø21Ø1, sattes på kontrollpanelens vippor, LOAD ADRESS och RUN intrycktes i nämnd ordning. Därvid skrev Tele-typen:

```
*LST
```

Förnyad tryckning av RUN gav:

```
.IOC.   16056
.SQT.   16033
.MEM.   16026
.BUFR   16224
START   02101
.MPY    02231

*LINKS
 01777 01777

*RUN
```

och ytterligare en tryckning startade programmet.

Resultat enligt tabell:

Körning	1:a	2:a	3:e	medelvärde
Tid 50 ggr(s)	32.8	37.0	35.3	35.0
1 gg (ms)	656	740	706	700

Vid beräkning av körtider blev kortaste tid 286 och längsta tid 870. Mellan körningarna har begynnelsevärdena ändrats. Vid en ytterligare körning med extremt avvikande värden tog det 142 s. innan HLT. Detta ligger markant utanför längsta beräknade tid. P.g.a. tidsbrist hittade vi ej något fel, men sannolikt har innehållet i någon programcell ändrats.

Totalt minnesutrymme

FIX	152	celler
.MPY	73	"
Relocatable assembly modes	3072	" - finns inte vid assemblering, ej vid körning.
Basic Binary Loader	64	
Serial Teleprinter Driver	187	
Totalt	okänt antal	celler

I Relocatable assembly modes finns sannolikt plats för både FIX och .MPY. Se bilaga 4.

Startadressen ligger ju exempelvis i ØØØØØ. Det är därför med de informationer som finns omöjligt att göra ens en tillnärmelsevis riktig uppskattning av antalet celler.

IV. 2 PI-REGULATOR I FORTRAN - VERSION. FIX RÄKNING

Program enligt nedan:

```
FTN,B,L,A
      PROGRAM PIREG
      READ(5,*)IXT,IA,IB,IREF,IYT
100   FORMAT(I15)
      DO 10 I=1,50
      IYT=IYT-IREF
      IXT=IX+IYT
      IUT=-(IA*IXT+IB*IYT)
10    CONTINUE
      WRITE(2,100)IUT
      STOP 7777
      END
      END$


500,-1,-1,-399,400

PIREG
```

Anm: Deklarationen INTEGER finns inte i HP:s FORTRAN-version, varför vi istället får börja heltalsvariabelnamnen med I,J,K,L,M,N. Programmet ASSEMBLY-listades se bilaga 3. När programmet kompilerades och binärtapen lästes in stannade remsan och Tele-typen konstaterade "Check Sum Error". Orsaken till parity check-felet var remsstansen som fungerade otillfredsställande.

Minnesutrymmen (se bilaga 5)

Det är även härom möjligt att ge något tillförlitligt mått på minnesutrymmet. Följande minnesutrymmen upptas dock

PIREG	77	celler
.MPY	73	"
.IOC.	142	"
.STOP	17	"
FRMTR	1392	"
Externa subr. till FRMTR	174	"
	1725	celler

IV.3 Kommentarer beträffande multiplikationstider, Extended Arithmetic Unit med mera.

Avsaknaden av Extended Arithmetic Unit ökar multipliktonstiden från 24 us till 187 us, dvs ungefär en faktor 8. Samtidigt förlängs programmet avsevärt genom att lång-skift ej kan användas. Vid en programmering där lång-skift användes blev instruktionslistan 107 order medan antalet rena instruktioner här är 134. (3)

Multiplikationstiderna varierar f.ö. något mellan de olika referenserna. Nö[illegible]:

Referens:	3	4	5	6
Basic 2115	187	190	160	190
2115 wEAU	24	-	-	-
2116A Basic	1150[1]	-	120	150
Divisionstider.				
Basic 2115 A	387	390	375	385
2115A wEAU	26	-	-	-
2116A Basic	310	-	300	310

Någon fullt adekvat jämförelse mellan den egna Assembly-listningen och dem listning som gjorts av FORTRAN-versionen är inte möjlig att göra. Det åtgår ex.vis 84 celler för den modifierade versionen, medan FORTRAN-versionen kräver 77 celler. Då ryms i den modifierade versionen 5 celler för in- och utmatning. När det gäller själva aritmetiken är listningarna relativt lika. Det tyder på att FORTRAN-kompilatorn är effektiv.

1/ siffran 1150 är sannolikt ett feltryck; bör rimligtvis vara 150.

IV.4 Huvudprogram i Fix räkning för HP 2115, Modifierad form utan skalning.

```
ASMB,L,B,R                  /Control statement
        NAM MODI            /Programmets namn
        ENT START           /entry-punkt
        EXT SPILL, LOOP     /externa subrutiner från "FIX"
MREF    OCT 0               /konstanter
MA      OCT 0
MB      OCT 0
MAXI    OCT 077777
MINI    OCT 177777
SKAL    OCT 0
FYRA    OCT 4
XT      BSS 1               /variabler
YT      BSS 1
MAX     BSS 1
TALM    BSS 1
SLAS    BSS 1
START   STC SC              /inläsning
        SFS SC
        JMP * - 1
        LIA SC,C
        ASR 4               /skifta 4 steg åt höger med en gång
        ADA MREF            /och räkna med skalfaktorn 4.
                            /
        STA YT
        ADA XT              / för övriga kommentarer se bil 1 med
                            / samma program skalat.
        STA XT
        MPY MA
        JSB SPILL
        STA MAX
        LDA YT
        JSB LOOP
        MPY MB
```

```
                JSB SPILL
                STA MBY
                JSB LOOP
                ADA MAX
                CLE,ELA
                SEZ
                CMA
                LSL SKAL        /skifta tillbaka SKAL-faktorn +
                LSL 4           /de fyra steg vi började med att skifta
                SEZ
                CMA
                SZB
                JMP SIGN
                ERA
EXIT            OTA SC
                STC SC
KANAL           HLT
                JMP START
SIGN            SEZ
                JMP MINUS
                LDA MAXI
                JMP EXIT
MINUS           LDA MINI
                JMP EXIT
                END
```

IV.5 Huvudprogram i dubbel precision för HP 2115.

```
        ASMB,L,B,R          /Control Statement
        NAM DOUBL           /Programmets namn
        ENT START           /entry-punkt
        (EXT CARRY, SIGN )
ETTOR   OCT 177777          /konstanter
MREF    OCT 0
MA      OCT 0
MB      OCT 0
BIG     OCT 077777
X       BSS 1               /variabler
Y       BSS 1
MAX1    BSS 1
MAX2    BSS 1
MAX3    BSS 1
MAX4    BSS 1
RES1    BSS 1
RES2    BSS 1
RES11   BSS 1
RES12   BSS 1
RES21   BSS 1
RES31   BSS 1
SLASK   BSS 1
```

```
START STC SC          /starta med att sätta kontrollbit
      SFS SC          /skip om flagga satt
      JMP * - 1       /hoppa tillbaka och vänta på flagga
      LIB SC          /läs in i B-ack.
      ASR 16          /lång-skifta höger 16 steg.
      ADA MREF + 1    /addera och lagra enligt nedanstående figur
      SEZ             /om E-ack är satt innebär det carry-siffra
      JSB CARRY       /hoppa till subrutin som adderar denna och
      ADB MREF        /nollställer E-registret
      DST Y
      ADA X + 1
      SEZ
      JSB CARRY
      ADB X
      DST X
      MPY MA + 1
      DST MAX1
      LDA X + 1
      MPY MA
      DST MAX2
      LDA X
      MPY MA + 1
      DST MAX3
      LDA X
      MPY MA
      DST MAX4
      LDA Y + 1
      MPY MB + 1
```

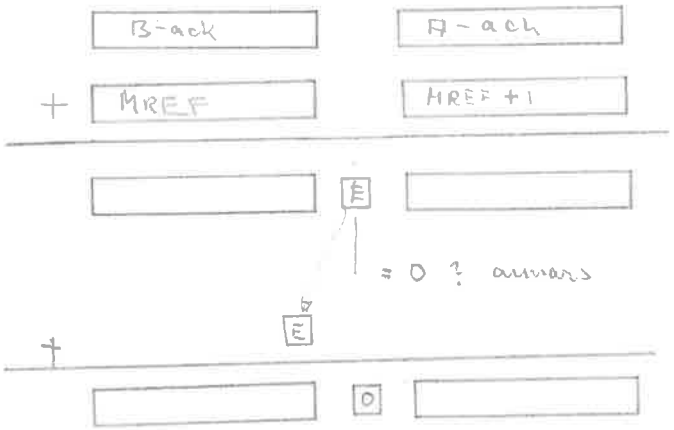
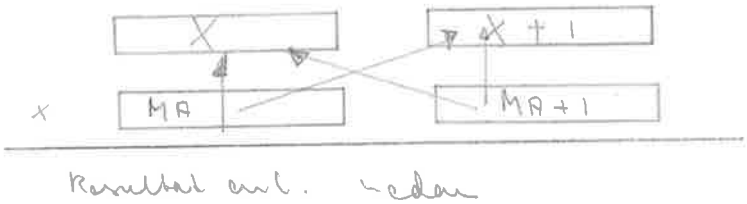
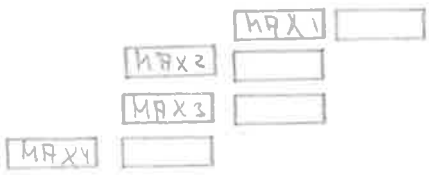


Fig. 1.





P.S.S med MB × Y

```
        ADA MAX1+1
        SEZ
        JSB CARRY
        ADB MAX1
        DST RES1
        SSB
        JMP MINUS
BACK    LDA Y + 1
        MPY MB
        ADA MAX2 + 1
        SEZ
        JSB CARRY
        ADB MAX2
        DST RES2
        SSB
        JMP MINU2
BACK2   LDA Y
        MPY MB + 1
        ADA MAX3 + 1
        SEZ
        JSB CARRY
        ADA RES1
        SEZ
        JSB CARRY
        ADA RES2 + 1
        SEZ
        JSB CARRY
        STA RES1
        STB RES3
        SSB
        JMP MINU3
```

NÄR MB × Y BERÄKNAS, ADDERAS MAX1 TILL

/kommentarer framgår bäst ur figurer på denna och föregående sida.

```
BACK3   LDA Y                   /kommentarer se föregående sidor.
                                 Speciellt figur 1. 2 och 3.
        MPY MB
        ADA MAX4 + 1
        SEZ
        JSB CARRY
        ADA RES3
        SEZ
        JSB CARRY
        ADA RES2
        SEZ
        JSB CARRY
        ADA RES12
        SEZ
        JSB CARRY
        ADB MAX4
        ADB RES11
        ADB RES21
        ADB RES31
        SZB
        JSB SIGN
        ASL 16
        LDA RES1
        SZB
        JSB SIGN
        ASL 16
        LDA RES1 + 1
        SZB
        JSB SIGN
```

```
EXIT    OTA SC          /utmatning till DA-omvandlare
        STC SC          /sätt kontrollbit
KANAL   HLT             /halt
        JMP START
MINUS   LDA ETTOR       /looper för attfylla RES11, RES12,
        STA RES11       RES21 och RES31 med ettor om talen
        STA RES12       blir negativa när de adderas.
        JMP BACK
MINU2   LDA ETTOR
        STA RES21
        JMP BACK2
MINU3   LDA ETTOR
        STA RES31
        JMP BACK3
SIGN    NOP             /subrutin för justering av tecken vid
                        utmatning
        CMB
        SZB
        JMP TECK
        JMP SIGN,I
TECK    SSB
        JMP PLUS
        LDA ETTOR
        JMP EXIT
PLUS    LDA BIG         ∅ talet är för stort
        JMP EXIT
CARRY   NOP             /carry-siffra se figur 1 för
                        förklaring
        STB SLASK
        CLB
        ELB,CLE
        ADB SLASK
        JMP CARRY, I
        END
```

IV.6 Huvudprogram i Flytande räkning för HP 2115

```
        ASMB,L,B,R          /Control Statement för kompilatorn, relocatable, binary output, list output
        NAM FLOAT           /programmets namn

        ENT START           /entry-punkt för programmet
        EXT FLOAT,IFIX,     /yttre använda subrutiner
MREF    OCT 0               /konstanter

MA      OCT 0               /

MB      OCT 0               /

FRY     BSS 1               / variabelutrymmen

FRX     BSS 1               /

MAX     BSS 1               /

START   STC SC              /Starta med att sätta kontrollbit

        SFS SC              /skip om flagga satt

        JMP * - 1           /hoppa tillbaka och vänta på flagga

        LIA SC,C            /läs in i A-ack, 0's flagga

        JSB FLOAT           /förvandla till flytande tal

        FAD MREF            /flytande addition

        DST FRY             /dubbellagring

        FAD FRX             /flytande addition

        DST FRX             /dubbellagring

        FMP MA              /flytande multiplikation

        DST MAX             dubbellagring

        DLD FRY             /hämta FRY

        FMP MB              /flytande multiplikation

        FAD MAX             /flytande addition

        JSB IFIX            /förvandla till fixt tal

        OTA SC              /mata ut från A-ack. till buffert

        STC SC              /sätt kontrollbit

KANAL   HLT                 /stanna och vänta på ny kanal, när SC har ändrats tryck på RUN

        JMP START           /hoppa till START

        END                 / stopp för kompilering
```

IV.7 Huvudprogram i Fix räkning för HP 2115 med skalning.

```
ASMB,L,B,R
        NAM FIX
        ENT START
XT      OCT 500
YT      OCT 400
B141    OCT 40000
M16     OCT 20
MAXI    OCT 177777
MINI    OCT -177776
NOLL    OCT 0
SKAL    OCT 0
SKALX   OCT 0
SLAS    BSS 1
SC      BSS 1
MAX     BSS 1
MBY     BSS 1
TALM    BSS 1
MREF    OCT 1000
MA      OCT -1
MB      OCT -1
COUNT   DEC -50
LOOP    NOP
RETUR   ARS
        INB
        SZB
        JMP RETUR
        LDB TALM
        ADB SKAL
        STB SKAL
        JMP LOOP,I
SKIFT   NOP
BACK    RAL
        INB
        SZB
        JMP FTAM
        JMP SKIFT,I
FTAM    SSA
        JMP SKIFT,I
        JMP BACK
SPILL   NOP
        STA SLAS
        LDA NOLL
        STA TALM
        LDA SLAS
        CLE,ELB
        SEZ
        JSB NEGAT
        ERB
BECK    SZB
        JMP A1XX
        SSA
        JMP A1XX
        SEZ
        JSB NEGAT
        ALS,ERA
        LDB TALM
        CMB
        JMP SPILL,I
```

/Control statement
/program namn
/entry-punkt
/konstanter; även vissa variabler
/har här tilldelats begynnelse-
/värden som ändras under programmets
/gång.

/variabler

/räknare,ställd -50,+1 varje loop
/subrutin,här lagras återhoppadr.
/aritmetiskt höger 1 bit
öka B-ack med 1
/skip om B=0
/hoppa tillbaka
/ladda B-ack med TALM
/addera SKAL
/lagra i SKAL
/hoppa tillbaka via LOOP till huvudpr
/subrutin
/rotera A vänster
/öka B-ack
/skip om B=0
/hoppa FTAM
/hoppa till huvudpr.
/skip om sign A = 0
/hoppa till huvudpr.
/hoppa BACK
/subrutin
/lagra A-ack i SLAS
/nollställ A-ack
/nollställ TALM
/ladda A-ack med SLAS
/0´s E,rotera B vänst. med E
/skip om E = 0
/hoppa subr. NEGAT
/rotera E höger m B
/skip om B=0
/hoppa A1XX
/skip om sign A = 0
/hoppa A1XX
/skip om E = 0
/hoppa subr. NEGAT
/A arit, vänst, Aroteras h. m E
/ladda TALM
/kompl. B
/hoppa SPILL indirekt

```
A1XX   ISZ TALM          /öka TALM 1
       ERB               /rotera B h m E
       ERA,CLE           /rotera A h m E,0's E
       ELB               /rotera B v m E
       JMP BECK          /hoppa BACK
NEGAT  NOP               /subr.
       CLE               /0's E
       CMA               /kompl. A
       CMB               /kompl. B
       JMP NEGAT,I       /hoppa tillbaka
START  LDA XT            /startadress; ladda A-ack med XT
       LDB SKALX         / ladda B med SKALX
       CMB               /kompl. B
       CLE,ELA           /0's E, rotera A v m E
       SEZ               /skip om E  = 0
       CMA               /kompl. A
       SSA,RSS           /Skip sign A=0, tvärtom
       JSB SKIFT         /hoppa subr. SKIFT
       SEZ               /skip om E = 0
       CMA               /kompl A
       ERA               /rotera A h m E
       STA XT            /lagra A i XT
       STB SKALX         /lagra B i SKALX
       CMB               /kompl. B
       STB SKAL          /lagra B i SKAL
       NOP               /markeringsorder för inmatning
       NOP               /
       NOP               /
       LDA XT            /ladda A med XT
       ADA MREF          /addera MREF
       LDB XT            /ladda B med XT
       SOS C             /skip om overflow, clear
       JMP ANOG          /hoppa ANOG
       ISZ SKAL          /öka SKAL m 1
       ARS               /aritm. skift höger
       IOR B141          /inklusive OR
       BRS               /B aritm. skift höger
ANOG   STB SLAS          /lagra B i SLAS
       LDB SKALX         /ladda B med SKALX
BRA    SZB               /skip om B=0
       JMP YSKFT         /hoppa YSKFT
       LDB SLAS          /ladda SLAS
       STA YT            /lagra A i YT
       ADB YT            /addera YT
       SOS C             /skip om overflow set, clear
       JMP BNOG          /hoppa
       ISZ SKAL          /öka SKAL m 1
       BRS               /B aritm. skift höger
       IOR B141          /inclusive OR
       ARS               /A aritm skift höger
BNOG   STB XT            /lagra XT
       LDB SKAL          /ladda SKAL
       STB SKALX         /lagra SKALX
       MPY MB            /multiplicera MB
       JSB SPILL         /hoppa subr. SPILL
       STA MBY           /lagra MBY
       LDA XT            /ladda XT
       SZB               /skip om B=0
       JSB LOOP          /hoppa subr. LOOP
```

```
        MPY  MA
        JSB  SPILL
        STA  MAX
        LDA  MBY
        SZB
        JSB  LOOP
        ADA  MAX
        ELA
        SEZ
        CMA
        LDB  SKAL
        CMB
RUNT    SZB
        JMP  TILL
        SSA
        JMP  SIGN
        INB
        ALS
        JMP  RUNT
TILL    SEZ
        CMA
        ERA
EXIT    NOP
        NOP
        JMP  KANAL
SIGN    SEZ
        JMP  MINUS
        LDA  MAXI
        JMP  EXIT
MINUS   LDA  MINI
        JMP  EXIT
YSKFT   ARS
        INB
        JMP  BRA
KANAL   ISZ  COUNT
        JMP  START
        HLT
        END
```

```
/mult MA
/hoppa subr. SPILL
/lagra MAX
ladda MBY
/skip om B=0
/hoppa subr. LOOP
/addera MAX
/rotera A v m E
/skip om E=0
/kompl.A
/ladda SKAL
/kompl. B
/skip om B=0
/hoppa
/skip om sign A = 0
/hoppa
/öka B med 1
/skifta A v
/hoppa
/skip om E=0
/kompl A
/rotera A v m E
/markering av utmatning

/hoppa
/skip om E=o
/hoppa
/ladda MAXI
/hoppa
/ladda MINI
/hoppa
/skifta A höger
/öka B
/hoppa
/räknare; öka  m 1
/hoppa
/halt
/slut (kompilering)
```

```
 PAGE 0002 #01


0001                        ASMB,L,B,R
0002  00000                       NAM FIX
0003                               ENT START
0004  00000 000500  XT       OCT 500
0005  00001 000400  YT       OCT 400
0006  00002 040000  B141     OCT 40000
0007  00003 000020  M16      OCT 20
0008  00004 177777  MAXI     OCT 177777
0009  00005 000002  MINI     OCT -177776
0010  00006 000000  NOLL     OCT 0
0011  00007 000000  SKAL     OCT 0
0012  00010 000000  SKALX    OCT 0
0013  00011 000000  SLAS     BSS 1
0014  00012 000000  SC       BSS 1
0015  00013 000000  MAX      BSS 1
0016  00014 000000  MBY      BSS 1
0017  00015 000000  TALM     BSS 1
0018  00016 001000  MREF     OCT 1000
0019  00017 177777  MA       OCT -1
0020  00020 177777  MB       OCT -1
0021  00021 177716  COUNT    DEC -50
0022  00022 000000  LOOP     NOP
0023  00023 001100  RETUR    ARS
0024  00024 006004             INB
0025  00025 006002             SZB
0026  00026 026023R            JMP RETUR
0027  00027 066015R            LDB TALM
0028  00030 046007R            ADB SKAL
0029  00031 076007R            STB SKAL
0030  00032 126022R            JMP LOOP,I
0031  00033 000000  SKIFT    NOP
0032  00034 001200  BACK     RAL
0033  00035 006004             INB
0034  00036 006002             SZB
0035  00037 026041R            JMP FTAM
0036  00040 126033R            JMP SKIFT,I
0037  00041 002020  FTAM     SSA
0038  00042 126033R            JMP SKIFT,I
0039  00043 026034R            JMP BACK
0040  00044 000000  SPILL    NOP
0041  00045 072011R            STA SLAS
0042  00046 062006R            LDA NOLL
0043  00047 072015R            STA TALM
0044  00050 062011R            LDA SLAS
0045  00051 004066             CLE,ELB
0046  00052 002040             SEZ
0047  00053 016074R            JSB NEGAT
0048  00054 005500             ERB
0049  00055 006002  BECK     SZB
0050  00056 026067R            JMP A1XX
0051  00057 002020             SSA
0052  00060 026067R            JMP A1XX
0053  00061 002040             SEZ
0054  00062 016074R            JSB NEGAT
0055  00063 001025             ALS,ERA
0056  00064 066015R            LDB TALM
0057  00065 007000             CMB
0058  00066 126044R            JMP SPILL,I
```

```
0059  00067 036015R A1XX   ISZ TALM
0060  00070 005500         ERB
0061  00071 001540         ERA,CLE
0062  00072 005600         ELB
0063  00073 026055R        JMP BECK
0064  00074 000000  NEGAT  NOP
0065  00075 000040         CLE
0066  00076 003000         CMA
0067  00077 007000         CMB
0068  00100 126074R        JMP NEGAT,I
0069  00101 062000R START  LDA XT
0070  00102 066010R        LDB SKALX
0071  00103 007000         CMB
0072  00104 000066         CLE,ELA
0073  00105 002040         SEZ
0074  00106 003000         CMA
0075  00107 002021         SSA,RSS
0076  00110 016033R        JSB SKIFT
0077  00111 002040         SEZ
0078  00112 003000         CMA
0079  00113 001500         ERA
0080  00114 072000R        STA XT
0081  00115 076010R        STB SKALX
0082  00116 007000         CMB
0083  00117 076007R        STB SKAL
0084  00120 000000         NOP
0085  00121 000000         NOP
0086  00122 000000         NOP
0087  00123 062000R        LDA XT
0088  00124 042016R        ADA MREF
0089  00125 066000R        LDB XT
0090  00126 103301         SOS C
0091  00127 026134R        JMP ANOG
0092  00130 036007R        ISZ SKAL
0093  00131 001100         ARS
0094  00132 032002R        IOR B141
0095  00133 005100         BRS
0096  00134 076011R ANOG   STB SLAS
0097  00135 066010R        LDB SKALX
0098  00136 006002  BRA    SZB
0099  00137 026223R        JMP YSKFT
0100  00140 066011R        LDB SLAS
0101  00141 072001R        STA YT
0102  00142 046001R        ADB YT
0103  00143 103301         SOS C
0104  00144 026151R        JMP BNOG
0105  00145 036007R        ISZ SKAL
0106  00146 005100         BRS
0107  00147 032002R        IOR B141
0108  00150 001100         ARS
0109  00151 076000R BNOG   STB XT
0110  00152 066007R        LDB SKAL
0111  00153 076010R        STB SKALX
0112  00154 016001X        MPY MB
      00155 000020R
0113  00156 016044R        JSB SPILL
0114  00157 072014R        STA MBY
0115  00160 062000R        LDA XT
```

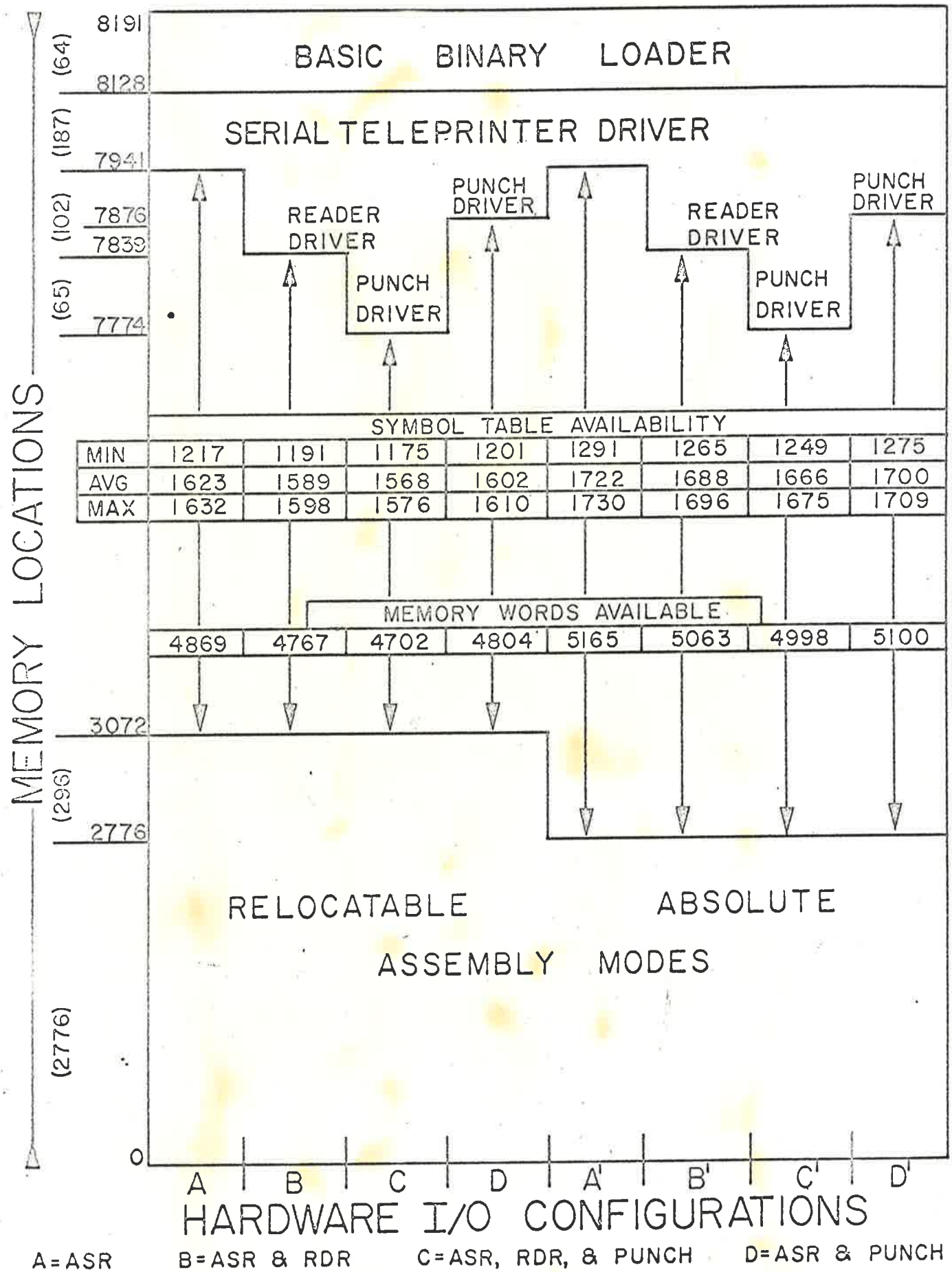
```
 PAGE 0004 #01


0116   00161 006002          SZB
0117   00162 016022R         JSB LOOP
0118   00163 016001X         MPY MA
       00164 000017R
0119   00165 016044R         JSB SPILL
0120   00166 072013R         STA MAX
0121   00167 062014R         LDA MBY
0122   00170 006002          SZB
0123   00171 016022R         JSB LOOP
0124   00172 042013R         ADA MAX
0125   00173 001600          ELA
0126   00174 002040          SEZ
0127   00175 003000          CMA
0128   00176 066007R         LDB SKAL
0129   00177 007000          CMB
0130   00200 006002  RUNT    SZB
0131   00201 026207R         JMP TILL
0132   00202 002020          SSA
0133   00203 026215R         JMP SIGN
0134   00204 006004          INB
0135   00205 001000          ALS
0136   00206 026200R         JMP RUNT
0137   00207 002040  TILL    SEZ
0138   00210 003000          CMA
0139   00211 001500          ERA
0140   00212 000000  EXIT    NOP
0141   00213 000000          NOP
0142   00214 026226R         JMP KANAL
0143   00215 002040  SIGN    SEZ
0144   00216 026221R         JMP MINUS
0145   00217 062004R         LDA MAXI
0146   00220 026212R         JMP EXIT
0147   00221 062005R MINUS   LDA MINI
0148   00222 026212R         JMP EXIT
0149   00223 001100  YSKFT.,ARS
0150   00224 006004          INB
0151   00225 026136R         JMP BRA
0152   00226 036021R KANAL   ISZ COUNT
0153   00227 026101R         JMP START
0154   00230 102000          HLT
0155                         END
**   NO ERRORS*
```

IV.9 LISTNING AV FORTRAN-VERSIONEN.

```
PAGE 0001  PIREG


00000  000000  BSS 000000
00000  000000  OCT 000000
00001  016001X JSB CLRIO
00002  000003R DEF 000003
00003  062111R LDA 000111
00004  006404  OCT 006404
00005  016002X JSB .DIO.
00006  000000  OCT 000000
00007  100104R DEF 000104,I        Inläsning
00010  016003X JSB .IOI.
00011  072074R STA 000074          IXT
00012  016003X JSB .IOI.
00013  072075R STA 000075          IA
00014  016003X JSB .IOI.
00015  072076R STA 000076          IB
00016  016003X JSB .IOI.
00017  072077R STA 000077          IREF
00020  016003X JSB .IOI.
00021  072100R STA 000100          IYT
00022  126106R JMP 000106,I
00023  024040  OCT 024040
00024  044461  OCT 044461
00025  032451  OCT 032451
00026  062112R LDA 000112          Hämta 1
00027  072101R STA 000101          och lagra i 101
00030  062100R LDA 000100          ladda IYT
00031  003004  OCT 003004
00032  042077R ADA 000077          addera IREF
00033  003004  OCT 003004
00034  072100R STA 000100          lagra IYT
00035  062074R LDA 000074          ladda IXT
00036  042100R ADA 000100          addera IYT
00037  072074R STA 000074          lagra IXT
00040  062075R LDA 000075          ladda IA
00041  016004X JSB .MPY            multiplicera
00042  000074R DEF 000074
00043  072103R STA 000103          lagra i MAX
00044  062076R LDA 000076          ladda IB
00045  016004X JSB .MPY            multiplicera
00046  000100R DEF 000100
00047  042103R ADA 000103          addera IMAX
00050  003004  OCT 003004
00051  072102R STA 000102          lagra IUT
00052  062101R LDA 000101          hämta 101
00053  002004  OCT 002004
00054  072101R STA 000101          lagra i 101
00055  003004  OCT 003004
00056  042113R ADA 000113
00057  002021  OCT 002021
00060  026030R JMP 000030
00061  062114R LDA 000114
00062  006400  OCT 006400
00063  016002X JSB .DIO.
00064  100105R DEF 000105,I
00065  100110R DEF 000110,I
00066  062102R LDA 000102          mata ut
00067  016003X JSB .IOI.
00070  016005X JSB .DTA.
```

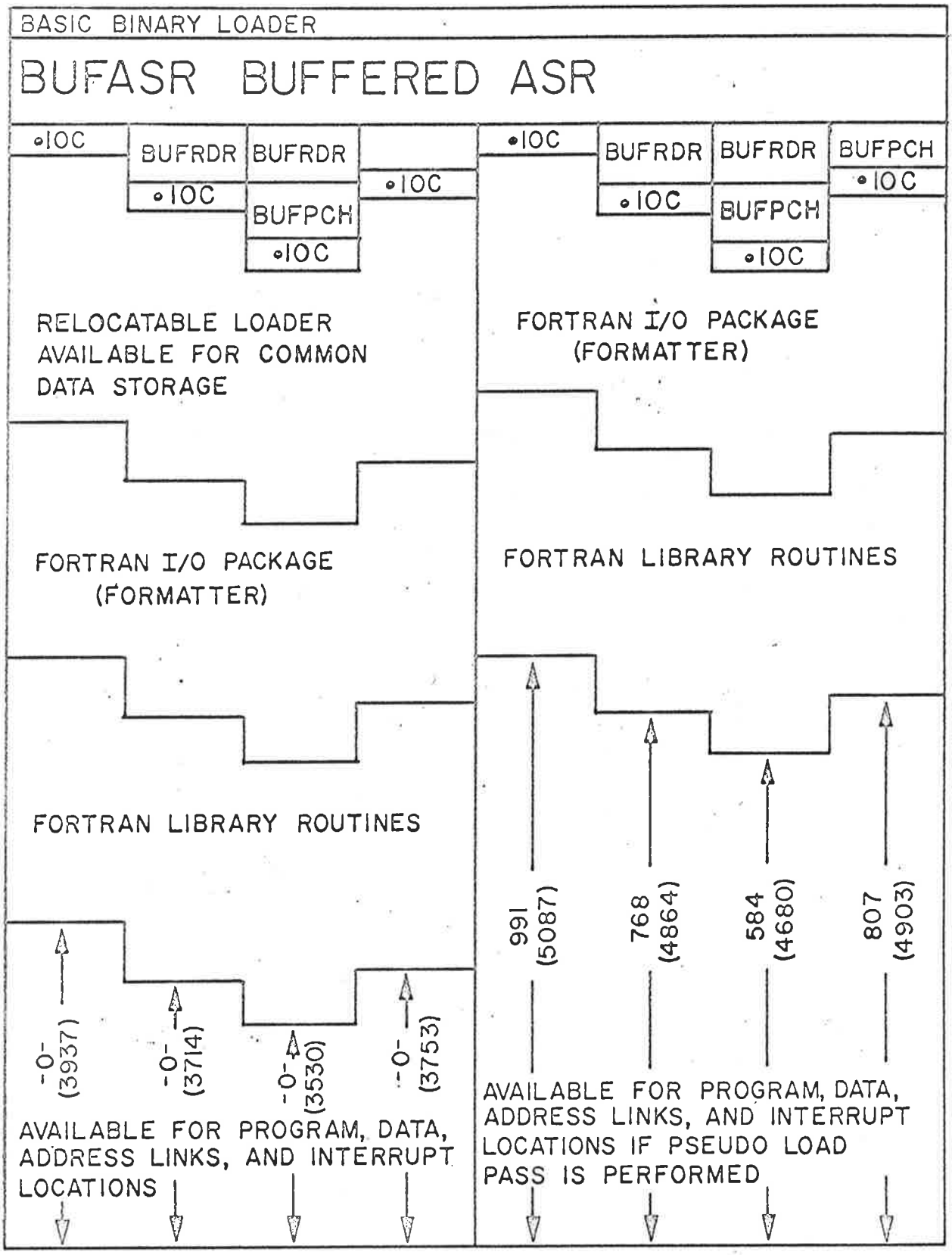
```
PAGE 0002  PIREG


00071   062115R LDA 000115        ladda 777
00072   016006X JSB .STOP         Stopp
00073   016006X JSB .STOP
00074   000000  BSS 000007
00103   000000  OCT 000000        IMAX
00104   000022R DEF 000022
00105   000023R DEF 000023
00106   000026R DEF 000026
00107   000052R DEF 000052
00110   000071R DEF 000071
00111   000005  OCT 000005
00112   000001  OCT 000001        1 oktalt
00113   000062  OCT 000062        50 oktalt
00114   000002  OCT 000002
00115   007777  OCT 007777
                TRA PIREG

*** END
```

HP-ASSEMBLY TIME-8K MEMORY MAP



3-16

FIGURE 4



RUN TIME MEMORY ALLOCATION FOR OBJECT PROGRAMS WRITTEN IN FORTRAN WITH ENTIRE FORTRAN LIBRARY RESIDENT

NUMBERS IN BRACKETS ARE FOR 8K SYSTEMS

3-6

IV.1 REFERENSLISTA för HP 2115.

1. Specifikations and Basic Operation Volume 1.
2. Assembler
3. 2116/2115 Computer Extended Arithmetic Unit option M9
4. 2115 A Technical Data 10 nov. 67
5. FORTRAN/ALGOL library Subroutines. Ingår i Memory Allocation
6. Computer Training Manual, HP

KAPITEL V

SEL 810A. PROGRAMMERING AV PI-REGULATOR.

SEL 810A är en snabb, 'general purpose', 16-bitars datamaskin. På grund av sin relativt låga kostnad, snabba cykel-tid (1.75 us) och flexibla I/O-utrustning lämpar den sig väl för reelltidskörning vid processreglering. Elektroniken är helt uppbyggd med integrerade kretsar. Minnet kan utökas i steg om 4K upp till 32 K.

SEL 810A har den kraftfullaste instruktionsrepertoaren av de undersökta maskinerna. Bland de instruktioner som inte har någon motsvarighet hos HP och IBM kan nämnas två 'branch/skip' instruktioner, CMA och SAS, som orsakar tre-vägshopp. Vidare är normaliseringsordern SNO och "carry-ordern" CSB synnerligen användbara.

Multiplikationsordern MPY är betydligt snabbare än hos HP och IBM (7.0 us mot resp 187 och 15 us).

I/O-instruktionerna är särskilt kraftfulla därför att varje instruktion orsakar flera funktioner att utföras. Den genomsnittliga tid som åtgår för att utföra den kompletta "koppla upp", testa, överför och "koppla ner" operationen är bara tre maskin-cykler(5.25 us). Motsvarande tid för IBM 1800 är 20 us och för HP 8 us (4 maskincykler).

SEL 810A intar vad det gäller indexregister en mellanställning mellan IBM 1800 och HP 2115. SEL har ett 'hard-ware'-mässigt indexregister (B-registret), men det kan också användas som ett aritmetiskt adresserbart register.

I de manualer som konsulterats nämns inget om någon ALGOL-kompilator, men i övrigt verkar SEL ha ett väl utbyggt och väl dokumenterat 'soft-ware system'.

V.1 SEL 810 A. ENKEL PRECISION

```
STRT    CLA          / 0's A-ack
        STA SKAL     / 0's SKAL
        STA SAL2       0's SAL2
        AIP 1,W      / läs in till A-ack. från enhet 1. Vänta på flagga.

        LBA XT       / Ladda B-ack med XT

        SMA REF      / Minska med REF

        SOF          / Skip om ej overflow

        SPB OVER     / lagra adress och hoppa till OVER, som skiftar XT och
                       YT ett steg häger. ökar SKAL-faktor med 1.
        STA YT       / lagra YT

        CLA          / 0's A-ack

        FLA SAL1     / lång-skifta vänster SAL1 gånger

        SAZ          / skip om A=0

        SPB FLOW     / lagra adress och hoppa till FLOW, som skiftar tillbaka
                       XT och YTså att XT får plats. SKAL ökas

        IAB          / Växla så att XT - A-ack och YT - B-ack.

        AMA YT       / Addera XT + YT

        SOF          / Skip om ej overflow

        SPB OVER     / se ovan

        STA XT       / lagra XT

        LAA SKAL     / Ladda SKAL i A-ack

        STA SAL1     / Lagra i SAL1

        MPY MB       / multiplicera (-B) y(t)

        SPB SPIL     / lagra adress och hoppa till SPIL, som normaliserar
                       resultatet och beräknar antal bitar utanför A-ack
        STA MBY      / lagra i MBY

        LAA TALM     / Ladda TALM i A-ack

        STA SALY     / lagra A-ack i SALY

        LBA XT       / ladda XT i B-ack

        MPY MA       / multiplicera (-A)·XT
```

```
        SPB SPIL    / se ovan

        STA MAX     / lagra A-ack i MAX

        LAA TALM    / ladda TALM i A-ack.-TALM anger skalfaktorn för att
                      MAX skulle rymmas i A-ack.
COMP    CMA SALY    / Jämför minne och A-ack. Hoppa n+1 om (A) - (M) < 0,
                      n+2 om =0 och n+3 om > 0.
        BRU SKFY    / skalfaktorn för X störst, hoppa

        BRU EXIT    / skalfaktorerna lika, hoppa

        LAA MAX     / skalfaktorn för Y störst, ladda A-ack med MAX
        LBA SALY    / ladda B-ack m. SALY
        IMS SALY    / öka skalfaktorn för Y, dvs minska absolut

        RSA ETT     / skifta A-ack höger
        STA MAX     / lagra i MAX
        BRU COMP    / hoppa tillbaka

SKFY    LAA MBY     / ladda A-ack med MBY
        LBA TALM    / lagra B-ack i TALM
        IMS TALM    / öka skalfaktorn för X, dvs minska absolut

        RSA ETT     / skifta höger ett steg
        STA MBY     / lagra MBY
        BRU COMP    / hoppa tillbaka

EXIT    IAB         / VÄXLA

        NEG         / complementera

        AMA SKAL    / öka med SKAL

        STA SKAL    / lagra i SKAL

        LBA MAX     / ladda B-ack med MAX

        AMB MBY     / addera MBY
        CLA         / 0's A-ack
        FLA SKAL    / skifta vänster SKAL gånger

        SAZ         / skip om A=0

        BRU UTG     / hoppa till utgång

        IAB         / växla A och B

        BRU ADO     / hoppa till ADO

UTG     IAB         / växla A och B

        SAS         / skippa 0 order om A neg, 1 order om A=0, 2 order om
                      A positiv
        BRU MIN     / hoppa

        BRU ADO     / hoppa

        LAA BIG     / ladda A-ack med BIG

        BRU ADO     / hoppa
```

```
MIN     LAA SMAL    / ladda A-ack med minsta tal.

ADO     AOP 1,W     / mata ut till enhet 1 från A-ack
        HLT
SPIL    NOP         / lagra återhoppsadress här

        STA SLAS    / lagra A-ack i slaskcell

        LAA M15     / ladda A-ack med -15
        STA TALM    / lagra -15 i TALM
DIT     SNO         / Skip om A ej är normaliserat


        BRU* SPIL   / hoppa indirekt tillbaka till huvudprogrammet

HIT     FLA ETT     / skifta A och B-ack vänster ett steg

        IMS TALM    / öka TALM med ett, skip om TALM = 0

        BRU DIT     / hoppa

        BRU* SPIL   / hoppa indirekt tillbaka till huvudprogrammet

OVER    NOP         / här lagras återhoppsadress

        IMS SKAL    / öka SKAL med ett

        RSA ETT     / Skifta A-ack höger ett steg

        STB SLAS    / lagra B-ack i Slaskcell

        LBA B141    / ladda B141 i B-ack

        OBA         / "inclusive or" A och B

        LBA SLAS    / ladda tillbaka innehållet i B-ack

        IAB         / växla A och B

        RSA ETT     / skifta A-ack höger

        IAB         / växla

        BRU* OVER   / hoppa indirekt tillbaka

FLOW    NOP         / här lagra återhoppsadress

BACK    FRA ETT     / skifta A och B höger ett steg
        IMS SAL2    / öka SAL2 m.1
        SAZ         / skip om A = 0

        BRU BACK    / hoppa till BACK

        LAA SAL2    / ladda A med SAL2

        AMA SKAL    / addera SKAL

        STA SKAL    / lagra i SKAL
```

```
        LAA YT          /ladda A-ack med YT
        RSA SAL2        /skifta höger SAL2 gånger
        STA YT          /lagra YT
        BRU*+ FLOW      /hoppa indirekt tillbaka till huvudprogrammet
REF     DATA '0
MA      DATA '
MB      DATA '
ETT     DATA '000001
BIG     DATA '077777
SMAL    DATA '177777
B141    DATA '040000
M16     DATA '177760
YT      BSS 1
XT      BSS 1
SAL1    BSS 1
SAL2    BSS 1
SALY    BSS 1
TALM    BSS 1
SLAS    BSS 1
MAX     BSS 1
MBY     BSS 1
```

V.2 SEL 810A. ENKEL PRECISION. MODIFIERAD VERSION.

Insignalen y(t) skalas direkt med faktorn 4. REF-värdet finns lagrat med samma skalfaktor, medan MA och MB lagrats utan skalfaktor.

```
        AIP 1,W     /läs in A-ack från enhet 1, vänta på flagga
        RSA 4       /skifta 4 steg höger
        SMA REF     /minska med REF
        STA YT      /lagra i YT
        TAB         /överför YT till B-ack
        ADA XT      /addera A-ack + XT
        STA XT      /lagra XT
        MPY MB      /multiplicera B-ack(YT) med MB
        SPB SPIL    /hoppa till subrutin SPIL
        STA MBY     /lagra MBY
        LAA TALM    /ladda A-ack med TALM
        STA SALY    /lagra i SALY
        LBA XT      /ladda B-ack med XT
        MPY MA      /multiplicera med MA
        SPB SPIL    /hoppa till subrutin SPIL
        STA MAX     /lagra i MAX
        LAA TALM    /ladda A-ack med TALM
COMP    CMA SALY    /jämför TALM - SALY
        BRU SKFY    / <0, hoppa till SKFY
        BRU EXIT    / =0, hoppa till EXIT
        LAA SALY    /> 0, ladda A-ack med SALY
        NEG         /komplementera
        STA SKAL    /lagra i SKAL
        LAA MAX     /ladda A-ack med MAX
        IMS SALY    /öka SALY med 1
        RSA 1       /skifta A-ack höger 1
```

```
        STA MAX         /lagra MAX
        BRU COMP        /hoppa till COMP och gör om jämförelsen
SKFY    LAA TALM        /ladda TALM
        NEG             /komplementera
        STA SKAL        /lagra SKAL-faktor
        LAA MBY         /ladda MBY
        IMS TALM        /ökatTALM m 1
        RSA 1           /skifta höger 1 steg
        STA MBY         /lagra MBY
        BRU COMP        /hoppa tillbaka och gör om jämförelsen
EXIT    LBA MAX         /ladda B-ack med MAX
        AMB MBY         /addera MBY till B-ack
        CLA             /0´sA-ack
        FLA SKAL        /skifta vänster SKAL gånger
        FLA 4           /skifta v 4 gånger
        SAZ             /skip om A=0
        BRU ADO         /hoppa till ADO
UTG     IAB             /växla A och B
        SAS             /Skip on A sign
        BRU MIN         /hoppa till MIN om negativt
        NOP             /fortsätt ty om A=0 här så är talet för stort
        LAA BIG         /ladda A-ack med BIG
        BRU ADO         /hoppa till ADO
MIN     LAA SMAL        /ladda A-ack med SMAL
ADO     AOP 1,W         /mata ut till enhet 1 från A-ack
REF     DATA '          /konstanter
MA      DATA '
MB      DATA '
```

```
BIG  DATA '077777          /konstanter
SMAL DATA '177777
YT   BSS  1               /variabler "block starting by symbol"
XT   BSS  1
MBY  BSS  1
TALM BSS  1
SALY BSS  1
MAX  BSS  1
SKAL BSS  1
     END                  /end-statement för kompilatorn
```

I programmet ingår också subrutinen SPIL, angiven och använd i SEL 810A, ENKEL PRECISION.

V.3 SEL 810A. DUBBEL PRECISION

```
STRT AIP 1,W          /läs in till enhet 1, vänta på flagga
     FRA 15           /skifta höger 15 steg, långskift
     AMB MREF + 1       se figurer
     CSB
     AMA MREF
     STA YT
     STB YT + I
     AMB XT + I
     CSB
     AMA XT
     STA XT
     STB XT + I
     MPY MA + I
     STA GGI
     STB MAX4
     LDB XT + I
     MPY MA
     STB GG2 + I
     FRA 15
     STA GG2 - I
     STB GG2
     LDB XT
     MPY MA + I
     STB GG3 + I
     FRA 15
     STA GG3 - I
     STB GG3
```

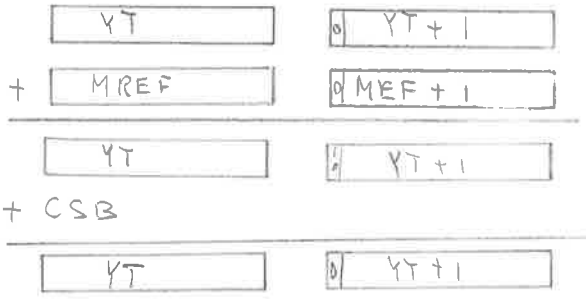


fig. 1

P.s.s med XT + YT = XT

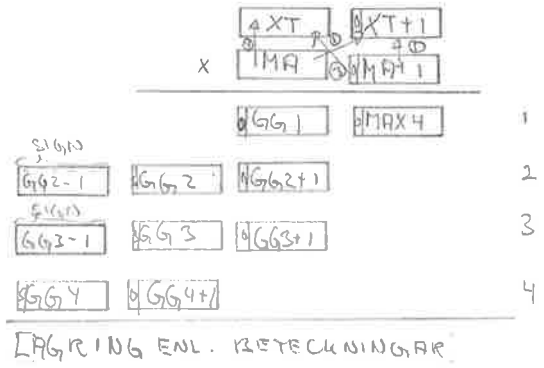


fig 2

```
LDB XT
MPY MA
STA GG4
STB GG4 + I
LDB YT +I
MPY MB + I
AMB MAX4
CSB
AMA GG1
STB RES1
FRA 15
AMB GG2 + I
CSB
AMA GG2
STB RES2
STA RES3
LDB YT + I
MPY MB
AMB RES2
CSB
AMA RES3
STB RES2
STA RES3
LDB YT
MPY MB + I
AMB RES2
CSB
AMA RES3
AMB GG3 + I
CSB
AMA GG3
```

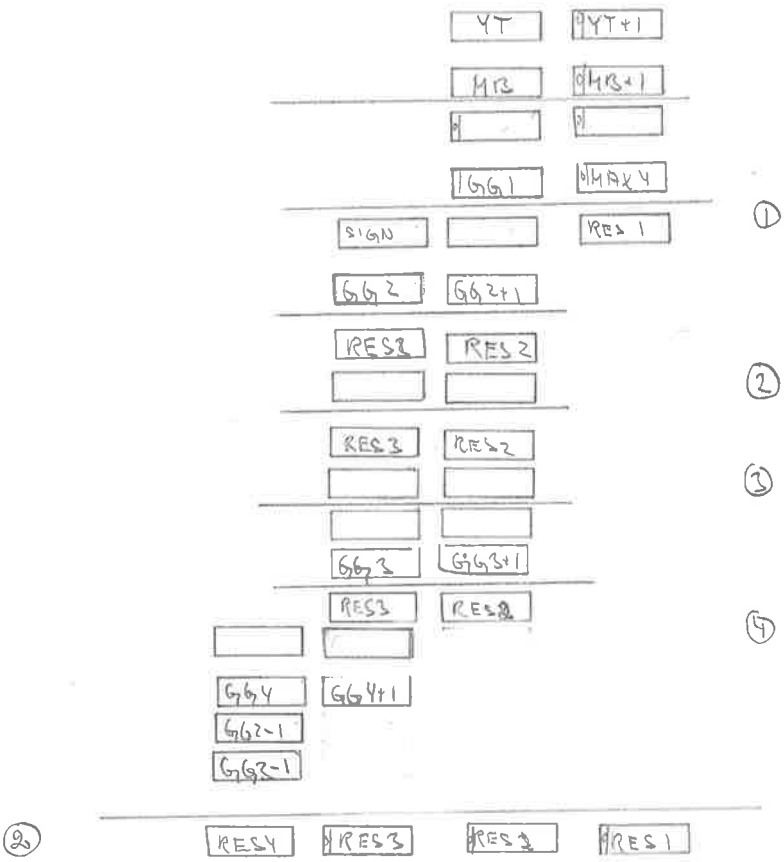


fig 3.

```
      STA RES3
      STB RES2
      LDB MB          /Förklaring enligt figur 3.
      MPY YT
      AMB RES3
      CSB
      AMA GG2-1
      AMA GG3-1
      AMB GG4 + 1
      CSB
      AMA GG4         /Här är multiplikationer och additioner
      SAS              klara och resultatet finns lagrat i den
      BRU NEGA         form som anges i figuren.
      BRU NOLL         Vi ser nu på RES4 och skippar ber. på tecken
STOR  LDA BIG         /GG4 = 0 men ej neg. dvs förstort pos.
EXIT  AOP 1,W         /mata ut till enhet 1.
      HLT             /halt
NOLL  IAB             /GG4 = 0 vi ser på GG3 i B-ack
      SAZ             /skip om A-ack är noll
      BRU STOR        /hoppa STOR
      LDA RES2        ladda A-ack m RES2
      LDB RES1        /ladda B-ack med RES1
      SAZ             /skip om A=0
      BRU STOR        /hoppa till STOR
      BRU EXIT        /hoppa till EXIT
NEGA  CMA             /GG4 negativ,komplementera
      SAZ             /skip om A=0
      BRU LITE        /hoppa
      CMA             /komplementera  för rätt tecken
      FLA 15          /skifta vänster 15 steg
```

```
        CMA             /komplement
        SAZ             /skip om A=0
        BRU LITE        /hoppa
        LDB RES2        /ladda B-ack med RES2
        CMA             /komplement
        FLA 15          /långskifta A v 15 steg
        CMA             /komplement
        SAZ             /skip om A=0
        BRU LITE        /hoppa
        LDB RES1        /ladda B-ack med RES1
        CMA             /komplement
        FLA 15          /långskifta  vänster 15 steg
        BRU EXIT        /hoppa
LITE    LDA SMAL        /ladda A med SMAL
        BRU EXIT        /hoppa
MA      DATA '          /konstanter
MB      DATA '
MREF    DATA '
BIG     DATA '077777
SMAL    DATA '177777
GG1     BSS 1           /variabler
MAX4    BSS 1
GG2     BSS 3
GG3     BSS 3
GG4     BSS 2
RES1    BSS 1
RES2    BSS 1
RES3    BSS 1
RES4    BSS 1
END
```

V.5 SEL 810 A. FLYTANDE RÄKNING.

En stor uppsättning subrutiner för operationer i flytande räkning, enkel precision finns att tillgå. Tyvärr var informationen om dessa i de manualer som fanns att tillgå för exarbetaren synnerligen bristfällig, där den inte helt saknades.

```
STRT   MIP 1,W YT     /läs in till YT från enhet 1, vänta om flagga ej är satt
       CALL FLOAT     /heltal i argumentet förutsättes förvandlas till
                       normaliserat, flytande tal.
       DAC YT         /argument
       CALL F$S       /
       DAC REF        /  } y(t) - REF
       CALL F$H       /
       DAC YT         /  } lagras i y(t)
       CALL F$A       /
       DAC XT         /  } y(t) + x(t)
       CALL F$H       /
       DAC XT         /  } lagra XT
       CALL F$M       /
       DAC MA         /  } (-A) · x(t)
       CALL F$H       /
       DAC MAX        /  } lagras i MAX
       CALL F$L       /
       DAC YT         /  } hämta y(t)
       CALL F$M       /
       DAC MB         /  } (-B) · y(t)
       CALL F$A       /
       DAC MAX        /  } addera MAX + MBY
       CALL F$H       /
       DAC UT         /  } lagras i UT
       CALL IFIX      /  } Reellt tal i argumentet resulterar i heltal i arg.
       DAC UT         /
       MOP 1,W UT     /  Resultatet matas ut från minnescellen till enhet 1
       END
```

```
(konstanter)
MA    DATA '
MB    DATA '
REF   DATA '
(minnesutrymmen, vars innehåll ändras under programmets körning)
XT    BSS 1
YT    BSS 1
MAX   BSS 1
UT    BSS 1
```

V.5 AD- och DA-omvandling för SEL 810A.

En lämplig AD-omvandlare är SEL model 521 med omvandlingshastighet upp till 50 KHz. Den finns en-polig (10 volt input) eller bipolär (±10 volt input) Utgång ing upp till 15 bitar. Omvandlingen sker genom succesiva approximationer. Upplösning upp till 300 microvolt.

STANDARD OUTPUT CODE för max upplösning, 15 bitar

+ Full skala	111111111111111
+ ½ skala	110000000000000
noll	100000000000000
- ½ skala	010000000000000
- full skala	000000000000000

DA-omvandlingstiden är enligt uppgift 10 /us.

REFERENSER

1. SEL 810A General Purpose Computer Reference Manual
2. SEL 810A General Purpose Computer Operating Instructions
3. SEL 810A Interface Design Manual
4. SEL Proposal No. S-570634-60, March 24, 1967 (Kockums)

KAPITEL VI

IBM 1800. Programmering av PI-regulator.

IBM´s motsvarighet till HP 2115 och SEL 810 A är IBM 1800-systemet. Detta har konstruerats för att användas på en bred skala av reelltidstillämpningar, processkontrol och snabb datainsamling. En stor mängd kringutrustning finns och systemen kan byggas upp till stor komplexibilitet. Genom att IBM/1800 kan anslutas direkt till system/360 vinner denna PDM ytterligare i styrka.

IBM/1800 har fyra minnesstorlekar - 4, 8, 16 eller 32 K bitar med 16 bitars ordlängd. Minnescykeltiden är 2 eller 4 us. 26 grundinstruktioner finns men flera av dessa har multipla funktioner.

'High-Speed I/O'operationer kan utföras under 'cycle steal' via datakanal med hastigheter upp till 500 000 ord per sekund. Det finns både indexregister (tre styck) och dessutom möjlighet till indirekt adressering. IBM/1800 har ett mångnivåigt 'interrupt system' och tre'interval timers' med grundperioder .125,.25,.5, 1, 2, 4, 8 eller 16 millisekunder. Tiderna är programmerbara.

Skillnaden mellan IBM 1800 och de två andra undersökta PDM ligger främst på programmeringssidan. Den huvudsakliga skillnaden ligger i att IBM har en orderuppsättning som lämpar sig för dubbelaritmetik, dvs alla aritmetiska uppgifter kan utföras med ett register som utsträcks till 32 bitar. Dessa order är synnerligen användbara.

I gengäld saknar man en del skift-, rotate- och normaliseringsorder och framförallt en order som ökar någon räknare med ex.vis ett och skippar om = 0. Sedan man lärt sig utnyttja indexregistren innebär detta dock ingen större nackdel.

På soft-ware-sidan är IBM/1800 vid en ytlig undersökning minst i klass med de andra undersökta PDM. Framhållas bör dock den möjlighet till 'Fill-in-the-Form-programmering' som finns. Se vidare om detta i kapitel

VI.1 IBM 1800. Enkel Precision. Skalad version.

```
START XIO L  MPXER write  /ge adress till multiplexer
      XIO L  YT    read   /LÄS in till YT
      LD     XT           /ladda XT
      LDX  2 SKALX        /ladda indexreg 2 med SKALX
      BSI L  SPILL        /hoppa till subrutin SPILL
      STX  2 SKALX        /lagra XR2 i SKALX
      LDX  1 SKALX        /ladda XR1 med SKALX
      STO    XT           /lagra XT
      LD     YT           /ladda YT
      S      REF          /subtrahera REF
      BSC L  SKIFT,O      /hoppa till sorderin Skift om overflow
      SRT    1            /skifta höger ett steg
      OR     B141         /V-addera 1 i position 1
      MDX X1 1            /öka XR1 med ett
SKIFT STO    YT           /lagra YT
FLOW  STX  1 SKALY        /lagra XR1 i SKALY
      STX  3 SKALX        /lagra XR3 i SKALX
      LD     SKALX        /ladda SKALX
      CMP    SKALY        /jämför SKALX-SKALY
      BSC L  INY          / >0; hoppa till INY
      BSC L  INX          / <0; hoppa till INX
      LD     XT           / =0; ladda XT
      A      YT           /addera YT
      STO    XT           /lagra imXT
      BSC L  SPILX,O      / hoppa till SPILX om overflow
      MDX X3 1            /öka XR3 med ett
      SRT    1            /skifta höger ett steg
```

```
        OR      B141        / V-addera 1 i position 1
        STO     XT          / lagra XT
        LD      YT          / ladda YT
        SRT     1           / skifta höger ett steg
        STO     YT          / lagra YT
        LD      XT          / ladda XT
SPILX   M       MA          / multiplicera MA x XT
        STD     MAX         / dubbellagra MAX
        LD      YT          / ladda YT
        M       MB          / multiplicera MB x YT
        AD      MAX         / dubbeladdera MAX
        LDX X2  SKALX       / ladda XR2 med SKALX
        BSI L   SPILL       / hoppa till SPILL
        STX  2  SKAL        / lagra XR2 i SKAL
        STO     UT          /lagra UT
        LD      SKAL        / ladda SKAL
        BSI L   SIGN,Z      / hoppa till subrutin SIGN om ≠ 0
        XIO     UT write    / mata ut UT
        WAIT                / vänta ;programmet kan återstartas
        BSC L   START       / hoppa till START
SIGN    NOP                 / här lagras återhoppsadress
        LD      UT          / ladda UT
        BSC L   MINUS,Z+    / hoppa till MINUS om negativ
        LD      BIG         / ladda BIG
        STO     UT            lagra i UT
        BSC L   SIGN,I      / hoppa till SIGN indirekt
MINUS   LD      SMAL        / ladda SMAL
        STO UT                lagra i UT
        BSC L   SIGN,I      / hoppa indirekt tillbaka
INY     LD      YT          / ladda YT
        SRT     1           / skifta höger ett steg
        STO     YT          / lagra i YT
```

```
        MDX X1 1            /öka XR1 med ett
        BSC L  FLOW         /hoppa till FLOW
INX     LD     XT           /ladda XT
        SRT    1            /skifta höger ett steg
        STO    XT           /lagra XT
        MDX X2 1            /öka XR2 med ett
        BSC L  FLOW         /hoppa till FLOW
SPILL   NOP                 /här lagras återhoppsadress
        BSC L  NEGAT,Z+     /hoppa till NEGAT om negativ
        SLT    1            /skifta vänster ett steg
        SLC    2            /skifta vänster och räkna ned XR2
        SRT    1            /skifta höger ett steg
        BSC L  SPILL,I      /hoppa tillbaka
NEGAT   SLT    1            /skifta vänster ett steg
BACK    BSC L  OUT,Z-       /hoppa till OUT om positivt
        SLT    1            /skifta vånster ett steg
        MDX X2 (-1)         /öka XR2 med -1
        BSC L  BACK         /hoppa till BACK
OUT     SRT    1            /skifta höger ett steg
        OR     B1000        /V-addera tecken-bit
        BSC L  SPILL,I      /hoppa tillbaka

MPXER   BSS    1            /konstanter
YT      BSS    1            /
XT      BSS    1            /
UT      BSS    1            /
MAX     BSS    1            /
SKAL    BSS    1            /
SKALX   BSS    1            /
SKALY   BSS    1            /
                            /
```

```
B14L   DEC    16384        /konstanter
B1000  DEC    32768
MA     DC
MB     DC
REF    DC
BIG    DC
SMAL   DC
       END
```

VI.2 IBM 1800. Enkel precision. Modifierad form.

```
START XIO L   MPXER write /GE ADRESS TILL MULTIPLEXERN
      XIO L   YT    read  /läs in till YT
      LD      YT          /ladda YT
      LDX X1  4           /ladda XR1 med 4
      SRT  1              /skifta höger det antal steg som XR1 anger
      S       REF         /minska med REF
      STO     YT          /lagra i YT
      A       XT          /addera XT
      STO     XT          /lagra i XT
      M       MA          /multiplicera med MA
      STD     MAX         /lagra i MAX
      LD      YT          /ladda YT
      M       MB          /multiplicera MB
      AD      MAX         /addera MAX dubbelt
      LDX X2  20          /ladda XR2 med 20
      BSI L   SPILL       /hoppa till subrutin SPILL
      STX  2  SKAL        /lagra XR2 i SKAL
      STO     UT          /lagra UT
      LD      SKAL        /ladda SKAL
      BSI L   SIGN,+-     /hoppa till subrutin SIGN om SKAL ≠ 0
      XIO L   UT write    /mata ut UT
      HLT                 /HALT!
      BSC L   START       /hoppa till START
SIGN  NOP                 /här lagras återhoppsadress
      LD      UT          /ladda UT
      BSC L   MINUS,+Z    /hoppa till MINUS om negativ
      LD      BIG         /ladda BIG
      BSC L   SIGN,I      /hoppa tillbaka
      LD      SMAL        /ladda SMAL
```

```
        BSC L   SIGN,I      /hoppa indirekt till huvudprogrammet
OMPXER  BSS     1           /variabler
YT      BSS     1           /
XT      BSS     1           /
UT      BSS     1           /
MAX     BSS     1           /
SKAL    BSS     1           /
REF     DC                  /konstanter
BIG     DC                  /
SMAL    DC                  /
        END                 /SLUT!
```

I programmet ingår också subrutinen SPILL, angiven i den skalade versionen.

VI.3 IBM 1800. Dubbel Precision.

Subrutinen XMD är en multiplikationsordef i dubbel precision med binärpunkten placerad mellan teckenbiten och MSB. Ett dubbel-ord i FAC (Se IBM 1800,flytande räkning) multipliceras med ett dubbel-ord i A-och Q-ackumulatorn. Produkten plaseras i A- och Q-ackumulatorn.

```
START XIO L  MPXER write/ge adress till multiplexer
      XIO L  YT   Read  /läs in till YT
      LDD    YT         /ladda dubbelt YT
      LDX X1 16         /ladda XR1 med 16
      SRT  1            /skifta höger innehållet i XR1
      SD     REF        /dubbelsubtraherar REF
      STD    YT         /dubbellagrar i YT
      LDX X2 32         /ladda XR2 med 32
      BSI L  SPILL      /hoppa till SPILL
      LDX X3 4094       /ladda XR3 med 4094; detta för att bilda FAC
      STD I3            /dubbellagra i cellen vars första adress
                         står i XR3
      LDD    MA         /dubbelladda MA
      BSI L  SPILL      /hoppa till SPILL
      CALL   XMD        /multiplicera
      STD    MAX        /dubbellagra i MAX
      LDD    YT         /dubbelladda YT
      LDX X2 32         /ladda XR2 med 32
      BSI L  SPILL      /hoppa till SPILL
      STD I3            /dubbellagra i de celler vars första adress
                         står i XR3
      LDD    MB         /dubbelladda MB
      BSI L  SPILL      /hoppa till SPILL
```

```
          CALL XMD     / (-B) y(t) i A och Q

          AD MAX       / addera (-A) x(t)

          CMP TEST     / jämförvmed innehållet i TEST

          BSC L STOR   / hoppa

          BSC L LITEN  / hoppa

          SLT 1        / skifta in resultatet i A-reg.

EXIT      XIO UT       / exekvera DA-omvandling

STOR      LD MAXX      / lagra maximalt tal i A-reg
          STO UT       / lagra i UT
          XIO UT       / mata ut från UT

LITEN     LD MIN       / lagra minimalt tal i A-reg.
          STO UT       / lagra i UT
          XIO UT       / mata ut i UT

          MPXER BSS1   / konstanter o. variabler
YT        BSS 1

          REF DC

XT        BSS 1

          MA DC

          MB DC

MAX       BSS 1

TEST      DC 0000

MAXX      DC 7FFF

MIN       DC 8001
UT        BSS 1
          END
```

I programmet ingår också subrutinen SPILL, angiven inden skalade versionen.

VI.4 IBM 1800. FLYTANDE RÄKNING.

IBM 1800 subrutiner för reella tal kräver ibland ett register eller en ackumulator som kan innehålla tal i reell form. Eftersom alla "hardware" register bara har 16 bitar, måste man skaffa sig en pseudoackumulator. Denna kan vara två eller tre ord lång. Pseudoackumulatorn FAC (floating accumulator) är ett tre-ords register i "över föringsvektorns" tre högsta positioner. (C 26-5882 s.77)

Karakteristika	Mantissa	Mantissa

Fig. Pseudoackumulatorns format.

Program.

```
M.S     ORG 3000
8-10    START   XIO L MPXER /(write)Se fix räkning
8-10            XIO L YT    /(read)      -"-
4'/4            LD YT       /ladda y(t)
165             CALL FLOAT  /Anrop av FLOAT medför att heltal i A-reg förvandlas
                            till reellt tal i FAC.
130             CALL NORM   /Reellt onormaliserat tal i FAC resulterar i att
                            mantissan i FAC skiftas tills MSB finns i bit 1.
                            Karakteristikan ändras till sitt rätta värde.
290             CALL FSUB   /
                DC REF      /   } y(t) - REF
 90             CALL FSTO   /
                DC YT       /   } lagras i y(t)
230             CALL FADD   /
                DC XT       /   } y(t) + x(t)
 90             CALL FSTO   /
                DC XT       /   } lagras i x(t)
```

```
280         CALL FMPY   /
                        }  (-A)  x(t)
            DC MA       /

 90         CALL FSTO   /
                        }  lagras i MAX
            DC MAX      /

 90         CALL FLD    /
                        }  hämta y(t)
            DC YT       /

280         CALL FMPY   /
                        }  (-B)  y(t)
            DC MB       /

230         CALL FADD   /
                        }  Addera MAX & MBY
            DC MAX      /

 70         CALL IFIX   /Reellt tal i FAC resulterar i heltal i A-registret.
            STO UT      /lagra i UT
 9-10       XIO UT      /läs ut till yttre enhet.


            MPXER BSS 1 / konstanter o. variabler

            YT    BSS 1

            REF   DC

            XT    BSS 1

            MA    DC

            MAX   BSS 1

            MB    DC

            UT    BSS 1
```

~2050 minnesl

```
            END
```

VI.5 AD/DA-omvandlingstider för IBM/1800

De två modeller som finns för analog/digital-omvandling för 1800-systemet omvandlar till bipolära analoga signaler inom intervallet $\pm$ 5 volt.

Modell 1 inkluderar en buffertförstärkare och kan programmeras för upplösning 8, 11 eller 14 bitar. Nominella omvandlingshastigheten är 10 kHz. Ingångsimpedans 10 M.

Modell 2 innehåller dessutom en sample- och hållkrets vilken avsevärt minskar aperturtiden och därmed ökar systemets hastighet. nominell omvandlingshastighet 20 kHz. Ingångsimpedans 0.1 M

Omvandlingstiden påverkas av antalet bitar:

antal bitar	8+S	11+S	14+S
omvandlingstid(μs)	29	36	44

Hela 1800-systemets omvandlingshastighet varierar mellan 9000 upp till 25 000 samplingar per sekund.

Negativa tal representeras i 2-komplementform.

VI.6 REFERENSER

1. IBM 1800 DATA Acquisition and Control System Reference Manual. Form A26-5918-1

2. IBM 1800 Assembler Language Form C26-5882-1

KAPITEL VII. Sammanställning över antal minnesceller som utnyttjas och av teoretiska körtider.

VII.1 Enkel Precision. Skalad version.

Maskintyp	HP 2115	SEL 810A	IBM 1800
Antal instruktioner	134[1]	100	97
Cykeltid(µs)	2.0	1.75	2.0(4.0)
Effektiv programtid i memory-cycles (m.c)	172	187	172
kortaste genomlopp	143 m.c	112	113
drygaste "	435 m.c	494	320

VII.2 Enkel Precision. Modifierad version.

Maskintyp	HP 2115	SEL 810A	IBM 1800
Antal instruktioner	72	61	53
Effektiv programtid m.c.	127	126	90
kortaste genomlopp	138 m.c.	88	74
längsta genomlopp	331	316	190

Fotnot: 1. P.g.a. att programmet körts i maskin utan Extended Arithmetic har programmet blivit något längre.

VII.3 Dubbel Precision.

Maskintyp	HP 2115	SEL 810A	IBM 1800 [2]
Antal instruktioner	158	141	129
Effektiv programtid m.c.	298	265	239
Kortaste genomlopp	231	211	345
Längsta genomlopp	410	250	495

Kommentarer om Dubbel Precision: Se nedan

VII.4 Flytande Räkning

Maskintyp	HP 2115	SEL 810A	IBM 1800
Antal instruktioner	555	- [1]	504
Körtid	3050 us	-	2050 us

Dubbel Precision: Tiderna för IBM 1800 synes väl långa då denna maskin är i högre grad än de andra anpassad för dubbelordsaritmetik (dubbel-lagring, dubbel-addition osv.) Orsaken till dessa långa tider är subrutinen XMD, som är 130 m.c. lång. Den anropas två ggr i programmet. Den har också 66 instruktionsord och ökar därmed tot. antalet instruktioner med en faktor 2. I de övriga D.P. programmen har inga subrutiner konstruerats för dubbelt mult. utan multiplikationerna har programmerats var för sig. Det betyder att antalet instruktioner ökar medan körtiden blir kortare.

Fotnot 1. De subrutiner som använts finns enbart angivna till namnet och (i någon mån) gagnet i ref V.4 Tider och antal minnesceller finns inte medtaget,

2. Orsaken till den långa körtiden synes vara att instruktionen XMD kräver 520 µs dvs 260 m.c. eftersom tiderna räknats med 2 µs kärnminne.

VII.5

Omvandlingstider för AD-omvandlare. Översikt.

Antal bitar	9	12	14	15	aperturt.
HP 2115basic.		51kHz[1]	19kHz[1']	28[1]	50 µs
" wSoH			"		50 ns
SEL 810A	upp till 15 bitar; hastighet upp till 50kHz				
IBM 1800[2]	34kHz	28kHz		23kHz -	

Fotnot: 1. Tider med 1 och 1´som indicering härrör från olika modeller av AD-omvandlare.
2. IBM/1800 har också en modell med SoH-krets som avsevärt sänker aperturtiden. Se sid 26 ff.

KAPITEL VIII. Sammanfattning.

Första avdelningen av detta examensarbete är ett försök att ge en bild av de olika modeller av analog/digital och digital/analog omvandling som tillämpas idag. De fel som kan göra sig gällande och begränsa det använda systemets tillförlitlighet analyseras.

Tre principiella sätt att generera styrsignaler för ställorgan undersöks. Det visar sig att analog styrning med DA-omvandling av styrspänningen är snabbast ur programsynpunkt. För små förflyttningar har stegmotorn uppenbara fördelar när det gäller upplösning och snabbhet, medan digital styrning av en synkronmotor är användbar vid större styrvinklar.

Lyhörda programmerare har uppmärksammat systemingenjörerenas högljudda rop på ett enkelt process-styrspråk som inte kräver några djupare insikter i programmeringens ädla konst. Ett av svaren blev PROSPRO/1800 som kort presenteras.

Huvudvikten i arbetet ligger på programmering av PI-regulator för tre olika processdatamaskiner, HP 2115, SEL 810A och IBM 1800. PI-regulatorn har programmerats upp för enkel precision, dubbel precision och flytande räkning. I enkel precision har programmet skrivits i två versioner, där jag i den ena versionen redan från början skalat insignalen med en faktor fyra. Genom denna skalning redan från början, istället för när det krävs p. g. a. spill, minskar minnesutrymmet generellt för alla maskinerna till nästan hälften. Effektiva körtiderna sjunker i samma grad.

I samtliga skrivna program kräver IBM 1800 minsta utrymmet. HP 2115 kräver i genomsnitt 30 % mer, medan SEL 810A ligger mitt emellan. Orsaken synes vara IBM:s annorlunda orderuppsättning som enklare tillåter operationer i dubbel precision.

Effektiva programtiden är också kortare för IBM 1800, även om försprånget till SEL 810 A här har krympt. Det beror framför allt på att SEL 810 A har kortare cykeltid (1.75 µs mot 2.0 µs).

SUMMARY

The first part of this work is an attempt to assemble comprehensive information of the different kinds of analog/digital and digital/analog converters which is in use today. Since the end result of conversion is the representation of a given value in different terms, it is important to know how accurate the representation is. An analys of the various types of errors and their causes is made.

Three basic techniques to generate control signals in processing systems are examined. The result points out that analog control with D/A conversion of the steering voltage is the fastest in programming view. When only small changes in parameters is asked for, the stepping motor technique has certain advatages, specially due to better resolution. Digital steering of a synchronous motor is useable when greater changes in parameters is demanded.

Programmers with a sensitive ear has payed attention to the processing engineers cry for a simple language for process control, which do not require any deeper knowledge in the noble art of programming. One of the answers was PROSPRO/1800, which is shortly presented.

The main part of the work is the programming of a PI-regulator for three process-controllers,HP 2115, SEL 810 A and IBM 1800.The program is written for fixed point data (single precision and double precision) and floating point data. In single precision the program is written in two versions, where the starting values in the second version has been given an exponent 4 from the beginning. This reduce the memory space almost to the half. The effective running time is reduced to the same degree.

In all the assembly-written programs IBM 1800 demands the smallest memory space. HP 2115 demands in average 30 % more, while SEL 810 A is between. The reason seems to be IBM:s different instruction list, which allows operation in double precision. The running time is shorter for IBM 1800, even if the advantage to SEL 810 A has grown less. That applies especially on the fact that SEL 810 A has shorter cycle time (1.75 us vs. 2.0 us)