# STUDIUM AV EN STYRALGORITM SAMT PROGRAMSYSTEM FÖR PROCESSDATAMASKINER

ANDERS YDREMARK

Rapport RE - 28 maj 1968

# STUDIUM AV EN STYRALGORITM SAMT PROGRAMSYSTEM FÖR PROCESSDATAMASKINER.

Examensarbete i regleringsteknik

Anders Ydremark

INNEHÅLLSFÖRTECKNING.

# I. INLEDNING.

Kapitel II innehåller en ganska allmän orientering om programsystem för processdatamaskiner. Saknaden av entydigt språkbruk fabrikanterna emellan har gjort studierna av deras manualer relativt svårbemästrat. Ett typiskt monitorpaket för IBM-1800 beskrivs därför ingående och får gälla som exempel och jämförelsegrund för eventuellt studium av andra lösningar av programproblemet.
I kapitel II.4 redogörs för ett förslag till programspråk, RTL. Detta högre nivå språk skulle göra det ekonomiskt lönsamt att konstruera speciella styrsystem till den rika flora av processer, som förekommer.
Resultaten av ett programsystem för en PDP-8 maskin där användarens program kan skrivas i ett ALGOLliknande språk, utvecklat vid tekniska högskolan i Trondheim, finns redovisade i kapitel II.5.

Kapitel III innehåller programmet för en styralgoritm, för generering av en PI-regulator. Programmeringsarbetet är utfört i assemblerkod för tre olika maskiner med olika ordlängd: PDP-8 , 12 bitar , IBM-1800, 16 bitar och GE-PAC 4020 24 bitar. Styrlagen är skriven dels för fix räkning enkel och dubbel precision dels för flytande räkning. Kapitlen inleds med en kort beskrivning av maskinerna.
Avsikten med programmeringen har till viss del varit att få en uppfattning om de olika maskinernas prestanda genom att jämföra minnesutrymmen och exekveringstider för programmen samt att få en allmän jämförelse av programmeringsarbetet.
Att hitta informationen i fabrikanternas manualer och handböcker har varit den största svårigheten. Inte alltför utförliga beskrivningar av assemblerkodens användning och problemet att rättvist jämföra olika funktioner hos maskinerna, har varit andra stötestenar. Det rekommenderas att ha instruktionslistan till hands för en njutbar genomläsning av programmen.

Bifogad ordlista skall vara till hjälp för den oinvigde läsaren, och för vidare studier hänvisas till respektive referenser, angivna inom klammer med en nummerbeteckning. Referenslistan liksom ordlistan finns i slutet av uppsatsen

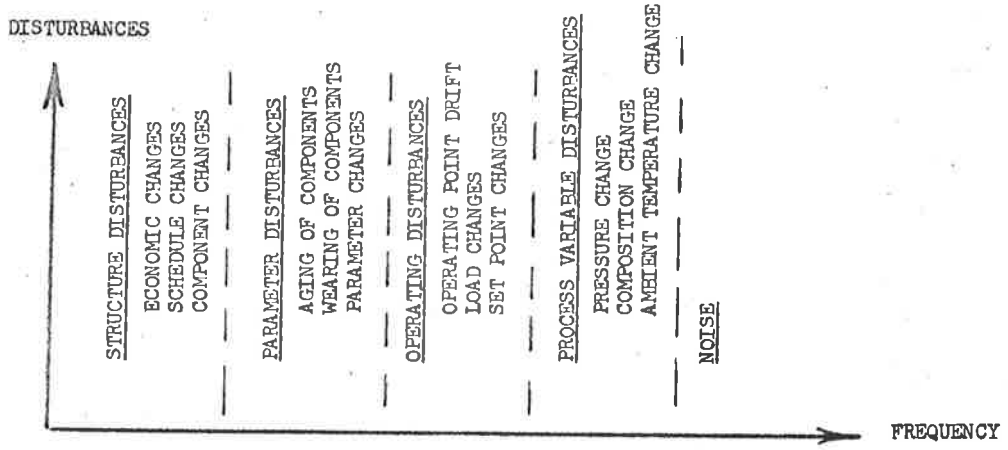
## II. STUDIUM AV PROGRAMSYSTEM FÖR PROCESSDATAMASKINER.

### II .1 ALLMÄN BETRAKTELSE AV PROGRAMMERINGSSYSTEMETS UPPBYGGNAD.

Tanken att processkontrollmaskiner skulle kräva små ansträngningar på programsidan ersattes snabbt då man insåg att man inte enbart hade att ta hänsyn till de konventionella programkraven. Reelltidsaspekten på processkontrollproblemen måste ochså ingå i programvaran. Detta har lett till ett fullständigt tiddelningssystem med ett subrutinbibliotek för kommunikation mellan maskin och process. Under den tid processen inte sysselsätter maskinen tillåter styrprogrammet (eng. executive) behandling av off-line uppgifter under kontroll av en off-line monitor för att skilja behandlingen från styrprogrammet själv. /2/

Programvarans sammansättning förstås bäst genom att definiera processkontrollproblemet och sedan bestämma vilka programmeringsproblem det genererar. Det finns emellertid en rik flora av olika industrier, vilka kräver olika processkontroll. Vissa drag är dock gemensamma för alla industrier och processer, exempelvis syftet att kontrollsystemet skall styra processen tillfredssyällande i den omgivning av störningar som förekommer. Med störningar menas här variationer i processvariabler, operationspunkter, processkarakteristiken och ändringar i ekonomiska betingelser för processens resultat. Processproblemet inkluderar således inte bara DDC utan också övervakande kontroll, driftsinformation och driftskontroll. Dessa kontrollsystem påverkar programproblemet och måste därför tas i beaktande. De varierande processtörningarna kategoriseras i förhållande till sin relativa frekvens. Genom att karakterisera den typ av kontroll, som används för varje frekvensordning, samt dess krav på kommunikation med operatör och andra maskiner, är det möjligt att ganska explicit definiera programproblemet.

Det har visat sig att detta leder till en hierarki bestående av nivåer klart skilda åt (se fig.1) där varje kontrollnivå




Decomposition of Disturbances According to Frequency

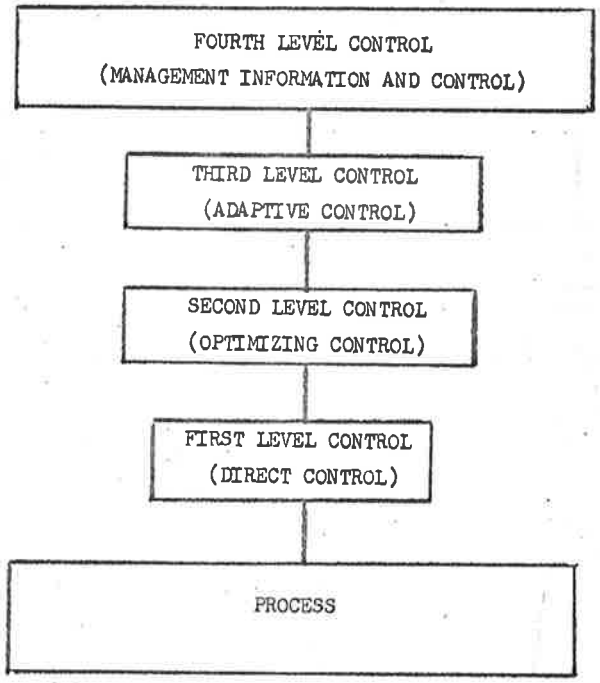



FIGURE 1.

Multilevel Control Hierarchy

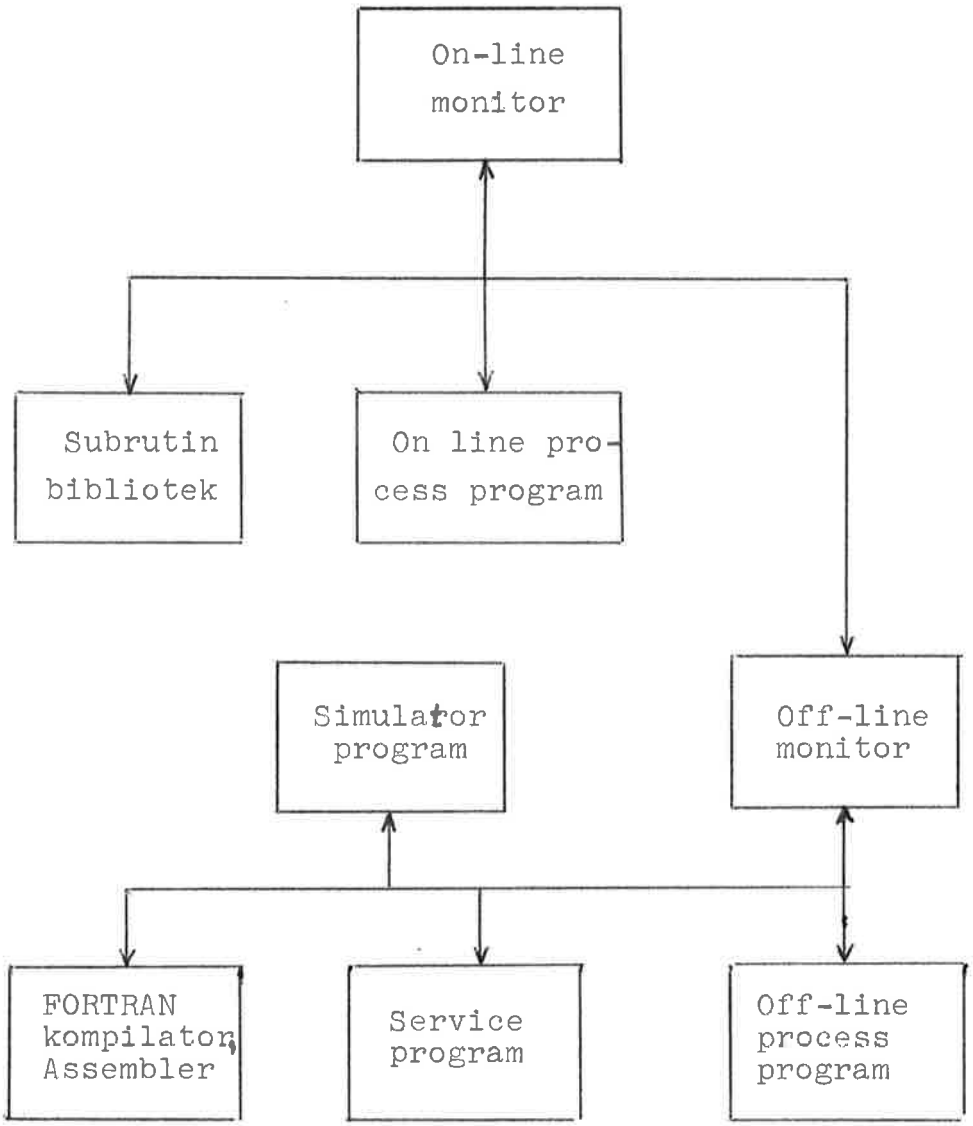
svarar mot en frekvensordning. Beräkningsproblemet för varje nivå är relativt oberoende och kan göras allmänt.
Att märka är att varje nivås funktion är att använda data, som lämnats av alla de lägre nivåerna tillsammans med speciell information från den omedelbart över belägna nivån, för att skapa beslut, som påverkar nivån omedelbart under. Ju högre kontrollnivå desto mera sällan inträffar störningar, som är förbundna med den och desto svårare blir själva kontrollproblemet.
Programkraven, som är genererade av kontrollproblemet skisserat ovan, beskrivs bäst genom att först betrakta de individuella kontrollnivåerna och sedan problemet att kombinera programmen för varje nivå.
De tre funftionerna hos den första nivån är övervakning, DDC och framläggandet av data. Den senare funktionen inbegriper omvandling av processdata, kontrollparametrar och liknande till lämpliga enheter samt utskrivning av dessa på utorgan för att informera operatören. Den övervakande funktionen består i att göra periodiska avläsningar av processvariabler, kontrollera riktigheten av dessa avläsningar, alarmera och eventuellt taga del i situationer där variablerna överskridit sina gränser.
Ovan beskrivna funktioner använder maskinens perifera utrustning. Eftersom analog och digital ingång och utgång ofta medför stora skillnader i operationshastigheten, blir programproblemet ganska svårt och viktigt. Till exempel blir problemen helt olika om man använder en reläscanner med låg hastighet än om man använder en elektronisk scanner med mycket hög hastighet. I det första fallet rör sig tiden, det tar för multiplexern att gå från en punkt till en ny, om ungefär 5-20 ms medan i det senare fallet motsvarande tid är ungefär 50 $\mu$s. I första fallet kan man inte tillåta maskinen vara sysslolös under tiden multiplexern arbetar och man förstår att multiprogrammering blir nödvändig för att på ett effektivt sätt utnyttja maskinens kapacitet.
Ofta förekommande beräkningar utföres vanligen i fix aritmetik såvida inte flytande räkning kan utföras i maskinvara.

I allmänhet måste programmeringsarbetet på den första nivån göras mycket effektivt om man ska nå god utnyttjandegrad av maskinvaruutrustningen. Kontrollen på denna upptar i allmänhet största delen av maskinräknetiden.
Varje signifikant ineffektivitet exempelvis en subrutin i flytande räkning skulle kräva en ökning av maskinvarumöjligheterna Det stora antalet variabler kräver ofta hög avbrottsfrekvens och snabb svarstid. Problemen på den första nivån varierar dessutom starkt mellan tillämpningarna varför det är svårt att åstadkomma ett program av allmän natur.
Funktionen hos den andra nivån är att styra processen optimalt under rådande omständigheter. Optimering innefattar maximering, minimering av någon analytisk funktion eller kanske ett slumpmässigt eller styrt sökande i en systemmodell efter ett bättre operationstillstånd.
Input till den andra nivån består både av data insamlade av den första nivån och de matematiska modeller och mål, som satts upp av tredje och fjärde kontrollnivåerna. Beräkningar sker ofta i flytande räkning p g a variablernas obestämda natur. Output från nivån är information till de lägre nivåerna.
Programmen för detta kontrollsystem är ofta av subrutin karaktär. Resultatet blir att programmeringen på denna kontrollnivå är mera kompakt och uppbyggd i segment än fallet är på den första nivån. Den är mindre maskinbunden i mening av intim kommunikation med de varierande maskinvarorna ovh överhuvudtaget väldigt skild från programmeringen på den första nivån.
Maskinarbetet på den tredje kontrollnivån medför ofta mycket lagring av data exempelvis på yttre minnen. Denna nivå har liten kommunikation med yttervärlden. Den hämtar sin information från första och fjärde nivåerna och levererar resultat huvudsakligen till de andra kontrollnivåerna.
Den fjärde nivån handhar bokföringsuppgifter, rapportgenerering och andra problem av administrativ karaktär. Dessutom är denna nivå ansvarig för sambandet mellan olika opererande, processer och maskiner.

Programmeringsarbetets viktigaste egenskap är alltså att programmen för dessa fyra nivåer växelverkar men kan skrivas oberoende av varandra.
Det är följaktligen nödvändigt att förse sig med ett styrsystem, som matar fram dessa program och kontrollerar deras växelverkan. Exempelvis ett multiprogrammerat styrsystem, som delar upp maskintiden mellan nivåerna enligt en prioritetstabell. /1/,/2/,/3/.

## II.2 MONITORSYSTEM.

Ändamålet med ett monitorsystem är att avlasta programmeraren från de mest arbetsamma uppgifterna i samband med reelltidsoperationer. Det besvarar avbrott, lagrar och tar upp delvis utförda program, anslår primära och sekundära minnesutrymmen och svarar på felsignaler. Dessutom tillåter monitorsystemet off-line körningar om tid finns tillgänglig efter slutfört processkontrollprogram. Icke-processbundna program assembleras, kompileras, simuleras, testas och rättas utan att processen störs.Strukturen av ett typiskt monitorsystem visas nedan:



Nyckelprogrammet är en residentmonitor, som svarar på avbrottssignaler, bestämmer lämplig åtgärd med hänsyn till avbrottets art och prioritet, delar upp program och tar i allmänhet hand om samordnande uppgifter mellan programmen i ett reell-tids system. Vid ledig maskin anropar monitorn off-line monitorn. All assemblering, kompilering, testning och felkorrigering göres under kontroll av off-line monitorn. Simulatorn tillåter testning av program utan att den aktuella processen berörs. Under kontroll av off-line monitorn står också utility routines (service program) som tillåter ändringar i program för on- och off-line processkontroll och som kontrollerar program lagrade i sekundärminnet.
Huvuduppgiften för styrprogrammet är genomlöpandet av processprogram skrivna av användaren; program, som är förbundna med de många in/ut organen. Medan styrprogrammet inte tillåter direkt programmering av in/ut organen, möjliggöres kontroll av de senare genom anrop av ett subrutinbibliotek. Kombinationen FORTRAN och dessa reell-tidssubrutiner resulterar i någonting närbesläktat med ett processkontrollspråk, som tillåter programmering av de förut omtalade högre nivåernas kontrollproblem. Dragen hos dessa språk varierar bland olika maskiner, men grundfunktionen är densamma. /2/,/4/.

## II.3 IBM 1800 TIME-SHARING EXECUTIVE SYSTEM.

IBM 1800 Time-sharing Executive System (TSX) är i allmänna ordalag en grupp program, som genererar, organiserar, testar och utför program skrivna av användaren för processkontroll, datainsamling och icke-processbundna operationer. TSX systemet är ett illustrativt exempel på ett styr- och övervakningssystem med reelltids faciliteter, och ska därför beskrivas ingående.

Nödvändig maskinutrustning för användande av TSX systemet:

1. Centralenhet med minst 8K kärnminne.
2. Skivminne (disk).
3. Printer.
4. Kortstans eller remsstans och remsläsare.

Ytterligare maskinenheter ger naturligtvis större möjligheter att utnyttja TSX systemet.

Användarens program kan skrivas i FORTRAN eller symboliskt språk och dessa kompileras eller assembleras, testas och lagras på disk medan TSX systemet övervakar processen. Icke-processbundna program såsom tekniskt vetenskapliga beräkningar eller normala dataprocessarbeten kan utföras simultant med styrning av processen.

TSX systemetskomponenter är System Director, Non-process Monitor och Subrutin Library (se fig. 2).

System Director (executive program) utgör hjärtat av TSX systemet. Det finns alltid lagrat i kärnminnet och hela kärnminnets permanenta area är minnesskyddat. Kontrollen överförs till System Director genom följande händelser:

1. TSX CALL statement i användarens program.
2. Avbrott.
3. Felupptäckt.

System Director består i sitt grundutförande av fem kontrollprogram och två dataareor. Kontrollprogrammen är :

Program Sequence Control (PSC).
Master Interrupt Control (MIC).
Interval Timer Control (ITC).
Time-Sharing Control (TSC).
Error Alert Control(EAC).

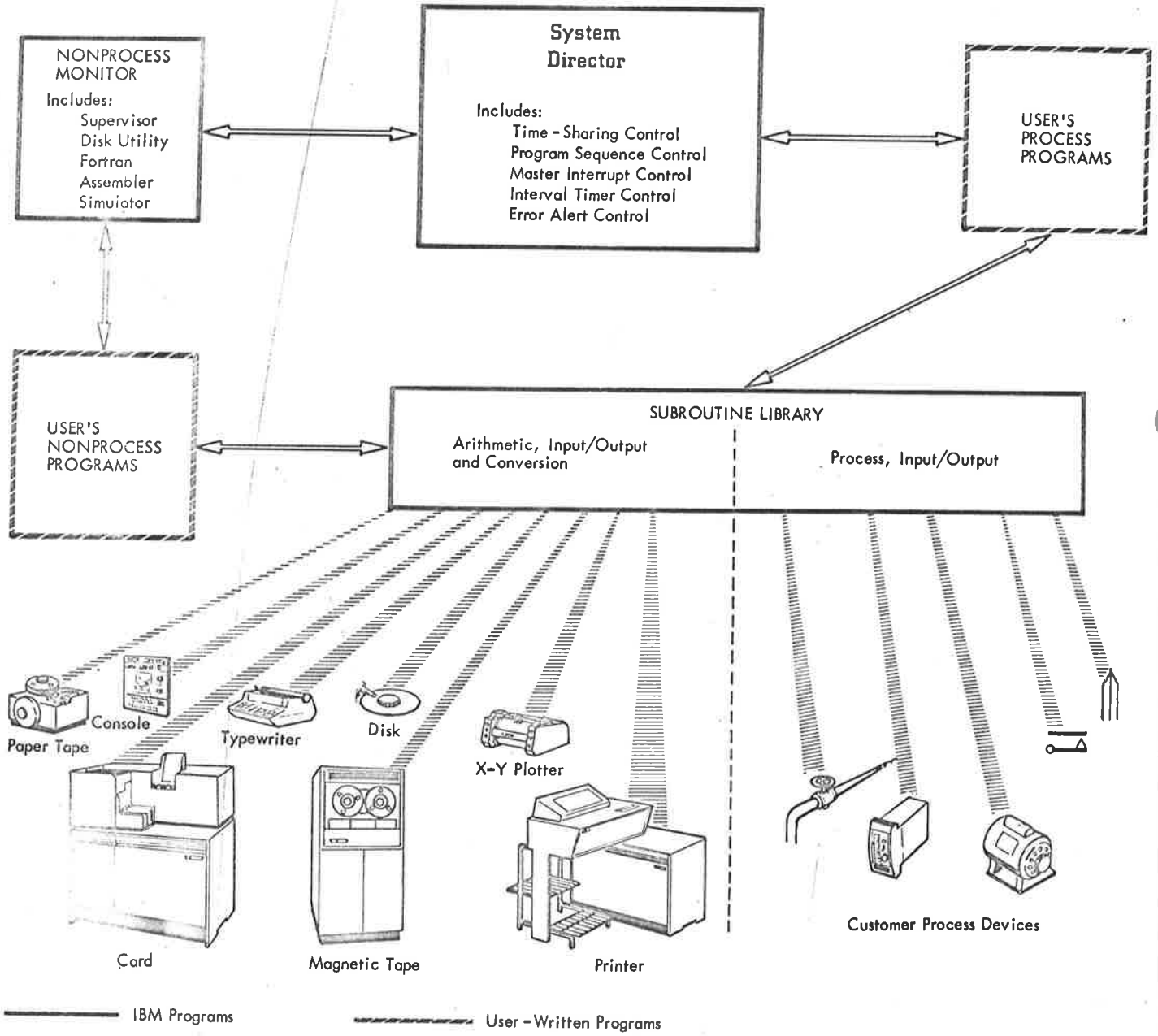



Figure 2. IBM 1800 Time-Sharing Executive System

Program Sequence Control (PSC) är ett kontrollprogram, som handhar kontrollflödet från en core load till nästa. Kärnminnet är i allmänhet inte tillräckligt stort för att innehålla alla processkontrollprogram på en gång. Dessa program måste därför sammansättas i mindre enheter kallade core loads. De finns lagrade på disk och innehåller: main program (interrupt, mainline eller non-process), alla de subrutiner, som används och inte behöver vara permanent lagrade i kärnminnet samt kommunikationstabeller för core loads.

PSC:s specifika funktioner är:

1. Utför nästa följande core load. Det nya core load packetet överlappar det som erhöll anropet.
2. Lagra på disk det core doad, som utföres, och ladda med ett speciellt core load för vidare utförande.
3. Ladda åter det core load, som lagrades under 2 och fortsätt utförandet där det blev avbrutet.
4. Ordna kö för core loads associerade med avbrott, vars inträffande har registrerats.
5. Utför det core load med högsta prioritet, vilket har listats i core load kön.
6. Placera startadressen för ett core load packet i core load kön. Alternativt plocka bort det.

Anrop för att utföra ovanstående funktioner sker till vissa subrutiner. Exempelvis:

CHAIN: Specifikation av nästa program, som ska utföras.
QUEUE: Placera ett program i en väntande kö.
UNQ: Tag bort startadressen till ett från kön.
VIAQ: Anropa programmet med högsta prioritet som väntar i kön.

Genom användandet av dessa instruktioner kan programmeraren kontrollera frekvensen och ordningen i vilken de förut omtalade nivåerna utföres. Av vikt är den lätthet med vilken ordningsföljden ändras när processkotrollproblemet ändras med tiden.

Master Interrupt Control (MIC) program kontrollerar betjäningen av avbrott. Ett avbrott kan inträffa när som helst men det registreras inte av

MIC om avbrottet tillhör en nivå, som maskas och om inte avbrottet är av högre prioritet än den närvarande maskinoperationens nivå. Användaren bestämmer själv prioriteten för ett särskilt avbrott, registreringen av dessa kan fördröjas genom att maska den nivå på vilken de befinner sig. Betjäningen av process- och programmerade subrutiner kan också fördröjas.
Avbrott kan indelas i två typer: interna och externa.
Interna avbrott associeras med I/O organ, interval timer (klocka) eller felupptäckt, och betjänas omgående av subrutiner.
Externa avbrott associeras med process- och programavbrotts egenskaperna. De betjänas eller registreras av fyra typer av rutiner skrivna av användaren (se fig.3, sid.19):

Skeleton interrupt routine.
Mainline interrupt routine.
Interrupt core load.
Mainline core load.

De olika typerna av rutiner möjliggör flexibilitet i användandet av kärnminnet och i kravet på svarstid för ett specifikt avbrott.
Skeleton interrupt routines ligger permanent i kärnminnets skeleton area och används normalt för att betjäna avbrott som kräver omedelbart svar, har hög prioritet eller inträffar ofta. Dessa rutiner utnyttjas endast om användaren anser det nödvändigt att alltid ha dem i kärnminnet. Externa avbrott, som inte betjänas av skeleton interrupt routines kan betjänas av rutiner, som är del av ett mainline core load. Svarstiden för mainline interrupt routine är samma som för ovanstående skeleton interrupt routine om det core load, som innehåller avbrottsrutinen ligger i kärnminnet när avbrottet inträffar. Ett mainline core load krävs vid externa avbrott, som måste tegistreras och betjänas senare. Interrupt core loads används då användaren specificerar avbrottsservice rutinen att vara på disk alternativt vara i kärnminnet som del av ett mainline core load.

Programmerade avbrott behandlas på samma sätt som process-avbrott. Det kan bara finnas en programmerad avbrottsrutin för varje avbrottsnivå.
MIC anrop sker genom följande operationer:

CALL LEVEL: Programmerat avbrott.

CALL INTEX: Alla externa avbrott betjänade på avbrottsnivå måste återlämna kontrollen till MIC.

CALL DPART: Om nuvarande nivå är en avbrottsnivå utföres CALL INTEX annars CALL VIAQ (se ovan).

Interval timer control (ITC) program möjliggör kontroll med hjälp av fyra olika typer av timers:

Två maskinbundna intervall timers.

Nio programmerade intervall timers.

En programmerad reell-tids klocka.

En timer för tid-delnings kontroll.

De två maskinbundna intervall timers användes för att indikera relativt korta tidsintervall och anropas genom:

CALL TIMER (NAME,I,INT),

där NAME är användarens subprogram utfört när tiden är ute. De programmerade intervall timers användes för att specificera långa tidsperioder. De kan speciellt användas för periodiskt utförande av program eller för att initiera utförande vid en senare tidpunkt.

Time-sharing control (TSC) program kontrollerar den tid, som avsätts för icke-processbundna operationer. Tid-delning kan initieras dels genom anrop av subrutin SHARE, i ett process mainline program dels genom anrop av VIAQ då core load kötabellen är tom. Hopp sker då från VIAQ till SHARE. SHARE indikerar ledig tid för off-line program. Då tid-delning önskas enligt första metoden lagras core load i sekundärminnet och off-line monitorn (mera senare) läses in i kärnminnet och utföres. Denna metod användes då tid-delning önskas vid vissa tider medan den andra metoden brukas då maskinen inte är sysselsatt med processen.

Alla avbrott, som inträffar uder en tid-delningsoperation tas om hand av MIC. Kontrollen återföres efter avbrottsbetjäningen till det icke-processbundna programmet. Om det icke-processbundna programmet inte fullbordas innan tiden är slut, lagras det och slutföres under nästa tid-delningsoperation.

Error alert control (EAC) program övertar kontrollen från:

1. Någon I/O subrutin när subrutinen inte kan rätta ett fel eller avbrottstillstånd.
2. Kösubrutinen när core load kötabellen blir överfull.
3. MIC programmet när ett internt maskinfel uppträder (paritet, minnesskydd,etc.)
4. Andra kontroll program.

En felsubrutin skriven av användaren kan valfritt ingå i varje process core load. Användaren övertar då kontrollen innan EAC i händelse av fel. Vid anrop av EAC programmet överförs det från disk till kärnminnets variabla area och användaren kan tänkas vilja bevara viss information innan detta sker.

Data areor, som användes av System director:

Mainline Core Load Queue Table innehåller namnen på de mainline core loads, som har blivit satta i kö för utförande samt dessas prioritet.

Level Work Areas innehåller avbrottsnivåinstruktioner, MIC sammanlänkningar samt arbetsareor.

En nivåarbetsarea krävs för:

1. Varje avbrottsnivå, som användes.
2. Process mainline program.
3. Icke-processbundna core loads.
4. Intern felnivå.

Förutom utförandet av användarens program måste TSX systemet matas in på disk och system skeleton byggas upp och laddas. Följande program finns tillgängliga:

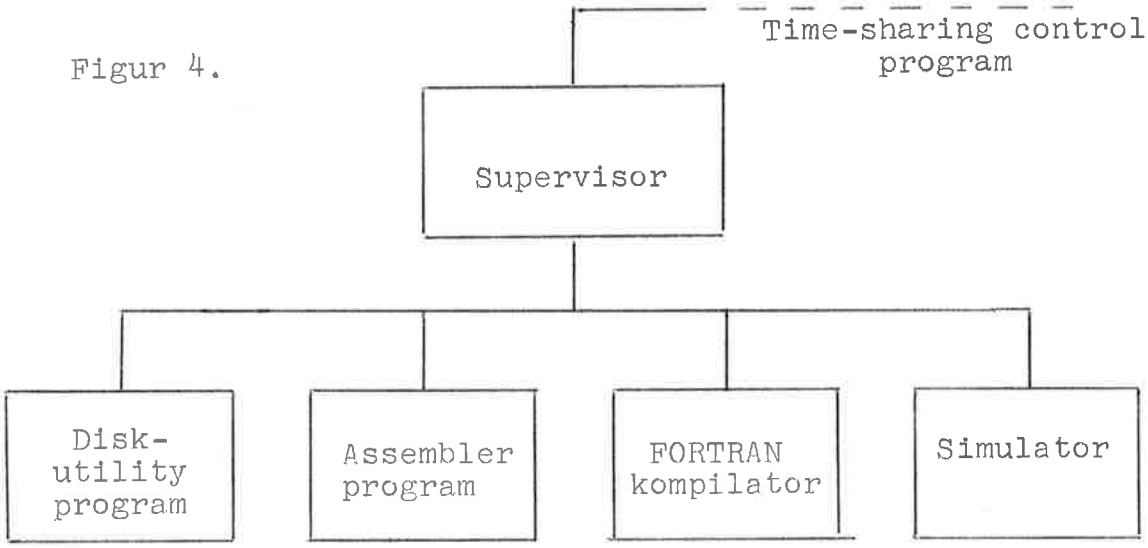
TASK
System loader
Skeleton builder
Core load builder

Temporary Assembled Skeleton (TASK) kontrollerar till en början de övriga tre innan det överlappas av system skeletons fasta plats i kärnminnet.
System loader har som primär uppgift att läsa in TSX systemets program, bygga upp en avbrottsanvisningstabell och förbereda disken för system operation.
Skeleton builder program erhåller information från användarbestämda kontroll poster, program och subrutiner och information anskaffad av system loader för att bilda system skeleton (i core-image form) och sedan lagra det på disk.
Core load builder är ett program, som omvandlar program och subrutiner skrivna av användaren till core loads för lagring på disk.

Non-process monitor är den andra av TSX systemets huvud komponenter. Den kan operera on-line under kontroll av System director eller off-line. Monitorns primära uppgift är attt ombesörja kontinuerliga processor-kontroll operationer under en serie av arbeten (jobs), som annars skulle kräva flera oberoende programsystem. Den koordinerar processor-kontroll aktiviteten genom upprättandet av en common communication area i kärnminnet, som användes av monitorns varierande program.
Non-process monitorn består av fem program (se fig.4):

1. Supervisor program
2. Disk utility program (DUP)
3. Fortran compiler
4. Assembler program
5. Simularor program

Figur 4.




Programmen beskrivs nedan i korthet.
Supervisor program övervakar alla icke-processbundna operationer. Det avkodar monitor kontrollposterna i den stackade ingången för icke-process arbeten och anropar monitor program för att utföra den önskade operationen.

Disk utility program (DUP) är en grupp rutiner, som är avsedda att hjälpa användaren vid utnyttjande av disken. De är kapabla att lagra, utplåna och skicka ut användarens program och definiera system- och maskinparametrar.

FORTRAN compiler översätter program skrivna i FORTRAN språk till maskinspråk och ombesörjer automatiskt anrop av nödvändiga aritmetiska, funktionella, omvandlings och I/O subrutiner.

Assembler program översätter program skrivna i symbolspråk till maskinspråk. Det råder en 1-1 korrespondans mellan språken dvs för varje symbolisk instruktion svarar en maskininstruktion. Man kan i sumbolspråk lätt använda I/O, omvandlings och aritmetiska subrutiner, som är en del av subrutinbiblioteket.

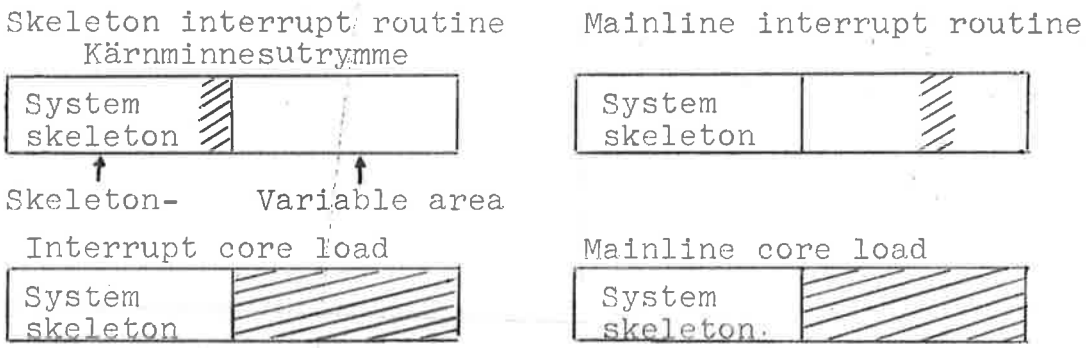
Simulator program är ett hjälpmedel vid simulering av processkontrollprogram, som arbetar utan att inverka på den löpande processen.

De subrutiner, som ingår i TSX systemet kan indelas i följande grupper:

1. I/O och omvandling.
2. Aritmetiska och speciallicerade.
3. Selective dump och miscellaneous.

I/O subrutiner gör det möjligt att snabbt och lätt referera till de varierande I/O organen för in- eller utdata.
Omvandlingssubrutiner överför data från en kod till en annan.
Aritmetiska och specialicerade surutiner är de mest använda vid matematiska beräkningar.
Selective dump subroutines tillåter utskrift av valda areor av kärnminnet under ett objektprograms utförande.
Miscellaneous subrutines tillåter användaren att skriva bitar av sitt FORTRAN program i maskinkod.
Totalt finns ungefär 230 subrutiner (1965) i TSX systemets bibliotek.

Figur 5 och 6 visar hur disk respektive kärnminness utrymmet disponeras av programmen.
/5/.




Figur 3.

| | Number of Sectors** | Sectors Reserved For | |
|---|---|---|---|
| 0000 | 1 | Disk Communication Area | File Protected Area |
| | 21 | Nonprocess Supervisor | |
| | 64 | Disk Utility Program | |
| | 40 | Assembler | |
| | 104 | FORTRAN Compiler | |
| | 100 | Simulator | |
| | 8 | LET-FLET | |
| | 48 | IBM Subroutines | |
| | UD* | Relocatable Program Area | |
| | | Nonprocess Work Storage | Non-Protected Area |
| | TC* | Error Dump Area | |
| | 6 | Error Save Area | |
| | VC* | Nonprocess Save Area | |
| | UD* | Message Buffer | |
| | UD* | Process Work Storage | |
| | UD* | F I/O Save Area | |
| | UD* | Interrupt Save Area | |
| | UD* | Core Load Area | |
| | VC* | Special Save Area | File Protected Area |
| | VC* | Process Save Area | |
| | UD* | Skeleton | |
| | 30 | Error Programs | |
| 1599 | 4 | Cold Start | |

*NOTE: TC indicates enough sectors to store total core; if error dump is not used, no space is reserved. VC indicates enough sectors to store variable core. UD indicates user defined. Relocatable program area boundary increases or decreases nonprocess work storage as programs are deleted or added, respectively.

**NOTE: Sector quantities are approximate figures.

Figure 5. Example of Disk Layout

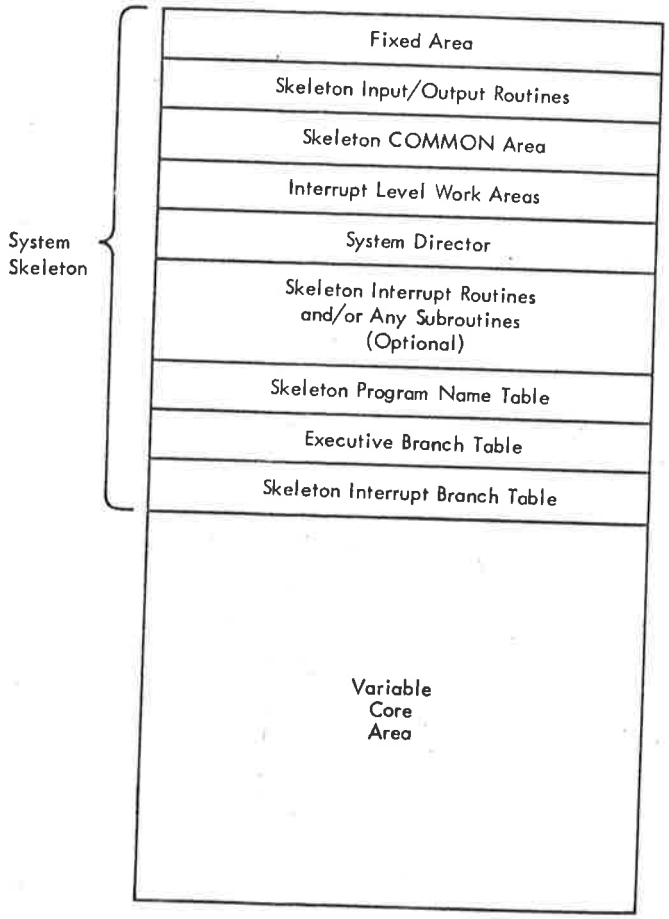



Figure 6. Sample Core Storage Layout

## II. 4 ETT REELL-TIDS PROGRAMMERINGSSPRÅK FÖR PROCESSKONTROLL:

Program skrivna i FORTRAN och ALGOL kompileras för att producera ett program i maskinkod, som arbetar under kontroll av ett styrsystem. När det gäller kontroll av industriella processer är det nödvändigt med ett styrsystem för nästan alla processkontroll situationer. Men inget styrsystem är tillräcklit för att klara alla applickationer. Det är dessutom inte ekonomiskt lönsamt att sätta ihop många olika styrsystem för en maskin.
Språk med specifika kontrolltillämpningar har utvecklats. Språket innehåller här antingen sitt eget styrsystem eller löper under kontroll av ett allmänt styrsystem. Sådana språk kallas Process Oriented Languages (POL). Det vore önskvärt att styrsystemen vore tillräckligt flexibla för att tillåta flera av dessa språk.
För att minska de speciella styrproblemen är det föreslaget att ett programmeringsspråk för industriell processkontroll bör konstrueras. Ett sådant måste vara starkt förbundet med reell-tids maskinvaran och processen, ty det måste vara så grundligt och flexibelt att styrsystemet kan inbäddas i det. Det betyder att språket måste kunna referera till all maskinvaruutrustning (indikatorer, avbrott, perifera organ, etc.) i ett maskinvarusystem för att i detalj specificera arbetet, som ska utföras av styrsystemet.
Fördelar man kan nå med detta språk:

1. Styrprogrammen kan skrivas i reell-tids språket hellre än i maskinkod och följaktligen kan de vara självdokumenterande i mening att koden kan läsas ungefär som vanlig språktext. Detta skulle förenkla rättning, felsökning och modifiering när mskinvaru- eller processkonfigurationen ändras.
2. Pga de stora minskningar i programmeringsansträngningar, som åtgår när ett högre nivå språk användes, kan styrsystemet produceras till lägre pris och följaktligen kan ekonomiskt lönsamma styrsystem göras för speciella applickationer.

3. Modifiering av styrsystemet, för att tillåta språk för speciella ändamål mer passande för en viss kontrollnivå, skulle bli lättare.
4. Decentralicering av maskinerna (central maskin med mindre maskinenheter som terminaler) med deras tjänande styrsystem skulle underlättas. Styrsystemen kunde nämligen skräddarsys till den speciella maskinvarukonfigurationen och mera viktigt är uppdelningen av kontrollproblemet mellan de olika maskinerna.
5. Möjligheten att felchecka varje anrop av systemsubrutiner under ett felsökande stadium av ett stort system och senare kompilera på nytt utan att checka det för att minska minnesutrymme och genomlöpningstid.

REAL-TIME LANGUAGE (RTL).

Nedanstående presentation av ett RTL ska speciellt sysselsätta sig med intressanta drag hos språket; drag,som gör det olikt mera konventionella språk typ FORTRAN. Terminologin är hämtad från vanliga programspråk såsom FORTRAN och ALGOL samt från processkontrollprogram använda av flera system tillverkare. De mest inflytelserika källorna är PL/1 och MULTICS konstruerat för vetenskapliga maskinsystem i reell-tid. PL/1 är inte ett reell-tidsspråk men är avsett att producera program, som löper under konventionella styrsystem. 95% av koden, som används för MULTICS är hämtad från PL/1. Återstående 5% är maskinkod eftersom inte PL/1 har reell-tids möjligheter, vilket RTL tänkts ha inbyggt. Karakteristik av RTL: Maskinvaruregister, in/utsignal buffertar, indikatorer etc. är variabler, som har reserverade identifierare. Dessa variablers attribut är implicit definierade i maskinvara. de kallas perifera variabler till skillnad från vanliga interna variabler.

Exempelvis om en maskin har operationen " testa om I/O kanalen upptagen " skulle detta i RTL associeras med en logisk perifer variabel IOCBUSY.
För utsagan:

```
IF IOCBUSY THEN .... ;
```

skulle RTL kompilatorn generera ett lämpligt " testa om I/O kanal upptagen och hoppa " kommando.
Om maskinen har operationen " skriv ut på skrivmaskin från ackumulatorregistret " har anspråksfulla kommandoord använts för att ordentligk initiera utorganet. I RTL skulle utskrift av tecknet 'R' kunna göras genom utsagan:

```
CHTR = 'R'
  :
TYPEOUT = CHTR ;
```

om man antar att CHTR tidigare har deklarerats som en symbol variabel.(CHTR är en intern variabel medan TYPEOUT är perifer variabel.)
Programavbrott operationer är väsentliga för processkontrollens många uppgifter. Styrrutinens effektivitet i reell-tid är till stor del beroende av avbrottssvaren, som därför måste kunna användas av styrsystemets programmerare.
Till grund för nedanstående utsagor (statements) ligger PL/1 språket. Varje avbrottskälla har en identifierare associerad med sig. Identifieraren kallas actuel inetrrup$ identifier, och är vanligtvis orienterad till det organ, som orsakat avbrottet. Exempelvis för ett avbrott, som fås från en Skrivmaskinsoperation föredras identifieraren TYPEWRITER eller TYPER framför namn såsom INT 8 (nivå 8 avbrott).
Svaret på ett avbrott specificeras av en ON utsaga.
Exempel:

```
200  ON TYPER DO
201  BEGIN TYPEOUT = FIRST (OUTSEQ,NONE);
202  GO TO EXIT; NONE: IOFF = FALSE;
203  EXIT: END
```

Rad 201 till 203 är utsagan för avbrottssvar och rad 200 anger detta som svar på det aktuella avbrottet TYPER. Nivån NONE är återvändspunkten när listan OUTSEQ är tom. Språket själv innehållar inga definitioner angående prioritet; detta är en del av maskinvaran. I maskiner, som tillåter maskning av avbrott måste emellertid dessa möjligheter göras tillgängliga genom att definiera dem som perifera variabler. Ett maskregister för avbrott kan definieras som ett logiskt fält.

Avbrott kan inträffa när som helst under utförandet av en RTL utsaga. Eftersom både det avbrutna programmet och svarsrutinen vanligtvis definieras med hänsyn till speciella register såsom instruktionsräknaren, ackumulatorn etc. måste maskinens "tillstånd" vid avbrottet lagras undan innan genomlöpandet av svarsrutinen. Vid fortsatt behandling av det avbrutna programmet tas det fram igen. Grundformen av ON utsagan medför därför att kompilatorn föreskriver 'save' operationer före svaret och 'restore' operationer vid slutet av utsagan.

Vid utförandet av vissa RTL utsagor kan man inte tillåta avbrott. Följande sekvens är ett exempel på två insruktioner, som måste utföras utan avbrott mellan dem:

```
MASK(10) = FALSE;
IC = ICSAVE;
```

Dvs, sätt bit 10 i MASK registret till noll och återplacera innehållet i instruktionsräknaren. För att indikera för kompilatorn att detta måste betraktas som en enda instruktion föregår en speciell symbol &, sekvensen och ersätter alla semikolon utom det sista. Möjligheten att använda dylika oavbrytbara utsagor är starkt maskinbundna.

Den normala användningen av I/O faciliteter i system är att en eller ett fåtal I/O subrutiner konstrueras för varje organ, och dessa anropas av andra program. RTL inkluderar list variabler, som kan användas som parametrar för dessa subrutiner. Dessa 'circular lists' tillfredsställer både köbildningskraven och behovet av list parametrar för I/O subrutiner. Ordet CIRCL användes för att deklarera 'circulat lists' och måste deklarera listans maximala längd.

Exempelvis:

```
DECLARE REAL SEQ4/CIRCL(128),
SEQ5/CIRCL(16);
```

deklarerar att SEQ4 och SEQ5 är circular lists med maximal längd på 128 respektive 16. Båda innehållande reella (flytande punkt) variabler.

Sammanfattning:

En analys av program problemet vid industriellaprocess kontroll har visat att det finns ett stort behov av ett reelltidsspråk på högre nivå. Ett passande RTL skulle uppmuntra utarbetandet av Process Oriented Languages. /1/

## II.5 PROCESS-ALGOL 1.

Avsikten med följande beskrivning är att komma fram till ett system för processtyrning där programmet kan skrivas i ett ALGOLliknande språk.
Systemet består av en PDP-8; en 12-bitars maskin med 4K kärnminne och ett skivminne (disk). Kompileringen blir alltför omfattande och måste göras på en särskild maskin. (I detta fall GIER.)

Fasta program i maskinen.
Det fasta programmet kan uppdelas i 3 huvuddelar:

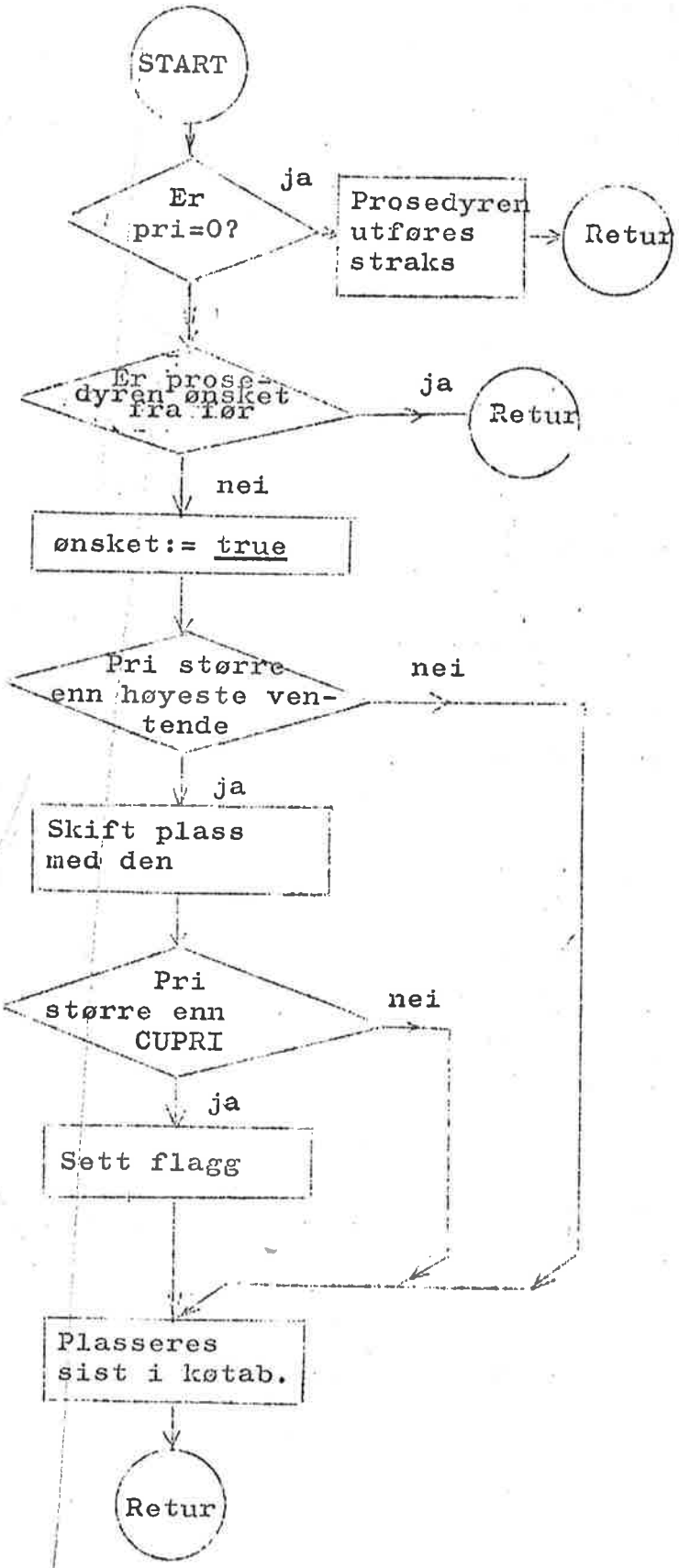
1. Run-systemet för ALGOL
2. Administrationsrutiner för reell-tidssystemet
3. Standardprocedurer

1. Runsystemet för ALGOL består till största delen av små subrutiner, till vilka hopp ständigt görs från programmet. Alla variabler placeras i en stack och refereras relativt till en stackreferens. När ett nytt block aktiveras, upprättas en ny stackreferens över den gamla stacken. I programmet föreligger aritmetiska uttryck på omvänd polsk form och det kräver då en arbetsstack på toppen av variabelstacken.
2. Reell-tidssystemet. Här följer en ingående studie av reell-tidstabellerna och hur de administreras. ALGOLprocedurer, som initieras av klocka eller avbrott placeras i var sina kolumner.

| Kanal (SKIP-order) | Tid fram till nästa utförande | Prioritet för avbr. procedur | Pri. för klock proc. | Adress avbrotts proc. | Adress klock proc. |
|---|---|---|---|---|---|
| klokkeint | | 0 | | | |
| TTI | | 0 | | | |
| TTO | | 0 | | | |
| SKP | 7777 | 4001 | 2001 | 0 | |

I första kolumnen är avbrottskanalerna representerade med sina skip-order. Vid ett avbrott startas en administrationsrutin, USINT, som söker igenom denna kolumn för att finna den kanal det gäller. När den är påträffad, sker hopp till subrutin TEST, som bestämmer vad som ska göras.(Se blockschema fig.8)
Det finns tre fasta kanaler:
Den första används för klockavbrott. Var 10:e ms kommer en puls, subrutinen KLOCCEINT startar och genomletar tidstabellen.
De två andra är servicerutiner för in- och utmatning via Teletype.
Sist i den första kolumnen står en skiporder, som har till uppgift att fånga upp eventuella avbrott från obrukade kanaler. Den andra kolumnen innehåller kvarvarande tid fram till klockprocedurenas utförande. Tiden sätts in som ett negativt tal. Subrutinen KLOKKEINT kallas var 10:e ms och räknar då upp alla talen i kolumnen med ett. När ett av talen nått 0, har tiden gått ut för tillhörande procedur. Den önskas då utförd och hopp sker till TEST. Talet 7777 sist i kolumnen markerar slutet på tabellen.
Den tredje kolumnen innehåller status och prioritet för avbrottsprocedurerna. Bit 0 innehåller den boolska tillståndsvariabeln "önskad". "Önskad" blir satt när tillhörande procedur önskas utförd och blir "återsatt" när den är färdigutförd.
Fjärde kolumnen innehåller status och prioritet för klockprocedurerna. Bit 0 innehåller "önskad". Bit 1 innehåller den boolska tillståndsvariabeln "aktiv", som sätts då proceduren ska kunna utföras. Om den inte är satt räknas variabeln som frånkopplad. Bit 2-11 innehåller prioriteten. Sista cellen innehåller prioriteten för block 0.
Femte och sjätte kolumnerna innehåller startadress för respektive proceduranrop.
Kötabellen innehåller referenser till de procedurer, som är önskade och väntar på att bli utförda.
Tabellen har tre hjälpstorheter:

1. CUPRI innehåller prioriteten för löpande program.
2. PRINULL innehåller referens till väntande pricedur med högst prioritet.
3. FLAGG är en boolsk variabel, som sätts när man önskar


START
Er pri=0?
ja
Prosedyren utføres straks
Retur
Er prosedyren ønsket fra før
ja
Retur
nei
ønsket:= true
Pri større enn høyeste ventende
nei
ja
Skift plass med den
Pri større enn CUPRI
nei
ja
Sett flagg
Plasseres sist i køtab.
Retur

Figur 8.

en procedur med högre prioritet än den CUPRI anger, dvs ett löpande program kan avbrytas.
Kötabellen administreras av två rutiner, TEST som anropar nya procedurer, och RTENTRY som avlägsnar proceduren när den ska utföras.
Snabba standardprocedurer såsom KLOKKEINT och servicerutinerna för Teletypen utföres omgående utan att gå genom kötabellen. (Prioriteten = 0; se flödesschema.)
När FLAGG blir satt betyder det att löpande program kan avbrytas till fördel för en procedur med högre prioritet. Omedelbart innan uthopp från subrutinerna för ALGOLs runsystem ligger en test på FLAGG. Är FLAGG satt sker hopp till RTENTRY, som lagrar återhopp. Det sker som följer:

1. 6 celler lagras i stacken som returinformation.
2. Nytt värde på CUPRI sätts.
3. Proceduren med högst prioritet letas fram i kötabellen och referensen flyttas till PRINULL.

När en procedur är färdigutförd, sker hopp till RTEXIT. Returinformationen, som lagrats av RTENTRY hämtas fram. "Önskad" blir återsatt och hopp sker tillbaks till det avbrutna programmet.

Standardrutiner, som arbetar på reell-tids-tabellen.

1. clock: proceduranropet förs upp i reell-tids-tabellen, så att det kan utföras vid bestämda tider
2. interrupt: proceduranropet får tilldelat en avbrottskanal. Ett avbrott på denna kanal leder till att tillhörande proceduranrop blir önskat utförd.
3. set: boolsk procedur som placerar tiden, fram till nästa utförande för ett proceduranrop, i tidstabellen. Om proceduren redan är deklarerad önskad eller håller på att utföras blir inte ny tid satt. Set får värdet false. Är procedur däremot ledig, kommer tiden att sättas enligt en parameter med enheten 10 ms. Är proceduren i tillståndet "passiv" överförs den till "aktiv". Set får värdet true.

4. inc: tiden fram till nästa utförande ökas.
5. chap: tillhörande procedur får ny prioritet.
6. delay: om tillhörande procedur är önskad eller under utförande får den boolska standardproceduren delay värdet true.
7. dscnct: proceduren överföres till "passiv".

Standardprocedurer för bitmanipulation:

1. bit: test om viss bit är satt.
2. setbit: tillhörande bit blir satt.
3. clearbit: tillhörande bit blir återsatt.
4. shiftr: logiskt skift.

In/ut procedurer från processen:

1. digin: innehållet i ingångsregistret överföres till den boolska variabeln.
2. digout: motsvarande för utgångsregistret.
3. alogin: via A/D omvandlaren läses innehållet i multiplexerkanalen till variabeln.
4. alogout: heltalsvariabeln går till D/A omvandlaren.

/6/,/7/.

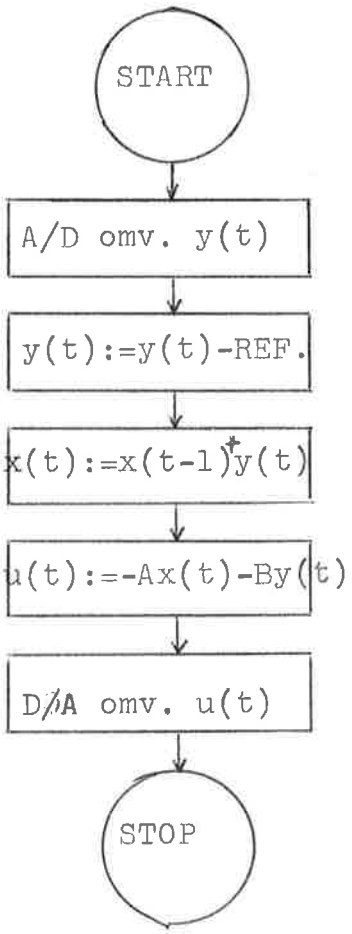
## III. JÄMFÖRELSE AV PROGRAM I OLIKA PROCESSDATAMASKINER FÖR GENERERING AV PI-REGULATOR.

### III.1 STYRLAGEN.

Signalen y(t) från en process A/D omvandlas.
Beräkningsgången är följande:

$$y(t) := y(t) - REF \;;$$
$$x(t) := x(t-1) + y(t) \;;$$
$$u(t) := -A\,x(t) - B\,y(t) \;;$$

Styrsignalen u(t) D/A omvandlas och skickas tillbaka till processen.
REF står för referensvärdet eller börvärdet, dvs det värde man önskar y(t) ska ha för korrekt styrning av processen.
x(t) är en tillståndsvariabel och A och B är integrationskonstanter.
Flödesschema:



III.2 PDP-8.

PDP-8 har en registeruppsättning enligt figuren nedan:

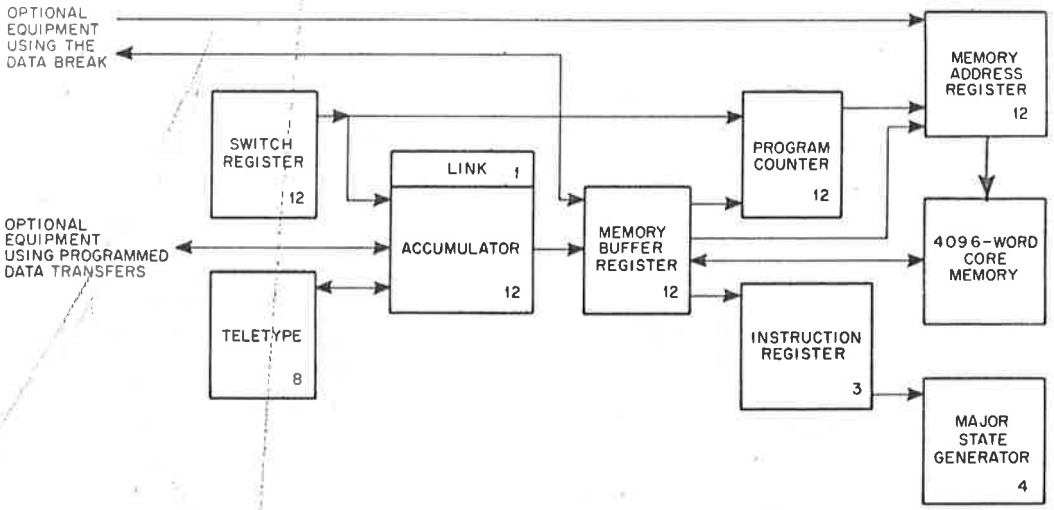



Figure Standard PDP-8 Block Diagram

Ackumulatorn är ett 12 bitars register där den första biten är teckenbit. För att kunna utföra multiplikationerna i programmet måste ett Extended Arithmetic Element (Typ 182) tillfogas. Detta innehåller en utvidgning av AC-registret med ett 12 bitars multiplikandregister (MQ). Maskinen har dessutom ett en-bits register -Link (L) - för att möjliggöra test av overflow och carry. Att notera är att PDP-8 likt de flesta maskiner representerar negativa tal med sitt tvåkomplement. Tänkta PDP-8 system innehåller också en A/D omvandlare (Typ 138E) samt en D/A omvandlare (Typ AA01A).
Analoga insignalen ligger mellan 0 och -10 volt och A/D omvandlaren, som arbetar med succesiva approximationer, ger ett digitalt värde av 6 till 12 bitar med tecken. 0 volt ger således $4000_8$, -5 volt $0000_8$ och -10 volt ger ett digitalt värde av $3777_8$. I programmet förutsättes en noggrannhet på 12 bitar dvs maximalt uttag.

Till A/D omvandlaren typ 138E användes en multiplexer (Typ 139E), som ger möjlighet att adressera upp till 64 ingångskanaler och avläsa var och en av dessa 415 ggr/sek. Denna funktion behöver dock inte utnyttjas här då styrlagen enbart genomlöpes för en ingång.
D/A omvandlaren omvandlar 12 bitars ord till analog spänning. Grundutrustningen består av 3 kanaler, som vardera innehåller ett 12 bitars digitalt buffert register och en D/A omvandlare (DAC).
Kärnminnet består av totalt 4096 ord (4K) och är uppbyggt av 32 sidor om 128 ord i varje. Om man med ett program beräknar fylla mer än en sida, måste man lämna ungefär 10 minnesceller i slutet av sidan för adresserings- och hoppinstruktioner till ny sida. Detta innebär alltså ett slöseri med minnesutrymme och beräkningstid. 12 binära positioner kan endast adressera 4K ($2^{12}$) celler men man kan tillfoga en logik om 3 bitar, som kan adressera 8 st 4K minnen dvs 32K. Maskinen blir då ungefär 4 ggr långsammare. Begränsningen med 12 bitar är således uppenbar. /8/.

III.2.1 PROGRAMMET I FIX RÄKNING, ENKEL PRECISION.

```
200  START, CLA        /0 till AC.
201         ADCC       /0 till kanaladressregistret (CAR).
202         TAD CNUM   /Kanalnumret adderas till det tomma AC.
203         ADSC       /Kanalmumret överföres från AC till CAR.
204         ADCV       /A/D omvandlingen startas och när den
                       /är klar sätts en flagga.
205         ADSF       /Test av flaggan. Om den innehåller en
                       /binär etta räknas program räknaren (PC)
                       /fram ett steg, dvs nästa instruktion
                       /slopas. Om flaggan innehåller en nolla
                       /utföres nästa instruktion.
206         JMP.-1     /Hoppa tillbaka ett steg och testa igen.
207         ADRB       /Omvandlingsbuffert till AC.(y(t)).
```

```
210         TAD BÖRV    /Referensvärdet lagrat med minustecken
                        /adderas till C(AC).y(t):= y(t) - ref.
211         DCA 306     /y(t) lagras i cell 306. 0 till AC.
212         TAD 306     /Addera y(t) till AC.
213         TAD XVAR    /x(t-1) i cell XVAR adderas till AC.
214         DCA XVAR    /x(t) lagras i cell XVAR.
215         TAD AAA     /Tag upp innehållet i cell AAA dvs talet
216         DCA 10      /305 och lägg ner det i cell 10. Cellerna
                        /10₈ till 17₈ på sida noll i kärnminnet
                        /har en speciell funftion. När de adres-
                        /seras indirekt (se nedan) ökas deras
                        /innehåll med ett och resulterande tal
                        /utgör sedan adress till den cell, som
                        /ska läsas. Denna funktion kallas auto-
                        /indexering och kommer att möjliggöra att
                        /teckenmultiplikation, som ska utföras
                        /två ggr kanskrivas som en subrutin.
217         TAD MINB    /Konstanten -B tas upp och lägges i cell
220         DCA 310     /310.
221         TAD BBB     /Se ovan
222         DCA 11
223         JMS MULT    /Hoppa till subrutin för teckenmultipli-
                        /kation.(Se nedan) Återhoppsadressen lag-
                        /ras undan i cell MULT.
224         TAD HIGH    /Tag upp den mest och den minst signifi-
225         DCA HBY     /kanta delen och lagra dem i HBY resp.
226         TAD LOW     /LBY.
227         DCA LBY
230         TAD XVAR    /Gör klart att hoppa till surutin genom
231         DCA 307     /att lägga x(t) och -A i cellerna 307
232         TAD MINA    /och 311 respektive.
233         DCA 311
234         JMS MULT    /Hoppa till subrutin MULT.
```

```
/ Följande programbit undersöker hur stort resultatet u(t) är .
/ Omresultatet är större än vad som kan representeras i maski-
/ nen på 12 bitar eller D/A omvandlas, måste man ge ut max 0
/ volt anolog spänningoch -10 volt om resultatet är för litet.
/ I övrigt sker direkt omvandling.
235         CLA         /0 till AC.
236         TAD LBY     /Den minst signifikanta delen av -B  y(t)
                        /till AC.
237         RAL         /Rotera bitarna ett steg vänster, dvs
                        /teckenbiten AC0 går till Link (L).
240         CLA         /0 till AC
241         RAR         /Rotera ett steg höger. Teckenbiten till
                        /AC0.
242         TAD LOW     /-A x(t) adderas till AC. Om produkterna
                        /har samma tecken får vi spill (L=1).
243         RAL         /Rotera ett steg vänster.
244         SZL         /Hoppa över nästa instr. om L=0.
245         JMP AA      /Hoppa till AA. Talen har samma tecken.
246         CLA         /Produkterna har olika tecken. De läggs
247         TAD LBY     /samman och hopp sker till CC+1.
250         TAD LOW     /(Ingen risk att u(t) för stort.)
251         JMP.CC+1
252  AA,    CLA CLL     /0 till AC och 0 till L. Simultant.
253         TAD LBY     /-B  y(t) till AC:
254         RAL         /ROtera ett steg vänster.
255         SNL         /Hoppa om L=1 dvs talet är negativt.
256         JMP BB
257         RAR         /Rotera tillbaka ett steg höger.
260         TAD LOW     /Addera -A  x(t).
261         RAL         /Rotera ett steg vänster för att bestämma
                        /tecken.
262         SZL         /Hoppa om L=0 dvs overflow,annars ej.
263         JMP CC
264         CLA
265         TAD MIN10   /Tag upp talet +2047 till AC.
266         JMP CC+1
```

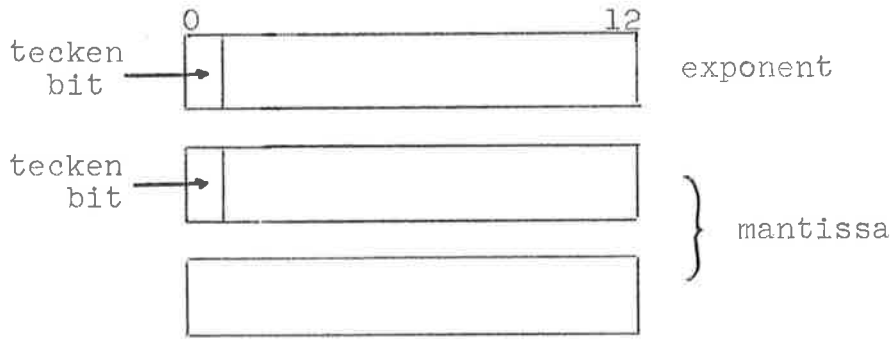
```
267  BB,    RAR          /Rotera höger. Båda talen positiva.
270         TAD LOW
271         RAL
272         SNL          /Hoppa om L=1 dvs overflow.
273         JMP CC
274         CLA
275         TAD MAXO     /Tag upp talet -2046 till AC.
276         JMP.+2
277  CC,    RAR          /ROtera tillbaka ett steg höger.
300  DAO,   DAL 1        /Innehållet i AC laddas i digitala buf-
                         /fertregistret kanal 1 för omvandling.
301         HLT          /Stoppa programmet.
302  BÖRV,
303  MINA,  -A
304  MINB,  -B
305  XVAR,  0
306       ,  y(t)
307       ,  x(t)
310       ,  -B
311       ,  -A
312  AAA,   305
313  BBB,   307
314  HIGH,  0
315  LOW,   0
316  HBY,   0
317  LBY,   0
320  MIN10, -2046
321  MAXO,  +2047
322  SIGN,  0
```

Subrutin för tecken-multiplikation:

```
323  MULT,  0            /Plats för återhoppsadress.
324         CLA CLL      /0 till AC och 0 till L.
325         TAD I Z 11   /Konstanten -B (eller -A) läses in i AC
                         /genom indirekt adressering. (Se ovan)
326         SPA          /Hoppa om AC positivt.
327         CMA CML IAC/Tvåkomplementera C(AC) och C(L).
330         MQL          /AC till MQ
331         TAD I Z 10   /y(t) plockas upp genom indirekt adress.
332         SPA          /Hoppa om AC positivt.
333         CMA CML IAC/Tvåkomplementera.
334         DCA MLTP     /Lagra multiplikanden. 0 till AC vid DCA.
335         RAL          /Rotera vänster. L till AC11 osv.
336         DCA SIGN     /Lagra teckenindikatorn.
337         MUY          /C(MQ) multipliceras med närmast följande
                         /cells innehåll. Resultatet blir ett 24
                         /bitars ord med de mest signifikanta bi-
                         /tarna i AC och resten i MQ.
340  MLTP,  0
341         DCA HIGH     /C(AC) till HIGH.
342         TAD SIGN
343         RAR          /Teckenindikatorn till L
344         MQA          /MQ till AC.
345         SNL          /Hoppa om L=1. (Produkten negativ)
346         JMP LAST
347         CLL CMA IAC/0 till L. Tvåkomplementera AC.
350         DCA LOW
351         TAD HIGH
352         CMA          /Ettkomplementera.
353         SZL          /Testa om L=0. (carrybit)
354         IAC          /Öka med ett om carry från LOW.
355         DCA HIGH
356         SKP          /"Skippa" nästa instruktion.
357  LAST,  DCA LOW
360         JMP 1 MULT   /Hoppa till cellen vars adress ligger i
                         /MULT, dvs tillbaka till programmet.
```

III.2.2 PROGRAMMET I FLYTANDE RÄKNING.

Ett tal med flytande punkt består av en mantissa och en exponent ( MANTISSA $2^{EXPONENT}$) där mantissans absolutvärde ligger mellan $\frac{1}{2}$ och 1. I PDP-8 representeras talet enligt fig. nedan:




Man använder således tre konsekutiva celler. Som anmärkning kan sägas att det förefaller vara ganska dålig balans mellan exponent och mantissa. Utrymmet för exponenten är för stort i förhållande till den 24 bitar långa mantissan (7 decimalers noggrannhet).
Flytande punkt systemet använder en pseudoackumulator (FAC), som består av tre register nämligen cellerna 44,45 och 46. De används i tur och ordning för exponenten och mantissans mest och minst signifikanta delar.
Programmet anropar en subrutin "floating point pakcage 8-5-s", ett interpreterande program, som tolkar de pseudoinstruktioner, som används. /8/,/9/.

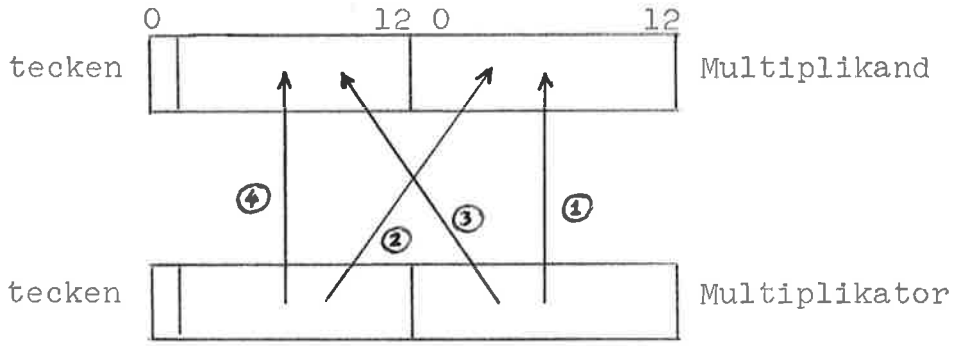
```
200   START, CLA
201          ADCC
202          TAD CNUM
203          ADCV
204          ADSC
205          ADSF
206          JMP.-1
207          ADRB      / Omvandlingsbuffert (12 bitar) till AC.
210          DCA 45    / AC till cell 45 (mantissans mest signi-
                       / fikanta del). 0 till AC.
211          DCA 46    / AC dvs 0 till cell 46.
212          TAD EXP   / Exponenten 11 (decimalt) tas upp i
213          DCA 44    / AC och läggs i cell 44.
```

```
214         JMS I 7     /Anropa interpreterane program. Adressen
                        /till den minnescell, som innehåller
                        /nästa pseudoinstruktion lagras undan
                        /vid detta anrop.
215         FNOR        /Innehållet i FAC (y(t)) normaliseras.
216         FSUB BÖRV   /Feferensvärdet minskas från FAC.
217         FPUT YVAR   /y(t) lagras.
220         FGET XVAR   /x(t-1) till FAC.
221         FADD YVAR
222         FPUT XVAR
223         FGET XVAR
224         FMPY MINA   /Flytande multiplikation.
225         FPUT SLASK
226         FGET YVAR
227         FMPY MNIB
230         FADD SLASK
231         FEXT        /Lämna det interpreterande programmet
                        /och hoppa till nästa instruktion. Re-
                        /sultatet ligger nu i den flytande acku-
                        /mulatorn dvs reg. 44, 45 och 46.
232         CLA         /0 till AC.
233         TAD 44      /Exponenten till AC.
234         SZA SMA     /Hoppa över nästa instruktion om expo-
                        /nenten är 0 eller negativ.
235         JMP.+3      /Hoppa 3 steg framåt om således exp.≥ 1.
236         CLA
237         JMP READY+1/Digitala utsignalen = 0; D/A omvandla-
                        /ren kan bara ta 12 bitar och mantissan
                        /ligger mellan ½ och 1. Exp. är 0 eller
                        /neg.
240         TAD MIN13   /Lägg till -11 dvs sätt binärpunkten till
                        /höger om flytande pkten.(11 platser).
241         SNA         /Hoppa om AC≠0
242         JMP READY   /Om AC=0 ligger u(t):s rätta värde i
                        /cell 45. Hoppa till READY.
            (SMA        /Testa om talet för stort. Hoppa i så
            JMP ERROR   /till felrutin.)
```

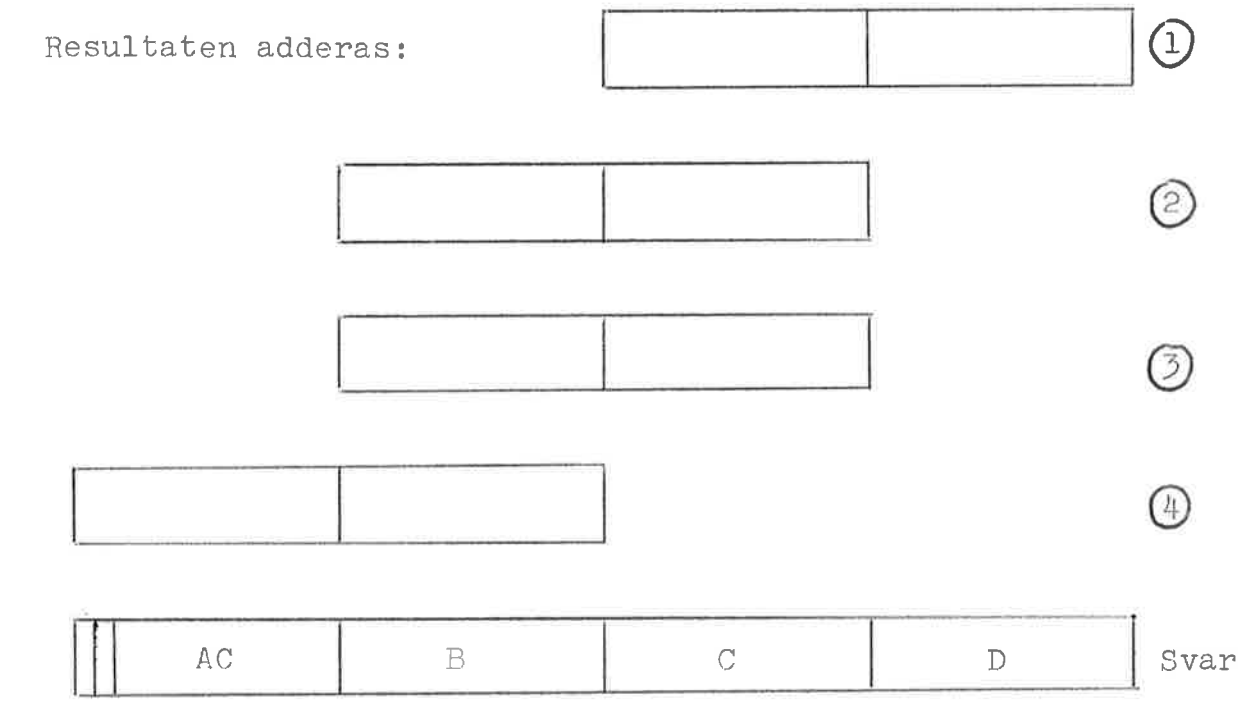
```
243          DCA 44      /Lägg tillbaka den nya exponenten i cell 44.
244  LABEL,  CLL         /0 till L.
245          TAD 45      /Mantissan till AC.
246          SPA         /Hoppa om AC större än noll.
247          CML         /Komplementera L dvs L=1.
250          RAR         /Rotera AC och L ett steg höger, dvs Lgår
                         /till AC0 och AC11 till L. OM AC neg. skif-
                         /tar vi in en 1:a i AC0 som sig bör.
251          DCA 45      /Lagra mantissan i cell 45.
252          ISZ 44      /Öka cell 44 med ett och testa om = 0. Om
                         /innehållet är 0 hoppa över nästa instr.
253          JMP LABEL   /Mantissan har inte skiftats tillräckligt.
                         /Genomlöp loopen än en gång.
254  READY,  TAD 45
255          DAL 1
256          HLT
257  CNUM,
260  BÖRV,               /Talen lagrade i flytande form.
263  XVAR,
266  YVAR,
271  MINA,
274  MINB,
277  SLASK,
282  EXP,    11
283  MIN13,  -11
```

III.2.3 PROGRAMMET I FIX RÄKNING, DUBBEL PRECISION.

Ett ord i dubbel precision lagras i två konsekutiva celler. Operationer, som addition, subtraktion, lagring och upphämtning måste utföras cell för cell dvs den mest signifikanta delen för sig och den minst för sig.
Vid multiplikation anropas emellertid en subrutin DMUL, som av två 23 bitars ord ger en produkt av 46 bitar enligt nedan: /8/,/10/.




Resultaten adderas:



```
400  START, CLA
401         ADCC
402         TAD CNUM
403         ADSC
404         ADCV
405         ADSF
406         JMP.-1
407         ADRB
410         TAD BÖRV1
411         DCA YVAR1
412         TAD BÖRV2
413         DCA YVAR2
414         CLL         /0 till L.
415         TAD YVAR1
416         TAD XVAR1
417         DCA XVAR1
420         RAL         /Rotera ett steg vänster. L(carry) till
                        /AC11.
421         TAD YVAR2
422         TAD XVAR2
423         DCA XVAR2   /x(t):= x(t-1) + y(t)
424         JMS I DMULTP/Hoppa indirekt till subrutin för tec-
425       , 446         /kenmult. i dubbel precision.
426       , 452
/ Indirekt adressering är nödvändig då subrutinen ligger på
/ en annan sida i minnet.
/ Det resultat, som är av intresse ligger efter mult. i AC
/ och cell B.
427         DCA HBY     /Lagra mest signifikanta delen.
430         TAD I CELLB/Indirekt adress.
431         DCA LBY
432         JMS I DMULTP
433       , 444
434       , 450
435         DCA HAX
436         TAD I CELLB
437         TADLBY
440         DAL 1
441         HLT
```

```
442  CNUM,
443  BÖRV1,
444  BÖRV2,
445  XVAR1,
446  XVAR2,
447  YVAR1,
450  YVAR2,
451  MINA1,
452  MINA2,
453  MINB1,
454  MINB2,
455  HBY,
456  LBY,
457  HAX,
460  CELLB, B (cell 341)
461  DMULTP,DMUL (cell 200)
```

III.3 IBM-1800.

IBM-1800 systemets registeruppsättning visas i figuren nedan. Ackumulatorregistret består av 16 bitar, där bit 0 är teckenbit och bitarna 14 och 15 markerar carry respektive overflow. Det finns två modeller av A/D omvandlare (se tabell 1), som omvandlar den analoga signalen mellan $\pm 5$ volt till alternativt 8, 11 och 14 bitar plus teckenbit. Modell 2 har dessutom en sample och håll förstärkare, som medför ökad omvandlingshastighet för systemet.
Av D/A omvandlare erbjudes fyra modeller. Modell 1 och 2 omvandlar 10 bitar till en respektive två analoga utgångar.

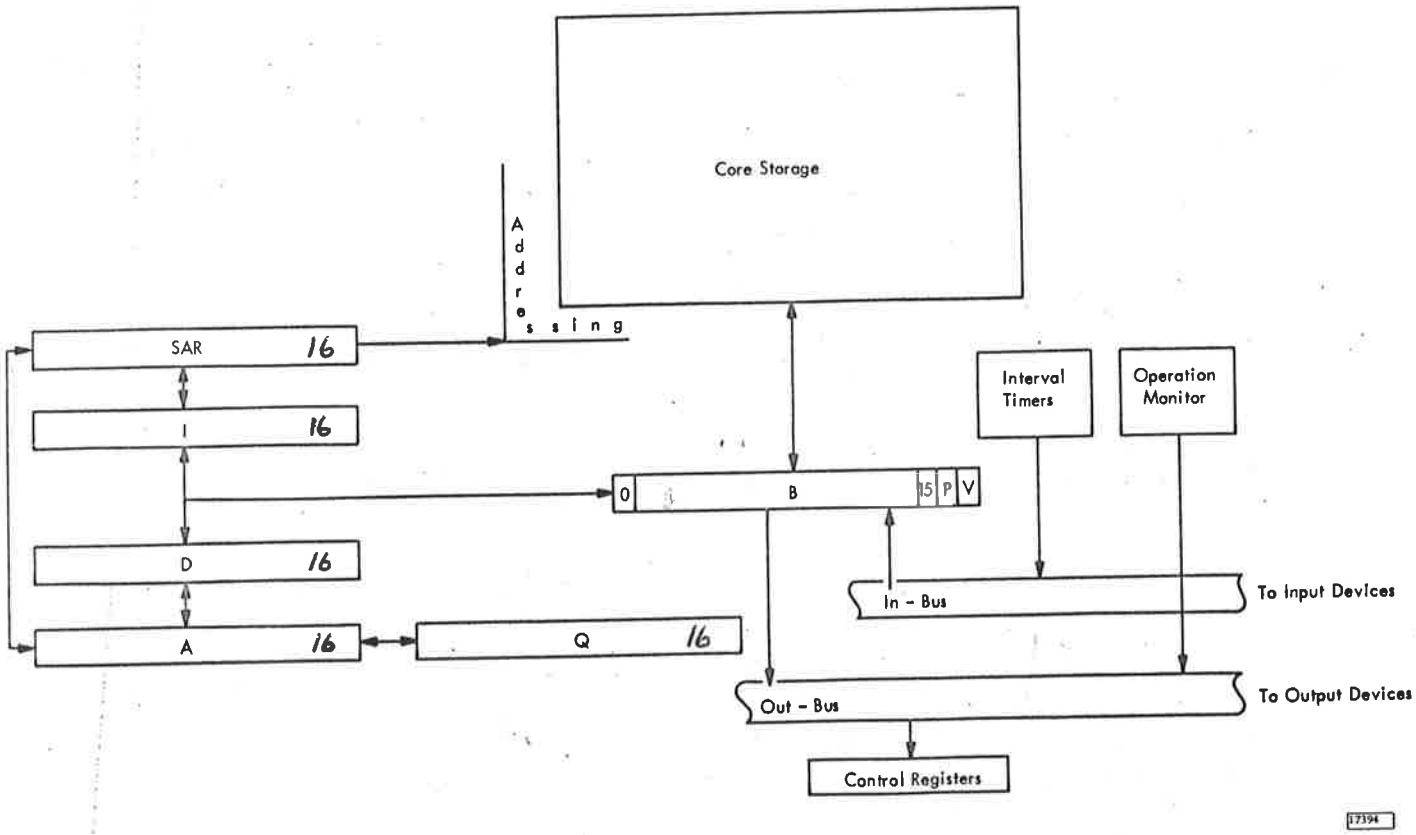


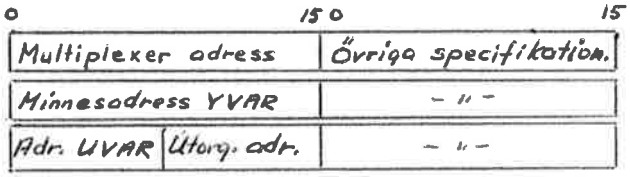

Figure 1800 P-C Data Flow

Modell 3 och 4 tar 13 bitar plus teckenbit till en respektive två utgångar. Omvandlingstiden är 6µs för alla modellerna. Kärnminnet är variabelt från 4K upp till 32K. Man måste emellertid ha minst 8K för att få rum med de monitorpacket, som erbjudes.
Vid kommunikation med I/O organ kan maskinen arbeta enligt två metoder: programstyrning och genom datakanaler.
Vid programstyrning sätts en flagga när något organ önskar uppmärksammas. Man frågar och finner ut vem som orsakat det och informationen överföres sedan till kärnminnet.
Datakanalerna arbetar på avbrottsbasis sk cycle stealing. /11/

III.3.1 PROGRAMMET I FIX RÄKNING ENKEL PRECISION. /11/,/12/.

```
            ORG 100       Definiera programmets början.
100 START,  XIO WRITE     Adressera multiplexern och välj analog
                          signal, som ska omvandlas.(se cell WRITE)
                          EN intern signal sändes från multiplex-
                          ern till A/D omvandlaren (ADC) för att
                          starta omvandlingen. När ADC:n är klar
                          sändes en avbrottssignal till program-
                          räknaren och en subrutin lokaliserar
                          orsaken till avbrottet om nödvändigt.
101         XIO READ      Den digitala signalen överförs till min-
                          nescell YVAR.(Se cell READ.)
102         LD YVAR       y(t) till A-reg; bitarna 0-14.
103         S BÖRV        y(t):= y(t) - referensvärdet.
104         STO YVAR      Lagra y(t).
105         A XVAR        x(t):= x(t-1) + Y(t).
106         STO XVAR
107         M MINA        x(t) -A. Teckenmultiplikation utföres
                          här i maskinvara (jmf PDP-8). Resulta-
                          tet i A-reg och Q-reg.
108         LDX 1 16      Ladda index register (XR) 1 med talet 16.
109         SLT 1         Skifta A och Q vänster så många steg som
                          XR 1 anger. Om styrlagen vettig innehål-
                          ler A innan skift enbart nollor, varför
                          vi genom skiften inte förlorar informa-
                          tion.
```

```
10A          STO SLASK    Lagra x(t) -A i cell SLASK.
10B          LD YVAR
10C          M MINB
10D          LDX 2 16     Ladda XR 2 med talet 16.
10E          SLT 2        Skifta Q och A 16 steg vänster.
10F          A SLASK      Addera SLASK: Resultat: y(t) -B +
                          x(t) -A.
110          BSC 0        Hoppa över nästa instruktion om over-
                          flow-biten inte satt.
111          BSC L SPILL  Overflow. Hoppa till nivå SPILL.
                          (L betecknar tvåinstruktionsord.)
113          STO UVAR     Lagra u(t) i cell UVAR.
114          BSC L OUT    Hoppa ovillkorligt ut.
116  SPILL,  BSC L NEG,+  Hoppa till NEG om A-reg 0 eller neg.
118          LD MAX       Ladda A med största positiva tal , ty
                          u(t) är för stort att representeras
                          på 16 bitar i maskinen.
119          STO UVAR
11A          BSC L OUT    Hoppa ovillkorligt ut.
11C  NEG,    LD MIN       Ladda A med minsta tal.
11D          STO UVAR
11E  OUT,    XIO DAO      Analogt utgångsregister väljes och digi-
                          tala värdet u(t) överföres till detta.
11F  BÖRV,   DC
120  MINA,   DC
121  MINB,   DC           Data specificerade som konstanter.
122  MAX,    DC
123  MIN,    DC
124  YVAR,   BSS 1
125  XVAR,   BSS 1        Reserverade arbetsutrymmen.
126  UVAR,   BSS 1
127  SLASK,  BSS 1
128  WRITE,
12A  READ,
12C  DAO,
12E          END
```



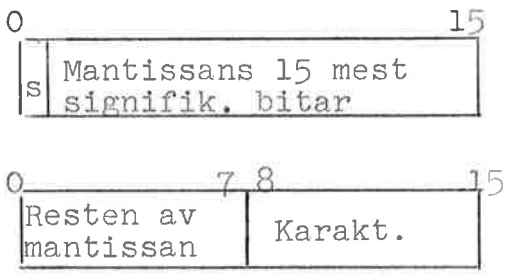

/11/,/4/.

III.3.2 PROGRAMMET I FLYTANDE RÄKNING.

För beräkningar i flytande räkning användes en pseudoackumulator (FAC) bestående av ett treordsregister. Registret utgöres av tre minnesceller belägna i slutet av en överföringsvektor, som användes för kommunikation mellan huvudprogrammet och subrutiner. FAC:s format visas i fig. nedan./11/,/12/,/13/.


Karakteristik FAC

Reella tal i flytande räkning lagras i kärnminnet i två konsekutiva celler:




Vid beräkning anropas olika aritmetiska och speciallicerade subrutiner, program, som utför aritmetiken i flytande räkning.

```
            ORG 100
100  START, XIO WRITE
101         XIO READ
102         LD YVAR      y(t) till A-reg; bitarna 0-14.
103         CALL FLOAT   Anrop av subrutin. Innehållet i A över-
                         förs till FAC i reellt tal.
104         CALL NORM    FAC normaliseras dvs mantissan skiftas
                         tills den mest signifikanta biten står
                         i position 1.
105         CALL FSUB    y(t) minskas med referensvärdet i fly-
106         DC BÖRV      tande räkning.
107         CALL FSTO    y(t) lagras i YVAR.
108         DC YVAR
109         CALL FLD     x(t-1) till FAC
10A         DC XVAR
```

```
10B          CALL FADD   x(t):= x(t-1) + y(t).
10C          DC YVAR
10D          CALL FSTO   x(t) lagras undan för nästa genomlöpan-
10E          DC XVAR     de av algoritmen.
10F          CALL FMPY   x(t) multipliceras med -A flytande.
110          DC MINA
111          CALL FSTO
112          DC SLASK
113          CALL FLD
114          DC YVAR
115          CALL FMPY   y(t) multipliceras med -B flytande.
116          DC MINB
117          CALL FADD   x(t) -A + y(t) -B till FAC.
118          DC SLASK
119          CALL IFIX   Anropet resulterar i att reella talet i
                         FAC övergår i heltal i det vanliga A-reg.
11A          STO UVAR    u(t) lagras i cell UVAR.
11B          XIO DAO
11C  BÖRV,   DC
11D          DC
11E MINA,    DC
11F          DC
120  MINB,   DC
121          DC
122  YVAR,   BSS 2
124  XVAR,   BSS 2
126  UVAR,   BSS 2
128  SLASK,  BSS 2
12A  WRITE,
12C  READ,
12E  DAO,
130          END
```

## III.3.3 PROGRAMMET I FIX RÄKNING DUBBEL PRECISION.

Tal i dubbel precision lagras i två konsekutiva celler. Som ackumulator användes A-reg. och dess utvidgning Q-reg. Add., sub., lagring och upphämtning i dubbel precision utföres med maskinvara.
För multiplikation i dubbel precision anropas subrutin XMD varvid multiplikanden ska vara i A och Q och multiplikatorn i FAC (Se tidigare.). Resultatet hamnar i Aoch Q. /11/,/12/,/13/.

```
            ORG 100
100  START, XIO WRITE
101         XIO READ
102         LD NOLL      0:: till A-reg.
103         STO YVAR+1   Lagra i cellen efter YVAR.
104         LDD YVAR     y(t) till A och Q i dubbel precision.
105         SD BÖRV      y(t):= y(t) - referens      -"-
106         STD YVAR     Lagra y(t)                  -"-
107         AD XVAR      x(t):= y(t) + x(t-1)        -"-
108         STD XVAR
109         LDX 3 8180   Ladda XR 3 med talet 8180 dvs adressen
                         till den näst sista cellen i ett 8K
                         kärnminne. (FAC)
10A         LDD XVAR
10B         STD I 3      Lagra i dubbel precision i den cell
                         vars adress står i XR 3 dvs FAC. I be-
                         tecknar tvåordsinstruktion med indi-
                         rekt adressering.
10D         LDD MINA
10E         CALL XMD     Anropa subrutin XMD.
10F         STD SLASK
110         LDD XVAR
111         STD I 3      Lagra indirekt i cell 8180.:(Se ovan.)
113         LDD MINB
114         CALL XMD
115         AD SLASK
```

```
116          LDX 2 16
117          SLC 2        Skifta A och Q 16 steg vänster.
118          STO UVAR
119          XIO DAO
11A  BÖRV,   DC
11B          DC
11C  MINA,   DC
11D          DC
11E  MINB,   DC
11F          DC
120  YVAR,   BSS 2
122  XVAR,   BSS 2
124  UVAR,   BSS 2
126  SLASK,  BSS 2
128  NOLL,   DC
129  WRITE,
12B  READ,
12D  DAO,
12F          END
```

## III.4 GE-PAC 4020.

Ge-Pac 4020 systemets register visas i blockdiagrammet på nästa sida.
Ackumulatorregistret (A) har liksom de tidigare maskinerna en förlängning i Q-registret bestående av 24 bitar plus teckenbit. Overflow i A-reg. markeras av en flip-flop (OVFR), som sätts.
Till 4020 systemet hör två olika A/D omvandlare, en integrerande och en som arbetar med succesiva approximationer. Tillsammans med dessa arbetar en scanner kontroll, kanalväljare och hög- och lågnivåförstärkare. Alltsammans går under namnet analog input scanner (AIS).
In- utorganen arbetar oberoende av varandra dvs exempelvis analoga signaler scannas samtidigt som analoga signaler distribueras. Förstnämnda A/D omvandlare ger 16 bitar plus teckenbit och spillbit, den senare ger 12 bitar plus teckenbit och spillbit.
Multiple output cotrol (MOC) tillåter analoga utsignaler i grupper om två (8 bitar) eller en (10 bitar). I programmet användes de in- utorgan, som ger största noggranhet.
4020 systemet med integrerande kretsar är det snabbaste i Ge-Pac familjen av maskiner. /14/,/15/.

### III.4.1 PROGRAMMET I FIX FÄKNING ENKEL PRECISION. /16/,/17/.

```
            ORG / 200
200  START, LDA SCWH    Kontrollordet, som väljer analog input
                        scanner (AIS) och analog ingång, över-
                        föres till A-reg.
201         OUT AIS     Kontrollordet överföres till Scan Com-
                        mand Register. Scann operation initieras
                        och A/D omvandlingen startar. När opera-
                        tionen är slutförd sätts en flagga.
202         JNR AIS     Hoppa två steg om flaggan ej satt, gå
                        annars till nästa instruktion.
203         BRU  +2     Hoppa ovillkorligen två steg framåt
204         BRU  -2                -"-              bakåt.
```

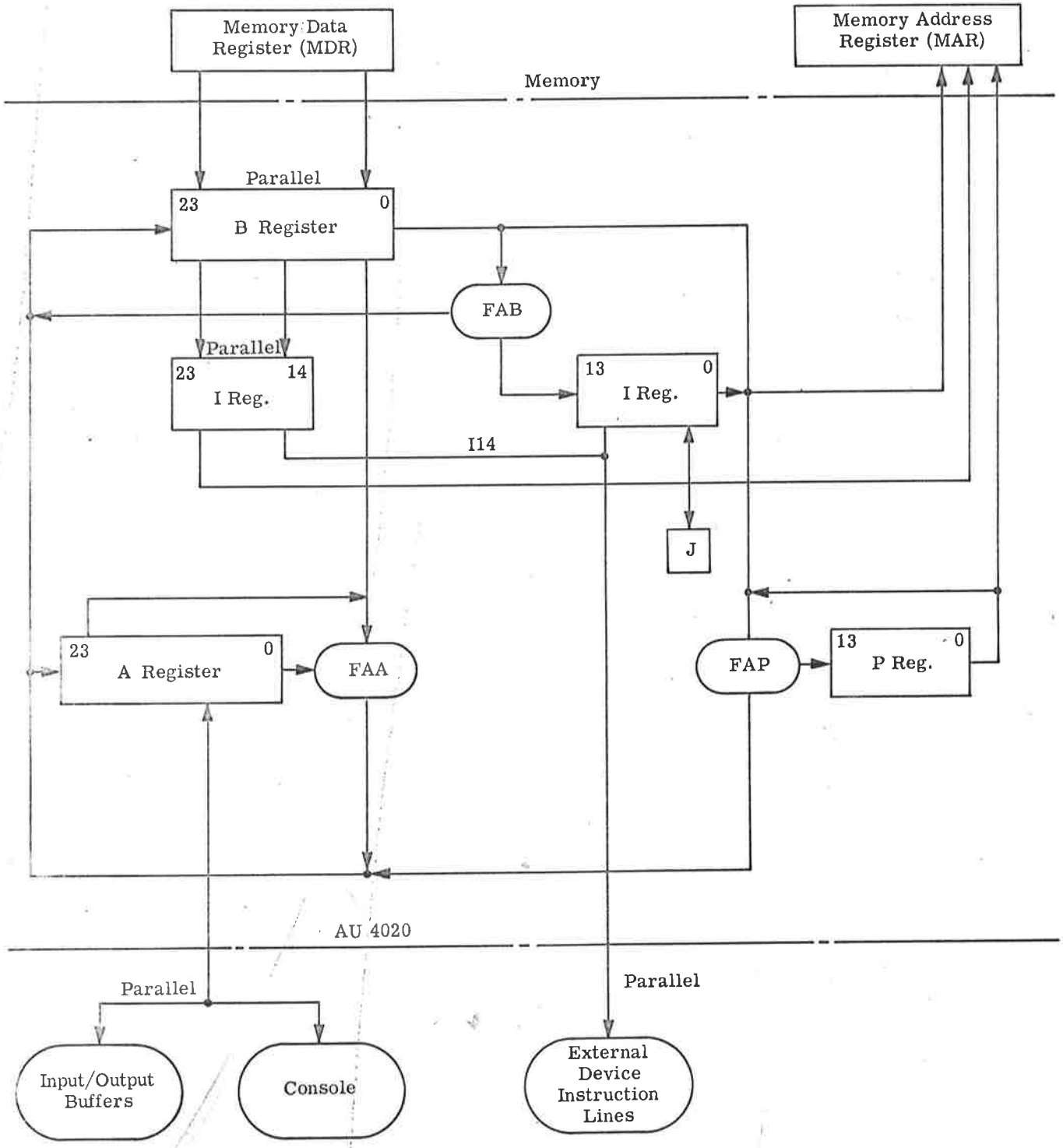


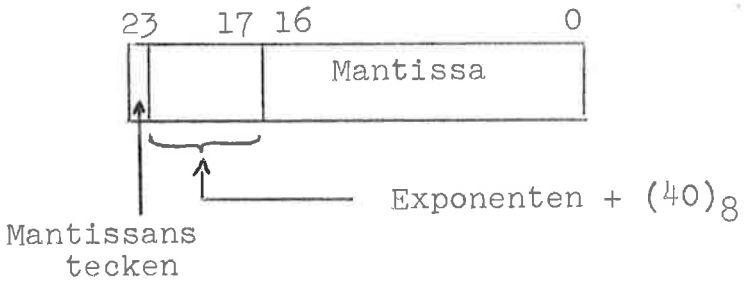

Figure ... . Simplified Block Diagram of AU 4020

| | | |
|---|---|---|
| 205 | IN AIS | Digitala värdet överförs från Scanner Converter Register till A-reg. bitarna 23-6. (Bitarna numreras här från höger till vänster.) |
| 206 | SRA 6 | Skifta A-reg 6 steg höger så att informationen hamnar så lång åt höger som möjligt. |
| 207 | SUB BÖRV | y(t):= y(t) - referensvärdet |
| 210 | STA YVAR | Lagra y(t). A-reg. ändras inte. |
| 211 | ADD XVAR | x(t):= x(t-1) + y(t). |
| 212 | STA XVAR | |
| 213 | MAQ | C(A) överföres till Q-reg. 0 till A-reg. |
| 214 | MPY MINA | Konstanten -A multipliceras med C(Q). Resultatet hamnar i Aoch Q med 48 bitar. |
| 215 | STQ SLASK | C(Q) lagras i cell SLASK. |
| 216 | LDQ YVAR | y(t) till Q-reg. |
| 217 | MPY MINB | |
| 220 | SLA 30 | Skifta 24 steg vänster, dvs C(Q) ersätter C(A). |
| 221 | ADD SLASK | u(t):= -A x(t) + -B y(t). |
| 222 | SLA 14 | Skifta 12 steg vänster. u(t) ligger nu i de 10 bitar, som D/A omvandlaren tar hand om dvs bitarna 23-12. |
| 223 | JNO | Hoppa över nästa instruktion om overflow vippan inte är satt. Om overflow gå till nästa instruktion. |
| 224 | BRU +2 | Hoppa ovillkorligen två steg framåt. |
| 225 | BRU UT | Hoppa ovillkorligen till cell UT. |
| 226 | TOR 27 | Testa om bit 23 i A-reg. är noll eller ett.(dvs om C(A) är pos. eller neg.) Om biten är en etta sätt Test Flip-Flop (TSTF) annars ej. |
| 227 | BTS +3 | Hoppa tre steg om TSTF är satt. |
| 230 | LDA MAX | Vi har alltså konstaterat overflow och att talet i A-reg. är pos. Ladda A-reg. med det största tal, som kan representeras med bitarna 23-12. |
| 231 | BRU UT | Hoppa ut. |

```
232             LDA MIN     Ladda med det minsta negativa tal, som
                            kan representeras med bitarna 23-12.
233  UT,        ADD MODO    Addera MOD-ord,(Multiple Output Distri-
                            buter) i bitarna 8-0 innehåller ut-
                            funktioner och andra krav i samband
                            med utmatningen.
234             OUT MOD     A-reg. överföres till Multiple Output
                            Controller (MOC) Command Register, u(t)
                            överföres till D/A omvandlaren och om-
                            vandlingen startar.
235  MODO       CON
236  SCWH       CON
237  BÖRV       CON         Konstanter.
240  MINA       CON
241  MINB       CON
242  YVAR       BSS 1
243  XVAR       BSS 1       Reserverat block i minnet.
244  SLASK      BSS 1
245  AIS        EQL/2100    Anvisar ett värde till en symbol under
246  MOD        EQL/4200    assembleringen.
247             END
```

III.4.2 PROGRAMMET I FLYTANDE RÄKNING.

Till skillnad från de föregående maskinerna har Ge-Pac 4020 inbyggd maskinvaruutrustning för flytande räkning. Enkelord i flytande räkning representeras i maskinen enligt figur:



```
                ORG / 200
200  START,     LDA SCWH
201             OUT AIS
202             JNR AIS
203             BRU  +2
204             BRU  -2
205             IN AIS
206             SRA 6
207             FLO 23      C(A) omvandlas från fixt tal med skal-
                            faktor 23 till flytande normaliserat tal.
210             FSU BÖRV    y(t):= y(t) - referens i flytande räkn.
211             STA YVAR    y(t) lagras. A-reg. oförändrat.
212             FAD XVAR    Addera x(t) i flytande räkning.
213             STA XVAR
214             FMP MINA    Flytande multiplikation.
215             STA SLASK
216             LDA YVAR
217             FMP MINB
220             FAD SLASK
221             FIX 23      Omvandla till fixt tal igen.
222             SLA 14      Skifta 12 steg vänster. (Se fixt progr.)
223             ADD MODO
224             OUT MOD
225  MODO       ØON
226  SCWH       CON
227  BÖRV       CON
230  MINA       CON
231  MINB       CON
232  YVAR       BSS 1
233  XVAR       BSS 1
234  SLASK      BSS 1
235  AIS        EQL/2100
236  MOD        EQL/4200
237             END
```

III.4.3 PROGRAMMET I FIX RÄKNING DUBBEL PRECISION.

Ett tal i dubbel ordlängd lagras i två konsekutiva celler. Addition , subtraktion, lagring och upphämyning av tal i dubbel precision sker med maskinvara. A-reg. och Q-reg. användes som det 48 bits register som krävs.
Multiplikation i dubbel precision har inte använts, som i de tidigare maskinerna. Med de 48 bitarna fås full jämförbar noggrannhet. /16/,/17/.

```
            ORG / 200
200  START, LDA SCWH
201         OUT AIS
202         JNR AIS
203         BRU  +2
204         BRU  -2
205         IN AIS
206         SRA 6
207         LDQ NOLL     0 till Q-reg.
210         DSU BÖRV     y(t):= y(t) - ref. i dubbel precision.
211         DST YVAR     Lagring                    -"-
212         DAD XVAR
213         DST XVAR
214         LDQ XVAR
215         LDZ          0 till A-reg.
216         MPY MINA
217         DST SLASK    Lagra A- och Q-reg.
220         LDQ YVAR
221         LDZ
222         MPY MINB
223         DAD SLASK    Dubbelords addition.
224         DDA 44       Skifta A och Q 36 steg vänster.
225         ADD MODO
226         OUT MOD
227-                     Samma celler som tidigare, dock i dubbel
  246                    precision.
247         END
```

## IV. SLUTSATS.

Eftesom totala antalet bitar i maskinerna är väsentligt måste 4K såsom i PDP-8 utgöra en klar begränsning. Program, som överstiger detta utrymme är inte ovanliga. Någon monitor får inte heller plats på 4K. IBM-1800 och Ge-PAC 4020 har däremot väl utvecklade monitorsystem.
PDP-8:s enbart 12 bitar ger besvär vid adressering. 128 celler kan nås med direkt adressering medan motsvarande för IBM-1800 är 256 celler.
De 24 bitar, som finns i Ge-Pac 4020 är bekväma att använda men nödvändigheten kan kanske diskuteras.
Operationerna på PDP-8 är mycket fundamentala och står mycket nära registerna till skillnad från IBM-1800, som har färre grundinstruktioner, vilka finns i omfattande modifieringar. Programmeringen av den senare blir härigenom mera lätthanterlig och kan göras på mindre utrymme. Detsamma gäller även Ge-Pac 4020, som med sin väl utbyggda maskinvaruutrustning tillåter förenklad programmering.
Någon anledning till att PDP-8 saknar upphämtningsinstruktion har inte hittats. Man har löst detta genom en additionsinstruktion sedan ackumulatorn automatiskt nollställes vid lagring.
Hur programmeringen av A/D omvandlingen genomföres på de olika maskinerna finns angivet i kommentarerna till programmen.
Vad minnesutrymmet beträffar kan programmet för IBM-1800 och Ge-Pac 4020 sägas kräva ungefär lika mycket medan PDP-8 konsekvent kräver större utrymme . Detta delvis beroende på den mindre maskinvaruutrustningen.(Se tabell 3.)
Exekveringstiderna för Ge- Pac är i allmänhet kortast, om man undantar fixräkning där den största tiden upptas av A/D omvandlingen.(Se tabell 1 och 4.)
Tidsfaktorn för dubbelordsaritmetiken i förhållande till flytande räkning är genomgående en faktor 10 till förmån för den förstnämnda. För maskinerna, som medger större noggrannhet kan detta vara en tidsvinst, som samtidigt ger god noggrannhet.
Ovanstående program gör inga anspråk på att vara optimala lösningar av programmeringsproblemet förknippat med styralgoritmen. Avsikten har enbart varit att fullgöra uppgiften på ett så likartat sätt som möjligt för de olika maskinerna och därigenom få fram ett jämförelsematerial.

Tilläggas kan att skalning intuitivt vore rimlig att genomföra i vissa operationssteg, men eftersom den analoga ingången och utgången måste vara synkroniserade vad gäller antalet skift är skalning inte meningsfull.
Hänsyn till att u(t) kan bli för stor för att rymmas i A-registret,har endast tagits i programmen för fix räkning enkel precision.
För direkta sifferjämförelser hänvisas till tabellerna i slutet av uppsatsen.

TABELL 1: Omvandlingstider för A/D omvandlare (μs).

| Antal bitar | | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 15 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PDP-8 | Typ 189 | 6 | 13 | 20 | 24 | 27 | 45 | 55 | | | |
| | Typ 138E | 9 | 10,5 | 12 | 13,5 | 17 | 25 | 35 | | | |
| IBM-1800 | Modell 1 | | | | 29 | | | 36 | | 44 | |
| | Modell | | | | 29 | | | 36 | | 44 | |
| Ge-Pac 4020 | Succ. approx. | | | | | | | | 700 | | |
| | Integr. omv. | | | | | | | | | | 20-40 ms |

( Teckenbiten medräknad.)

TABELL 2: Arbetsutrymmen för monitor och kompilator.

| | PDP-8 | IBM-1800 | GE-PAC |
|---|---|---|---|
| FORTRAN kompilator | ~ 2K | minst 8K | ~2K (F67) |
| Monitorsyem | saknas | 8K | 8K |

TABELL 3: Minnesutrymmen.

| PROGRAM I: | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| MASKIN | | PDP-8 | | IBM-1800 | | GE-PAC 4020 | | | |
| FIX RÄKNING | Antal instruk. att skriva | 102 | | 27 | | 30 | | | |
| | Antal minnes-celler | 113 | | 47 | | 40 | | | |
| | Antal slask-celler | 19 | | 15 | | 10 | | | |
| FLYTANDE RÄKNING | Antalet instr. att skriva | 48 | | 18 | | 22 | | | |
| | Antal minnes-celler | 705 (inkl. subr) | | 350 (inkl. subr) | | 327 | | | |
| | Antal slask-celler | 21 (exkl. subr) | | 30 | | 10 | | | |
| DUBBEL PRECISION | Antal instruk. att skriva | 30 | | 25 | | 24 | | | |
| | Antal minnes-celler | 175 (inkl. subr) | | 47 | | 40 | | | |
| | Antal slask-celler | 16 (inkl. subr) | | 20 | | 17 | | | |

TABELL 4: Exekveringstider (medeltal).

| | PDP-8 | IBM-1800 | | GE-PAC 4020 |
|---|---|---|---|---|
| Cykel tid | 1,5µs | 2µs | 4µs | 1,6µs |
| Program i fix räkning | 310µs | 170 µs | 290µs | 828 µs |
| Program i flyt. räkn. | ? | - | 4,25ms | 1,92ms |
| Program i dubbel prec. | 2,92ms | - | 1,36ms | 1,03 ms (ej subr) |
| Uppskattad programmerings-(förhållande) | 10 | 5 | | 4 |

TABELL 5: Minnesutrymme och exekveringstider för några vanliga subrutiner hos de olika maskinerna.

| Program karaktär | PDP-8 | | IBM-1800 | | GE-PAC 4020 | |
|---|---|---|---|---|---|---|
| | Tid (µs) | Ut-rymme | Tid (µs) | Ut-rymme | Tid (µs) | Ut-rymme |
| Flytande addition | 405 | 625 | 460 | 102 | 208 | H |
| Flytande subtraktion | ? | 625 | 560 | 102 | 208 | H |
| Flytande multiplikation | 530 | 625 | 560 | 152 | 151 | H |
| Flytande division | 590 | 625 | 760 | 86 | 182 | H |
| Flytande lagring | ? | 625 | 180 | 54 | 3,2 | H |
| Flytande upphämtning | ? | 625 | 180 | 54 | 3,2 | H |
| Normalisering | ? | 625 | 260 | 42 | - | - |
| Mult. i dubbel prec. | 1,4ms | 125 | 520 | 66 | ? | ? |
| Div. i dubbel prec. | 1,65ms | 105 | 1,76ms | 74 | ? | ? |
| Fixt till flytande | - | - | 330 | 10 | 123 | H |
| Flytande till fixt | - | - | 140 | 40 | 111 | H |

H betecknar att motsvarande funktion finns i maskinvara.

## TABLE 6 SOFTWARE: STATUS AND AVAILABILITY FOR

| | Computer system | Executive program | Availability | Assembler | Availability | Compiler off-line | Availability |
|---|---|---|---|---|---|---|---|
| 1. | Control Data Corp. CDC 1700 | Major real-time, on-line executive package and associated control programming is called the Control Operating System. Complete system available as noted. | 1/67 | Offers a Macro-Assembler (MA) as part of executive systems. A Utility Assembler (UA) is available for stand-alone operation. | MA 4th quarter 1966 UA Present | ASA FORTRAN II with byte handling but without complex or double precision arithmetic, stand alone operation. | 8/66 |
| 2. | Foxboro Co. PCP 88 (PDP 8) | Supervisory member of a multiple computer system contains equivalent of an executive program. Provides all generally included functions. | 8/66 | Off-line operation only Symbolic Assembler. | Present | Operates in 4K core system. | Present |
| 3. | General Electric Co. All GE/PAC 4000's Less 4020 | GE's process control executive package is labelled MONITOR. 4050-II and 4060 include a Free-Time System (FTS). | Present | Off-line assembler labelled as PAL (Process Assembler Language). | Present | Scientific FORTRAN II available for off-line use. | Present |
| 4. | General Electric Co. GE/PAC 4020 | MONITOR for supervisory system use available 11/66. Special adaptation for DDC available 12/66. | Standard 11/66 Adapter 12/66 | Process Assembler Language (PAL). | 12/66 | Process FORTRAN | 2/67 |
| 5. | Honeywell Inc. H 20 | Title of "Scheduler" preferred by Honeywell. | Present | Off-line assembler available now. On-line assembler as part of new process package. | Off-line Present On-line 9/66 | FORTRAN IV without complex and double precision arithmetic and data statements. | Present |
| 6. | International Business Machines IBM 1800 | Process control executive package is called the TSX or Time-Sharing Executive System. | Phase I 7/66 Phase II 11/66 | An assembler is available in either executive system. | Off-line 7/66 On-line 11/66 | Available with phase I. | Off-line 7/66 |
| 7. | Systems Engineering Lab. SEL 800 | Executive system is known as MONITOR. | Present | Mnemonic Macro-Assembler available for off-line use. | Present | FORTRAN IV | Present |
| 8. | Westinghouse Electric Corp. PRODAC p 50 | Extensive executive program available. | Present | WESAP (Westinghouse Assembly Program) operates off-line on a simulator. | Present | Available as part of simulator program. | Present |

Referens /3/.

# THIRD GENERATION PROCESS COMPUTER HARDWARE

| Compiler on-line | Availability | Debugging routines on-line | Availability | Simulator | Availability | Standardized DDC algorithms | Availability |
|---|---|---|---|---|---|---|---|
| Available only with the Control Operating System. | 1/67 | On-line debugging package available as part of Control Operating System, includes trace routines. | 4th quarter 1966 | A 1700 Simulator or Emulator will operate on all 3000 class CDC machines. | Present | Runs as part of the Control Operating System. | 8/68 |
| On supervisory machine only. Requires 12K core plus disc file. Produces interpretive code FORTRAN II. | 9/66 | Not available at present. | — | Simulation of PCP 88 on PDP 4 available at Foxboro | Present | Extensive routines including cascade, feedforward, ratio, noninteracting in addition to regular three-mode | 8/66 |
| Part of Free-Time System for 4050-II and 4060. Available also for 4040. | Present | Available as parts of Free Time System (FTS) for 4050-II and 4060. Available to limited extent for 4040. | Present | Information not available at present. | — | Available for special purpose or limited number of points. | Present |
| Process FORTRAN—part of Free Time System (FTS). | 2/67 | Part of the Free Time System. (FTS). | 2/67 | Information not available at present. | — | Extensive package with special algorithms. | 12/66 |
| Not available. | — | Includes trace and utility features. | 9/66 | Not available. | — | Not available at present. | — |
| Available only in the TSX system, phase II. | On-line 11/66 | On-line diagnostics and debugging available with the TSX system. | 11/66 | No public announcement. | — | No public announcement. | — |
| None | — | DIAG group of diagnostic routines which operates off-line only. | Present | None | — | None | — |
| None | — | Available on simulator only. | Present | PRODAC 580 operating system. | Present | Very extensive program packaging available. | Present |

REFERENSLISTA.

1. Programming languages for industriel process control; James D. Schoeffler m.fl.
2. Datamation febr. 1966; Schoeffler.
3. Control Engineering sept. 1966; T. J. Williams.
4. Andoff-Bodin; Examensarbete i regleringsteori,1967.
5. IBM 1800 Time-Sharing Executive System Specifications, 1965.
6. SINTEF 67-55-C Process-ALGOL; O. Hösöen och T. Matre, 1967.
7. SINTEF 67-38-C Generaliserat administrationsprogra i PAL; I. Westhagen, T. Endresen och O. Hösöien, 1967.
8. Small Coputer Handbook; (PDP-8,1967).
9. Floating-point System Programming Manuel; 8-5-S (1965).
10. Digital 8-13-F, Double-Precision Multiply Subroutine-Signed,(1965).
11. Data Acquisition and Control System Referens Manual, Form A26-5918-2.
12. Assembler Language, Form C26-5882-2,(1964).
13. Subrutin Library, Form C26-5880-3,(1966).
14. Systems Manual, GET-3201A,(1965).
15. Systems Manual, GET-3460A,(1967).
16. Programmimg Manual, PCP-102B, (1965).
17. Process Assembler Language, YPG 12M, (1965).
18. Datamaskiner och databehandling; Olle Dopping,1966.
19. IFIP-ICC Vocabulary of Information processing,1966.

## ORDFÖRKLARINGAR.

assemblerkod—— symbolspråk

block —— en mängd information, som i något avseende utgör en enhet.

carry —— minnesetta

cycle stealing —— tiduppdelad in- och utmatning

cykel tid —— tidsintervallet mellan starterna av succesiva läs-skriv cykler

DDC —— Direct Digital Control

dump routine —— program, som lagrar undan viss information, som hotas att överlappas

interpretativ kod —— kod, som tolkas till vanlig maskinkod av ett särskilt interpreterande program

interrupt program —— betjäningsprogram (IBM)

mainline program —— processprogram (IBM)

minnesskydd —— omöjliggör överlappning av program och förbjudna zoner i minnet

multiplexer —— enhet, som växlar mellan olika analoga insignaler, vilka enbart tar en A/D omvandlare i anspråk

multiprogrammering —— maskintiden delas mellan oberoende program av vilka inget kontinuerligt lägger beslag på den aritmetiska enheten.

nonprocess program —— icke-processbundna program

objekt program —— maskinprogram

off-line —— bredvid linjen; exempelvis kommunikation mellan maskin och process med manuell hjälp

omvänd polsk form —— adresseringsätt, en operator appliceras på de senast nämnda operanderna

on-line —— på linjen; dataöverföring utan manuell insatts.

overflow —— spill

post —— (eng. record) data, som relateras till en speciell del av ett job.

reell-tids programmering —— omedelbar bearbetning när transaktionen kommit in i datamaskinen

scanner —— avsökare

stack —— a) hög av program, b) en serie register eller minnesceller

System skeleton —— kärnminnesutrymme, som innehåller vissa program och arbetsarea (IBM)

Teletype —— skrivmaskin

Time-Sharing —— tid-delning; möjlighet att utföra icke-processbundna program när centralenheten inte är upptagen av processprogram

variable area —— kärnminnesutrymme för core loads (IBM)

( I övrigt hänvisas till referenserna /18/ och /19/.)