Övningar och laborationer i Datorer i Reglersystem

Rundqwist, Lars

1988

Document Version:
Förlagets slutgiltiga version

CODEN: LUTFD2/(TFRT-7386)/1-72/(1988)

Övningar och laborationer i Datorer i Reglersystem

Lars Rundqwist

Institutionen för Reglerteknik
Lunds Tekniska Högskola
April 1988

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Internal report

Date of issue
April 1988

Document Number
CODEN: LUTFD2/(TFRT–7386)/1-72/(1988)

Author(s)
Lars Rundqwist

Supervisor

Sponsoring organisation

Title and subtitle
Övningar och laborationer i Datorer i Reglersystem (Exercises and laboratories in Computers in Control Systems)

Abstract

This report documents the exercises and laboratories given in the course Computers in Control Systems (Datorer i Reglersystem) during the spring semester 1987. The main reason for developing new laboratories were a change from LSI-11 to IBM PC/AT computers.

The laboratories are organized as follows. Laboratory 1: Sequence control with a commercial PLC system, with an IBM PC interface. Laboratory 2: A simple control system with operator communication on IBM PC/AT, implemented with Modula-2 and a real time kernel. Laboratory 3: Digital control of a DC servo, using the control program TOOLBOX on IBM PC/AT. Laboratory 4: Commercial process computers, the same control tasks as in laboratories 1–2 are programmed in an Asea Master process computer. Both programming and operator interfaces use IBM PC/AT computers.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
72

Recipient's notes

Security classification

1. Övningar och laborationer

Denna rapport dokumenterar de övningar och laborationer som genomfördes i kursen Datorer i Reglersystem under vårterminen 1987. Övningarna utgör förberedelser till laborationerna. De organiseras på följande sätt.

Laboration 1 och övning 1: Sekvensstyrning med PLC-system; genomförs med tekokarprocessen och ett kommersiellt PLC-system från Mitsubishi. Programmering och operatörskommunikation sker via IBM PC/AT.

Laboration 2 och övning 2: Ett enkelt reglersystem; nivåreglering av nivån i övre tanken med en IBM PC/AT. Reglering och operatörskommunikation implementeras i Modula-2 mha institutionens realtidskärna.

Laboration 3 och övning 3–5: Digital reglering; ett DC-servo regleras med programmet TOOLBOX på en IBM PC/AT. På övningar genomgång av sampling, approximation av PID-regulatorer samt direkt laborationsförberedelse.

Laboration 4 och övning 6–7: Kommersiella processdatorer; samma uppgifter som i laboration 1 och 2, dvs sekvensstyrning och nivåreglering, genomförs med ett Asea Master-system. Konfigurering och operatörskommunikation sker via IBM PC/AT.

Grundmotivet för att ta fram de 4 laborationerna var att institutionens LSI-11-datorer hade pensionerats. Därför har jag på enklast möjliga sätt gjort nya laborationer genom att kopiera och omarbeta andra laborationer. Endast laboration 4 kan sägas vara ny. Laboration 1 har tidigare varit demonstration, men genomförs nu av teknologerna. Laboration 2 är en förenklad variant av en laboration från kursen Tillämpad realtidsprogrammering. Laboration 3 är en variant av en laboration i kursen Digital reglering. Laboration 4 ersätter dels en laboration på DDC-paketet på LSI-11 samt en demonstration av ett NAF-Unic-system.

Följdaktligen har jag stulit/lånat material från andra på institutionen, samt erhållit hjälp från diverse håll. Rolf Braun och Beijer Electronics, Malmö, var till stor hjälp vid laboration 1. Leif Andersson har implementerat realtidskärnan samt skrivit dokumentation om den och hanteringen av IBM PC-maskinerna. Michael Lundh har skrivit programmet TOOLBOX samt utvecklat DC-servo-laborationen. Asea AB och Rejlers Ingenjörsbyrå, båda i Lund, gav stort bistånd under framtagandet av laboration 4. Laborationen förutsätter tillgång till manualer om Aseas Master-system.

Sekvensstyrning med PLC-system

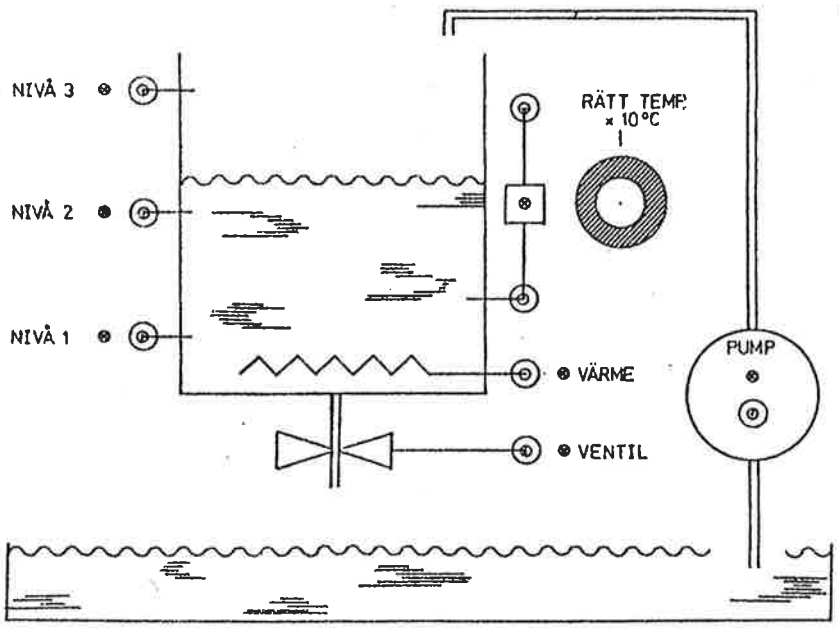
1. Inledning

Start- och stoppförlopp samt alarmövervakning är ofta förekommande inom alla industrigrenar. Denna typ av styrning har tidigare gjorts med hjälp av reläer. Det har inneburit att det har varit svårt att göra förändringar i förloppen. Vidare har utrustningen i regel levererats av andra firmor än de som har levererat den övriga reglerutrustningen. Sedan ett antal år är styrsystemen i regel uppbyggda med mikroprocessorer istället. Det innebär att de är programmerbara och att programmet också kan ändras utan större ingrepp. Sådan utrustning kallas i regel för PLC-system (Programmable Logical Control) eller PC-system.

Utvecklingen har nu kommit så långt att den analoga regleringen och logikstyrningen ofta görs i samma utrustning. Med PLC-systemen har man ofta möjlighet att implementera enkel reglering. Å andra sidan kan logiksystem ibland också byggas upp inom DDC-paket (Direct Digital Control). DDC-paketen har närmat sig logikstyrningen från reglersidan medan PLC-systemen har utvecklats ur relästyrning. Dessa två angreppssätt har också i många fall präglat den grundläggande filosofin för hur man implementerar logik- och sekvensstyrning.

Avsikten med Övning 1 och Laboration 1 är att illustrera hur man bygger upp logik- och sekvensstyrning för en enkel process. Laborationen utförs med hjälp av ett kommersiellt PLC-system av typ MELSEC F_2-40 från Mitsubishi. PLC-systemet kan programmeras och övervakas från ett hjälpprogram på IBM PC. Processen som skall styras är en tankprocess med ett värmeelement.

När man beskriver funktionen hos ett styrsystem kan man använda Booleska uttryck. I industriella sammanhang använder man ofta en grafisk beskrivning istället. Det finns flera sätt att illustrera ett logiskt (kombinatoriskt) nät. Ett sätt är med hjälp av reläsymboler, vilket är vanligt i amerikansk litteratur. Europeisk litteratur använder i regel logikscheman. Många PLC-system kan numera programmeras grafiskt med reläscheman eller logikscheman. I en del utrustningar är det möjligt att presentera ett logiskt nät på bägge sätten. I andra fall kan man även använda Booleska uttryck eller motsvarande maskininstruktioner när man programmerar.



Figur 1. Processchema.

2. Processen

Processen som skall styras är en vattentank med ett värmeelement. Logiska signaler anger vattennivån. En pump styr inflödet och en ventil styr utflödet. Ett processchema med definierade variabler finns i figur 1.

Enkel beskrivning av förloppet

Den enklaste sekvensen som PLC-systemet skall klara är följande. Efter att ha startat med tom tank, dvs nivå 1 (N1), skall tanken först fyllas till nivå 2 (N2) genom att utloppsventilen (VENTIL) stängs och pumpen (PUMP) startas. När nivå N2 är uppnådd skall uppvärmningen (VÄRME) startas. När önskad temperatur är uppnådd fås indikering genom mätsignalen RÄTT TEMP (kallas TEMP i fortsättningen). Därefter skall värmen stängas av och tanken fyllas till nivå 3 (N3). När N3 är uppnådd skall tanken tömmas.

Denna sekvens kan sedan utökas med fler villkor i mån av tid på laborationen.

Detaljer

1. Sekvensen startas efter kommando av operatören. Detta sker genom att en insignal sätts sann (hög).

2. Pumpen startas, ventilen öppnar och värmen slås till då styrsystemet läser ut en sann (hög) signal till respektive utgång.

3. Indikering av nivåerna N1–N3 och temperaturen T genom att respektive logisk signal blir sann.

3. Logik- och sekvensstyrning av processen

I ett logiskt (eller kombinatoriskt) nät bestäms utsignalerna direkt av insignalerna Nätet kan beskrivas med hjälp av de logiska operatorerna NOT (inte), AND (och) och OR (eller). NOT representeras i Boolesk algebra av ett streck över variabeln, $\overline{a}$, och AND och OR mellan två variabler av $a \cdot b$ respektive $a+b$. Prioritetsordningen är att alla inte-operationer utvärderas först, sedan och-operationer och till sist eller-operationer. Parenteser kan användas på vanligt sätt. Exempel på ett Booleskt uttryck är

$$x = \overline{a} + b \cdot c$$

Sekvensnät med två tillstånd

Förutom den kombinatoriska delen behöver vi kunna programmera sekvenser. För att göra detta krävs minnesfunktioner (tillstånd). Tillstånden kallas s_1 och s_2. Signalerna k_1 och k_2 är kvitteringar på att målet med respektive tillstånd är uppnått. Om systemet befinner sig i tillstånd s_1 skall det stanna kvar där tills kvitteringen k_1 uppfylls (självhållning), och om man skall gå till s_1 om k_2 uppfylls i s_2 (initialisering). Det betyder att vi får följande villkor

$$s_1 = s_2 \cdot k_2 + s_1 \cdot \overline{k_1}$$

För s_2 blir motsvarande uttryck

$$s_2 = s_1 \cdot k_1 + s_2 \cdot \overline{k_2}$$

Om systemet är uppbyggt så att s_1 direkt blir falsk då k_1 blir sann kommer inte heller s_2 att bli sann eftersom s_1 är falsk då s_2 utvärderas. Ett sätt att komma ifrån detta är att i stället skriva

$$s_1 = s_2 \cdot k_2 + s_1 \cdot \overline{s_2}$$

$$s_2 = s_1 \cdot k_1 + s_2 \cdot \overline{s_1}$$

Detta kommer visserligen att betyda att både s_1 och s_2 är sanna under en cykel av programmet. Om cykeln är kort kommer det inte att spela någon roll.

Vid fler än två tillstånd skall självhållningen gälla tills efterföljande tillstånd har blivit sant. Den allmänna strukturen för tillståndsövergångar blir då

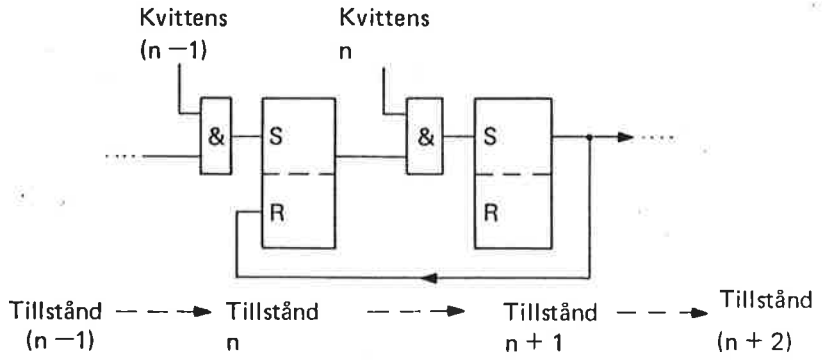
$$s_n = s_{n-1} \cdot k_{n-1} + s_n \cdot \overline{s_{n+1}}$$

För att sekvensnätet skall kunna startas, måste ett tillstånd initialiseras till sant då systemet startas upp.

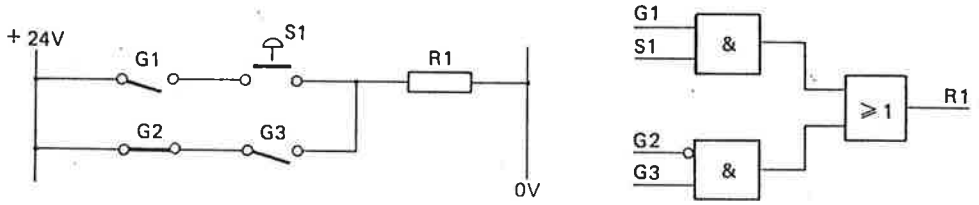
Sekvensstyrning med hjälp av SR-vippor

En SR-vippa är ett minneselement i ett logiksystem. Set-ingången (S) ställer utgången (L) hög, och den förblir hög med hjälp av självhållning (L i parentesen), till dess Reset-ingången (R) blir hög, då L ställs låg. Med algebra beskrivs funktionen på följande sätt.

$$L = (L + S) \cdot \overline{R}$$



Figur 2. Sekvensnät realiserat med SR-vippor.



Figur 3. Reläschema och logikschema.

Med hjälp av SR-vippor kan man programmera sekvensnät på det sätt som visas i figur 2.Nätet skall gå till tillstånd n när L_{n-1} är sann och kvitteringen k_{n-1} kommer. När tillstånd $n+1$ i sin tur blivit sant, skall tillstånd n återställas. Detta ger följande villkor för S_n och R_n.

$$S_n = L_{n-1} \cdot k_{n-1}$$
$$R_n = L_{n+1}$$

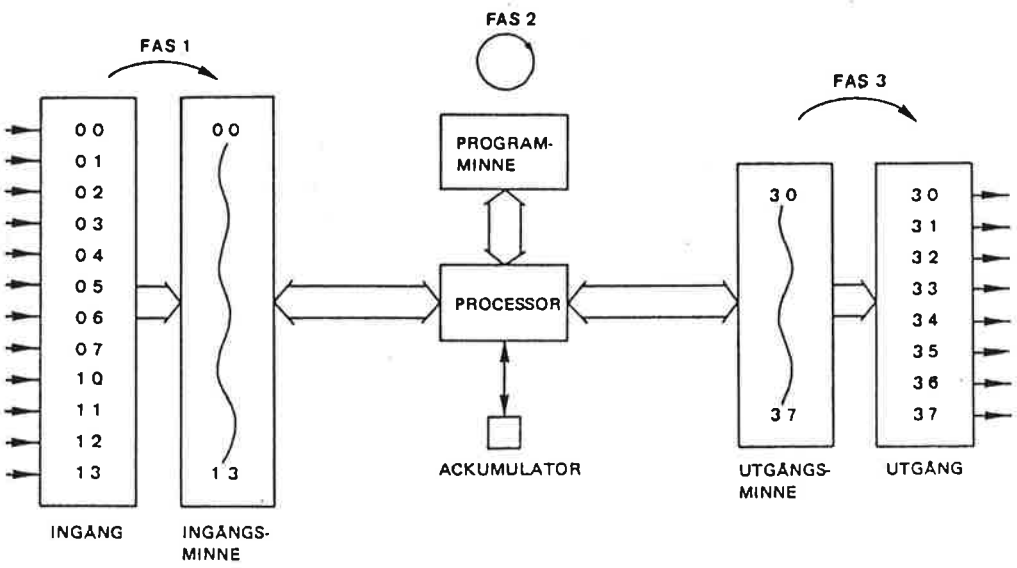
Observera att ett av tillstånden måste initialiseras till ett vid programstart.

Reläprogrammering

Vid reläprogrammering tänker man sig att signalerna som skall behandlas påverkar kontakter. En signal sluter en kontakt när den är sann, öppnar när den är falsk. En inverterad signal gör tvärtom, kontakten är sluten när signalen är falsk, öppen när den är sann, se figur 3., som även visar motsvarande logikschema.

AND-funktionen fås genom seriekoppling av kontakter, och OR-funktionen genom parallellkoppling. Vidare skall resultatet av operationerna påverka ett utgångsrelä, se figur 3. Med 24 Volt kopplat till den vänstra lodräta linjen, och 0 Volt till den högra lodräta linjen, så inser man att reläet drar när villkoren för kontakterna är uppfyllda. Reläet R1 kan i sin tur påverka fler kontakter i ett reläschema. Av konvention ritas alla sådan kontakter för fallet att reläet ej har dragit.

Ett reläschema går ofta under beteckningen Ladder-diagram. Det ritas i regel med stiliserade kontakt- och reläsymboler, se figurer i avsnittet om PLC-systemet.



Figur 4. PLC-systemets struktur.

4. PLC-systemet

I det PLC-system vi skall använda under laborationen, MELSEC F_2-40, sker exekveringen av ett program i tre faser. Först läses alla inkanaler in till ett inkanalminne. Sedan exekveras programmet ett varv, och därefter läses utkanalminnet ut till utgångsreläerna. Förloppet upprepas sedan cykliskt. I processorn finns ett ackumulatorregister, i vilket alla resultat lagras. Alla instruktioner sker mellan ackumulatorn och adresserad kanal. Instruktionsrepertoaren omfattar operationer för att ladda ackumulatorn (tex LD), logiska operationer mellan ackumulator och adresserad kanal (tex AND och OR), samt att lagra ackumulatorns innehåll i utkanal eller minne (OUT). Observera att ackumulatorn ej påverkas av OUT-instruktioner. Vidare finns några specialinstruktioner.

Kanaler

I tabell 1. visas hur kanalerna är fördelade och numrerade. Observera att numreringen är oktal, dvs siffrorna 8 och 9 ingår ej.

För att ange typen av kanal skriver man ofta X framför inkanaler, tex X111, Y framför utkanaler, M framför minneskanaler, T framför fördröjningskanaler, och C framför räknarkanaler. Specialkanalerna 70–77 har bestämda funktioner, och skall inte användas för att lagra resultat i. De tre viktigaste visas i tabell 2.

Specialinstruktioner

Förutom att lagra ackumulatorns innehåll i en kanal med OUT, kan man återställa räknare mm, genom att använda en specialinstruktion.

PLS Då ackumulatorn ändras till 1 genereras en puls i utkanalen. Pulsen varar ett varv i programmet.

Typ	Kanalnummer	Kommentar
Ingångar	400-413 500-513	24 st
Utgångar	430-437 530-537	16 st
Fördröjning	50-57	8 st, utsignal efter 0.1-999 s
Räknare	60-67	8 st, utsignal efter 1-999 inpulser
Special	70-72, 76-77	Se separat tabell
Minnen	100-377	För mellanlagring av resultat OBS! Tabellen ej fullständig

Tabell 1. Kanalnummer i MELSEC F_{2}-40.

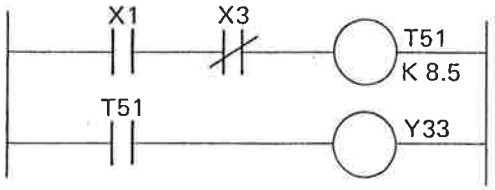
Kanal	Anmärkning
70	Kanalen har värdet 1 när PLC-systemet har startats (RUN). Kan tex användas för yttre RUN-indikering.
71	Kanalen har värdet 1 under första programvarvet efter programstart. Kan användas som initialiseringspuls till räknare och skiftregister.
72	Oscillator med frekvensen 10 Hz, symmetrisk fyrkant. Bör ej kopplas direkt till utgång.

Tabell 2. Specialkanalerna 70-72 i MELSEC F_{2}-40.

RST Återställning av räknare och skiftregister då insignalen är 1. Insignalen till RST genereras ofta med PLS, vilket medför att räknaren inte blockeras med återställningssignalen.

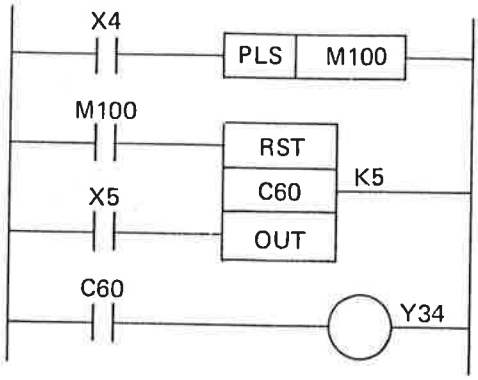
Fördröjningar och räknare

Ett exempel med fördröjning av en signal visas nedan. Tillslaget på kanal Y33 fördröjs 8.5 sekunder efter att X1 blivit sann och X3 falsk.



I nästa exempel visas hur en räknarkrets kan vara uppbyggd. Ingång X4 används för att återställa räknaren med en kort puls från PLS. När sedan 5 signaler har kommit in till räknaren C60 blir utgång Y34 sann. Observera

att räknarna bara räknar då insignalen ändras från falsk till sann.



Start och stopp av programmet

Programmet startas och stoppas med en omkopplare ansluten till RUN- och STOP-ingångar på PLC-n.

5. MEDOC

Med hjälpprogrammet MEDOC kan man programmera och övervaka PLC-systemet från en IBM PC. Programmeringen kan ske antingen genom att använda instruktioner eller genom att rita reläscheman. Den senare metoden kommer att användas i laborationen. MEDOC är i princip självinstruerande och innehåller en rejäl hjälpfunktion. Observera att i MEDOC kallas ett PLC-system för PC!

Benämningar

I MEDOC finns möjligheten att i stället för kanalnummer använda namn på in- och utgångar och minneskanaler. Tex kan man införa namnen `steg1`, `steg2`, etc. på tillstånden i ett sekvensnät. Vidare kan man kalla temperaturindikeringsingången `temp`, pumputgången `pump`, etc. Programmet blir dels mer läsbart, men man behöver heller inte känna till de fysiska kanalerna för in- och utgångar när man skriver programmet.

Menyn

MEDOC är till största delen menystyrt. I vissa fall används funktionstangenterna **F1-F10**. I menyn visas 2 rader. Den övre raden i menyn visar de olika alternativen, den undre visar vilka alternativ man kan välja i steget efter det markerade alternativet. Markören flyttas alltid med piltangenter, och med **ENTER**-tangenten väljer man ett alternativ. Med **Esc**-tangenten backar man upp i menyn. Alternativen på översta nivån är **START**, **EDIT**, **TRANSFER**, **PRINT**, **FILES** och **QUIT**.

Hjälpfunktion

Man kan överallt i programmet få instruktioner genom att trycka på **F1**. Ny sida fås genom **Pg-Down** och man lämnar hjälpfunktionen genom att trycka på **Esc**.

Inmatning av ett program

Följande stycke beskriver vilka val som skall göras i menyerna för att kunna skriva in ett nytt program, samt överföra det till PLC-syctemet.

I översta menyn väljer man först **START**, sedan **NEW_PROJ**. Välj ett unikt projektnamn (filnamn). PC-typen väljs till F2. Välj **EDIT**, och om benämningar önskas sedan **NAME**, fortsätt annars till **LADDER**. Spara benämningarna med **SAVE**. Välj därefter **LADDER** för att kunna skriva in ett reläschema. När programmet är färdigskrivet kan du kontrollera det formellt med **TEST** och spara det på fil med med **SAVE**. Välj sedan **TRANSFER** och **MEDOC>PC** för att föra över programmet till PLC-n.

För att skriva in både benämningar och reläschema flyttar man markören till arbetsfältet med **F2**. Använd hjälpfunktionen (**F1**) för att få fler instruktioner. En stor grön markör i arbetsfältet visar att reläeditering pågår och en brun markör att textinmatning pågår.

Övervakning vid exekvering av programmet

MEDOC kan övervaka PLC-programmet i drift. Välj först ut ett avsnitt i reläschemat med **EDIT** och **LADDER**. Tryck sedan på **F8** för att starta kommunikationen med PLC-n.

Övning 1. Logik och sekvensstyrning

Syftet med denna övning är att ni skall repetera Boolesk algebra, samt kunna översätta logiken till ett reläschema. Vidare skall ni förebereda laboration 1 genom att konstruera ett antal reläkopplingar, för till exempel en SR-vippa. Vidare skall ni göra ett flödesschema som beskriver styruppgiften, samt införa tillstånd i ett sekvensnät, för att realisera det önskade styrprogrammet på PLC-systemet.

I samtliga uppgifter (som avser reläscheman) skall ni använda korrekta kanalnummer för insignaler mm, till MELSEC F_2-40.

Uppgift 1.1

Rita ett reläschema för en NAND-operation, dvs en AND med inverterad utsignal.

Uppgift 1.2

Rita ett reläschema för en NOR-operation, dvs en OR med inverterad utsignal.

Uppgift 1.3

Rita ett reläschema för en XOR-operation (exklusiv eller). Utsignalen y ges av insignalerna a och b enligt

$$y = \bar{a} \cdot b + a \cdot \bar{b}$$

dvs insignalerna måste vara olika för att ge en sann signal ut.

Uppgift 1.4

Rita ett reläschema för en SR-vippa. Insignaler X410, X411, utsignal Y432.

Uppgift 1.5

Rita ett reläschema för ett sekvensnät med 2 SR-vippor. Tillstånd 1 skall aktivera utgång Y431 och får kvittering på ingång X401. Tillstånd 2 skall aktivera utgång Y432 och får kvittering på ingång X402. Tillstånd 1 skall initialiseras till 1. Vad händer om en kvitteringssignal är sann redan då övergången sker från föregående tillstånd.

Uppgift 1.6

Rita ett flödesschema för styrningen. Specifikation enligt första laborationsuppgiften. Bestäm vilka tillstånd som man måste ha för att genomföra den beskrivna sekvensen.

Det finns många sätt att välja tillstånden på. Programmeringen brukar blir lättare och överskådligare om man använder sig av många tillstånd. Tänk efter vilket det minimala antalet tillstånd är för att kunna beskriva processen.

Skriv upp de logiska villkoren som beskriver övergångar från ett tillstånd till nästa, samt hur utsignalerna beror av tillstånden.

Uppgift 1.7

Skriv ett reläschema som utför den beskrivna uppgiften. Använd gärna benämningar på åtminstone de olika in- och utgångarna. Vid laborationen kan du sedan koppla ihop namnen med I/O-kanalerna i **NAME**-delen av **EDIT**.

Laboration 1. Sekvensstyrning

Ett styrprogram skall programmeras in i PLC-systemet för att lösa en specificerad uppgift. För programmering och uttestning finns programmet MEDOC, skrivet för IBM PC. I laborationen ges också ett antal extra uppgifter, som ni kan lösa i mån av tid.

Krav

För att få laborera skall ni ha med ett förslag till styrprogram till uppgift 1. För att få laborationen godkänd skall ni ha tagit fram ett fungerande program till första uppgiften. Programmet skall kunna löpa genom uppgiften ett godtyckligt antal gånger.

Laborationsprocessen

Gör Dig bekant med de olika delarna och hur processen fungerar. Kontrollera vilka in- och utgångar som de olika signalerna är kopplade till. Gör en lista som kopplar ihop I/O-kanalerna med dina benämningar.

MEDOC

Starta MEDOC och lär dig hantera menyerna. För in eventuella benämningar för in- och utgångar.

Uppgift 1

Den enklaste sekvensen som PLC-systemet skall klara är följande. Efter att ha startat med tom tank, dvs nivå 1 (N1), skall tanken först fyllas till nivå 2 (N2) genom att utloppsventilen (VENTIL) stängs och pumpen (PUMP) startas. När nivå N2 är uppnådd skall uppvärmningen (VÄRME) startas. När önskad temperatur är uppnådd fås indikering genom mätsignalen RÄTT TEMP (kallas TEMP i fortsättningen). Därefter skall värmen stängas av och tanken fyllas till nivå 3 (N3). När N3 är uppnådd skall tanken tömmas.

Skriv in Ditt program och testa ut det, så att sekvensstyrningen uppfyller alla specifikationer. När detta är klart är ni godkända på laborationen.

Nedanstående uppgifter utgör kompletteringar av specifikationen ovan. Uppgifterna utförs i mån av tid.

Uppgift 2

Om aktiveringen lämnas i läge sann kommer en ny cykel att påbörjas så fort tanken har tömts. Komplettera programmet så att aktiveringen endast startar en cykel, sedan skall programmet vänta på ny aktivering.

Uppgift 3

Låt programmet vänta 60 sekunder efter uppnådd temperatur innan påfyllningen till N3 startar.

Uppgift 4

Låt programmet vänta 10 sekunder på nivå N3 startar innan tömningen startar.

Uppgift 5

Låt en aktivering innebära att programmet gör 3 cykler och sedan väntar på ny aktivering.

Uppgift 6

Inför temperaturreglering med till/från på uppvärmningen när tanken är fylld till nivå N2. För att du skall kunna se om regleringen fungerar måste programmet eventuellt vänta några minuter efter uppnådd temperatur på nivå N2, innan påfyllningen till N3 startar.

Ett enkelt reglersystem

1. Inledning

Avsikten med Övning 2 och Laboration 2 är att ni skall skriva och testa ett enkelt realtidsprogram. Det skall innehålla en proportionell regulator, samt operatörskommunikation för att kunna ändra tex regulatorförstärkningen.

Programmet skall skrivas i Modula-2 med utnyttjande av en realtidskärna för IBM-AT maskiner. I denna handledning finns dels en kort introduktion till realtidskärnan, samt en beskrivning av monitorbegreppet och av hur ett program kan stoppas snyggt.

Till övningen och laborationen behövs också särtrycket om Modula-2, med listningar av definitioner till biblioteksmoduler och dokumentation om IBM-PC.

2. Realtidskärnan

Realtidskärnan gör det möjligt att använda parallella processer i Modula-2 på IBM-AT. Både kärnan och Modula-2 finns dokumenterade i föreläsningssärtryck.

Processer deklareras som procedurer. Sådana procedurer får inte deklareras inuti andra procedurer. Processer terminerar aldrig. Man får därför se till att sista `END` i proceduren aldrig nås. När en process är klar med sin uppgift ger man den lägre prioritet än idleprocessen genom anropet `SetPriority(MaxPriority)`.

Ett enkelt exempel på ett realtidsprogram visas nedan.

```
MODULE Main;
  FROM Kernel IMPORT InitKernel, CreateProcess,
                     SetPriority, WaitTime, MaxPriority;
  FROM NumberConversion IMPORT CardToString;
  FROM Terminal IMPORT WriteString, WriteLn;
(* Process *) PROCEDURE InfiniteLoop;
VAR
  counter: CARDINAL;
  string: ARRAY [0..9] OF CHAR;
BEGIN
```

```
  counter:=0;
  SetPriority(100);
  LOOP
    INC(counter);                        (* INCrease by 1       *)
    CardToString(counter,string,10);     (* Convert into string *)
    WriteString(string); WriteLn;        (* Write on terminal   *)
    WaitTime(1000);                      (* Wait 1000 millisec. *)
  END;
END InfiniteLoop;
BEGIN
  InitKernel;
  CreateProcess(InfiniteLoop, 1000);
  SetPriority(MaxPriority);
END Main.
```

Huvudprogrammet blir en process med anropet av `InitKernel`. Efter att ha startat `InfiniteLoop` stannar exekveringen av huvudprocessen utan att programmet stannar.

3. Monitorprincipen

En monitor är en så kallad abstrakt datatyp. Den består av variabler (data), samt procedurer (monitor-procedurer) för att operera på dessa. Principen är att variablerna, tex en regulatorförstärkning, inte får ges ett nytt värde med en vanlig tilldelningssats (`k:=1.0;`), utan detta måste göras via en monitorprocedur (`SetK(1.0);`). De får heller inte läsas direkt. Den exakta implementationen av variablerna och procedurerna behöver inte vara känd utanför monitorn, men andemeningen bakom variablerna och procedurerna måste ha en klar innebörd. Om en semafor används kan procedurerna skrivas så att de garanterar ömsesidig uteslutning.

I ett Pascal-program måste alla gemensamma data (tex monitor-variablerna) deklareras i huvudprogrammet. De är då kända för alla programmerare som skriver olika delrutiner. De kan då frestas att bryta mot reglerna genom att utföra en tilldelning direkt. En sådan tilldelning kommer kompilatorn att godkänna, eftersom den inte är formellt förbjuden i språket.

I Modula-2 kan man däremot tvinga alla programmerare att ändra eller läsa de gemensamma variablerna på föreskrivet sätt. Detta görs genom att monitorn läggs i en modul som enbart exporterar monitor-procedurerna. Monitorvariablerna kan då inte refereras utanför modulen, ty kompilatorn kommer inte att godkänna direkta referenser till dem. I detta fall är dock implementationen känd för alla.

Den faktiska implementationen kan döljas genom att utnyttja separatkompilering. I en definitionsmodul exporteras enbart procedurerna medan variabler-

na deklareras i implementationsmodulen. Här visas ett enkelt exempel.

```
DEFINITION MODULE Monitor;
   EXPORT QUALIFIED Init, GetGain, SetGain;
   (* Entry *) PROCEDURE GetGain(VAR k:REAL);
   (* Entry *) PROCEDURE SetGain(k:REAL);
   PROCEDURE Init; (* Initialization of Monitor data *)
END Monitor.

IMPLEMENTATION MODULE Monitor;
   FROM Kernel IMPORT Semaphore, InitSem, Wait, Signal;
   VAR mutex: Semaphore;
       Gain: REAL;
   (* Entry *) PROCEDURE GetGain(VAR k:REAL);
   BEGIN
     Wait(mutex);
     k := Gain;
     Signal(mutex);
   END GetGain;
   (* Entry *) PROCEDURE SetGain(k:REAL);
   BEGIN
     Wait(mutex);
     Gain := k;
     Signal(mutex);
   END SetGain;
   PROCEDURE Init;
   BEGIN
     InitSem(mutex,1);
     k:= 0.0;
   END Init;
END Monitor.
```

Samma förfaranda kan användas i Ada. Där kallas definitionen för `package` och implementationen för `package body`. Liksom i Modula-2 kan de separatkompileras.

4. Programstopp

Om ett realtidsprogramm skrivs på det sätt som visas ovan kan det stoppas genom att trycka på antingen `Ctrl-Break` eller `Ctrl-C`. För ett reglerprogram

kommer den rådande styrsignalen då att ligga kvar tills programmet startar på nytt eller datorn stängs av.

Om man vill få ett snyggare programstopp får inte huvudprogrammet göra `SetPriority(MaxPriority)`, ty då kommer alltid åtminstone `Idle`-processen att exekvera. Huvudprogramet kan istället vänta på en semafor (här kallad `TheEnd`) som initialiseras till 0. Om operatörskommunikationen får ett stoppkommando, låter man den nollställa styrsignalens utgång och sedan göra `Signal(TheEnd)` på semaforen. När huvudprogrammet sedan når `END`-satsen stannar programmet med nollställd styrsignal.

5. Biblioteksmoduler

Observera att reglerprogrammet måste importera procedurer för att hantera realtid, terminal (in- och utmatning), analoga in- och utsignaler, samt för att konvertera variabler till/från textsträngar. Terminalrutinerna kräver textsträngar som argument, de klarar inte av tex `INTEGER`-variabler.

Övning 2. En enkel regulator

Syftet med denna övning är att ni skall bekanta er med Modula-2 och realtidskärnan för IBM-AT maskinerna. Ni skall skriva ett program för en P-regulator. Programmet skall också innehålla operatörskommunikation.

Uppgift 2.1

Programspecifikation enligt uppgift 1 på laborationen. Det är lämpligt att strukturera sitt system i två parallella processer, en regulatorprocess som exekverar med fast samplingsintervall och en process som hanterar inläsning från tangentbordet. Då ett värde lästs in kontrolleras det, och om det inte accepteras skrivs ett felmeddelande ut på skärmen. Sedan ges prompt för ny inmatning. En viktigt moment är ömsesidig uteslutning vid användning av data gemensamma för de båda processerna. Andra saker att tänka på är initialisering av data och hur de båda processerna startas. Lämpliga verktyg importeras från modulerna `Kernel`, `AnalogIO`, `Terminal`, `ConvReal` och `NumberConversion`.

Uppgift 2.2

Komplettera programmet enligt uppgift 2 på laborationen.

Uppgift 2.3

Ett sätt att skydda data är monitorer, som i Modula-2 lämpligen implementeras med hjälp av moduler. Se uppgift 3 på laborationen.

Uppgift 2.4

Komplettera programmet enligt uppgift 4 på laborationen.

Laboration 2. Ett enkelt reglersystem

Ett enkelt reglersystem för nivåreglering skall implementeras på IBM PC med hjälp av Modula-2 och en realtidskärna. Processen är de från AK-kursen välkända tankarna. Uppgiften går ut på att med hjälp av en digital P-regulator reglera nivån i den övre tanken. Operatören skall kunna påverka regulatorns parametrar. I laborationen ges också ett antal extra uppgifter som ni kan lösa i mån av tid. Som avslutning kan ni köra ett program som också utnyttjar grafik för att presentera signaler samt mus för operatörskommunikation.

Krav

Ni skall till laborationen medföra ett programförslag till första uppgiften. I detta program skall operatören kunna ändra regulatorförstärkningen och referensvärdet. Laborationen är godkänd när ni har tagit fram ett fungerande program till första uppgiften.

Praktiska tips

Pumpen kan inte gå baklänges och fungera som sug, varför det inte är meningsfullt att skicka ut negativa spänningar till den. Spänningsnivån skall ligga mellan 0 och 10 Volt, dvs argumentet `Value` till `DAOut` skall uppfylla $0.0 \leq \text{Value} \leq 1.0$.

För mätsignalen från tanken gäller att 0 Volt motsvarar tom tank och 10 Volt motsvarar full tank. Resultatet från `ADIn` kommer alltså att ligga mellan 0.0 och 1.0.

Låt både mätsignalen och styrsignalen använda kanal 0 i AD- respektive DA-omvandlarna.

Tidskonstanterna i systemet är sådana att man får en rimlig reglering genom att sampla 1–10 gånger per sekund.

Uppgift 1

Programmet skall realisera en digital P-regulator. Med operatörskommunikation skall man kunna ändra regulatorförstärkning och referensvärde. Ömsesidig uteslutning skall garanteras för de gemensamma variablerna, och regleringen får inte hindras av operatörskommunikationen. Eftersom tyngdpunkten i laborationen ligger på realtidsprogrammering gör det inget att utskrifter och inläsning via skärmen blir lite primitiv.

Skriv in Ditt program och testa ut det, så att sekvensstyrningen uppfyller alla specifikationer. När detta är klart är ni godkända på laborationen.

Nedanstående uppgifter utgör kompletteringar av specifikationen ovan. Uppgifterna utförs i mån av tid.

Uppgift 2

Komplettera programmet så att man kan ställa även samplingsintervallet, samt koppla in och koppla ur regulatorn utan att ändra förstärkningen. En avstängd regulator har utsignalen 0.

Uppgift 3

Samla de gemensamma variablerna i en monitor, implementerad som en separatkompilerad modul. Enbart monitorprocedurerna får importeras till regulator- och operatörsprocesserna.

Uppgift 4

Komplettera programmet så att det stannar snyggt utan hjälp av `Ctrl-C` eller **`Ctrl-Break`**, och med nollställd utgång.

Demonstrationsuppgift

Kör programmet **`Lab1main`**. Det löser samma regleruppgift men utnyttjar grafik och en mus för operatörskommunikationen.

Digital reglering

Detta avsnitt innehåller övningarna 3-5 samt laboration 3. Laborationen behandlar digital reglering av ett DC-servo. Övning 3 tar upp sampling av processer, stabilitet samt approximationer av tidskontinuerliga regulatorer. Övning 4 behandlar översättning av PID-regulatorer till diskret form, samt hur de kan skrivas som en allmän digital regulator. Övning 5 tar upp vissa förberedelseuppgifter till laborationen.

Övning 3. Samplade system

Målet för övningen är att gå igenom sampling av processmodeller, och att undersöka stabiliteten i några enkla fall. Vidare skall vi titta på hur man kan översätta en tidskontinuerlig regulator till en tidsdiskret.

Sampling av system

Ett tidskontinuerligt system på tillståndsform brukar skrivas

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

där tillstånden är $x = (x_1 \dots x_n)^T$, insignalen u, utsignalen y och A, B, C och D är matriser av lämplig dimension. I de flesta processer är $D = 0$. Överföringsfunktionen $G(s)$ ges då av

$$G(s) = C(sI - A)^{-1}B = \frac{Y(s)}{U(s)}$$

där $Y(s)$ och $U(s)$ är Laplace-transformerna av ut- och insignalerna.

Vid sampling är insignalen u konstant under samplingsintervallet $[kh\ kh + h]$. Lösningen till differentialekvationen ges i samplingstidpunkterna av

$$x(kh + h) = e^{Ah}x(kh) + \int_0^h e^{As}\,ds\,B\,u(kh) =$$
$$= \Phi x(kh) + \Gamma u(kh)$$

Ett sätt att beräkna e^{Ah} är att använda att

$$\mathcal{L}(e^{Ah}) = (sI - A)^{-1}$$

Med en Laplace-tabell kan man då erhålla elementen i e^{Ah} genom att inverstransformera motsvarande element i $(sI - A)^{-1}$.

Utsignalen y för det samplade systemet ges av samma uttryck som ovan. Pulsöverföringsfunktionen $H(q)$ ges analogt med överföringsfunktionen av

$$H(q) = C(qI - \Phi)^{-1}\Gamma = \frac{B(q)}{A(q)}$$

Skiftoperatorn q betyder skift ett steg framåt i tiden, dvs $qx(k) = x(k+1)$. Sekvenserna $\{y(k)\}$ och $\{u(k)\}$ av ut- och insignaler ges då av

$$y(k) = H(q)u(k)$$

eller

$$A(q)y(k) = B(q)u(k)$$

eller

$$y(k+n) + a_1 y(k+n-1) + \cdots + a_n y(k) = b_1 u(k+n-1) + \cdots + b_n u(k)$$

Polerna till det samplade systemet ges av egenvärdena till Φ, vilket är det samma som lösningen till den karakteristiska ekvationen $A(q) = 0$. Nollställen ges av lösningarna till $B(q) = 0$.

Uppgift 3.1

Antag att ett system beskrivs av

$$\dot{y} = -a \cdot y + b \cdot u$$

Låt insignalen u vara konstant under intervall av längden h. Beräkna hur y ser ut i tidpunkterna kh. Beräkna pulsöverföringsfunktionen $H(q)$. Hur varierar polen hos det samplade systemet då samplingsintervallet h varierar?

Uppgift 3.2

Sampla

$$\dot{x} = \begin{pmatrix} 0 & c_1 \\ 0 & -a \end{pmatrix} x + \begin{pmatrix} 0 \\ c_2 \end{pmatrix} u$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x$$

dvs beräkna Φ och Γ. a, c_1 och c_2 är positiva. Beräkna pulsöverföringsfunktionen $H(q)$. Vilka poler har det samplade systemet?

Stabilitet

För tidskontinuerliga system gäller att alla poler måste ha negativ realdel för att systemet skall vara stabilt. För tidsdiskreta (samplade) system gäller istället att polerna skall ligga innanför enhetscirkeln, dvs ha absolutbeloppet mindre än 1.

Uppgift 3.3

Bestäm stabilitetsvillkoren för öppna systemet i uppgift 3.1. Jämför med tidskontinuerliga fallet.

Uppgift 3.4

Återkoppla systemet i uppgift 3.1. med en proportionell regulator med förstärkningen K. Bestäm hur stabilitetsgränserna för regulatorförstärkningen K beror på samplingsintervallet h för fallet att $a > 0$. Jämför med tidskontinuerliga fallet.

Stabilitetsvillkor

För första ordningens system är frågan som synes lätt avgjord. För andra ordningens system gäller att den karakteristiska ekvationen

$$z^2 + a_1 z + a_2 = 0$$

har lösningar med absolutbeloppet mindre än 1 om följande villkor på koefficienterna är uppfyllda.

$$a_2 < 1$$
$$a_2 > -1 + a_1$$
$$a_2 > -1 - a_2$$

Uppgift 3.5

Bestäm hur den övre stabilitetsgränsen för regulatorförstärkningen K beror på samplingsintervallet h, då systemet i uppgift 2 återkopplas med en proportionell regulator. Approximera uttrycket för små h.

Tips: Det räcker att testa första villkoret av de tre. Vidare ger det tredje villkoret att $K > 0$.

Approximation av tidskontinuerliga regulatorer

Sampling av processmodeller bygger på att insignalen $u(t)$ är konstant under samplingsintervallen. För regulatorer gäller däremot att deras insignal $y(t)$, dvs processens utsignal, inte är konstant under ett samplingsintervall. Av den anledningen brukar man inte använda formlerna för Φ och Γ när en tidskontinuerlig regulator skall diskretiseras (samplas).

I stället brukar man använda olika approximationer av tidsderivatan, tex en framåtdifferens (Euler's metod)

$$\frac{dx}{dt} \approx \frac{x(t+h) - x(t)}{h} = \frac{q-1}{h} x(t)$$

eller en bakåtdifferens

$$\frac{dx}{dt} \approx \frac{x(t) - x(t-h)}{h} = \frac{q-1}{qh} x(t)$$

Derivering av tidsfunktioner motsvaras av multiplikation med s för funktionens Laplacetransform. En metod att gå från en kontinuerlig regulator $G(s)$ till en diskret regulator $H(q)$ blir då att låta

$$H(q) = G(s')$$

där man vid framåtdifferenser väljer

$$s' = \frac{q-1}{h}$$

eller vid bakåtdifferenser

$$s' = \frac{q-1}{qh}$$

Vidare förekommer att man använder Tustin's approximation,där

$$s' = \frac{2}{h} \frac{q-1}{q+1}$$

Uppgift 3.6

Approximera regulatorn

$$G(s) = K\frac{s+a}{s+b} \qquad a \neq b, \quad K, a, b > 0$$

med de tre approximationerna ovan. Den tidskontinuerliga regulatorn är stabil. Kan någon av approximationerna ge en instabil regulator?

Övning 4. PID-regulatorer

Syftet med övningen är att undersöka olika strukturering av diskreta PID-regulatorer, samt att ta fram en allmän digital regulator.

I förra övningen diskuterades några olika sätt att approximera tidskontinuerliga regulatorer. Dessa kan med fördel tillämpas på PID-regulatorer. I kompendiet (6.2) redovisas några vanliga varianter på PID-regulatorer. Dessa skiljer sig genom att antingen reglerfelet $e = u_c - y$ eller mätsignalen y deriveras, och huruvida derivatadelen innehåller ett filter. I kontinuerlig tid måste derivatadelen innehålla ett filter, men i diskret tid kan man klara sig utan.

Vid approximation av PID-regulatorer används vanligen framåtdifferenser till integraldelen och bakåtdifferenser till derivatadelen. Vid framåtdifferenser ersätts s i Laplacetransformen med

$$s' = \frac{q-1}{h}$$

och vid bakåtdifferenser med

$$s' = \frac{q-1}{qh}$$

Uppgift 4.1

Diskretisera I-delen med framåt- och bakåtdifferenser. Vad blir skillnaden?

Diskretisera D-delen (filtrerad) med framåt- och bakåtdifferenser. Vad blir skillnaden (vid små värden på T_d)?

Diskretisera D-delen (ofiltrerad) med bakåtdifferenser.

När denna uppgift är löst har du ett antal byggstenar för att sätta samman diskreta PID-regulatorer.

Uppgift 4.2

Härled en tidsdiskret PID-regulator där D-delen opererar på e utan någon filtrering. Använd framåtdifferenser till I-delen och bakåtdifferenser till D-delen. Skriv kod (i Fortran, Pascal eller Modula) som realiserar algoritmen. Vad skall ändras om D-delen opererar direkt på mätsignalen (y)?

Uppgift 4.3

Ange hur regulatorkoden i uppgift 4.2 skall ändras om D-delen istället filtreras.

En allmän digital regulator

De olika PID-regulatorerna består alla i att styrsignalen $u(k)$ skall beräknas från signalerna $u(k-1)$, $u(k-2)$, $y(k)$, $y(k-1)$, etc. En sådan regulator kan skrivas på följande sätt.

$$R(q^{-1})u(k) = -S(q^{-1})y(k) + T(q^{-1})u_c(k) \qquad (1)$$

där

$$R(q^{-1}) = 1 + r_1 q^{-1} + \cdots + r_n q^{-n}$$
$$S(q^{-1}) = s_0 + s_1 q^{-1} + \cdots + s_n q^{-n}$$
$$T(q^{-1}) = t_0 + t_1 q^{-1} + \cdots + t_n q^{-n}$$

Uppgift 4.4

Härled koefficienterna i polynomen R, S och T för PID-regulatorn i uppgift 4.2. Diskutera vad som förändras om D-delen innehåller ett filter, respektive skillnaden mellan att D-delen opererar på y eller e?

Beräkningstekniska aspekter

Betrakta en regulator på RST-form, tex den i föregående uppgift. Hur skall koden organiseras för att styrsignalen skall beräknas så snabbt som möjligt efter det att datorn har läst in mätvärdet $y(k)$ och börvärdet $u_c(k)$? Se kompendiet sid 6:11.

Lös motsvarande problem för PID-regulatorn i uppgift 4.2.

Uppgift 4.5

Skriv kod (i Fortran, Pascal eller Modula) för en regulator på RST-form. Gör i första hand ordningstalet begränsat till andra ordningen. Om tiden medger, så generalisera regulatorn till n:te ordningen. Ange hur regulatorparametrar och gamla signaler lagras. Se till att räknetiden mellan inläsning av mätvärden och utläsning av styrsignal blir kort.

Operatörskommunikation

Vilka parametrar skall operatören känna till? Var skall koefficienterna i R, S och T beräknas när en PID-regulator implementeras på RST-form i en dator? Hur bör parametrarna hanteras i en PID-regulator enligt uppgift 4.2?

Uppgift 4.6

Skriv (eller skissera) kod för hur parametrarna hanteras till en PID-regulator i de bägge fallen.

Reset-windup

I alla verkliga regulatorer är styrsignalen begränsad till ett visst intervall. I regulatorer med integraldel kan man då få ett fenomen som kallas reset-windup, eller integratoruppvridning. Det kan uppträda när reglerfelet har samma tecken under lång tid. För en PI regulator är problemet löst i kompendiet sid 6.10. För en regulator på formen R, S, T kan antireset-windup åstadkommas på följande sätt:

Lägg till $A_0(q^{-1})u(k)$ på båda sidor om uttrycket (1) och flytta $R(q^{-1})u(k)$ till höger sida. Vi får då

$$A_0(q^{-1})u(k) = -S(q^{-1})y(k) + T(q^{-1})u_c(k) + ((A_0(q^{-1}) - R(q^{-1}))u(k) \quad (2)$$

Man kan tex välja

$$A_0 = 1 - aq^{-1}$$

Det är viktigt att första koefficienten i A_0 är densamma som i R. Sedan låter man $u(k)$ i vänsterledet ersättas med $v(k)$, där $v(k)$ är den obegränsade styrsignalen och $u(k)$ den begränsade. En regulator med anti-reset windup får då följande utseende,

$$\begin{aligned} A_0 v(k) &= T u_c(k) - S y(k) + (A - R) u(k) \\ u(k) &= \text{sat}(v(k)) \end{aligned} \tag{3}$$

där sat($v(k)$) begränsar $v(k)$ till det tillåtna intervallet. I princip skall parametern a väljas mellan 0 och (nästan) 1. För en PI-regulator kan man välja a till 0, vilket betyder att man bara begränsar styrsignalen (på RST-form).

För en PI-regulator med explicit integraldel så motsvarar detta precis den metod för anti-reset windup som visats i kompendiet sid 6.10. Praktiskt betyder $a = 0$ att man på ett enda samplingsintervall återställer integraldelen till ett "tillåtet" värde.

För en PID-regulator fungerar det inte bra med $a = 0$. D-delen kan ge stora utslag pga mätbrus med följd att styrsignalen snabbt drivs till mättning om $a = 0$. Om istället a är större, tex 0.5-0.9, kommer inte styrsignalen att påverkas så snabbt av störningarna.

För en PID-regulator med explicit integraldel motsvarar det att man inte återställer integraldelen på ett enda samplingsintervall, utan betydligt fler. Priset för denna strategi är att man under normala driftfall kan få en större översläng på processens utsignal, jämfört med fallet $a = 0$.

Uppgift 4.7

Skriv ut (3) explicit så att tolkningen av skiftpolynomen framgår klart. Antag att ordningstalen för R, S och T är två, samt välj A_0 enligt ovan.

Att fundera över

Kan man i koden för PID-regulatorn helt koppla bort integraldelen ($T_i = \infty$) eller derivatadelen ($T_d = 0$)? Diskutera både RST-formen och PID med explicit integral- och derivatadel.

Ofta vill man att en regulator skall kunna köras manuellt, dvs u är en på något sätt given signal, tex från en analog ingång. Hur skall koden modifieras för att kunna hantera detta? Diskutera både RST-formen och PID med explicit integral- och derivatadel.

Diskutera mjuk övergång från t.ex. manuell reglering till automatik. Hur kan man se till att den inte blir stötig? Diskutera både RST-formen och PID med explicit integral- och derivatadel.

Kommer parameterbyten (i stationaritet) att bli stötfria? Diskutera både RST-formen och PID med explicit integral- och derivatadel.

Som ni ser finns det många aspekter på vad en diskret PID-regulator skall klara av, antingen den är skriven på RST-form eller som en traditionell PID-regulator.

Övning 5. Digital reglering av DC-servo

Denna övning är en direkt förberedelse till laboration 3. Där skall vi använda en allmän digital regulator,

$$R(q)u(k) = -S(q)y(k) + T(q)r(k) \tag{1}$$

där u är styrsignalen, y mätsignalen och r referensvärdet.

För DC-servot gäller följande tillståndsbeskrivning.

$$\begin{aligned} \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} &= \begin{pmatrix} 0 & 9.25 \\ 0 & -0.12 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 2.66 \end{pmatrix} u(t) \\ y(t) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \end{aligned} \tag{2}$$

Tillstånden är valda så att $x_1(t)$ är servots läge och $x_2(t)$ dess hastighet. Överföringsfunktionen från drivspänningen $u(t)$ till läget $x_1(t) = y(t)$ blir

$$G(s) = \frac{24.7}{s(s + 0.12)} \tag{3}$$

I övning 3.2 har ni beräknat dels pulsöverföringsoperatorn $H(q)$, dels den samplade tillståndsbeskivningen, dvs matriserna Φ och Γ, för ett system med samma struktur. I bägge fall är koefficienterna funktioner av samplingsintervallet h.

PD-reglering

Först skall en PD-regulator beräknas till den kontinuerliga överföringsfunktionen (3). Sedan skall den approximeras på olika sätt och testas med avseende på hur långa samplingsintervall den tål.

Uppgift 5.1

Beräkna K och T_d i en PD-regulator utan filter till (3), så att slutna systemet får det karakteristiska polynomet

$$s^2 + 2\zeta\omega s + \omega^2 = 0 \tag{4}$$

med $\omega = 6$ rad/s och $\zeta = 0.7$.

Uppgift 5.2

Förbered uppgifterna 1-3 på laborationen genom att beräkna koefficienterna i styrlagarna.

Tillståndsåterkoppling 1

I detta avsnitt skall vi testa några styrlagar baserade på tillståndsåterkoppling. De beräknas för den tidskontinuerliga modellen (2), men implementeras digitalt.

Uppgift 5.3

Beräkna l_1, l_2 och m i styrlagen

$$u(t) = -l_1 x_1(t) - l_2 x_2(t) + mr(t) \tag{5}$$

till (2), så att det slutna systemet får det karakteristiska polynomet (4), där $\omega = \{6, 10, 12\}$ rad/s och $\zeta = 0.7$. Konstanten m väljs så att det slutna systemets statiska förstärkning blir 1.

Tillståndsåterkoppling 2

I detta avsnitt skall koefficienterna i styrlagen

$$u(k) = -l_1 x_1(k) - l_2 x_2(k) + mr(k) \tag{6}$$

beräknas för det samplade systemet, dvs $\Phi = e^{Ah}$ och $\Gamma = \int e^{As} B\, ds$, så att slutna systemet får önskad karakteristisk ekvation.

Uppgift 5.4

Tillståndsbeskrivningen (2) har samplats i övning 3.2. Bestäm numeriska värden på koefficienterna i Φ och Γ för samplingsintervallet $h = 0.1$ s.

Uppgift 5.5

Visa att det tidskontinuerliga karakteristiska polynomet (4), vid sampling med samplingsintervallet h, motsvaras av det tidsdiskreta karakteristiska polynomet

$$z^2 - 2e^{-\zeta\omega h} \cos\left(\omega h\sqrt{1-\zeta^2}\right) z + e^{-2\zeta\omega h} = 0 \tag{7}$$

Tips: En tidskontinuerlig pol s_i motsvaras av en tidsdiskret pol $z_i = e^{s_i h}$.

Uppgift 5.6

Beräkna l_1, l_2 och m i styrlagen (6) så att det slutna systemet får det karakteristiska polynomet (7), när $h = 0.1$ s, $\omega = 6$ rad/s och $\zeta = 0.7$. Konstanten m väljs så att det slutna systemets statiska förstärkning blir 1.

Utsignalåterkoppling

I detta avsnitt skall ni beräkna regulatorpolynomen R, S och T för att erhålla ett önskat slutet system. Om vi känner processens överföringsfunktion $H(q) = \frac{B(q)}{A(q)}$ och använder regulatorn (1), så blir slutna systemet

$$y(k) = \frac{B(q)T(q)}{A(q)R(q) + B(q)S(q)} u_c(k) = \frac{B_m(q)}{A_m(q)} u_c(k) \tag{8}$$

$A_m(q)$ är det önskade karakteristiska polynomet. Vi vill i detta fall ha samma nollställen i slutna och öppna systemet. Detta innebär att

$$B_m(q) = t_0 B(q)$$

Valet av t_0 skall göras så att $\frac{B_m(1)}{A_m(1)}$ är lika med önskad statisk förstärkning, till exempel 1. I vårt fall skall då gälla

$$\frac{B_m(q)}{A_m(q)} = \frac{\frac{1+p_1+p_2}{b_1+b_2}(b_1 q + b_2)}{q^2 + p_1 q + p_2} \tag{9}$$

Vidare måste man välja

$$T(q) = t_0 A_o(q)$$

där $A_o(q)$ är ett extra polynom, observerarpolynomet, som vi måste välja. I så fall kan man beräkna R och S genom att lösa ekvationen

$$A(q)R(q) + B(q)S(q) = A_m(q)A_o(q)$$

Om det finns lösningar till ekvationen, så finns det i regel oändligt många. Alla dessa är emellertid inte lämpliga eller möjliga att använda, men genom att ge krav på gradtalen hos R och S får man en lämplig regulator.

För DC-servot görs följande val av polynom.

1) $A_m(q)$ väljs av gradtalet 2, dvs samma som för $A(q)$.

2) $A_o(q)$ valjs av gradtalen 1 eller 2, dvs antingen samma som för $A(q)$ eller 1 steg lägre.

3) En lösning till ekvationen får man då om $R(q)$ har samma gradtal som $A_o(q)$, och $S(q)$ har gradtalet 1 mindre än gradtalet för $A(q)$.

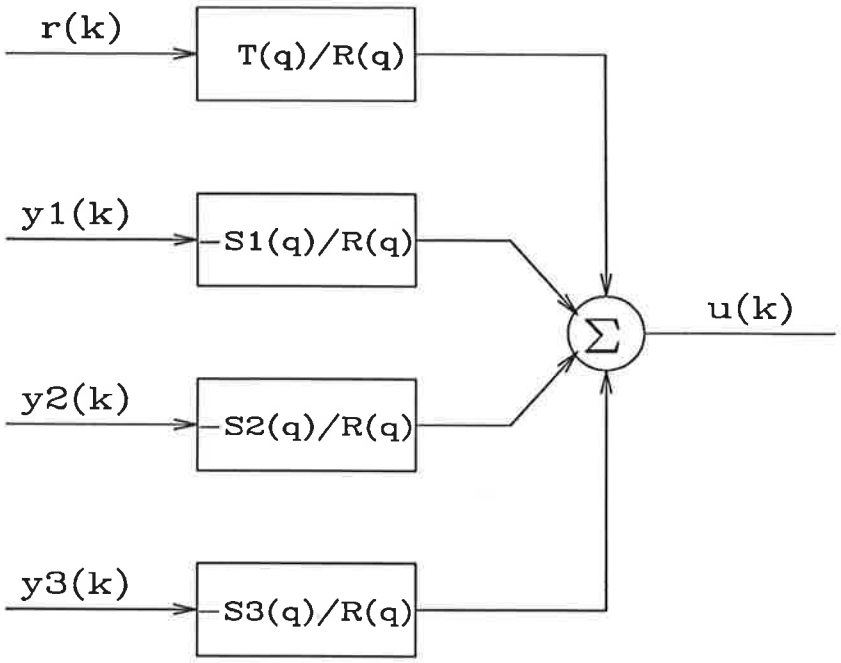
4) $T(q)$ beräknas, där t_0 väljs för att ge önskad statisk förstärkning.

Uppgift 5.7

I övning 3.2 har ni beräknat pulsöverföringsfunktionen $H(q)$. Bestäm numeriska värden på koefficienterna för samplingsintervallet $h = 0.1$ s, samt bestäm poler och nollställen. Tips: Polerna ges av $A(q) = 0$ och nollställena av $B(q) = 0$.

Uppgift 5.8

Beräkna R, S och T enligt metoden ovan för $H(q)$ i föregående uppgift, med $A_m(q)$ vald som ekvation (7) när $h = 0.1$ s, $\omega = 6$ rad/s och $\zeta = 0.7$. Använd $A_o(q) = q - e^{-1}$. Med $h = 0.1$ s, motsvarar detta ett tidskontinuerligt observerarpolynom $s + 10$.



Figur 1. Blockschema för MISO-regulatorn.

Laboration 3. Digital reglering av DC-servo

Avsikten med laborationen är att undersöka vad man kan åstadkomma med några olika metoder, då ett positionsservo styrs av en digital regulator. Ni skall i första hand studera avsnitten 1 och 2, där regulatorerna beräknats för tidskontinuerlig reglering. I andra hand skall ni studera digital tillståndsåterkoppling i avsnitt 3, och om tiden räcker till även utsignalåterkopplingen i avsnitt 4.

Krav

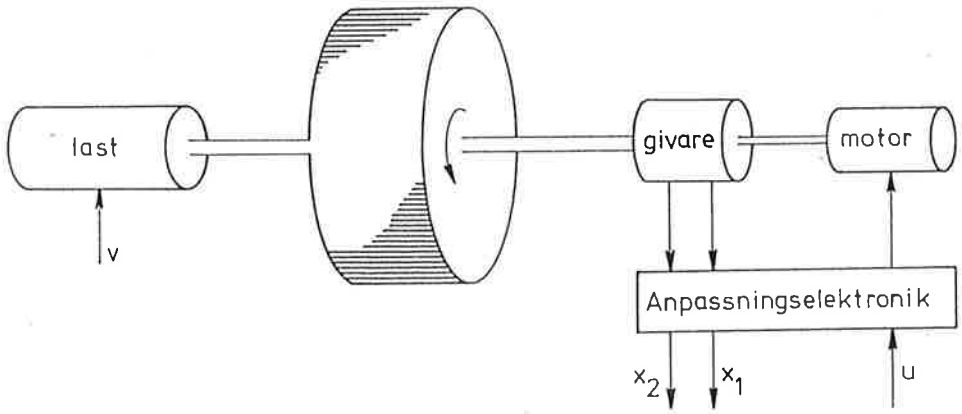
Till laborationen skall ni medföra tabeller med regulatorparametrar till uppgifterna 1-5. Detta har förberetts på övning 5.

Regulatorprogrammet

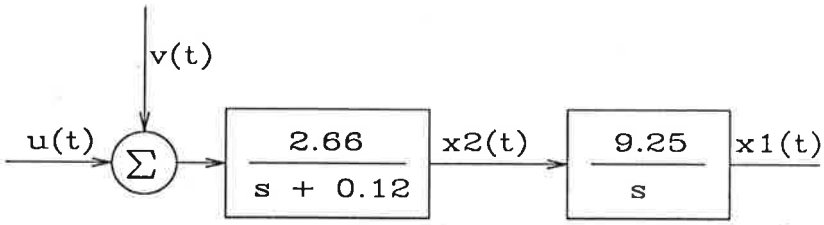
För att genomföra laborationen finns en IBM-PC AT utrustad med AD- och DA-omvandlare samt grafisk skärm. Processen är ett positionsservo, som skall regleras med datorns hjälp. I datorn finns program som realiserar regulatorn

$$R(q)u(k) = T(q)r(k) - S_1(q)y_1(k) - S_2(q)y_2(k) - S_3(q)y_3(k) \qquad (1)$$

där R, T, S_1, S_2 och S_3 är polynom av n:e graden i skiftoperatorn q. Signalen u är regulatorns utsignal, r är referensvärdet och y_1, y_2 och y_3 är mätvärden. Regulatorn kallas MISO-regulatorn (Multi Input Single Output) därför att den beräknar en styrsignal på basis av flera mätsignaler. Ett blockschema för regulatorn visas i Figur 1. Regulatorimplementeringen innehåller skydd mot uppvridning, om styrsignalen skulle mätta.



Figur 2. Processen



Figur 3. Blockschema för processen

Processen

Processen är ett DC-servo med svänghjul som skall fås att följa ett önskat vinkelläge. Anpassningselektronik kring processen gör snittet mot datorn enkelt. Styrsignalen u utgörs av en spänning $\pm 10V$ och mätsignalerna utgörs också av spänningar i detta intervall. Figur 2 visar en schematisk bild av processen.

Utifrån momentbalans för motoraxeln och med anpassningselektronikens konstanter erhåller man följande tillståndsbeskrivning av processen.

$$\begin{aligned} \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} &= \begin{pmatrix} 0 & 9.25 \\ 0 & -0.12 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 2.66 \end{pmatrix} u(t) \\ y(t) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \end{aligned} \tag{2}$$

Tillstånden är valda så att $x_1(t)$ är servots läge och $x_2(t)$ dess hastighet. Läget kan alltid mätas och ibland skall vi även mäta hastigheten. Processens blockschema visas i Figur 3. Överföringsfunktionen från drivspänningen $u(t)$ till läget $x_1(t) = y(t)$ blir

$$G(s) = \frac{24.7}{s(s + 0.12)} \tag{3}$$

En konstant momentlast v kan kopplas in med hjälp av en brytare. Hastighets- och lägesgivarna genererar ringa mätbrus som i de flesta fall ej ställer till några problem.

Mål

Under laborationen skall ni studera olika principer för digitala regulatorer. I de 2 första avsnitten studeras (approximationer av) tidskontinuerliga regulatorer. Framförallt skall ni där undersöka hur långa samplingsintervall h man kan använda. I de 2 följande avsnitten studeras regulatorer designade för tidsdiskreta system. Samplingsintervallet kan här väljas längre men måste ställas i relation till den önskade bandbredden (egenvinkelfrekvensen) för slutna systemet. Regulatorerna har till uppgift att få läget x_1 att följa referensvärdesvariationer. Detta problem brukar kallas servoproblemet.

Kraven på slutna systemet specificeras i form av egenvinkelfrekvensen ω, upp till $\omega = 15$ rad/s, och relativa dämpningen ζ för ett tidskontinuerligt system. I de två senare avsnitten översätts detta till motsvarande polplacering för tidsdiskreta system. Därigenom kan man jämföra de olika regulatorerna, eftersom de har samma prestandakrav.

Praktiska tips

Samplingsintervallet bör inte väljas kortare än 0.04 s. Vidare bör styrsignalen inte mätta, dvs överstiga 10 V, då referensvärdet ändras med 2 V. Detta ger ett krav på dels amplituden på den fyrkantvåg som servot skall följa, dels på koefficienterna s_0 och t_0 i styrlagen. Om inget annat anges väljs amplituden till 4 V.

Internt i regulatorprogrammet representeras spänningarna av reella tal mellan ± 1, dvs $+10$ V motsvaras av 1.0, och -10 V av -1.0.

Hantering av MISO-regulatorn

Innan du läser detta avsnitt bör du ha läst igenom appendix, vilket beskriver hanteringen och alternativen för de olika menyerna. Nedan beskrivs de viktigaste punkterna.

1) Under `SETUP`, välj samplingsintervall med `TSAMP` h, sedan `SAVE`.

2) Under `REGUL`, välj regulatorpolynom och gör sedan `SAVE`.

3) Under `SIGNAL`, välj amplitud på fyrkantvågen och gör `SAVE`.

4) Under `PLOT` kan du starta och stoppa regleringen, samt se alla dina utvalda signaler.

Under ett par av menyerna kan du både starta och stoppa regleringen, samt se börvärde och ärvärde plottade på en mindre yta än under `PLOT`-menyn. Då finns alltid `RUN/STOP` alternativet med som muskommando.

1. PD-reglering

Först antar vi att bara lägessignalen x_1 finns tillgänglig för mätning. Det visar sig att proportionell reglering av servot går mycket dåligt. Stabilitetsgränsen för förstärkningen blir låg, och vidare kommer torrfriktionen i servot att ställa till problem.

Slutna systemets prestanda förbättras drastiskt om styrlagen förutom lägesfelet också innehåller en term proportionell mot vinkelhastigheten. Då kan

man också dra upp förstärkningen för lägesfelet. Eftersom vinkelhastigheten inte kan mätas så måste den skattas på något sätt. I en PD-regulator kommer den deriverande delen i styrlagen att skatta vinkelhastigheten.

I detta avsnitt skall ni testa digital PD-reglering av DC-servot. Först skall en PD-regulator beräknas till den kontinuerliga överföringsfunktionen (3). Sedan skall den approximeras på olika sätt och testas med avseende på hur långa samplingsintervall den tål.

I uppgift 5.1 har ni beräknat K och T_d i en PD-regulator utan filter till (3), så att slutna systemet har fått det karakteristiska polynomet

$$s^2 + 2\zeta\omega s + \omega^2 = 0 \tag{4}$$

med $\omega = 6$ rad/s och $\zeta = 0.7$. Dessa värden på K och T_d skall användas i de tre följande uppgifterna.

Uppgift 1

Studera först en PD-regulator utan filter, där derivatadelen opererar på reglerfelet. Approximera med bakåtdifferenser och skriv resultatet på RST-form. Mätsignalen för läget är kopplad till S_1-polynomet. Beräkna koefficienterna för samplingsintervallen $h = 0.04, 0.10, 0.15, 0.20$ s, och testa regulatorn.

Uppgift 2

Genomför samma uppgift för en PD-regulator med filter, som approximeras med framåtdifferenser. Välj $T_f = T_d/5$, och använd samplingsintervallen $h = 0.04, 0.06, 0.09, 0.10$ s. För vilket h blir regulatorn instabil? Testa regulatorn.

Uppgift 3

Som föregående uppgift, men använd Tustin's approximation och samplingsintervallen $h = 0.04, 0.10, 0.15, 0.20$ s.

Sammanfatta dina erfarenheter av de olika approximationerna. Notera att PD-parametrarna har valts för att ge samma slutna system, åtminstone i kontinuerlig tid.

2. Tillståndsåterkoppling 1

Antag nu att vinkelhastigheten x_2 kan mätas. Det är då möjligt att använda den direkt i styrlagen istället för att skatta den. Denna typ av reglering kallas tillståndsåterkoppling, och den baseras på att alla tillstånd kan mätas. x_2 är kopplad till S_2-polynomet.

I detta avsnitt skall vi testa några styrlagar baserade på tillståndsåterkoppling. De har beräknats för den tidskontinuerliga modellen (2), men implementeras digitalt.

I uppgift 5.3 har ni beräknat l_1, l_2 och m i styrlagen

$$u(t) = -l_1 x_1(t) - l_2 x_2(t) + mr(t) \tag{5}$$

till (2), så att det slutna systemet har fått det karakteristiska polynomet (4), där $\omega = \{6, 10, 12\}$ rad/s och $\zeta = 0.7$. Konstanten m har valts så att det slutna systemets statiska förstärkning blir 1.

MISO-regulatorn (1) innehåller tillståndsåterkoppling som ett specialfall. Sätt $S_1 = l_1$, $S_2 = l_2$, $T = m$ och $R = 1$ så erhålles den önskade regulatorn.

Uppgift 4

Testa de olika regulatorerna med avseende på samplingsintervallet. Börja med $h = 0.04$ s och gå uppåt, tex $h = 0.10$, 0.15, 0.20 s etc. Välj amplituden 2 V till fyrkantvågen då $\omega = 10$, 12 rad/s, och 4 V då $\omega = 6$ rad/s.

Sammanfatta dina erfarenheter av de olika valen av egenvinkelfrekvens respektive samplingsintervall.

3. Tillståndsåterkoppling 2

I detta avsnitt skall ni undersöka tidsdiskret tillståndsåterkoppling. Då skall koefficienterna i styrlagen

$$u(k) = -l_1 x_1(k) - l_2 x_2(k) + mr(k) \tag{6}$$

beräknas för det samplade systemet, dvs $\Phi = e^{Ah}$ och $\Gamma = \int e^{As} B\, ds$, med en önskad karakteristisk ekvation

$$z^2 - 2e^{-\zeta\omega h} \cos\left(\omega h\sqrt{1-\zeta^2}\right) z + e^{-2\zeta\omega h} = 0 \tag{7}$$

som placerar polerna i enhetscirkeln. Detta polynom motsvarar ett tidskontinuerligt karakteristiskt polynom (4) när man samplar med samplingsintervallet h, se uppgift 5.5. Därigenom kan man göra jämförelser med tidskontinuerlig design.

Det blir tämligen tidsödande att beräkna koefficienterna när både samplingsintervallet h och egenvinkelfrekvensen ω varieras. Därför har vi samlat värden på h, ω och ζ med tillhörande återkopplingskonstanter l_1 och l_2 i en tabell. De koefficienter som saknas har beräknats i uppgift 5.6. m skall som tidigare väljas så att det slutna systemets statiska förstärkning blir 1.

h	ω	ζ	l_1	l_2
0.04	6	0.7	1.2397	2.8726
	10		3.0783	4.5636
	12		4.1909	5.3394
	15		6.0197	6.4215
0.10	6	0.7		
	10		2.0284	3.7405
	12		2.5393	4.2041
	15		3.2185	4.7593
0.20	6	0.7	0.6386	2.0911
	10		1.0188	2.6802
	12		1.1187	2.8218
	15		1.1800	2.9158

Uppgift 5

Testa de olika styrlagarna. Jämför med resultaten från avsnitt 2 vad avser prestanda, stabilitet, mm.

Hur påverkas resultatet av samplingsintervallet? Hur hög egenvinkelfrekvens kan det slutna systemet ha innan det beter sig illa?

4. Utsignalåterkoppling

Antag på nytt att hastigheten x_2 ej kan mätas. Styrlagen måste då, liksom i avsnitt 1, innehålla någon form av hastighetsskattning för att slutna systemet skall få acceptabla prestanda.

Till skillnad från PD-regleringen, skall vi här beräkna styrlagar utifrån att vi känner processens överföringsfunktion $H(q) = \frac{B(q)}{A(q)}$, samt har specificerat ett önskat slutet system. Det visar sig att de erhållna styrlagarna i en del fall kommer att kunna tolkas som PD- respektive PID-regulatorer.

Programmet innehåller, förutom MISO-regulatorn, algoritmer för att beräkna R-, S- och T-polynomen. Det är således möjligt att ge en processmodell $\{B(q), A(q)\}$, samplingsintervallet h och tidskontinuerliga (eller tidsdiskreta) specifikationer till programmet, som sedan beräknar regulatorparametrarna. Lösningsmetoden beskrivs kortfattat i övning 5.

Hantering av designdelen i programmet

Innan du läser detta avsnitt bör du ha läst igenom appendix, vilket beskriver hanteringen och alternativen för de olika menyerna.

1) Under `SETUP`, välj samplingsintervall med `TSAMP` h, sedan `SAVE`.

2) Under `MODEL`, välj `SAMPLE`, vilket ger en samplad modell av DC-servot för aktuellt samplingsintervall h.

3) Under `DESIGN` välj

   a) ny karakteristisk ekvation med `CSAM` ω ζ,

   b) observerarpolynom med `CSAO` ω_o (ζ_o),

   c) `INTON` eller `INTOFF`,

   d) `TOREG`, vilket laddar över de nya regulatorparametrarna.

4) Under `REGUL` kan du inspektera parametrarna.

5) Under `PLOT` kan du se alla dina utvalda signaler.

Uppgift 6

Upprepa försöken från uppgift 5 med samma ω och h och studera uppförandet hos systemet. Undersök speciellt det fall ni har beräknat i uppgift 5.8. Prova med observerarpolynom A_o av grad ett och två. Normalt bör $\omega_o > \omega$, men prova även $\omega_o < \omega$. Välj $\zeta_o = 1$ när grad $A_o = 2$. Hur varierar gradtalen för R, S och T med gradtalet för A_o? Vilken typ av regulator motsvarar detta? Anteckna regulatorparametrarna (framför allt när $\omega = 6$ rad/s) och jämför dels parametrar dels prestanda med försöken i avsnitt 1.

Hur påverkas resultatet av samplingsintervallet? Hur hög egenvinkelfrekvens kan det slutna systemet ha innan det beter sig illa? Hur påverkas systemets uppförande av valet av observerarspecifikationer ω_o?

Integralverkan

Vid försöken uppkommer stationära avvikelser mellan börvärde och uppmätt läge. Detta fel beror på torrfriktion, vilken påverkar processen på samma sätt som en momentlast. För att eliminera det stationära felet måste vi ha en integrerande regulator.

Uppgift 7

Utför några av försöken i föregående uppgift, men använd istället en integrerande regulator. Observera att du måste välja A_o av gradtalet 2 eller 3 när du använder `INTON`-alternativet för design. Välj då både ζ_o (och α) till 1, samt $\omega_o > \omega$.

Vilka gradtal får R, S och T vid integrerande reglering. Vilken typ av regulator motsvarar detta? Förklara reglerfelets uppträdande.

APPENDIX

Beskivning av programmet TOOLBOX

Programmet som skall användas vid laborationen är ett generellt verktyg för polynomdesign och reglering av linjära SISO system. Det kan dessutom användas för reglering av SIMO system dvs för tillståndsåterkoppling.
Programmet implementerar en regulator som samplas med intervallet h, som kan väljas mellan 0.01 och 10 sekunder.
Programmet är interaktivt och användaren kommunicerar med det med mus och tangentbord. Operatörskommunikationen är menystyrd och meny väljs med musen. Ett val med musen görs genom att placera musmarkören på texten på kommandoraden och då trycka på en av musens knappar. Den nedre kommandoradens utseende beror på aktuell meny. Val på denna aktiverar menyspecifika funktioner. Programmet skiljer ej på gemener och versaler.
Här följer en uppräkning av de kommandon som kan ges med tangentbord och mus för de olika menyerna. Kommandon som ges med tangentbordet är alltid någon inmatning av tal. I uppräkningen nedan betyder r reellt tal, i heltal och p polynomkoeffecienter.

SETUP-meny

Här sätts en del övergripande parametrar.

Kommandon från tangentbord

CHANREF i	Val av kanal för referenssignal r som kan läsas in.
CHANY1 i	Val av kanal för mätsignalen y_1.
CHANY2 i	Val av kanal för mätsignalen y_2.
CHANY3 i	Val av kanal för mätsignalen y_3.
CHANU i	Val av kanal för styrsignalen u.
TSAMP r	Val av samplingintervall.
HPT r	Val av tidshorisont för plottningen.
ULIM r r	Val av styrsignalbegränsningar.
NUMINP i	Val av antal insignaler som skall läsas in med AD-omvandlare.

Muskommandon

SAVE	Lagra de aktuella data som syns på skärmen.
BETWEEN	Val av plottning mellan samplingsintervallen.
EVERY	Val av plottning varje samplingsintervall.
EVERY2	Val av plottning vart annat samplingsintervall.
EVERY4	Val av plottning vart fjärde samplingsintervall.

MODEL-meny

Här matas en samplad processmodell in till programmet.

$$H(q) = \frac{B(q)}{A(q)}$$

För polynomen $B(q)$ och $A(q)$ matas deras koeffecienter in.

Kommandon från tangentbord

A p	Val av $A(q)$.
B p	Val av $B(q)$.

Muskommandon

RUN/STOP	Starta/Stoppa regulatorn.
SAVE	Lagra de aktuella data som syns på skärmen.
SAMPLE	Genväg för laborationen. Beräknar och lagrar $H(q)$ för vårt servo. Funktionen utnyttjar härvid samplingsintervallet som valdes på SETUP-menyn.

DESIGN-meny

Här väljer man parametrar för och utför design enligt kapitel 10.5 i Computer Controlled Systems.

Kommandon från tangentbord

BPLUS p	Val av $B^+(q)$.
BMINUS p	Val av $B^-(q)$.
BMPRIM p	Val av $B'_m(q)$.
AM p	Val av $A_m(q)$.
AO p	Val av $A_o(q)$.
CSAM r (r (r))	Val av $A_m(q)$ utifrån tidskontinuerliga specifikationer. Polynomets ordning beror på antalet parametrar. En parameter ω ger samplad version av $(s+\omega)$. Två parameter ω ζ ger samplad version av $(s^2+2\zeta\omega s+\omega^2)$. Tre parameter ω ζ α ger samplad version av $(s+\alpha\omega)(s^2+2\zeta\omega s+\omega^2)$.
CSAO r (r (r))	Som CSAM men istället defineras $A_o(q)$.

Muskommandon

RUN/STOP	Starta/Stoppa regulatorn.
COMPUTE	Utför design-beräkningarna.
TOREG	Utför design-beräkningarna och för över polynomen till den körande regulatorn.
INTON	Val av integrerande regulator.
INTOFF	Val av icke integrerande regulator.

REGUL-meny

Här väljs regulatorpolynomen direkt.

Kommandon från tangentbord

R p	Val av $R(q)$.
S1 p	Val av $S_1(q)$.
S2 p	Val av $S_2(q)$.
S3 p	Val av $S_3(q)$.
T p	Val av $T(q)$.
AO p	Val av $A_o(q)$.

Muskommandon

RUN/STOP	Starta/Stoppa regulatorn.
SAVE	Överför polynom från väntande till körande regulator.

SIGNAL-meny

Här väljer man parametrar för referenssignalen.

Kommandon från tangentbord

MEAN r	Val av medelvärde för intern referenssignal.
AMPLITUDE r	Val av amplitud för intern referenssignal.
PERIOD r	Val av periodtid för intern referenssignal.

Muskommandon

RUN/STOP	Starta/Stoppa regulatorn.
SAVE	Lagra de aktuella data som syns på skärmen.
EXTERN	Val av extern referenssignal.
SQUARE	Val av fyrkantvåg som referenssignal.
TRIANGLE	Val av triangelvåg som referenssignal.
SINE	Val av sinusvåg som referenssignal.
STEP	Val av steg som referenssignal. (Single shot)
RAMP	Val av ramp som referenssignal. (Single shot)

PLOT-meny

Här väljer man vilka signaler som man plottar i denna meny.

Muskommandon

RUN/STOP	Starta/Stoppa regulatorn.
RY1	Plotta signalerna r och y_1.
RY2	Plotta signalerna r och y_2.
RY3	Plotta signalerna r och y_3.
RY12	Plotta signalerna r, y_1 och y_2.
RY13	Plotta signalerna r, y_1 och y_3.
RY23	Plotta signalerna r, y_2 och y_3.
RY123	Plotta signalerna r, y_1, y_2 och y_3.

Bra att veta

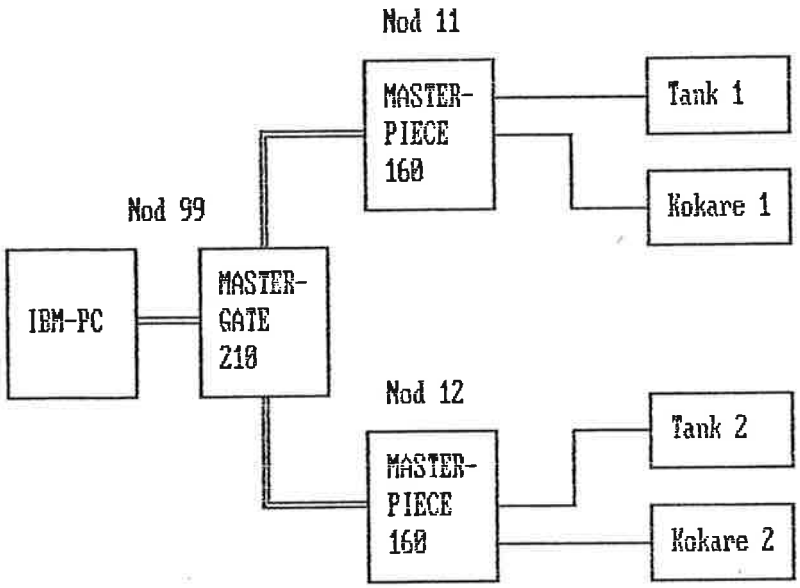
Programmet startas med kommandot TOOLBOX.

Programmet är ganska nyskrivet så buggar kan förekomma. Ni kan väl vara vänliga att skiva ner de buggar ni upptäcker så att de kan rättas.
Räknetidsproblem kan uppkomma då man samplar med korta samplingsintervall. Då man plottar signaler mellan samplingstillfällena uppträder problemen tidigare. Använd därför vanligen steg som referenssignal så kommer programet att hinna med att sampla tillräckligt fort.

Kommersiella processdatorer

Laboration 4.

Målet med denna laborationen är att ni skall lösa en styruppgift med en kommersiell processdator. I uppgiften ingår programmering, testning och man-maskin-kommunikation. En översikt över laborationsutrustningen visas i figur 1. Laborationen förbereds på övningarna 6 och 7.



Figur 1. Översikt över laborationssystemet

Asea's MasterPiece-system

Uppgiften löses med en MasterPiece 160. Man-maskin-kommunikationen sker med programmet PC-Operatör, vilket går på en IBM-PC. Till vårt förfogande finns två MasterPiece-enheter och MasterGate-enhet. Den senare samordnar och styr kommunikationen. Via MasterGate-enheten kan PC-Operatör kommunicera med MasterPiece-enheterna. Det blir sålunda gemensam man-maskin-kommunikation för de bägge MasterPiece-enheterna.

För att programmera MP160 används hjälpprogrammet MA120 (MasterAid 120) vilket också går på IBM-PC. Observera dock att vid programmering måste en kabel anslutas direkt mellan IBM-PC och MP160-enheten. Utdrag ur Asea's dokumentation om MP-systemet och handhavandet av MA120 bifogas.

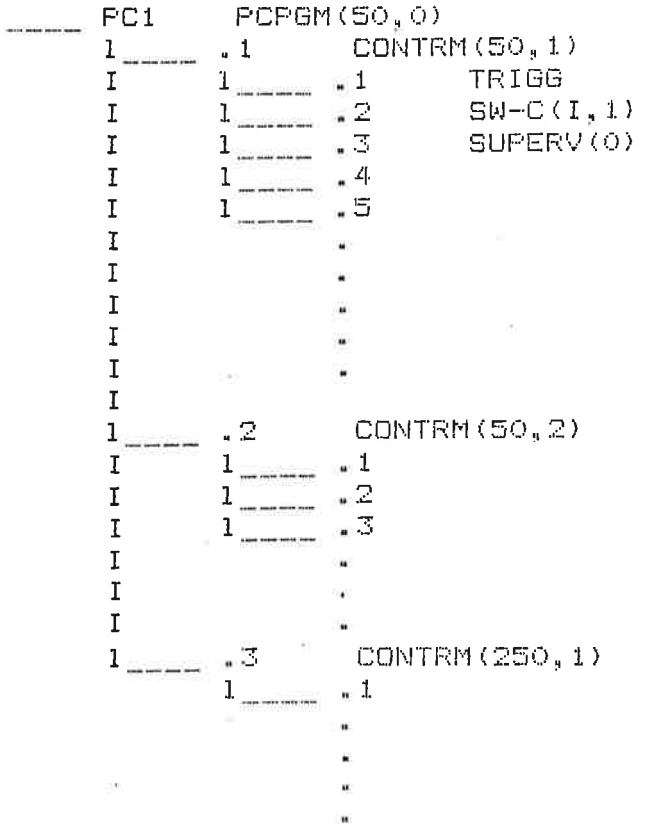
Programspecifikation

Programmet skall lösa 3 uppgifter.

1. Sekvensstyrning av tekokarprocessen enligt lab 1.
2. PI-reglering av en tank (jämför med lab 2).
3. Kommunikation med IBM-PC. Via denna länk görs alla operatörsingrepp, t.ex. start av sekvens; börvärde, förstärkning etc. till tankreglering.

Programstruktur

Programmet kan struktureras enligt figur 2.



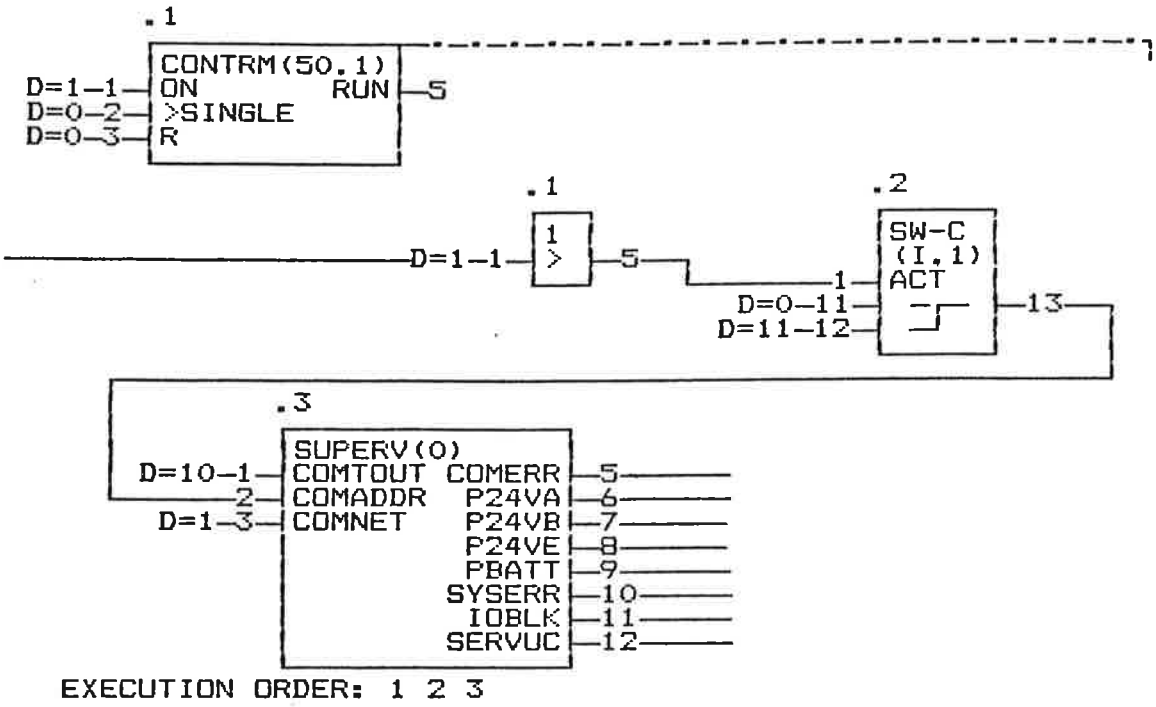
Figur 2. Programstruktur

Initialisering av kommunikation

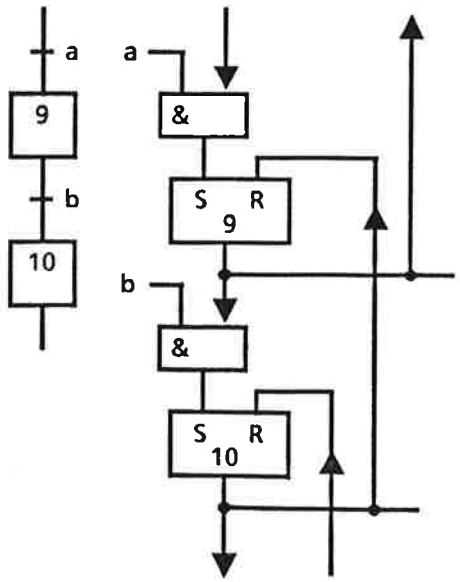
Med ett SUPERV()-element talar en MP160 om vilket nodnummer den har. Av ett tekniskt skäl måste nodnumret först sättas till noll, sedan till det 'verkliga'. Problemet löses med programdelen i figur 3.

Sekvensstyrning

Liksom i PLC-systemet sköts sekvensstyrningen med SR-vippor. Figur 4 visar principen för tillståndsövergångar. Villkoren för dessa kan ni hämta direkt från laboration 1. Koden för sekvensstyrning skall placeras under PC1.2 = CONTRM(50,2). Via kommunikationen skall ingången ON och R till PC1.2 kunna påverkas så att sekvensstyrningen kan stoppas helt och alla signaler



Figur 3. Initialisering för nod 11. Nod 12 hanteras på motsvarande sätt.

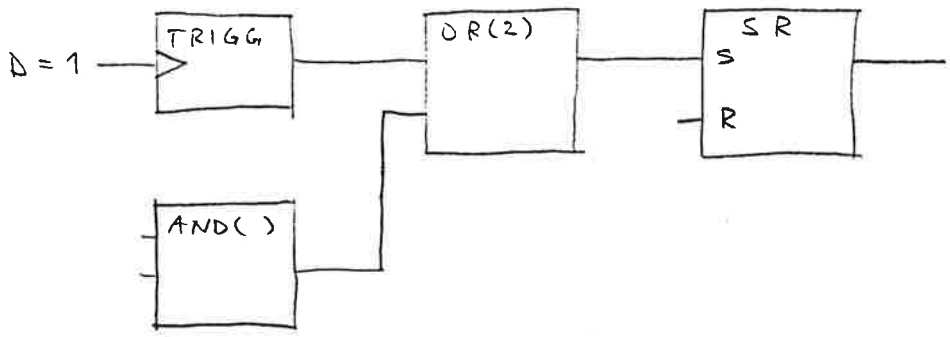


Figur 4. Tillståndsövergång

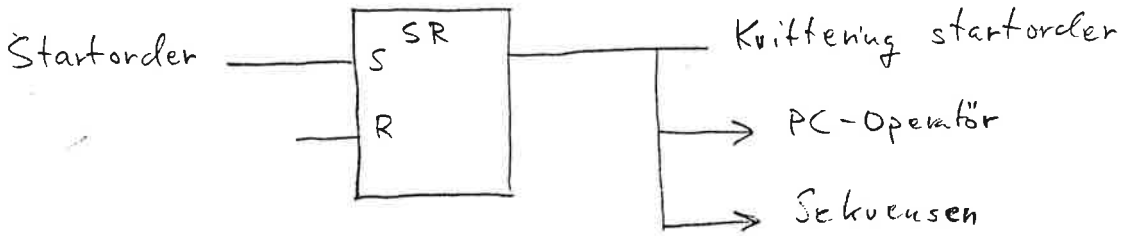
nollställs. Dess utgång, RUN, övervakas av PC-Operatör som skall visa status för styrningen.

Initialisering av första tillståndet i sekvensen

Liksom i PLC-systemet måste första tillståndet initialiseras till 1. Detta görs med samma teknik som i figur 3. Vid start har alla element initialtillstånd noll. Med TRIGG-elementet kommer då första tillståndet att ettställas. Via AND-elementet sker sedan den normala ettställningen av tillståndet, se figur 5.



Figur 5. Initialisering av första tillståndet



Figur 6. Start av en cykel sker via en SR-vippa

Start av en cykel

Start av en cykel i sekvensen skall ske enligt figur 6. PC-Operatör skickar signalen "Startorder" för att starta en cykel i sekvensen, samt övervakar kvitteringen. Kvitteringssignalen används sedan som direkt startsignal för sekvensen. Med hjälp av reset-ingången på SR-vippan skall kvitteringen nollställas automatiskt (dvs av programmet). Detta observeras då också av PC-Operatör.

PI-reglering

Använd PI-elementet. Observera att ingången TI (integrationstid) kräver datatypen TR, vanlig R går inte.

Koden placeras under PC1.3 = COMTRM(250,1). ON- och R-ingången till CONTRM-elementet skall kunna påverkas med kommunikationen så att regleringen stoppas helt och utsignalerna nollställs då signalen "Reglering till" är nollställd. Dess utgång, RUN, övervakas av PC-Operatör som skall visa status för regleringen. Vidare skall Reset på integraldel, med flera signaler i kommunikationen anslutas till PI-elementet.

Kommunikation

Med elementen EXT-I och EXT-O kan MP-systemet kommunicera med t.ex. IBM-PC eller andra MP-enheter. Vid kommunikation med externa datorer gäller vissa begränsningar i hur signalen får överföras. Följande konventioner används i laborationen.

Kommunikationsnätet (NET) har nummer 1. Övre MP160-enheten är nod 11, dess undre nod 12. Kommunikation till IBM-PC riktas till nod 99, informa-

tionen lagras då i MasterGate-enheten.

Information paketeras i ett "Dataset", som får ett visst nummer. Vid kommunikation med IBM-PC får högst 5 signaler transporteras i ett dataset. Alla måste vara av samma typ, antingen R eller IL. Upp till 32 Booleska variabler kan packas ihop till en IL-signal, som sedan kan skickas iväg. På motsvarande sätt kan en IL-signal delas upp i högst 32 Booleska variabler. Till detta används elementen PACK resp. UNPACK.

Kommunikation till/från nod 11

För laborationen används följande konvention.

Reella insignaler kommer från elementet EXT-I(41,0,4), vars utgångar innehåller följande.

Signal nr	Signal
11	Börvärde
12	Förstärkning
13	Integrationstid
14	Manuell styrsignal

Logiska insignaler kommer från EXT-I(11,1,0) och UNPACK(5), vars utgångar innehåller följande.

Signal nr	Signal
11	Sekvenstyrning till
12	Startorder
13	Reglering till
14	Reset på integration
15	Manuell reglering

"Sekvenstyrning till" startar PC1.2 och "Reglering till" startar PC1.3.

Logiska utsignaler skall anslutas till PACK(13) och EXT-O(21,1,0,0) enligt följande.

Signal nr	Signal
11	Sekvensstyrning exekvererar
12	N1
13	N2
14	N3
15	T
16	Pump
17	Ventil
18	Värme
19	Kvittering startorder
20	Reglering exekverar
21	Övre begränsning
22	Undre begränsning
23	PI-Error

11-19 hör till sekvensstyrningen och 20-23 hör till PI-regleringen. "Sekvensstyrning exekverar" kvitterar att PC1.2 exekverar och "Reglering exekverar" kvitterar att PC1.3 exekverar.

Sätt "dödbandet" till 1, se dokument om EXT-O.

Reella utsignaler skall anslutas till EXT-O(51,0,3,0) enligt följande

Signal nr	Signal
11	Styrsignal
13	Tanknivå
15	Reglerfel

Sätt "dödbandet" till 0.05, se dokumentationen.

Kommunikation till/från nod 12

För nod 12 gäller datasetnummer 42, 12, 22 respektive 52 istället för ovanstående. I övrigt samma som för nod 11.

In- och utsignaler till processen

Digitala insignaler ansluts till korttyp 1 med cykeltid 50 ms enligt följande.

Position	Signal
1.3.1	N1
1.3.2	N2
1.3.3	N3
1.3.4	T

Digitala utsignaler ansluts till korttyp 10 enligt följande.

Position	Signal
1.4.1	Pump
1.4.2	Ventil
1.4.3	Värme

Analoga insignaler ansluts till korttyp 23 med cykeltid 250 ms enligt följande.

Position	Signal
1.1.1	Tanknivå

Analoga utsignaler ansluts till korttyp 24 enligt följande.

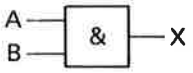
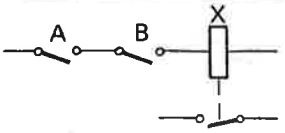
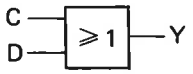
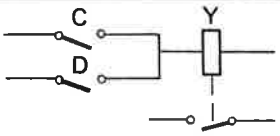
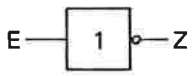
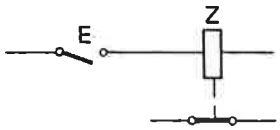
Position	Signal
1.1.1	Styrsignal till pump

De analoga in/ut-korten arbetar mellan ±10 Volt, men tanken använder bara 0-10 Volt. Internt skall 0 Volt (tom tank respektive pumpen stilla) motsvaras av 0.0 och 10 Volt (full tank respektive högsta pumphastighet) av 1.0. Valet av CONV_PARAM påverkar valen av RANGE-värden och LIMIT-värden, se dokumentationen om AI- och AO-korten.

2. Lösningar och assistenthandledningar

I den mån lösningar och assistenthandledningar har tagits fram så redovisas de på följande sidor.

De vanligaste logiska operationerna framgår av följande tabell.

Logisk operation	Booleskt samband	Logiksymbol	Reläkoppling
OCH (AND)	$X = A \cdot B$		
ELLER (OR)	$Y = C + D$		
INTE (NOT)	$Z = \overline{E}$		

1-2 INTERNAL CONFIGURATION OF PC

The PC is micro processor based, but can be equivalently regarded as an aggregate comprising general relay, timer, counter, etc.

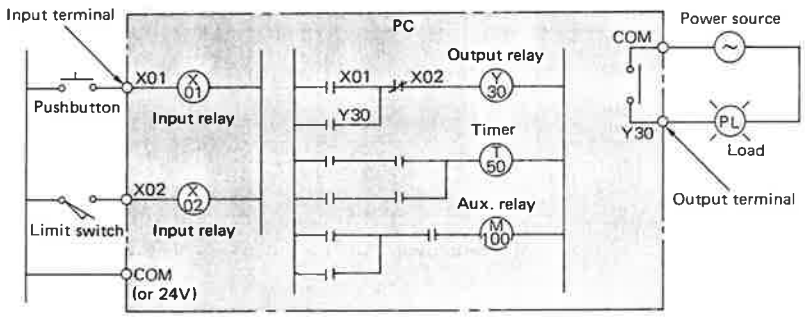
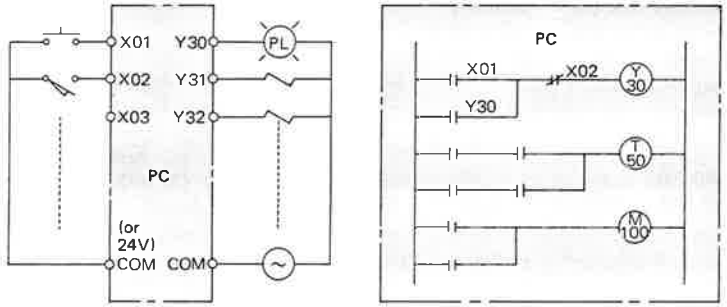


Fig. 1.2 Equivalent circuit of PC

The input/output terminal is prepared as a port for the PC to exchange the signal with the external sensor or load.
The input relay (X) built in the PC is driven by the external switch through the input terminal.
The input relay coil is prepared in DC24V rating, AC100V rating, AC200V rating according to the PC models.
The contact of the output relay (Y) in the PC is connected to the output terminal, and so designed as to drive the external load.
The output relay used comes in mechanical contact type and non-contact type (SSR, transistor).
In addition to the external output contact (1a), the PC is provided with multiple normally-open contacts and normally-close contacts which can be used in the PC.
Furthermore, the PC is incorporated with various elements such as timer (T), auxiliary relay (M), counter (C), etc.
Each of these elements (input relay-X, output relay-Y, Auxiliary relay-M, timer-T, counter-C, etc.) consists of several electronic normally-open contacts (a-contact) and normally-close contacts (b-contact) which can be used optionally within the PC.

1-3 GENERAL CONCEPT OF PROGRAM AND INSTRUCTION

When designing the control panel using the PC, the circuit shown in Fig. 1.2 can be broken down to the circuit as shown in Fig. 1.3.



(a) Input/output wiring diagram **(b) Ladder diagram**

Fig. 1.3 Break-down of equivalent circuit

Fig. 1-3-(a) shows the allotment and wiring of input/output.
The wiring needs to be executed with wires by the use of a screwdriver, pliers, etc. in the same manner as the conventional relay panel.
On the other hand, Fig. 1-3-(b) shows the relative connection of each element within the PC.
The connection for these contacts and coils can be executed through key operation by the use of the progtamming panel.
In Fig. 1-3-(b), for instance, the output coil Y30 is driven by the parallel circuit comprising the contact X01 of the input relay and other contact Y30 of the output relay, and series circuit made up of normally-close contact X02 of the input relay X02.
To perform the key operation for the above connection example by the programming panel, it is necessary to provide the instructions corresponding to series connection/parallel connection of contact, normally-open contact, normally-close contact, etc.
In this case, it is also necessary to provide the division of each element and element number at the same time.
In this respect, the instruction used for the PC are made up in combination of instruction showing the connection procedure, and element number.
These instructions are integrated to form a program.

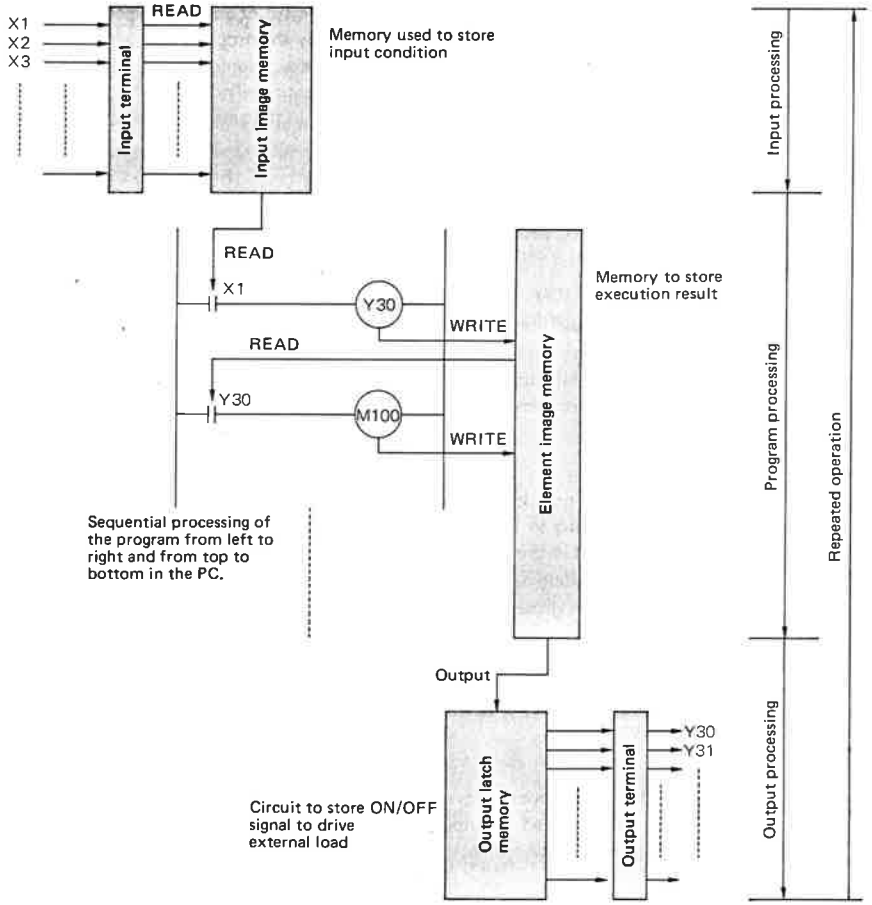


Fig. 1.4 Input/output and program processings

(2) Input/output Response Lagging

The PC may have response lagging due to influence by operation cycle in addition to the electrical lagging due to input filter or the mechanical lagging of output relay.
As an example, such a case is to be taken, where the input terminal X0 is changed from OFF to ON immediately after the input processing has been completed in the sequence as shown in Fig. 1.5.

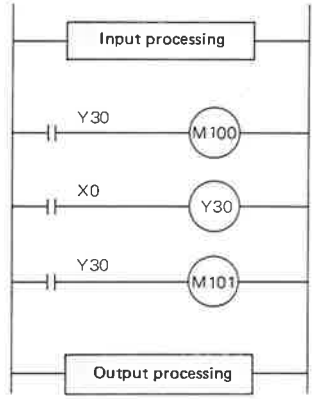


Fig. 1.5 Input/output response lagging

- **1st cycle**
  X0 of the input image memory is turned off, therefore, all the Y30, M100 and M101 are turned off consequently.

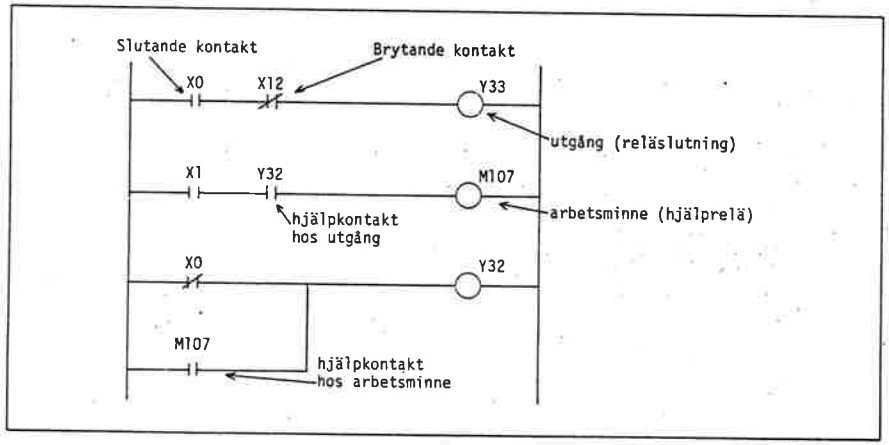
- **2nd cycle**
  With the input processing started, X0 of the input image memory will be turned on.
  M100 cannot be turned on as contact Y30 is still kept turned off.
  Coil Y30 will be turned on, as the input image memory X0 is turned on, however, the actual output will not be turned on yet, as the element image memory of Y30 is turned on at the moment.
  M101 will be turned on, as the image memory of Y30 is turned on.
  As the output processing is executed, the actual output Y30 of the PC will be turned on.
  Therefore, Y30 and M101 will be lagged in response by a maximum of two cycles.

- **3rd cycle**
  M100 will be turned on as the image memory of Y30 has been turned on. As described above, the M100, etc. of which contact (Y30) has been programmed before the coil (Y30) will operate in one cycle lagging.

1.2 Programmering

Underlag

Som underlag för programmering använder man t.ex. ett reläschema, där funktionen i styrsystemet beskrivs. Vid den elektriska inkopplingen till PC-systemet identifieras givare och manöverorgan med var sitt oktalt kanalnummer. För att förtydliga beteckningarna inleds kanalnumret med ett X för ingångarna och ett Y för utgångarna.

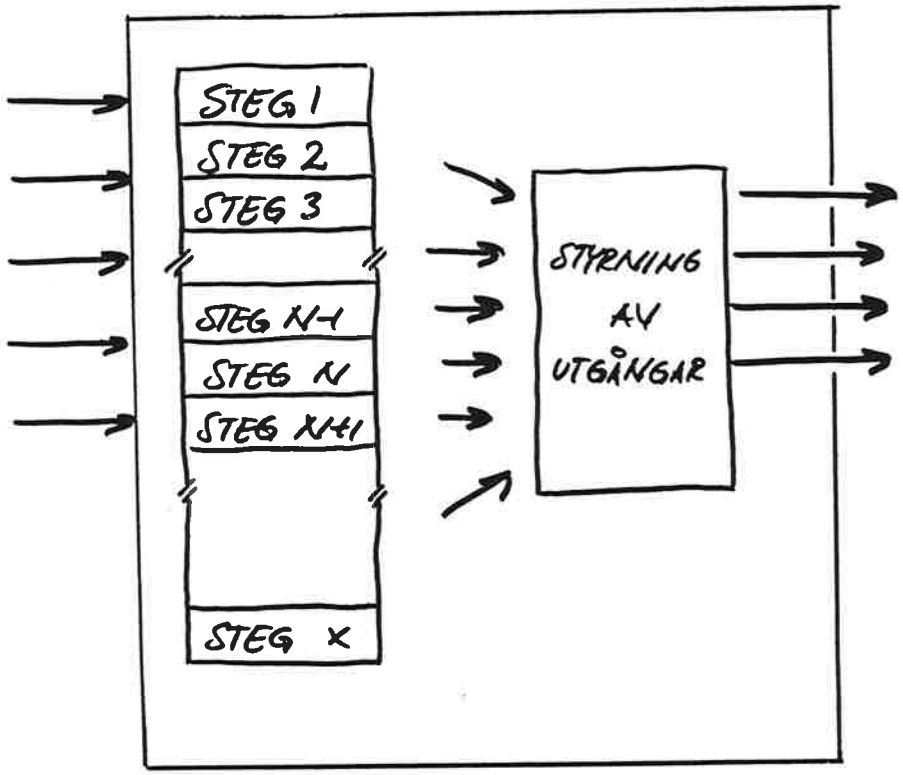


En givare ansluts normalt bara till en ingång, men kan trots detta användas som flera kontaktvillkor i reläschemat. Det är dessutom tillåtet att utnyttja statusen hos en utgång som ett kontaktvillkor, utan att någon yttre hjälpkontakt är ansluten till PC-systemet. Båda dessa funktioner blir möjliga genom den minnesbaserade funktionen i PC-systemet.

I ett reläsystem har man ofta behov av hjälpreläer som minneselement för olika tillstånd. PC-systemet utnyttjar för detta ändamål särskilda interna arbetsminnen, som fungerar som utgångar utan yttre kontakter. Totalt finns 64 olika arbetsminnen, som identifieras med oktala nummer från M100 till M177. Ett arbetsminne kan användas som kontaktfunktion på godtyckligt antal platser i reläschemat (obegränsat antal hjälpkontakter).

BEIJERS
BEIJER ELECTRONICS

SEKVENSSTYRNING

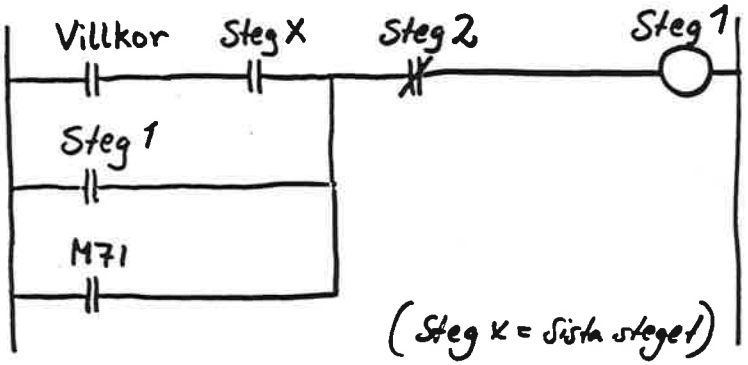


G. & L. Beijer AB, Box 325, 201 23 Malmö
Telefon 040/735 60

BEIJERS
BEIJER ELECTRONICS

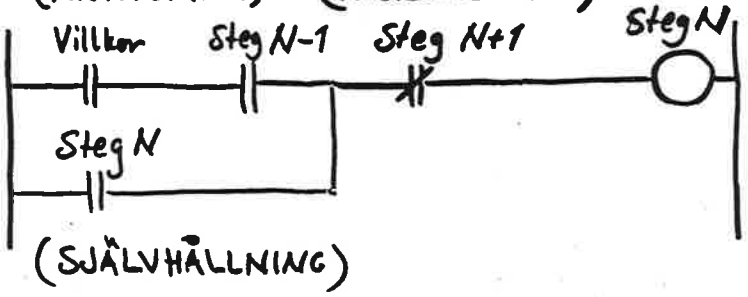
SEKVENSKEDJA

STEG 1



STEG N

(AKTIVERING) (ÅTERSTÄLLNING)



(SJÄLVHÅLLNING)

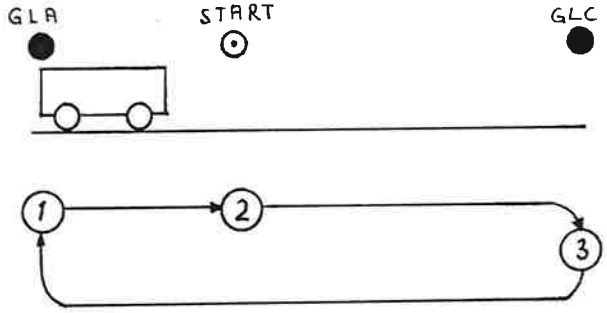
G. & L. Beijer AB, Box 325, 201 23 Malmö
Telefon 040/735 60

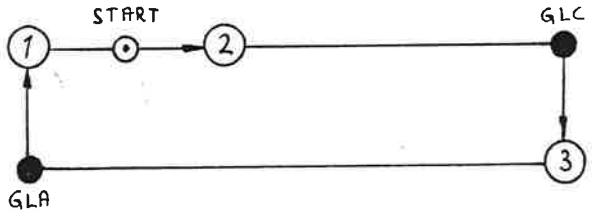
SEKVENSSTYRNING

En styrfunktion kan oftast utföras som en sekvensstyrning.

De flesta sekvenserna kan delas in i ett antal tillstånd som alltid avverkas i samma följd. Endast ett tillstånd är aktivt åt gången. Övergången från ett tillstånd till nästa initieras av en givarsignal från processen.

Exempel:

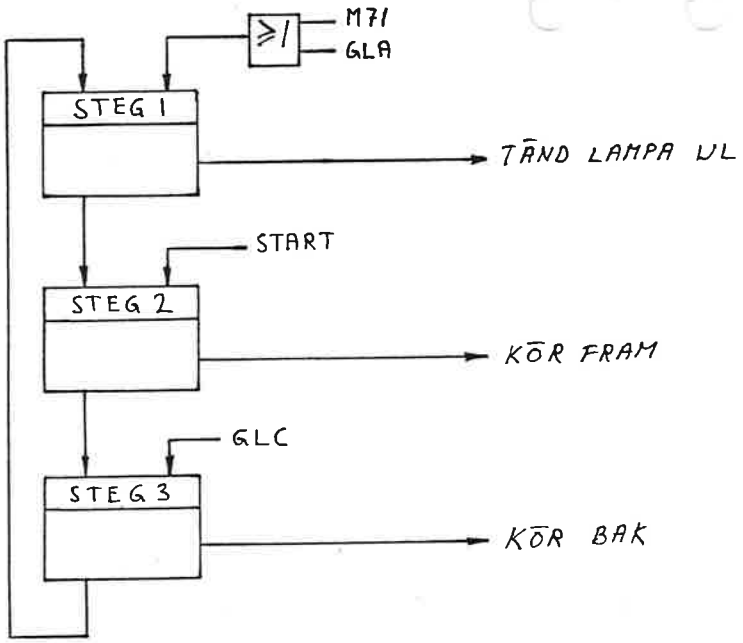


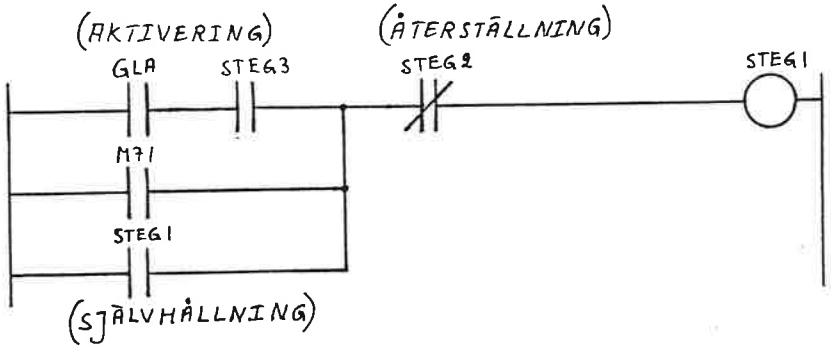


1 = Hemmaläge

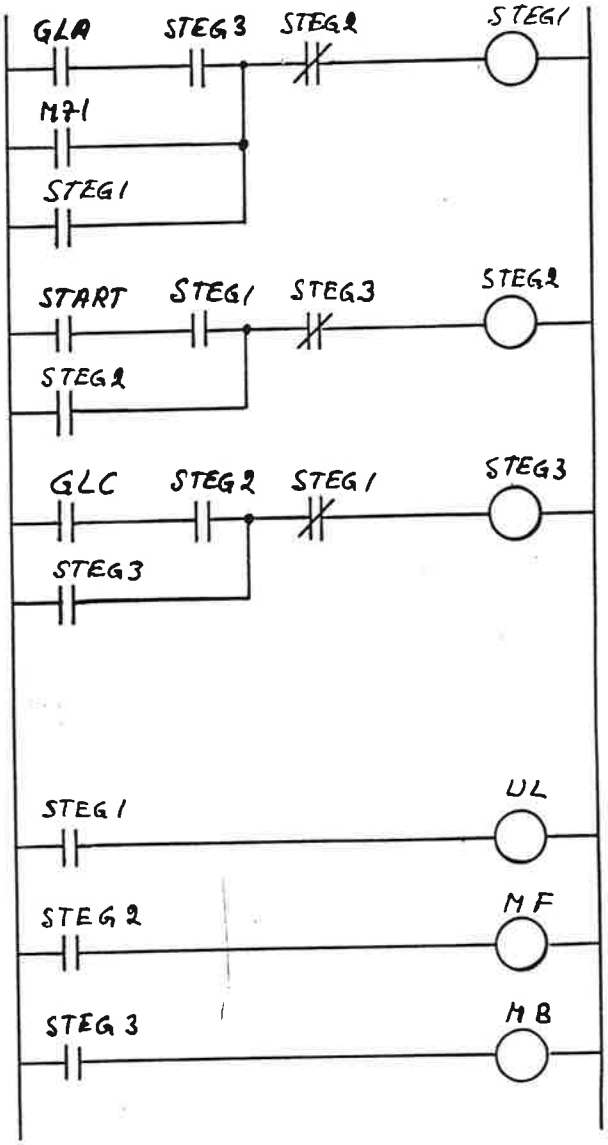
2 = Kör fram

3 = Kör bak





Självhållning krävs för att bli kvar i ett tillstånd även om givarsignalen försvinner. Återställning sker vid övergång till nästa tillstånd.



1. Assistenthandledning till övning 2

På föreläsning genomgång av Modula-2, Kernel, mfl moduler. Uppgifterna kräver att programmet har 2 processer

1. Regulatorprocess, med proportionell regulator $u = K * (y_r - y)$.
2. Operatörskommunikation, för förstärkning K och referensvärde y_r.

Således K och y_r gemensamma variabler, vilka kräver ömsesidig uteslutning för referenser. Använd semafor från Kernel.

Programskiss

```
MODULE Main;
  FROM ... IMPORT .... ;
  (* egna deklarationer, tex k, ref *)
  (* Process *) PROCEDURE Regulate;
        (* deklarationer och satser *)
  END Regulate;
  (* Process *) PROCEDURE ReadTerminal;
        (* deklarera  lokala variabler för k, ref *)
        (* satser *)
  END ReadTerminal;
  BEGIN
    (* Initialiseringar av variabler *)
    InitKernel;
    InitSem(...)
    CreateProcess(Regulate, 2000);
    CreateProcess(ReadTerminal, 2000);
    SetPriority(MaxPriority);
END Main.
```

I **Regulate** är det inte nödvändigt att lagra lokala variabler om de globala finns direkt gripbara. Regleralgoritmen kan då se ut på följande sätt.

```
y=ADIn(..);
Wait(mutex);
u=k*(ref-y);
Signal(mutex);
DAOut(..);
```

Vid anropet av **InitKernel** skapas ett antal processer.

1. Huvudprogrammet blir en process med prioritet 1.

2. Processen **Idle** skapas med prioritet **MaxPriority-1**.

3. Klockprocess med flera

Vid anropet av **CreateProcess(p1, 2000)** skapas en process **p1** med 2000 bytes egen dataarea och prioritet 2. Huvudprocessen kan därför starta alla andra processer utan att bli avbruten. När detta är klart måste den sänka sin prioritet till **MaxPriority**, eller bli väntande på annat sätt. Alla övriga processer skall också, efter att ha initialiserat sina variabler, sänka sina prioriteter till åtminstone 10.

Vid övningen

Ta upp att två processer är nödvändiga. Dra programskissen. Förklara vad som händer vid anropen **InitKernel** och **CreateProcess**, och att prioriteterna hos processerna måste sänkas. Låt dem själva fundera över prioriteter för **Regulate** respektive **ReadTerminal**. Ge inte för detaljerade instruktioner till uppgifterna. Låt eleverna tänka själva. Svara på frågor. Lämna frihet att utforma operatörskommunikationen.

Dessutom

bifogas lösningar till 2 uppgifter i Tillämpad Realtidsprogrammering. I dessa ändras endast K med operatörskommunikation. I den andra ligger gemensamma variabler i en Monitor-modul.

```
MODULE Skeleton;
FROM Kernel   IMPORT
  InitKernel, CreateProcess, SetPriority, MaxPriority,
  Time, IncTime, GetTime, WaitUntil, Semaphore, Wait, Signal, InitSem;
FROM AnalogIO IMPORT ADIn, DAOut;
FROM Terminal IMPORT ReadString, WriteString, WriteLn;
FROM ConvReal IMPORT StringToReal, RealToString;

VAR  mutex: Semaphore;
     K:     REAL;          (* Global data to be protected *)

PROCEDURE SetPar(value: REAL);
BEGIN

END SetPar;

PROCEDURE GetPar(VAR value:REAL);
BEGIN

END GetPar;

(* ------------------------------------------- *)

(* Process *) PROCEDURE Regul;
VAR  next:      Time;
     y,ref,u,K: REAL;
BEGIN

  SetPriority(..);
  GetTime(next);    (* Get current system time in ms *)
  LOOP

    y:=ADIn(0);

    DAOut(0,u);

    IncTime(next,1000);
    WaitUntil(next);     (* compare WaitTime(100); *)
  END;
END Regul;

(* Process *) PROCEDURE Reader;
VAR  row:  ARRAY [0..79] OF CHAR;
     K:    REAL;
     pos:  CARDINAL;
BEGIN

  SetPriority(..);
  LOOP

    WriteLn; WriteString('K= '); ReadString(row);
    pos:=0; StringToReal(row,pos,K);
    IF ( pos <> 0 ) THEN  SetPar(K)  END;

  END;
END Reader;
```

```
BEGIN (* Skeleton *)
  InitKernel;
  InitSem(mutex,1);                (* Initialisation of global data *)
  K:= 0.0;
  CreateProcess(Regul,5000);
  CreateProcess(Reader,5000);
  SetPriority(MaxPriority);    (*  Suspends main process *)
END Skeleton.
```

Lösningar till

Övning 4. PID-regulatorer

Uppgift 4.1

Integraldelen med framåtdifferenser:

$$I(s) = \frac{K}{sT_i}E(s) \rightarrow i(k) = \frac{K \cdot h}{(q-1)T_i}e(k) = H_{if}(q)e(k)$$

eller

$$i(k) = i(k-1) + \frac{K \cdot h}{T_i}e(k-1)$$

Bakåtdifferenser:

$$I(s) = \frac{K}{sT_i}E(s) \rightarrow i(k) = \frac{K \cdot q \cdot h}{(q-1)T_i}e(k) = H_{ib}(q)e(k)$$

eller

$$i(k) = i(k-1) + \frac{K \cdot h}{T_i}e(k)$$

Derivatadelen med framåtdifferenser:

$$D(s) = \frac{K \cdot T_d \cdot s}{1 + s \cdot T_f}E(s) \rightarrow d(k) = \frac{K \cdot T_d\frac{q-1}{h}}{1 + T_f\frac{q-1}{h}}e(k) = \frac{K \cdot T_d(q-1)}{h + T_f(q-1)}e(k)$$

Med $T_f = T_d/G_d$ så blir

$$d(k) = K \cdot G_d\frac{q-1}{q-1+\frac{h \cdot G_d}{T_d}}e(k) = H_{df}(q)e(k)$$

eller

$$d(k) = \Big(1 - \frac{h \cdot G_d}{T_d}\Big)d(k-1) + K \cdot G_d\Big(e(k) - e(k-1)\Big)$$

Bakåtdifferenser ger pss:

$$d(k) = \frac{K \cdot T_d \cdot G_d}{(h \cdot G_d + T_d)}\frac{(q-1)}{(q - \frac{T_d}{h \cdot G_d + T_d})}e(k) = H_{db}(q)e(k)$$

eller

$$d(k) = \frac{T_d}{h \cdot G_d + T_d}d(k-1) + \frac{K \cdot T_d \cdot G_d}{h \cdot G_d + T_d}\Big(e(k) - e(k-1)\Big)$$

D-delen (ofiltrerad) med bakåtdifferenser:

$$d(k) = K \cdot T_d\frac{q-1}{q \cdot h}e(k) = H_d(q)e(k)$$

eller

$$d(k) = \frac{K \cdot T_d}{h}(e(k) - e(k-1))$$

Analys:

I-delen: Vid framåtdifferens ingår inte $e(k)$ i $i(k)$, vilket den däremot gör vid bakåtdifferens.

D-delen: Bakåtdifferens alltid stabil om G_d, $h > 0$. Framåtdifferens stabil bara om $0 < h \cdot G_d < 2T_d$.

Notera att I-delen alltid ligger på stabilitetsgränsen och att D-delen utan filter saknar dynamik.

Uppgift 4.2

$$U(s) = K \cdot E(s) + \frac{K}{sT_i}E(s) + K \cdot T_d \cdot s\, E(s) = P(s) + I(s) + D(s)$$

$$\rightarrow u(k) = K \cdot e(k) + i(k) + d(k)$$

där $i(k)$ och $d(k)$ ges i föregående uppgift.

Kod kan då se ut som följer:

```
PROCEDURE Regulator;
  VAR next: Time;
BEGIN
  e1:=0; (* gamla reglerfelet *)
  i:=0;  (* integraldel *)
  GetTime(next);
  LOOP
    (* läs in r, y *)
    e:=r-y;
    u:=K*e + i + K*Td/h*(e-e1);
    i:=i + K*h/Ti*e;
    e1:=e;
    (* läs ut u *)
    IncTime(next,h);
    WaitUntil(next);
  END;
END;
```

Uppgift 4.3

Derivatadelen behöver ett tillstånd som uppdateras före u beräknas.

```
d:=Td/(Td+Gd*h)*d+K*Td*Gd/(Td+Gd*h)*(e-e1);
u:=K*e+i+d;
```

d måste initialiseras före loopen.

Uppgift 4.4

$$u(k) = K \cdot e(k) + H_{if}(q)e(k) + H_{db}(q)e(k) = K\left(1 + \frac{h}{T_i}\frac{1}{q-1} + \frac{T_d}{h}\frac{q-1}{q}\right)e(k)$$

$$(q^2 - q)u(k) = K\left(q^2 - q + \frac{h}{T_i}q + \frac{T_d}{h}(q-1)^2\right)e(k) =$$

$$= K\left((1 + \frac{T_d}{h})q^2 + (\frac{h}{T_i} - 1 - 2\frac{T_d}{h})q + \frac{T_d}{h}\right)e(k)$$

Då $e = u_c - y$ följer att

$$S(q) = T(q) = K(1 + \frac{T_d}{h})q^2 + K(\frac{h}{T_i} - 1 - 2\frac{T_d}{h})q + K\frac{T_d}{h}$$
$$R(q) = q^2 - q$$

eller om man vill använda bakåtskiftoperatorn

$$S^*(q^{-1}) = T^*(q^{-1}) = K(1 + \frac{T_d}{h}) + K(\frac{h}{T_i} - 1 - 2\frac{T_d}{h})q^{-1} + \frac{T_d}{h}q^{-2}$$
$$R^*(q^{-1}) = 1 - q^{-1}$$

Beräkningstekniska aspekter

Väsentligen skall enbart den del i RST-algoritmen som beror senaste y och u_c beräknas mellan in- och utläsning. Resten är känt i förväg.

För PID-regulatorn i uppgift 4.2 räcker det att flytta utläsningen till omedelbart efter beräkningen av styrsignalen.

Uppgift 4.5

Tex kan loopen se ut som följer.

```
LOOP
  (* läs in uc och y *)
  u:=t0*uc-s0*y+state;
  (* läs ut u *)
  state:= t1*uc+t2*uc1-s1*y-s2*y1-r1*u-r2*u1;
  uc1:=uc; y1:=y; u1:=u;
  (* vänta ett samplingsintervall *)
END;
```

Loopen kan ingå i en procedur, men parametrar och tillstånd måste lagras utanför den proceduren.

Operatörskommunikation

Operatören hanterar enbart K, T_i, T_d, G_d och eventuellt h. Översättning till RST-parametrar sker då nya parametrar lagras. Alla sammansatta "parametrar" i PID-regulatorn i uppgift 4.2 skall beräknas i operatörskommunikationen.

Uppgift 4.6

För PID-regulatorn i uppgift 4.2 kan algoritmen ändras till

```
d:=alfa*d+beta*(e-e1);
u:=K*e+i+d;
i:=i + gamma*e;
```

där parametrarna beräknas i operatörskommunikationen enligt

```
alfa:=Td/(Td+Gd*h);
beta:=K*Gd*alfa;
gamma:= K*h/Ti;
```

Uppgift 4.7

$$
\begin{aligned}
v(k) =& a \cdot v(k-1) + t_0 u_c(k) + t_1 u_c(k-1) + t_2 u_c(k-2) - \\
& - s_0 y(k) - s_1 y(k-1) - s_2 y(k-2) - (a + r_1) u(k-1) - r_2 u(k-2) \\
u(k) =& \text{sat}(v(k))
\end{aligned}
$$

Lösningar till

Övning 5. Digital reglering av DC-servo

Uppgift 5.1

Ny karakteristisk ekvation

$$s^2 + (a + bKT_d)s + bK = s^2 + 2\zeta\omega s + \omega^2 = 0$$

Då blir

$$K = \frac{\omega^2}{b} \qquad T_d = \frac{2\zeta}{\omega} - \frac{a}{\omega^2} \approx \frac{2\zeta}{\omega}$$

dvs $K = 1.44$, $T_d = 0.23$, dvs $KT_d = 0.34$.

Uppgift 5.2

Alla polynomen ges här i framåtskiftoperatorn q. Observera att R skall vara moniskt, dvs högstagradskoefficienten skall vara 1. Vidare skall R, S och T ha samma gradtal för att inte få fördröjningar (eller icke-kausalitet) i styrlagen.

PD utan filter, bakåtapproximation

$$R = q \qquad S = T = K(1 + \frac{T_d}{h})q - \frac{KT_d}{h} =$$
$$= (1.44 + \frac{0.34}{h})q - \frac{0.34}{h}$$

Parametrar

h	r_1	s_0	s_1
0.04	0	9.94	-8.50
0.10	0	4.84	-3.40
0.15	0	3.71	-2.27
0.20	0	3.14	-1.70

PD med filter, framåtapproximation

$$R = q + \frac{5h - T_d}{T_d} \qquad S = T = 6Kq + K\frac{5h - 6T_d}{T_d}$$

Stabilitetskrav: $h > 0$ och $T_d > \frac{5}{2}h$. Med valt värde på T_d skall $h < 0.092$ s.

Parametrar

h	r_1	s_0	s_1
0.04	-0.13	8.64	-7.39
0.06	0.30	8.64	-6.76
0.08	0.74	8.64	-6.14
0.09	0.96	8.64	-5.82
0.10	1.17	8.64	-5.51

PD med filter, Tustin's approximation

$$R = q + \frac{5h - 2T_d}{5h + 2T_d} \qquad S = T = K\frac{5h + 12T_d}{5h + 2T_d}q + K\frac{5h - 12T_d}{5h + 2T_d}$$

Parametrar

h	r_1	s_0	s_1
0.04	-0.394	6.458	-5.585
0.10	0.042	4.890	-3.390
0.15	0.240	4.177	-2.392
0.20	0.370	3.708	-1.736

Tillståndsåterkoppling 1

Uppgift 5.3

Ny karakteristisk ekvation

$$s^2 + (l_2c_2 + a)s + c_1c_2l_1 = 0$$

vilket ger

$$l_1 = \frac{\omega^2}{c_1c_2} = \frac{\omega^2}{b} \qquad l_2 = \frac{2\zeta\omega - a}{c_2}$$

dvs l_1 blir samma som K för PD-regulatorn, men l_2 blir proportionell mot T_d.

I stationaritet ($\dot{x} = 0$) ser man att $x_2 = 0$ och $c_2l_1x_1 = c_2mr$, dvs $m = l_1$ ger stationär förstärkning 1.

Parametrar

ω	l_1	l_2
6	1.4631	3.1128
10	4.0642	5.2180
12	5.8525	6.2707

Tillståndsåterkoppling 2

Uppgift 5.4

$$\Phi = \begin{pmatrix} 1 & 0.9195 \\ 0 & 0.9881 \end{pmatrix} \qquad \Gamma = \begin{pmatrix} 0.1225 \\ 0.2644 \end{pmatrix}$$

Uppgift 5.5

$$(s - s_i)(s - s_j) = 0 \rightarrow (z - z_i)(z - z_j) = 0$$

Uppgift 5.6

Önskad karakteristisk ekvation

$$z^2 - 1.1953z + 0.4317 = 0$$

Detta ger

$$m = l_1 = 0.9666 \qquad l_2 = 2.5503$$

Utsignalåterkoppling

Uppgift 5.7

$$H(q) = \frac{0.123(q+1)}{q^2 - 1.988q + 0.988}$$

Poler: 1 och 0.988, nollställe: -1 (ca).

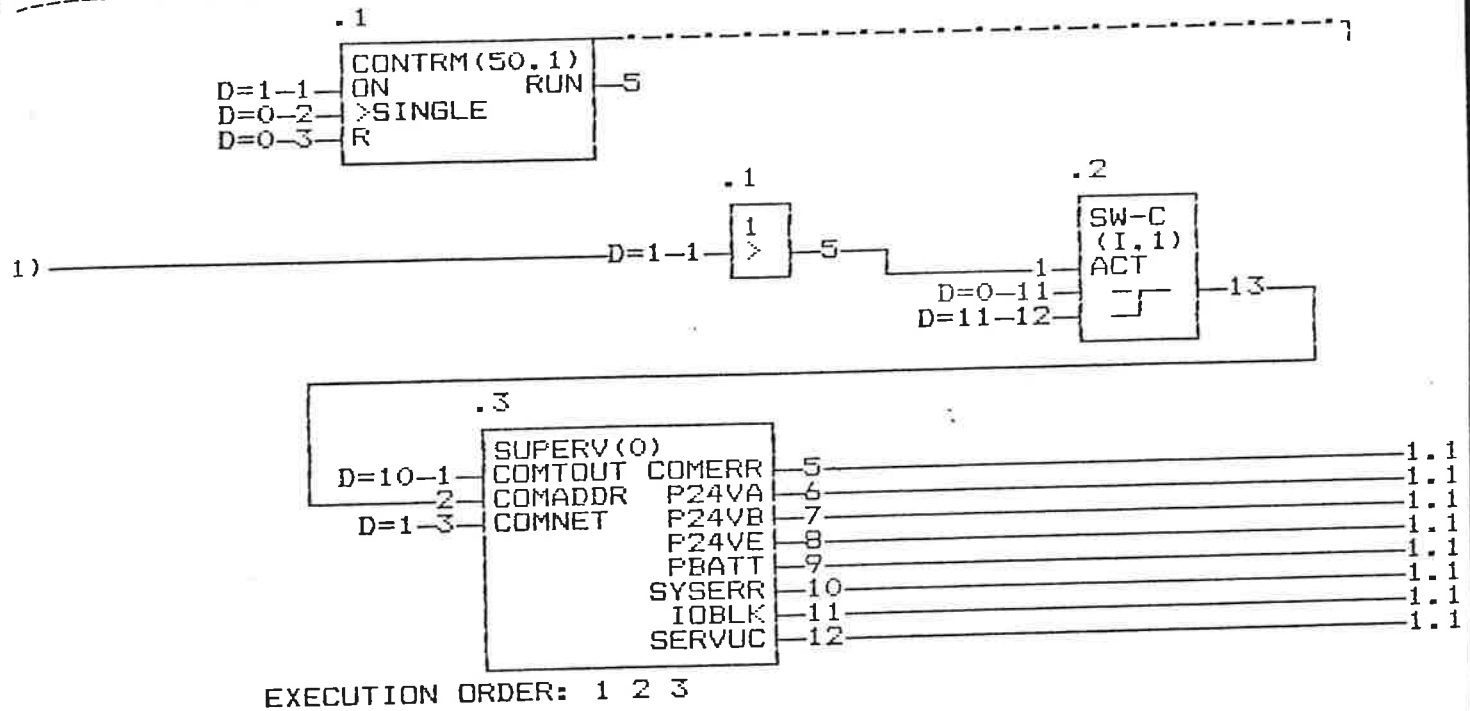
Uppgift 5.8

A_m väljs som ovan, $A_o = q - 0.368$. Detta ger

$$R = q + 0.096 \qquad S = 2.677q - 2.068 \qquad T = 0.963q - 0.354$$

INSTRUCTION	Datum	Reg.
Från	1987-01-27	
ILTP		
Utfärdare, tfn-nr		
Harald Davidsen	9439	

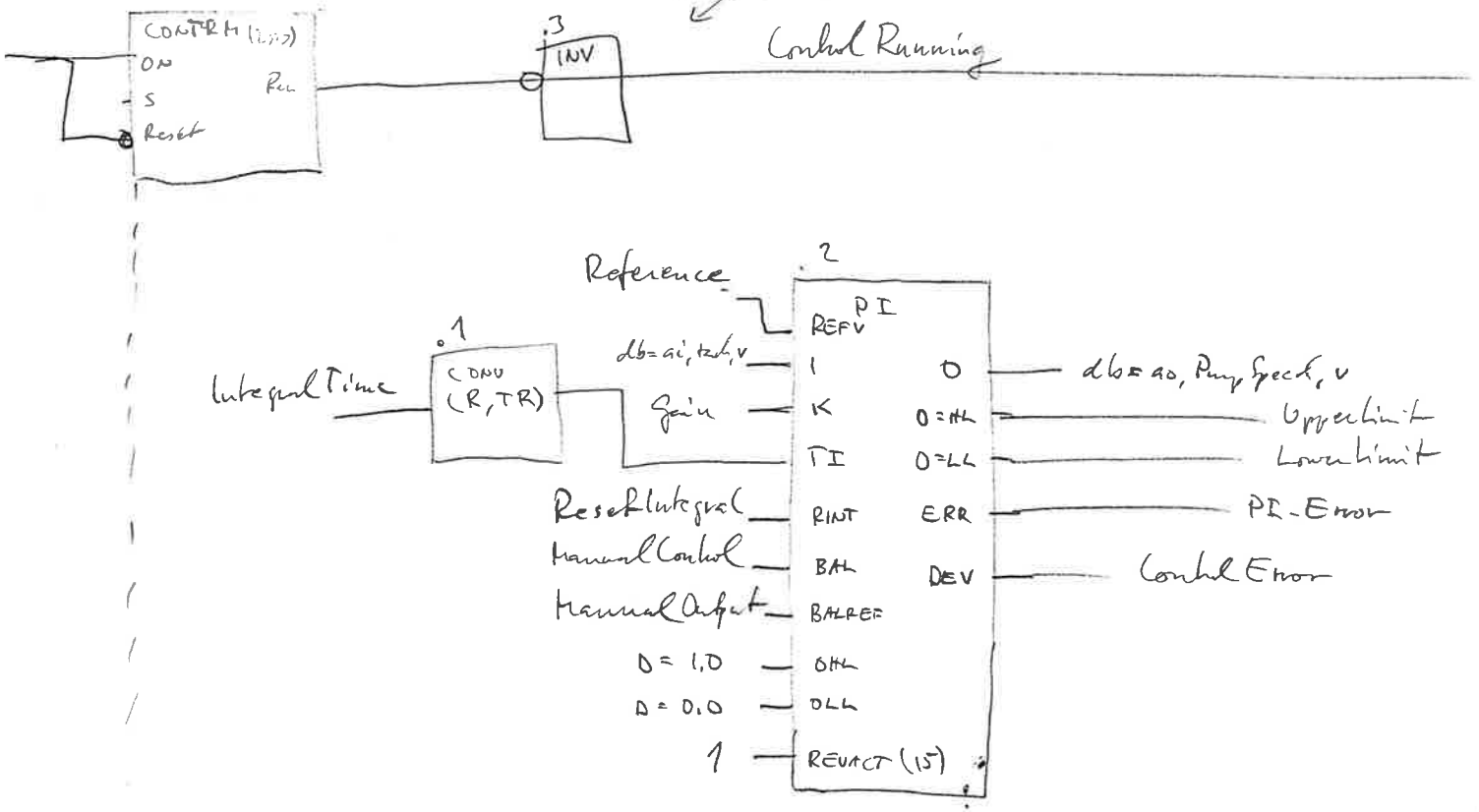
COMMON - 101



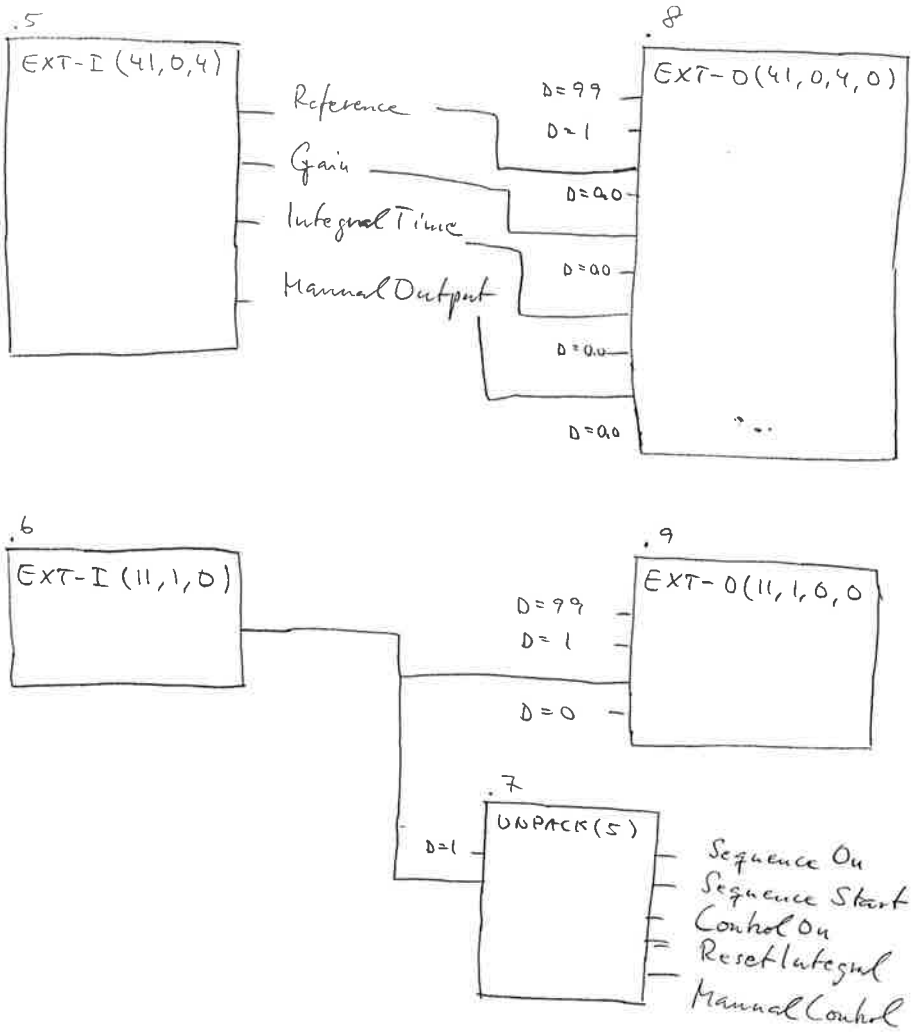
PI-reglering av tank

97-04-27

fix för att få korrekt indikering!

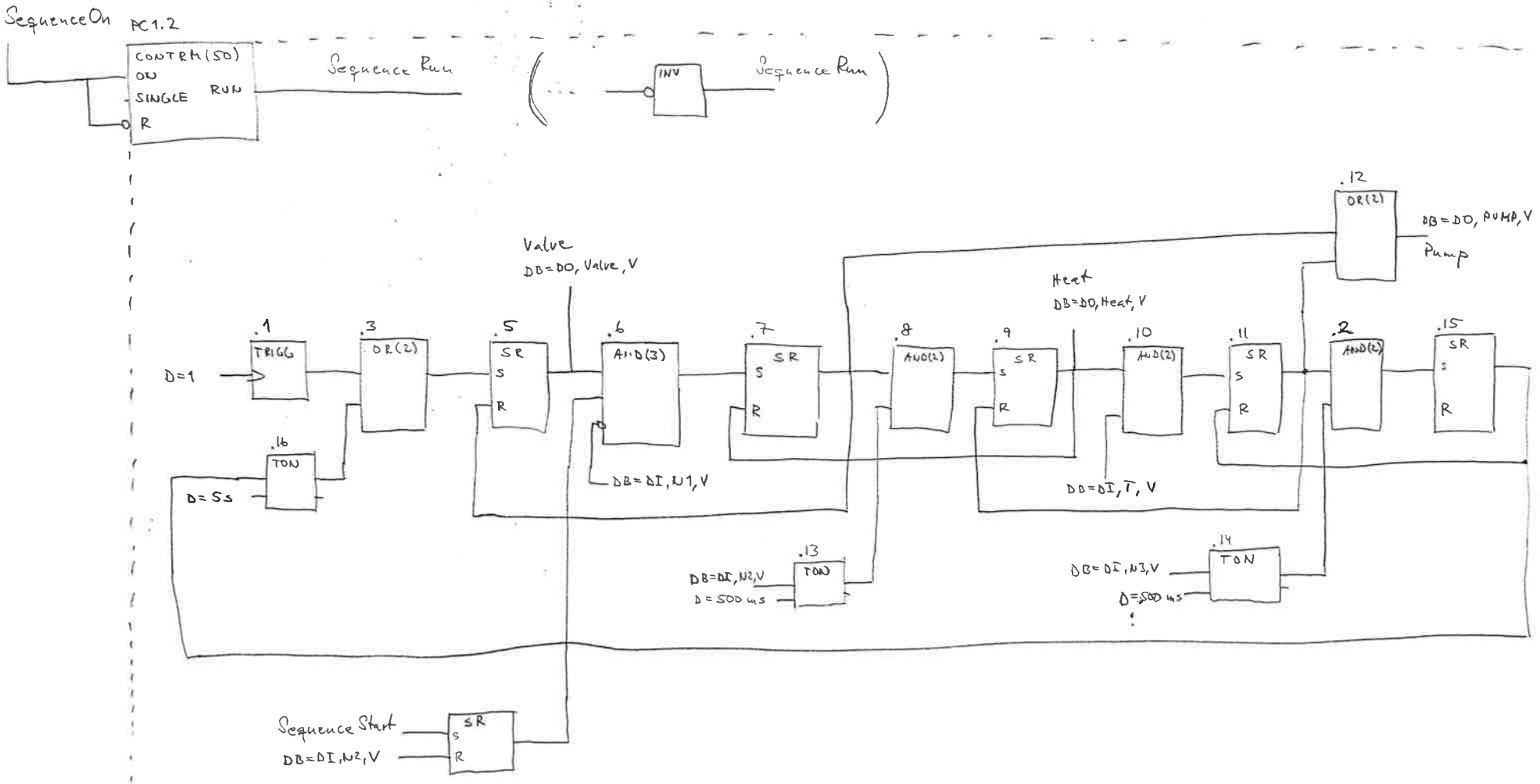


Kommunikationsdelen i PC1.1.m



Sekvensstyrning

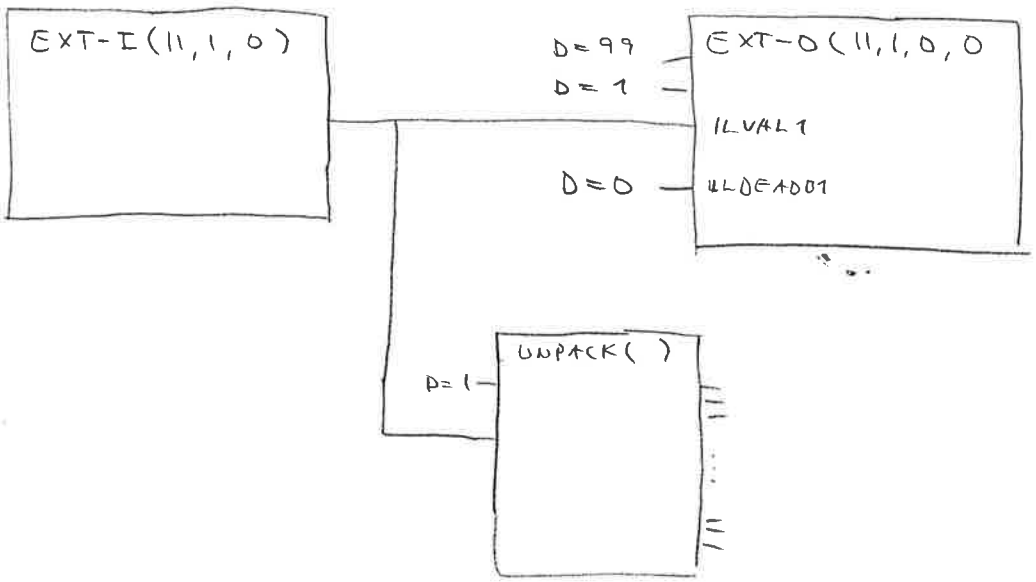
87-06-02
Lars Rg



Exekveringsordning 1 3 4 5 6 7 14 8 9 10 11 12 13 2 15 16

Kommunikation med PC-Operatör

All inkommande kommunikation måste omedelbart behandlas enligt följande



Välj dödbandet 0 resp 0.0 för alla sådana EXT-O-element.

Följande sidor innehåller listningar av program, databas mm från MP-160-systemet.

```
____ PC1     PCPGM(50,0)
     1____ .1      CONTRM(50,1)
     I     1____ .1      TRIGG
     I     1____ .2      SW-C(I,1)
     I     1____ .3      SUPERV(O)
     I     1____ .4      IND(1)
     I     1____ .5      EXT-I(41,0,4)
     I     1____ .6      EXT-I(11,1,0)
     I     1____ .7      UNPACK(5)
     I     1____ .8      EXT-O(41,0,4,0)
     I     1____ .9      EXT-O(11,1,0,0)
     I     1____ .10     PACK(13)
     I     1____ .11     EXT-O(21,1,0,0)
     I     1____ .12     EXT-O(51,0,3,0)
     I
     1____ .2      CONTRM(50,2)
     I     1____ .1      TRIGG
     I     1____ .3      OR(2)
     I     1____ .4      SR
     I     1____ .5      SR
     I     1____ .6      AND(3)
     I     1____ .7      SR
     I     1____ .14     TON
     I     1____ .8      AND(2)
     I     1____ .9      SR
     I     1____ .10     AND(2)
     I     1____ .11     SR
     I     1____ .12     OR(2)
     I     1____ .13     TON
     I     1____ .2      AND(2)
     I     1____ .15     SR
     I     1____ .16     TON
     I
     1____ .3      CONTRM(250,1)
           1____ .1      CONV(R,TR)
           1____ .2      PI
           1____ .3      INV
```


IDENTITY		TERMINAL	TYPE	SOURCE/NAME	UNIT	PAGE	COMMENTS
PC1		PCPGM(50,0)					
PC1.1		CONTRM(50,1)					
	:1	ON	IB	1			
	:2	SINGLE	IB	0			
	:3	R	IB	0			
	:5	RUN	OB				
PC1.1.1		TRIGG					
	:1		IB	1			
	:5		OB			1	PC1.1.2:ACT
PC1.1.2		SW-C(I,1)					
	:1	ACT	IB	PC1.1.1:5	1		
	:11		II	D=0			
	:12		II	D=11			
	:13		OI			1	PC1.1.3:COMADDR
PC1.1.3		SUPERV(O)					
	:1	COMTOUT	IT	D=10			
	:2	COMADDR	II	PC1.1.2:13	1		
	:3	COMNET	II	D=1			
	:5	COMERR	OB			1	PC1.1.4:I1
	:6	P24VA	OB			1	PC1.1.4:I2
	:7	P24VB	OB			1	PC1.1.4:I3
	:8	P24VE	OB			1	PC1.1.4:I4
	:9	PBATT	OB			1	PC1.1.4:I5
	:10	SYSERR	OB			1	PC1.1.4:I6
	:11	IOBLK	OB			1	PC1.1.4:I7
	:12	SERVUC	OB			1	PC1.1.4:I8
PC1.1.4		IND(1)					
	:1	I1	IB	PC1.1.3:COMERR	1		
	:2	I2	IB	PC1.1.3:P24VA	1		
	:3	I3	IB	PC1.1.3:P24VB	1		
	:4	I4	IB	PC1.1.3:P24VE	1		
	:5	I5	IB	PC1.1.3:PBATT	1		
	:6	I6	IB	PC1.1.3:SYSERR	1		
	:7	I7	IB	PC1.1.3:IOBLK	1		
	:8	I8	IB	PC1.1.3:SERVUC	1		
	:9	I9	IB	Valve	1		PC1.2.5:5
	:10	I10	IB	State1	1		PC1.2.7:5
	:11	I11	IB	Heat	1		PC1.2.9:5
	:12	I12	IB	State3	1		PC1.2.11:5
	:13	I13	IB	SequenceRunning	1		PC1.2.4:5
	:14	I14	IB	SequenceRun	1		PC1.2:RUN
	:15	I15	IB	ControlRunning	1		PC1.3.3:5P
	:16	I16	IB	ManualControl	1		PC1.1.7:05

```
C1.1.5          EXT-I(41,0,4)
                :31  RVAL1     OR        Reference                    1     PC1.1.8:R
                                                                             VAL1
                                                                      1     PC1.3.2:R
                                                                             EFV G
                :32  RVAL2     OR        Gain
                                                                      1     PC1.1.8:R
                                                                             VAL2
                                                                      1     PC1.3.2:K
                                                                              G
                :33  RVAL3     OR        IntegralTime
                                                                      1     PC1.1.8:R
                                                                             VAL3
                                                                      1     PC1.3.1:I
                                                                              G
                :34  RVAL4     OR        ManualOutput
                                                                      1     PC1.1.8:R
                                                                             VAL4
                                                                      1     PC1.3.2:B
                                                                             ALREF G
C1.1.6          EXT-I(11,1,0)
                :11  ILVAL1    OIL                                    1     PC1.1.7:I
                                                                      1     PC1.1.9:I
                                                                             LVAL1
C1.1.7          UNPACK(5)
                :1   ACT       IB        1
                :10  I         IIL       PC1.1.6:ILVAL1               1
                :11  O1        OB        SequenceOn
                                                                      1     PC1.2:ON
                                                                      1     PC1.2:R
                ;12  O2        OB        SequenceStart
                                                                      1     PC1.2.4:S
                :13  O3        OB        ControlOn
                                                                      1     PC1.3:ON
                                                                      1     PC1.3:R
                :14  O4        OB        ResetIntegral
                                                                      1     PC1.3.2:R
                                                                             INT G
                :15  O5        OB        ManualControl
                                                                      1     PC1.1.4:I
                                                                             16
                                                                      1     PC1.3.2:B
                                                                             AL G
C1.1.8          EXT-O(41,0,4,0)
                :1   DESTNODE  II        D=99
                :2   DESTNET   II        D=1
                :31  RVAL1     IR        Reference                    1     PC1.1.5:R
                                                                             VAL1
                :32  RDEADB1   IR        D=0.0
                :33  RVAL2     IR        Gain                         1     PC1.1.5:R
                                                                             VAL2
                :34  RDEADB2   IR        D=0.0
                :35  RVAL3     IR        IntegralTime                 1     PC1.1.5:R
                                                                             VAL3
                :36  RDEADB3   IR        D=0.0
                :37  RVAL4     IR        ManualOutput                 1     PC1.1.5:R
                                                                             VAL4
                :38  RDEADB4   IR        D=0.0
C1.1.9          EXT-O(11,1,0,0)
                :1   DESTNODE  II        D=99
                :2   DESTNET   II        D=1
                :11  ILVAL1    IIL       PC1.1.6:ILVAL1               1
                :12  ILDEADB1  IIL       D=0
PC1.1.10        PACK(13)
                :1   ACT       IB        1
                :11  I1        IB        SequenceRun                  1     PC1.2:RUN
                :12  I2        IB        DB=DI,N1,VALUE                     DB=1,1,2
                                                                             G
                :13  I3        IB        DB=DI,N2,VALUE                     DB=1,2,2
                                                                             G
                :14  I4        IB        DB=DI,N3,VALUE                     DB=1,3,2
                                                                             G
                :15  I5        IB        DB=DI,T,VALUE                      DB=1,4,2
                                                                             G
                :16  I6        IB        Pump                         1     PC1.2.12:
                                                                             20
                :17  I7        IB        Valve                        1     PC1.2.5:5
                :18  I8        IB        Heat                         1     PC1.2.9:5
                :19  I9        IB        SequenceRunning              1     PC1.2.4:5
                :20  I10       IB        ControlRunning               1     PC1.3.3:5
                                                                             P
                :21  I11       IB        UpperLimit                   1     PC1.3.2:0
                                                                             =HL P
                :22  I12       IB        LowerLimit                   1     PC1.3.2:0
                                                                             =LL P
                :23  I13       IB        PI_Error                     1     PC1.3.2:E
                                                                             RR P
                :50  O         OIL                                    1     PC1.1.11:
                                                                             ILVAL1
PC1.1.11        EXT-O(21,1,0,0)
                :1   DESTNODE  II        D=99
                :2   DESTNET   II        D=1
                :11  ILVAL1    IIL       PC1.1.10:O                   1
                :12  ILDEADB1  IIL       D=0
PC1.1.12        EXT-O(51,0,3,0)
                :1   DESTNODE  II        D=99
                :2   DESTNET   II        D=1
                :31  RVAL1     IR        DB=AO,PumpSpeed
                                         ,VALUE                             DB=4,1,2
                                                                             G
                :32  RDEADB1   IR        D=.02
                :33  RVAL2     IR        DB=AI,TankLevel
                                         ,VALUE                             DB=3,1,2
                                                                             G
                :34  RDEADB2   IR        D=.02
                :35  RVAL3     IR        ControlError                 1     PC1.3.2:D
                                                                             EV P
                :36  RDEADB3   IR        D=.02
PC1.2           CONTRM(50,2)
                :1   ON        IB        SequenceOn                   1     PC1.1.7:O
                                                                             1
                :2   SINGLE    IB        0
                :3   R         IB-       SequenceOn                   1     PC1.1.7:O
                                                                             1
                :5   RUN       OB        SequenceRun
                                                                      1     PC1.1.4:I
                                                                             14
                                                                      1     PC1.1.10:
                                                                             I1
PC1.2.1         TRIGG
                :1             IB        1
                :5             OB                                     1     PC1.2.7:1
```

C1.2.3	OR(2)					
	:1		IB	PC1.2.1:5	1	
	:2		IB	OpenValve	1	PC1.2.16:0
	:20		OB		1	PC1.2.5:S
C1.2.4	SR					
	:1	S	IB	SequenceStart	1	PC1.1.7:02
	:2	R	IB	DB=DI,N2,VALUE		DB=1,2,2G
	:5		OB	SequenceRunning		
					1	PC1.1.4:I13
					1	PC1.1.10:I9
					1	PC1.2.6:2
C1.2.5	SR					
	:1	S	IB	PC1.2.3:20	1	
	:2	R	IB	State1	1	PC1.2.7:5
	:5		OB	Valve		
				DB=DO,Valve,VALUE		DB=2,2,2P
					1	PC1.1.4:I9
					1	PC1.1.10:I7
					1	PC1.2.6:1
					1	PC1.2.15:R
C1.2.6	AND(3)					
	:1		IB	Valve	1	PC1.2.5:5
	:2		IB	SequenceRunning	1	PC1.2.4:5
	:3		IB-	DB=DI,N1,VALUE		DB=1,1,2G
	:20		OB		1	PC1.2.7:S
C1.2.7	SR					
	:1	S	IB	PC1.2.6:20	1	
	:2	R	IB	Heat	1	PC1.2.9:5
	:5		OB	State1		
					1	PC1.1.4:I10
					1	PC1.2.5:R
					1	PC1.2.8:1
					1	PC1.2.12:2
C1.2.14	TON					
	:1	I	IB	DB=DI,N2,VALUE		DB=1,2,2G
	:2	TD	IT	D=0.500		
	:5	O	OB	N2Delayed		
C1.2.8	:1	AND(2)	OB	N2Delayed	1	PC1.2.8:2
					1	PC1.2.14:5
	:20		OB		1	PC1.2.9:S

PC1.2.9	SR					
	:1	S	IB	PC1.2.8:20	1	
	:2	R	IB	State3	1	PC1.2.11:5
	:5		OB	Heat		
				DB=DO,Heat,VALUE		DB=2,3,2P
					1	PC1.1.4:I11
					1	PC1.1.10:I8
					1	PC1.2.7:R
					1	PC1.2.10:1
PC1.2.10	AND(2)					
	:1		IB	Heat	1	PC1.2.9:5
	:2		IB	DB=DI,T,VALUE		DB=1,4,2G
	:20		OB		1	PC1.2.11:S
PC1.2.11	SR					
	:1	S	IB	PC1.2.10:20	1	
	:2	R	IB	Wait5Seconds	1	PC1.2.15:5
	:5		OB	State3		
					1	PC1.1.4:I12
					1	PC1.2.9:R
					1	PC1.2.12:1
					1	PC1.2.2:1
PC1.2.12	OR(2)					
	:1		IB	State3	1	PC1.2.11:5
	:2		IB	State1	1	PC1.2.7:5
	:20		OB	Pump		
				DB=DO,Pump,VALUE		DB=2,1,2P
					1	PC1.1.10:I6
PC1.2.13	TON					
	:1	I	IB	DB=DI,N3,VALUE		DB=1,3,2G
	:2	TD	IT	D=0.500		
	:5	O	OB	N3Delayed		
					1	PC1.2.2:2
	:6	TE	OT			
PC1.2.2	AND(2)					
	:1		IB	State3	1	PC1.2.11:5
	:2		IB	N3Delayed	1	PC1.2.13:O
	:20		OB		1	PC1.2.15:S
PC1.2.15	SR					
	:1	S	IB	PC1.2.2:20	1	
	:2	R	IB	Valve	1	PC1.2.5:5
	:5		OB	Wait5Seconds		

				1	PC1.2.11:R
				1	PC1.2.16:I
C1.2.16	TON				
	:1 I	IB	Wait5Seconds	1	PC1.2.15:5
	:2 TD	IT	D=5		
	:5 O	OB	OpenValve	1	PC1.2.3:2
	:6 TE	OT			
C1.3	CONTRM(250,1)				
	:1 ON	IB	ControlOn	1	PC1.1.7:03
	:2 SINGLE	IB	0		
	:3 R	IB-	ControlOn	1	PC1.1.7:03
	:5 RUN	OB	OCR	1	PC1.3.3:1
C1.3.1	CONV(R,TR)				
	:1 I	IR	IntegralTime	1	PC1.1.5:RVAL3 G
	:5 O	OTR	TI s	1	PC1.3.2:TI
	:6 ERR	OB			
C1.3.2	PI				
	:1 REFV	IR	Reference	1	PC1.1.5:RVAL1 G
	:2 I	IR	DB=AI,TankLevel,VALUE		DB=3,1,2 G
	:3 K	IR	Gain	1	PC1.1.5:RVAL2 G
	:4 TI	ITR	TI s	1	PC1.3.1:0
	:5 RINT	IB	ResetIntegral	1	PC1.1.7:04 G
	:6 BAL	IB	ManualControl	1	PC1.1.7:05 G
	:7 BALREF	IR	ManualOutput	1	PC1.1.5:RVAL4 G
	:8 OHL	IR	D=1.0		
	:9 OLL	IR	D=0.0		
	:10 O	OR	DB=AO,PumpSpeed,VALUE		DB=4,1,2 P
	:11 O=HL	OB	UpperLimit	1	PC1.1.10:I11 P
	:12 O=LL	OB	LowerLimit	1	PC1.1.10:I12 P
	:13 ERR	OB	PI_Error	1	PC1.1.10:I13 P
	:14 DEV	OR	ControlError	1	PC1.1.12:RVAL3 P
	:15 REVACT	IB	1		
C1.3.3	INV				
	:1	IB-	OCR	1	PC1.3:RUN
	:5	OB	ControlRunning	1	PC1.1.4:I15 P
				1	PC1.1.10:I10 P

DI,1

1	NAME	N1
2	VALUE	0
3	UPD_BLK	0
4	BOARDTYPE	1
5	POSITION	1.3.1
6	CYCLE	50

DI,2

1	NAME	N2
2	VALUE	0
3	UPD_BLK	0
4	BOARDTYPE	1
5	POSITION	1.3.2
6	CYCLE	50

DI,3

1	NAME	N3
2	VALUE	0
3	UPD_BLK	0
4	BOARDTYPE	1
5	POSITION	1.3.3
6	CYCLE	50

DI,4

1	NAME	T
2	VALUE	0
3	UPD_BLK	0
4	BOARDTYPE	1
5	POSITION	1.3.4
6	CYCLE	50

DO,1

1	NAME	Pump
2	VALUE	0
3	UPD_BLK	0
4	BOARDTYPE	10
5	POSITION	1.4.1

DO,2

1	NAME	Valve
2	VALUE	1
3	UPD_BLK	0
4	BOARDTYPE	10
5	POSITION	1.4.2

DO,3

1	NAME	Heat
2	VALUE	0
3	UPD_BLK	0
4	BOARDTYPE	10
5	POSITION	1.4.3

AI,1

1	NAME	TankLevel
2	VALUE	.0034688
3	UPD_BLK	0
4	BOARDTYPE	16
5	POSITION	1.1.2
6	CYCLE	250
7	CONV_PARAM	4
8	LIN_CODE	0
9	RANGE_MIN	0.0
10	RANGE_MAX	1.0
11	FILTER_PARAM	0
12	ERROR	

AO,1

1	NAME	PumpSpeed
2	VALUE	0.0
3	UPD_BLK	0
4	BOARDTYPE	24
5	POSITION	1.2.1
6	CONV_PARAM	4
7	RANGE_MIN	0.0
8	RANGE_MAX	1.0
9	MAX_LIM	1.0
10	MIN_LIM	0.0

DI/1	POS=1.3.1 N1	I PC1.1.10:12
		I PC1.2.6:3
DI/2	POS=1.3.2 N2	I PC1.1.10:13
		I PC1.2.4:2
		I PC1.2.14:1
DI/3	POS=1.3.3 N3	I PC1.1.10:14
		I PC1.2.13:1
DI/4	POS=1.3.4 T	I PC1.1.10:15
		I PC1.2.10:2

DO/1	POS=1.4.1 Pump	O PC1.2.12:20
DO/2	POS=1.4.2 Valve	O PC1.2.5:5
DO/3	POS=1.4.3 Heat	O PC1.2.9:5

AI/1	POS=1.1.2 TankLevel	I PC1.1.12:33
		I PC1.3.2:2

AO/1	POS=1.2.1 PumpSpeed	I PC1.1.12:31
		O PC1.3.2:10

TRANSLATOR VERSION=AC

FREE MEMORY (BYTES):
 DATA: 2568 OF 3116   (82%) PD=2564 (LARGEST CONTINUOUS PART=2564)
 PROGRAM: 5889 OF 8192   (71%)

TARGET SYSTEM IDENTITY=0.151
TARGET ELEMENT LIBRARY=MP16* MB200 & MV100   /A50A

PCPGM	CONTRM	FUNCM	ADD	SUB	MUL	DIV	SQRT
ABS	COMP-I	COMP-R	COUNT-L	CONV	CONV-BI	PACK	UNPACK
MUX-N	SHIFT-L	FIFO	REG	REG-G	SW-C	EXT-I	EXT-O
FUNG-1V	DISP	DISP-SEG	KEYB-FU	KEYB-N	OR	AND	INV
SR	SR-AO	IND	SUPERV	LOAD	PI	INT	RAMP
DER	FILT-1P	PIP	PDP	P-1	P-DEADB	TON	TRIGG
MONO	FAULT	COUNT-PU	CON-PU1				

TARGET TRANSLATOR VERSION=AC
TARGET HARDWARE CONFIGURATION=1   NOT CHECKED (DCM=0 MP=1)

INTERFACEMODULE TYPES:

BASEBOARD (1):

IM1	IM2	IM3	IM4	IM5	IM6	IM7	IM8
TYPE=16	TYPE=24	TYPE=1	TYPE=10	(free)	(free)	(free)	(free)
DSAI 301	DSAO 302	DSDI 301	DSDO 301				
(free)	AO,1	DI,1	DO,1				
AI,1		DI,2	DO,2				
(free)		DI,3	DO,3				
(free)		DI,4	(free)				
(free)		(free)	(free)				
(free)		(free)	(free)				
(free)		(free)	(free)				
(free)		(free)	(free)				

1ST EXT.BOARD (2):

IM1	IM2	IM3	IM4	IM5	IM6	IM7	IM8
(free)	(free)	(free)	(free)	(free)	(free)	(free)	(free)