CODEN: LUTFD2/(TFRT-5188)1-041/(1976)

DATORSTYRD SVETSAUTOMAT

CLAES-HÅKAN MÅNSSON

Inst. för Reglerteknik
Lunds Tekniska Högskola
November 1976

Dokumentutgivare
04T0
Lund Institute of Technology
Handläggare Dept of Automatic Control
Johan Wieslander
Författare
08T0
Claes-Håkan Månsson

Dokumentnamn
04T4
REPORT
Utgivningsdatum
06T4
Nov 1976

Dokumentbeteckning
04T6
LUTFD2/(TFRT-5188)/1-041/(1976)
Ärendebeteckning
06T6

10T4

Dokumenttitel och undertitel
18T0
Datorstyrd svetsautomat

Referat (sammandrag)
26T0
Development of software for a micro-computer controlled automatic welding machine. The software is divided into a learning part and a production part.

Referat skrivet av
42T0
J Wieslander

Förslag till ytterligare nyckelord
44T0

Klassifikationssystem och -klass(er)
50T0

Indextermer (ange källa)
52T0

Omfång
41 pages
Språk
54T0
Swedish

Övriga bibliografiska uppgifter
56T2

Sekretessuppgifter
60T0

ISSN
60T4

ISBN
60T6

Dokumentet kan erhållas från
62T0
Department of Automatic Control
Lund Institute of Technology
P O Box 725, S-220 07 LUND 7, Sweden

Mottagarens uppgifter
62T4

Pris
64T0

DOKUMENTDATABLAD enligt SIS 62 10 12

SIS-DB 1

Blankett LU 11:25 1976–07

DATORSTYRD SVETSAUTOMAT

Mjukvara

CLAES-HÅKAN MÅNSSON

Handledare: JOHAN WIESLANDER

Sammanfattning.

Examensarbetet består av mjukvarudelen av styrsystemet till en kurvlinjestyrd automatsvets i tre dimensioner. Styrsystemets kärna är ett mikrodatorsystem med en Intel 8080 som centralenhet.

Programmeringen har gjorts i mellannivåspråket HILP 80.

Programmen är uppdelade i tre huvuddelar, två inlärningsdelar och en "användningsdel". I den ena inlärningsdelen lär man in den kurva man senare skall svetsa. I den andra sker inlärning av svetshastighet och övriga svetsparametrar. Den egentliga svetsningen av detaljer sker i "användningsdelen".

Abstract.

This report contains the software of a control unit to a curvi-linear controlled automatic welding machine. The core of the control unit is a microcomputer system with an Intel 8080 as the central processor unit.

The programming is made in the middlelevellanguage HILP 80.

The programs are divided into three principal parts, two learning parts and one performing part. In one learning part you learn the curve you latter will weld. In the other you learn the welding velocity and other welding parameters.

The real welding will take place in the performing part.

INNEHÅLLSFÖRTECKNING

1.Inledning.

Examensarbetet har utförts på institutionen för Reglerteknik vid Lunds Tekniska Högskola.
Problemet bestod i att ta fram ett styrsystem för kurvlinjestyrning,i tre dimensioner,av munstycket till en svetsautomat. Det stod ganska snart klart för oss att styrsystemet borde bestå av ett datorsystem eftersom man då relativt enkelt kan bygga ut systemet till att exempelvis styra själva svetsautomaten,fixturer,löpande band och dylikt. För att klara sin uppgift borde systemet bestå av fem frihetsgrader,tre koordinater plus två vridningar.
Målet har varit att få en enkel inlärning av kurvor,att kunna korrigera delar av dem och att sedan kunna styra ut svetsmunstycket längs den inlärda kurvan. Allt detta har det ej varit möjligt att utföra inom ramen för examensarbetet. Vridningarna har ej medtagits och ej heller möjligheten att korrigera delar av de inlärda kurvorna.
Styrsystemet består av ett mikrodatorsystem som är uppbyggt kring en Intel 8080. Att vi valt Intel 8080 beror på att systemet redan fanns på institutionen och att man utvecklat ett mellannivåspråk HILP 80 för Intel 8080. HILP 80 har varit ett utmärkt hjälpmedel för att optimera assemblerkoden.
När det gäller den mekaniska uppbyggnaden finns för varje koordinat en kulmutterskruv med motor. Motorenheten består av en PM motor försedd med tachometer och kraftaggregat med reglerkretsar.Reglerkretsarna är uppbyggda som en PID regulator vilken får sitt börvärde från datorn och sitt ärvärde från tachometern. En kodskiva på varje motor ger datorn indikation på

läget av kulmutterskruven.

Inlärningen av en kurva sker med teach-in metoden. Det finns därför ett handtag med vilket man kan få svetsmunstycket att följa den önskade kurvan.

I övrigt finns en styrpanel där vissa parametrar ställs in och varifrån man ger kommando till styrsystemet.

Av ekonomiska skäl kom testerna av systemet att ske i två dimensioner utan svetsmunstycke.

Arbetet har delats upp i två delar en hårdvarudel och en mjukvarudel. Mjukvarudelen har fallit på min lott och rapporten behandlar först de teoretiska huvudprinciper som använts och sedan en genomgång av programmen. Hårdvarudelen har utförts av Gert Hultqvist och arbetet har gjorts i samarbete med honom. De algoritmer för flytande räkning som använts fanns redan tillgängliga och kommenteras inte här. Det enda som bör nämnas är att mantissan representeras av ett 16-bits ord i tvåkomplement och exponenten av ett 8-bits ord i binär offset med teckenbiten i sjunde positionen.

2.Inlärning av svetskurva.

Det finns olika sätt att lära in en kurva. Ett sätt är att programmera från ritning,det vill säga man sitter vid sitt skrivbord och med ledning av detaljritningen bestämmer de punkter vars koordinater skall lagras. Ett annat sätt är Teach-in metoden. Den innebär att man följer kurvan med svetsmunstycket och vid lämpliga tillfällen lagrar punkter, det vill säga de koordinater som positionsräknarna anger. Vi har i detta fall valt teach-in metoden eftersom den i mindre utsträckning kräver tekniskt utbildad personal.

2.1.Teach-in metoden.

Kurvan kommer i detta fall att approximeras med en polygon, det vill säga varje liten kurvsträcka approximeras med en rät linje. Alternativet skulle vara att approximera varje liten kurvsträcka med en cirkelbåge med en viss radie.

2.1.1.Utstyrning av svetsmunstycket.

Utstyrning av svetsmunstycket vid teach-in sker med ett handtag som kan påverka två mikrobrytare för varje koordinatriktning. Brytarna är då tillslagna beroende på vilken riktning man vill att motorn för en koordinat skall gå i. Om ingen brytare är tillslagen står motorn still. För att få in en viss tröghet i systemet låter man signalerna från mikrobrytarna påverka en enkel regleralgoritm.

$$K(t)=(1-\alpha)\cdot K(t-1)+\alpha\cdot u(t)$$

u(t) är signalen från mikrobrytaren och antar värdena ett,noll eller minus ett. Ett om motorn skall gå i framriktningen,noll

om den skall stå still och minus ett om den skall gå i backriktningen. α är ett valfritt tal mellan noll och ett beroende på vilken tröghet man vill ha i systemet. Resultatet K(t) kommer att ligga mellan minus ett och ett. Det multipliceras sedan med 12.7 för att få rätt hastighet till motorerna. Att vi valt 12.7 beror på följande. Till vårt förfogande för utsignalen till varje motor har vi ett 8-bits ord. En bit används till teckenbit,detta för att ange i vilken riktning motorn skall gå. De övriga sju bitarna kan högst anta talet 127,ettor i alla sju positionerna. Talet 127 motsvarar då max. hastighet. Max. hastighet har vi valt till 30 mm/s,vilket används vid förflyttningar av svetsmunstycket transportsträckor. Hastigheten vid teach-in är vald till max. 3 mm/s. Talet blir alltså $\frac{127 \cdot 3}{30} = 12.7$.

2.1.2.Inlärning av koordinater.

Inlärning av koordinater kan ske på två sätt,manuell inlärning eller automatisk inlärning. Manuell inlärning går till på så sätt att man helt enkelt trycker på en knapp på styrpanelen om man vill lagra den punkt där munstycket befinner sig.

Automatisk inlärning går till enligt följande.(Se fig.2.1.)

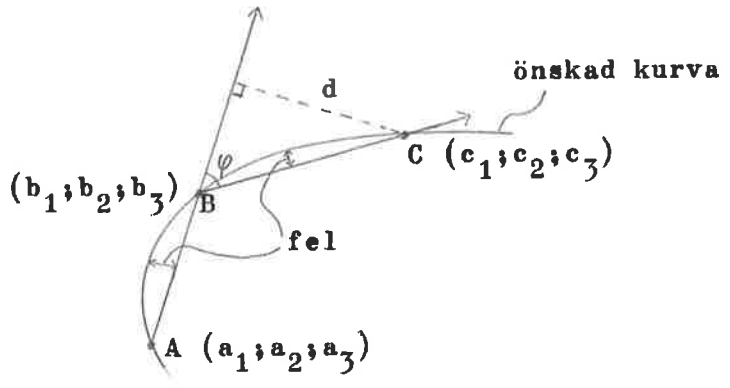


Fig.2.1.

Vi antar att punkt A och B är de senast lagrade punkterna och att munstycket befinner sig i punkt C. Man bestämmer då riktningen för vektorn $\overline{AB}$, $(b_1-a_1;b_2-a_2;b_3-a_3)$ och riktningen för vektorn $\overline{BC}$, $(c_1-b_1;c_2-b_2;c_3-b_3)$. Genom att använda skalär produkt

$$\overline{AB}\cdot\overline{BC}=|\overline{AB}|\cdot|\overline{BC}|\cdot\cos\varphi$$

$$(a_1-b_1)(b_1-c_1)+(a_2-b_2)(b_2-c_2)+(a_3-b_3)(b_3-c_3)=$$

$$=\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+(a_3-b_3)^2}\cdot\sqrt{(b_1-c_1)^2+(b_2-c_2)^2+(b_3-c_3)^2}\cdot\cos\varphi$$

$$\cos\varphi=\frac{(a_1-b_1)(b_1-c_1)+(a_2-b_2)(b_2-c_2)+(a_3-b_3)(b_3-c_3)}{\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+(a_3-b_3)^2}\sqrt{(b_1-c_1)^2+(b_2-c_2)^2+(b_3-c_3)^2}}$$

kan vinkeln mellan vektorerna beräknas. Sträckan d kan sedan beräknas genom att multiplicera längden av BC med sinus för vinkeln φ, $d=|\overline{BC}|\cdot\sin\varphi$. Om d är större än 0.25 mm lagras punkten C. Det totala felet vid svetsningen får högst vara 0.50 mm. I normala fall blir felet mycket mindre,se fig.2.1. Anledningen till att testparametern vid lagring valts till 0.25 mm beror på att i vissa extremfall kan felet bli 2 0.25 mm alltså 0.50 mm.

2.2.Korrigering.

En enklare korrigering vid teach-in finns för tillfället i systemet. Den går till enligt följande. Om man under inlärningen av en kurva märker att den senaste inlärda biten inte överenstämmer med den önskade kurvan, kan man genom att trycka på en knapp på styrpanelen initiera ett korrigeringsprogram. Korrigeringsprogrammet är så uppbyggt att man backar längs den inlärda kurvan tills man släpper knappen på panelen. Teach-in programmet fortsätter sedan som vanligt.

Vid en eventuell vidarutveckling av systemet kommer med all sannolikhet en förbättring av korrigering vid teach-in att införas.

3.Inlärning av svetshastighet och svetsparametrar.

Inlärning av hastighet och svetsparametrar går till på så sätt att man medan man följer den inlärda kurvan lär in hastigheter och svetsparametrar. Förflyttningen sker med den hastighet man sist lärt in.

På styrpanelen finns förnärvarande möjlighet att ställa in fyra olika kombinationer av trådmatningshastighet, strömstyrka, skyddsgas och dylikt. Förutom dessa finns möjlighet att helt slå ifrån svetsningen.

Man har tio olika hastigheter att välja på genom att ställa in ett tal mellan noll och nio. Detta tal motsvarar en tidigare lagrad hastighet. De lagrade hastigheterna kan väljas fritt mellan 0-30 mm/s.

De på panelen inställda svetsparametrarna och talet, som motsvarar en hastighet, lagras i en vektor. När man passerar en lagrad punkt kontrolleras om den senast lagrade hastigheten och de senast lagrade svetsparametrarna är identiska med de som är inställda på panelen. Om de är identiska inkrementeras en räknare som senare vid körning håller reda på hur många lagrade punkter som skall passeras innan en ny hastighet eller nya svetsparametrar skall användas. Om de ej är identiska lagras den räknare som hör till den föregående hastigheten och svetsparametrarna. Därefter inkrementeras den pekare som är index dels till den vektor där räknarna lagras och dels till vektorn för hastigheter och svetsparametrar. Sedan lagras den nya hastigheten och de nya svetsparametrarna.

Som exempel kan vi ta att om vi för tillfället använder den hastighet och de svetaparametrar som finns lagrade i V0(9) så finns i L0(9) det tal som talar om hur många lagrade punkter

som skall passeras med denna hastighet och dessa svetsparametrar, se fig.3.1.

Fig.3.1.

Vi tänker oss att V0(9) skall börja användas vid punkt två och att V0(10) skall börja användas vid punkt fem. Vilket tal finns då i L0(9)? Jo, L0(9) kommer att innehålla talet två, ty två punkter nämnligen punkt tre och fyra passeras med parametrarna V0(9). Om t.ex. V0(10) bara skall gälla mellan punkt fem och sex kommer L0(10) att vara lika med noll ty ingen punkt kommer att passeras med parametrarna V0(10).

4.Utstyrning av svetsmunstycket längs den inlärda kurvan.

Utstyrning av svetsmunstycket längs den inlärda kurvan är hela tiden en lineär förflyttning från punkt till punkt. En enklare koordinatkorrigering finns inlagd. Den kan, om man så önskar, kopplas bort. Hastigheter till varje motor beräknas med hjälp av absoluthastigheten och deltavärden. En test sker så att man kan kontrollera om den önskade koordinatpunkten är uppnådd.

4.1.Koordinatkorrigering.

Anledningen till att man behöver någon form av koordinatkorrigering är att man aldrig kan komma till en koordinatpunkt exakt. Korrigeringen går till enligt följande, se fig.4.1.

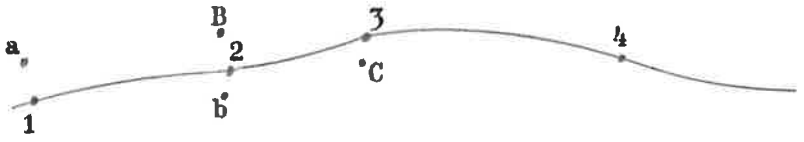


Fig.4.1.

Vi antar att vi har kommit till punkten a istället för till punkten 1. När vi sedan befinner oss mellan punkten 1 och 2 skall vi beräkna den hastighet vi skall ha mellan punkt 2 och 3. Vi korrigerar då utgångsläget för beräkningen det vill säga punkten 2 med det fel vi gjorde vid punkten 1. Vi beräknar alltså hastigheten mellan punkten B och 3 istället. Sedan fortsätter vi på samma sätt genom att korrigera utgångsläget för punkt 3 med det verkliga fel vi gjorde vid punkten 2. Anta att vi kommer till punkten b istället för till punkten 2. Då ändras utgångsläget för punkt 3 till C osv.

4.2.Hastighetsberäkning.

Egentligen är det inte hastighet som beräknas utan det tal mellan 0-127 som skickas via en D/A-omvandlare till respektive motor.

Beräkningen börjar med att man bestämmer deltavärden det vill säga $\Delta x=x(T)-x(T-1)$, $\Delta y=y(T)-y(T-1)$, $\Delta z=z(T)-z(T-1)$ och $\Delta s=\sqrt{\Delta x^2+\Delta y^2+\Delta z^2}$. Vid dessa beräkningar är x(T-1), y(T-1) och z(T-1) korrigerade enligt kap. 4.1.

Talen som skickas ut till respektive motor beräknas sedan enligt följande.

$$x_f=\frac{\Delta x\cdot v}{\Delta s\cdot 6,3} \quad \text{till x-motorn}$$

$$y_f=\frac{\Delta y\cdot v}{\Delta s\cdot 6,3} \quad \text{till y-motorn}$$

$$z_f=\frac{\Delta z\cdot v}{\Delta s\cdot 6,3} \quad \text{till z-motorn}$$

v är absoluthastigheten.

På varje motor finns en kodskiva som ger 80 pulser per varv. Hastigheten v är angiven i dessa pulser/s. Divisionen med 6,3 är till för att omvandla hastigheten från pulser/s till det tal mellan 0-127 som D/A-omvandlaren kräver.

Talet 127 motsvarar 30 mm/s eller 800 pulser/s eftersom ett varv ger 80 pulser och en förflyttning av 3 mm. Detta ger just division med 6,3 , $\frac{800}{127}=6,3$.

4.3.Test om en koordinatpunkt är uppnådd.

Test om en koordinatpunkt är uppnådd sker genom att i princip bestämma avståndet mellan den punkt där munstycket befinner sig och den punkt vi är på väg till.

Avståndet minskar först hela tiden när man närmar sig punkten. När avståndet börjar öka igen anser vi att punk-

ten är uppnådd och vi sänder ut nya hastigheter till motorerna. Avståndet beräknas på ett förenklat sätt genom att ta skillnaden mellan x-,y- och z-koordinaterna för den punkt där munstycket befinner sig och den punkt vi vill komma till.

Absolutbeloppet av skillnaderna summeras och används som detta avstånd. Som exempel låt oss tänka oss två punkter A och B. A är den punkt vi vill komma till med koordinaterna $(x_A;y_A;z_A)$ och B är den punkt där svetsmunstycket befinner sig med koordinaterna $(x_B;y_B;z_B)$. Avståndet s är då lika med

$$s= |x_B-x_A| + |y_B-y_A| + |z_B-z_A| \ .$$

4.4.Uppdelning i delsträckor.

Uppdelning i delsträckor är ett sätt att lägga in icke lagrade punkter mellan två punkter som finns lagrade. Uppdelningen i kombination med koordinatkorrigeringen bidrager till att minska det fel som uppkommer på grund av den gräns motorernas upplösning sätter.(se fig.4.2.)

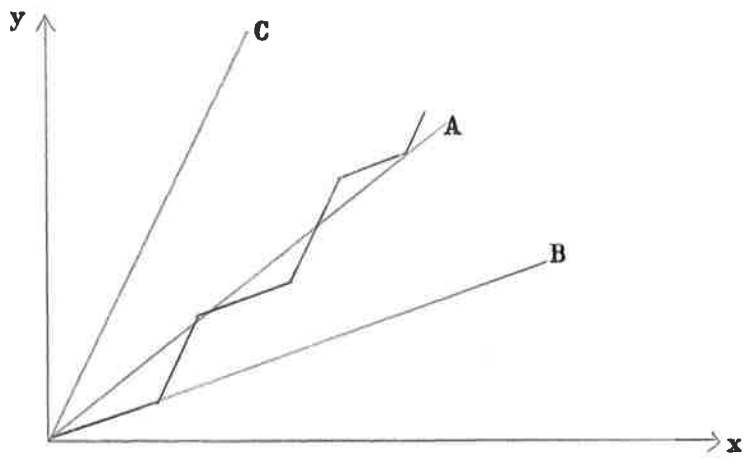


Fig.4.2.

Anta att vi vill förflytta oss längs linje A, men motorernas upplösning gör att vi ej kan förflytta oss den vinkel linje A representerar utan har B och C att välja på.

Uppdelningen i kombination med koordinatkorrigeringen gör då att vi kommer att sick-sacka längs kurva A. Felet blir då mycket mindre än om vi skulle ha följt kurva B eller C.

Om uppdelning skall ske beror dels på avståndet s mellan två lagrade punkter och dels på den hastighet som skall användas vid tillfället. Det som slutgiltigt bestämmer om uppdelning skall ske är den tid det tar för de beräkningar som sker vid munstyckets förflyttning från en punkt till en annan. Denna tid är i värsta fall omkring 80 ms. Om avståndet $s > v \cdot 80$ ms sker uppdelning i delsträckor. Sträckan delas i så fall upp i det antal delsträckor som bestämmes av heltalsdelen av $\frac{s}{v \cdot 80 \text{ ms}}$. Låt oss som exempel anta att $\frac{s}{v \cdot 80 \text{ ms}} = 3$. (se fig.4.3.)

Fig.4.3.

Uppdelningen av sträckan AB sker då i tre lika långa delsträckor, sträckorna Aa, ab och bB.

Utstyrningen av svetsmunstycket kommer liksom tidigare att vara lineära rörelser mellan de lagrade och icke lagrade punkterna. Rörelsen bestämmes av hastigheterna till de olika motorerna.

5.Lagring av data.

5.1.Lagring av koordinater.

Lagring av koordinater sker punkt för punkt. Först lagras en punkts x-koordinat sedan dess y-koordinat och därefter dess z-koordinat. Konsekutivt följer sedan nästa punkts x-koordinat, dess y-koordinat och dess z-koordinat osv. (se fig.5.1.)

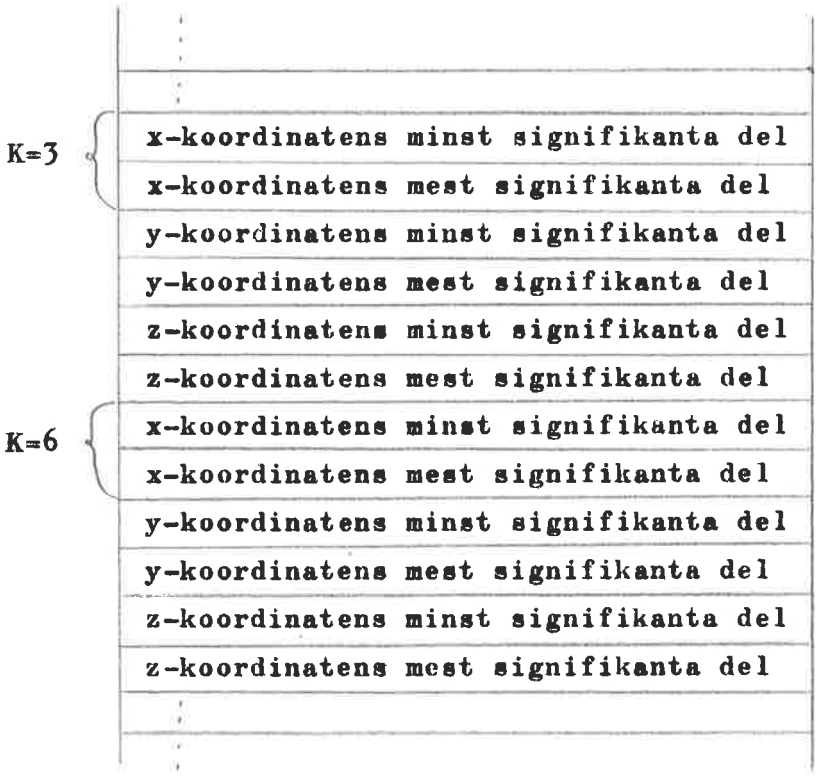


Fig.5.1.

Eftersom koordinaterna är lagrade i dubbelord sker i minnet uppdelning i mest signifikanta del och minst signifikanta del. Koordinatpekaren K används vid adressering av koordinaterna och detta är möjligt tack vare att vi vet adressen till den första punktens x-koordinat. Koordinatpekaren pekar hela tiden på x-koordinaten.
HILP 80 har minnesutrymmet uppdelat i banker om 256 bytes. Jag har ej använt språkets möjlighet att lägga koordinater-

na i en vektor eftersom man då är bunden av att inte överskrida bankgränserna och kan högst lagra 128 punkter, med en bank för varje koordinat.

5.2.Lagring av hastigheter och svetsparametrar.

De hastigheter eller rättare sagt det tal som motsvarar en hastighet och de svetsparametrar som läses in från styrpanelen lagras i vektorform. På samma sätt lagras de räknare som anger hur många lagrade punkter som skall passeras med samma hastighet och svetsparametrar. Det index som används i programmen för vektorerna betecknas RC.
I vektorform lagras också de verkliga hastigheterna. Vektorn har som index det tal mellan noll och nio som man vid inlärningen ställer in på styrpanelen.

5.3.Övrigt.

Man lagrar även ett tal som anger hur många koordinatpunkter som finns lagrade för en detalj.
I övrigt används en bank för interna hjälpvariabler.

6.Huvuddragen i programmen.

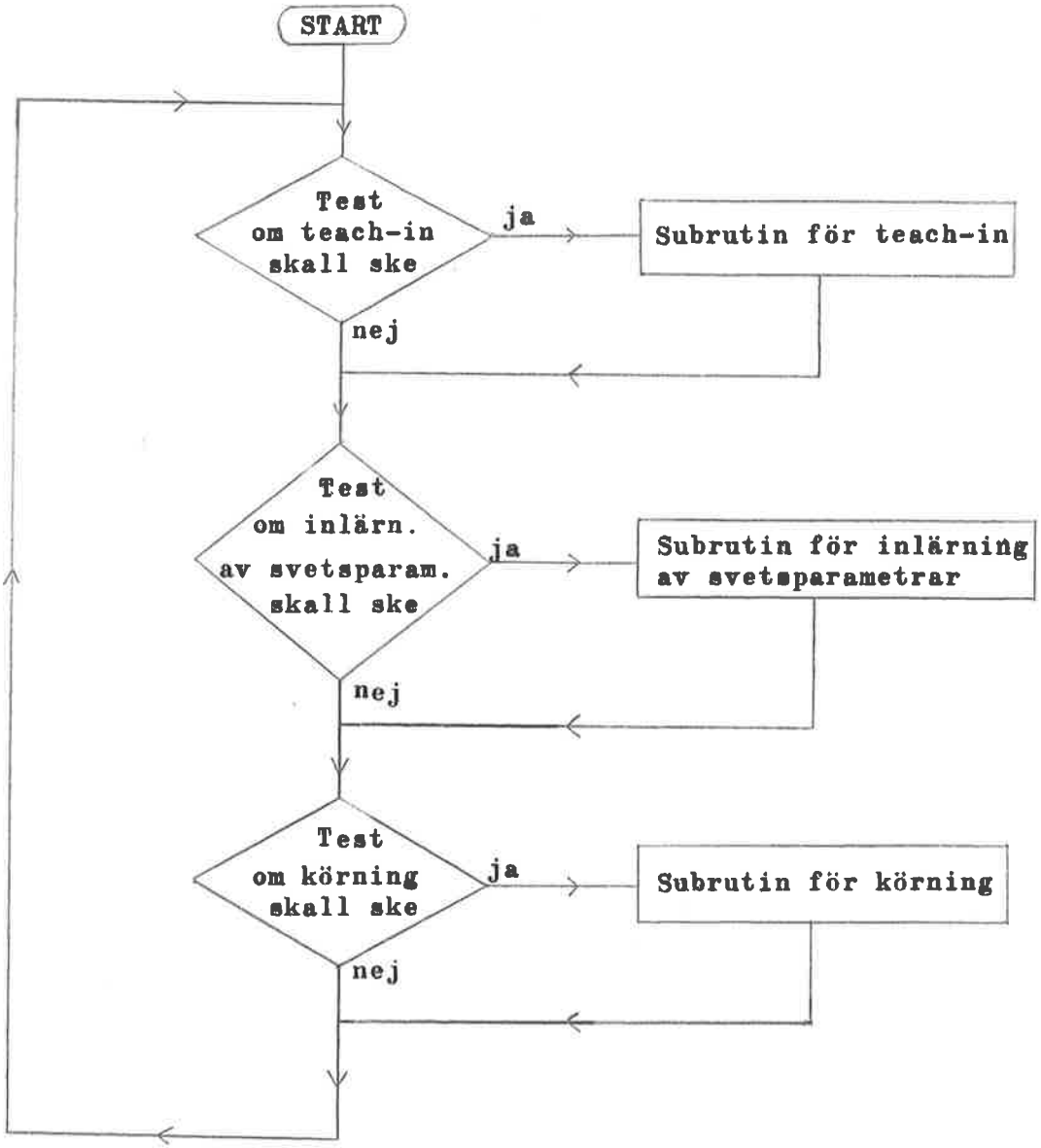
6.1.Huvudprogram.

I huvudprogrammet görs villkorliga subrutinanrop till antingen Teach-in, Inlärning av hastigheter och svetsparametrar eller Körning. Vilken av de tre subrutinerna som skall anropas ställes in på styrpanelen.

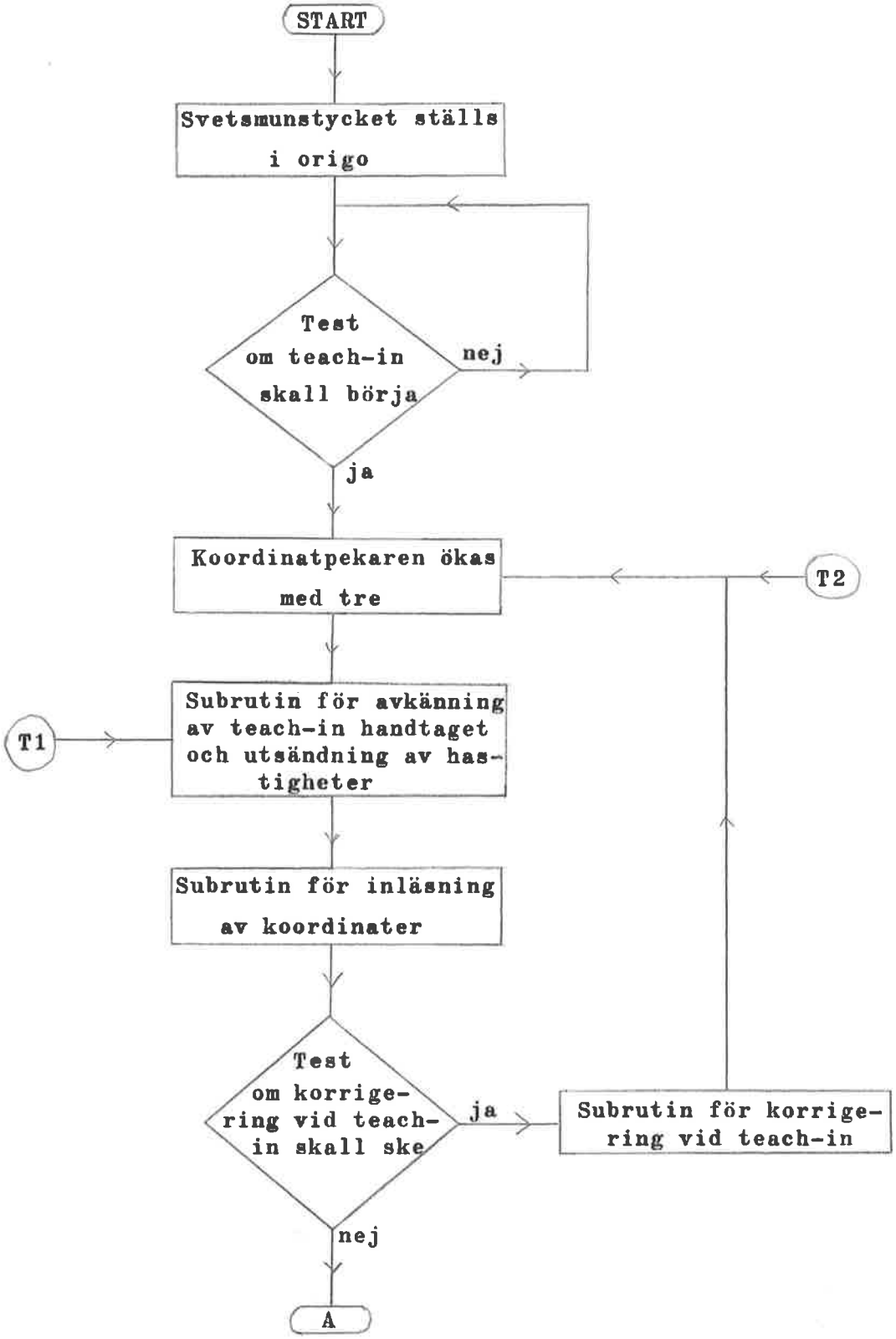
6:2.Subrutinen TEACH.

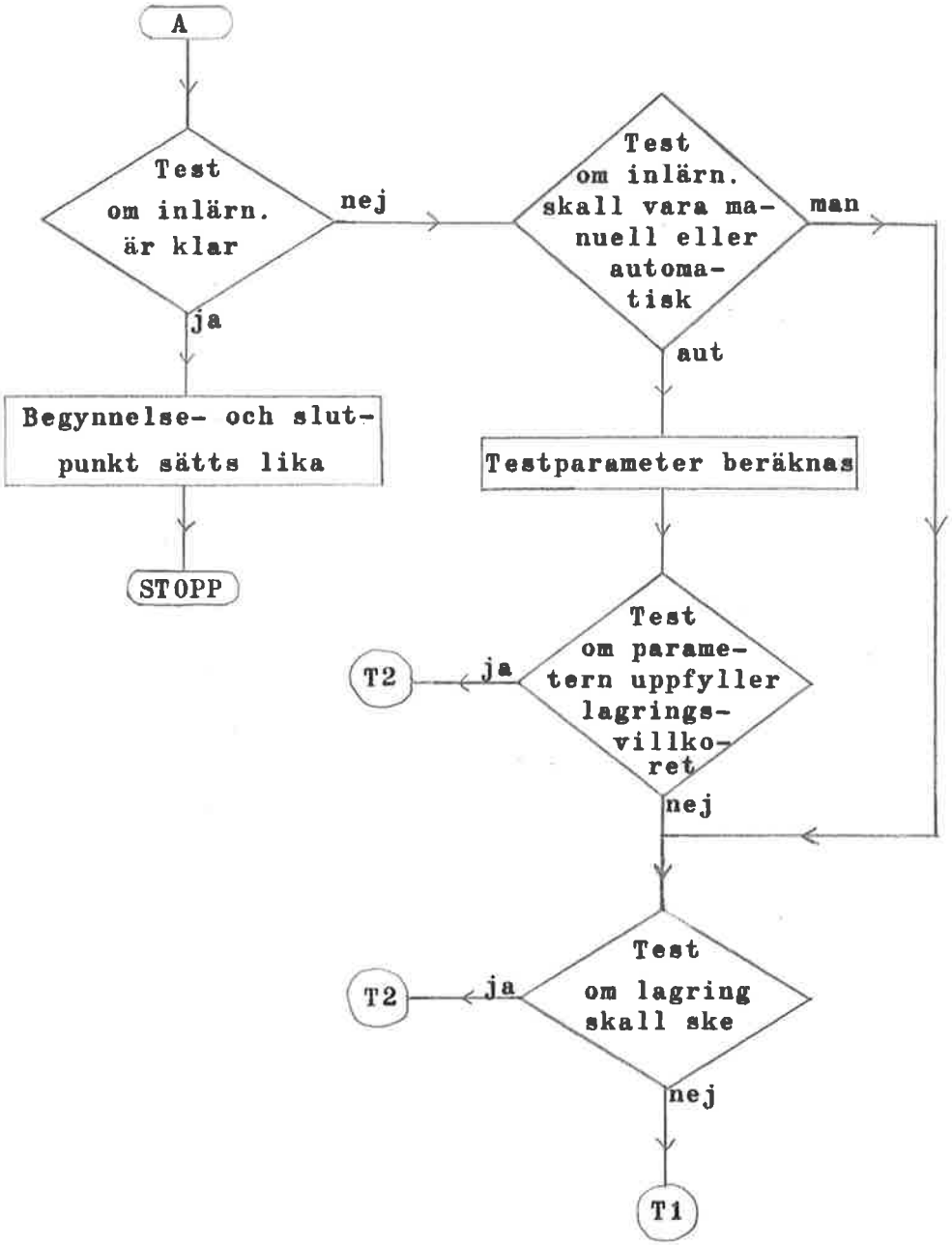
Subrutinen börjar med att anropa subrutinen HAST. Man kan då förflytta svetsmunstycket med teach-in handtaget till ett valfritt origo som är lämpligt för just den detalj det gäller. Koordinatpekaren K nollställes och (0;0;0) lagras.
Man ställer sedan in en begynnelseriktning för att ha något att räkna med när man första gången skall beräkna testparametern vid automatisk inlärning.
Koordinatpekaren K ökas med tre för att peka på nästa lediga minnescell. Därefter sker en uppdatering av koordinater och deltavärden. Exempelvis x(T-1)=x(T) , y(T-1)=y(T) osv.
Nu börjar den egentliga inlärningen av koordinater genom att anropa subrutinen HAST. Med hjälp av teach-in handtaget kan man nu förflytta svetsmunstycket längs den önskade kurvan. Subrutinen READ läser in koordinater från positionsräknarna och en beräkning av deltavärden sker. Inläsningen av koordinater betyder inte att de är lagrade, vill man inte lagra dem läser man nästa gång in koordinater i samma minnesceller.
Nu sker en del tester. Man testar parametrar som är inställ-

Flödesschema för huvudprogrammet.



Flödesschema för subrutinen TEACH.




A
Test om inlärn. är klar
nej
ja
Test om inlärn. skall vara manuell eller automatisk
man
aut
Begynnelse- och slutpunkt sätts lika
Testparameter beräknas
STOPP
Test om parametern uppfyller lagringsvillkoret
T2
ja
nej
Test om lagring skall ske
T2
ja
nej
T1

da på styrpanelen. Först testas om korrigering vid teach-in skall ske. I nästa test kontrolleras om man börjar överskrida det minnesutrymme för koordinater som man har till sitt förfogande. Om man gör det finns det ytterligare några minnesceller så att man kan avsluta en kurva. Avslutningen kan bara ske genom manuell inlärning. Detta är medtagit för att underlätta testkörning av systemet då man kanske har begränsat minnesutrymme till sitt förfogande. En kontroll sker så att samma punkt inte lagras två gånger.

Därefter kontrolleras om man är färdig med teach-in. I så fall sätts begynnelse- och slutpunkterna lika. Detta för att redan vid avslutandet av svetsningen av en detalj skall befinna sig i utgångsläget för svetsning av nästa detalj, se fig.6.1.

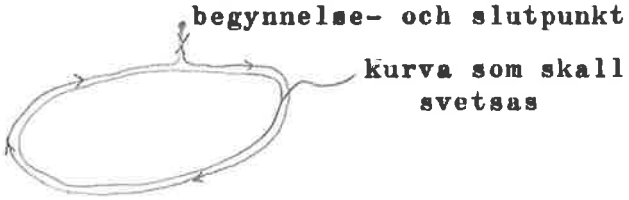


Fig.6.1.

I den sista testen bestämmer man om man skall ha manuell- eller automatisk inlärning. Vid manuell inlärning sker en test, på en på styrpanelen inställd parameter, om lagring skall ske eller ej. Vid automatisk inlärning beräknas testparametern och därefter testas om lagring skall ske. Man kan vid automatisk inlärning tvångslagra en punkt även om lagringskriteriet ej är uppfyllt.

6.3.Subrutinen HAST.

Subrutinen HAST beräknar hastigheterna, eller rättare sagt de tal som sänds ut till motorerna, vid teach-in. Denna

beräkning är helt förknippad med den enkla regleralgoritm som används.

$$K(t)=(1-\alpha)\cdot K(t-1)+\alpha u$$

Föregående K värde dvs K(t-1) lägges i en arbetscell och u-värde läses in. Se kap.2.1.1 Därefter anropas subrutinen HAS där beräkningen av regleralgoritmen sker. Resultatet sänds ut till motorn och K-värdet uppdateras.
Detta upprepas tre gånger en för varje koordinatriktning.

6.4.Subrutinen HAS.

I subrutinen HAS beräknas regleralgoritmen

$$K(t)=(1-\alpha)\cdot K(t-1)+\alpha u$$

Först beräknas $(1-\alpha)\cdot K(t-1)$. Beroende på om u är +1 eller -1 adderas eller subtraheras α till detta resultat. Sedan multipliceras resultatet med 12,7 enligt kap.2.1.1.
Som avslutning konverteras resultatet från tvåkomplement till binär offset. Detta sker på grund av att signalen till motorn via D/A-omvandlaren skall vara i binär offset.

6.5.Subrutinen READ.

Subrutinen READ läser in koordinater från positionsräknarna och lagrar dem dels enligt kap.5.1. och dels i arbetsceller.
Man börjar med att bestämma adressen till första lediga minnescell. X-koordinaten läses sedan in från positionsräknaren. Eftersom koordinaterna är i dubbelord lagras dess minst signifikanta del i den minnescell som anges av den beräknade adressen och den mest signifikanta delen i nästa minnescell. Koordinatem lagras även i en arbetscell.

Y- och Z-koordinaterna lagras på likartat sätt.

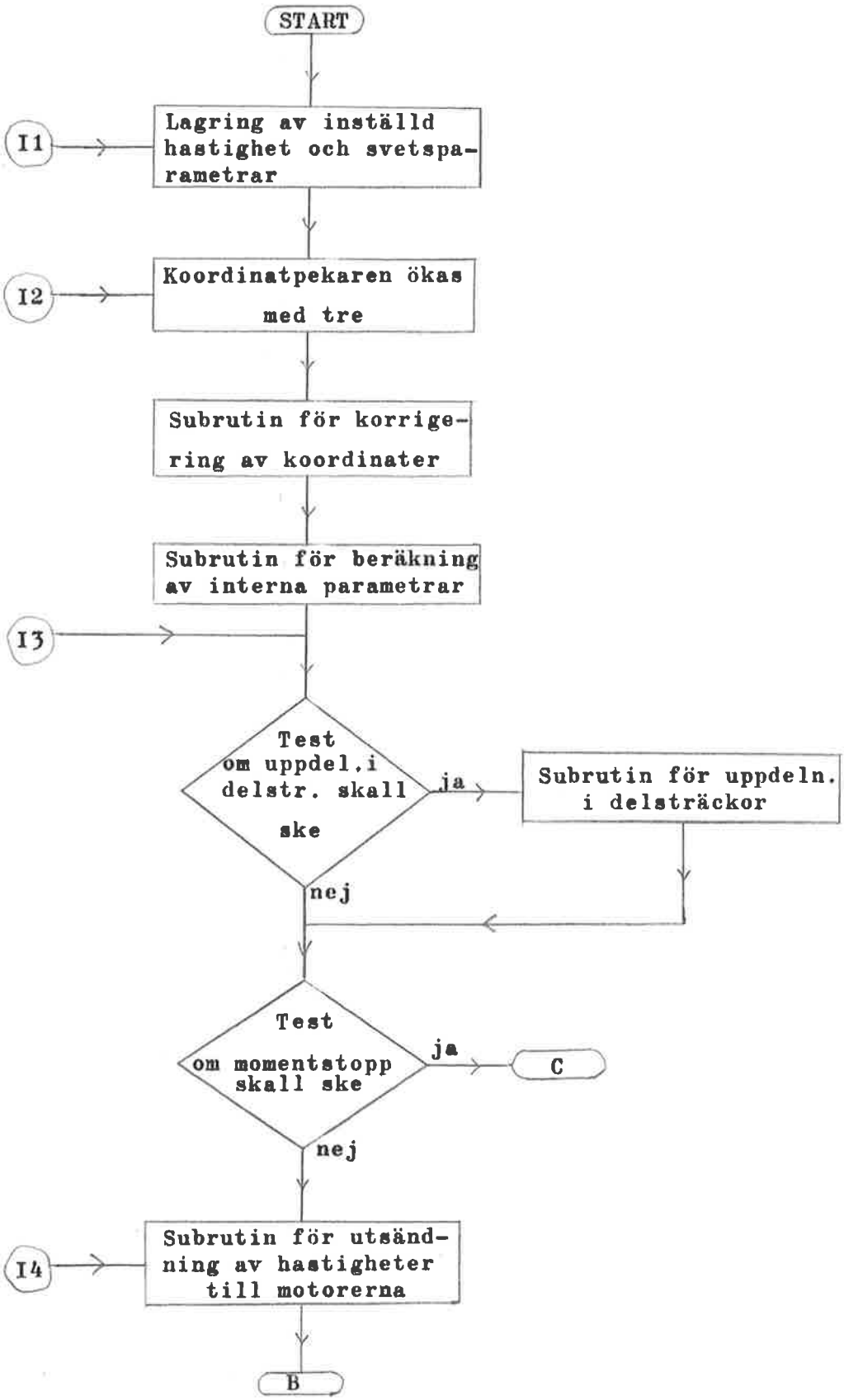
6.6.Subrutinen TKORR.

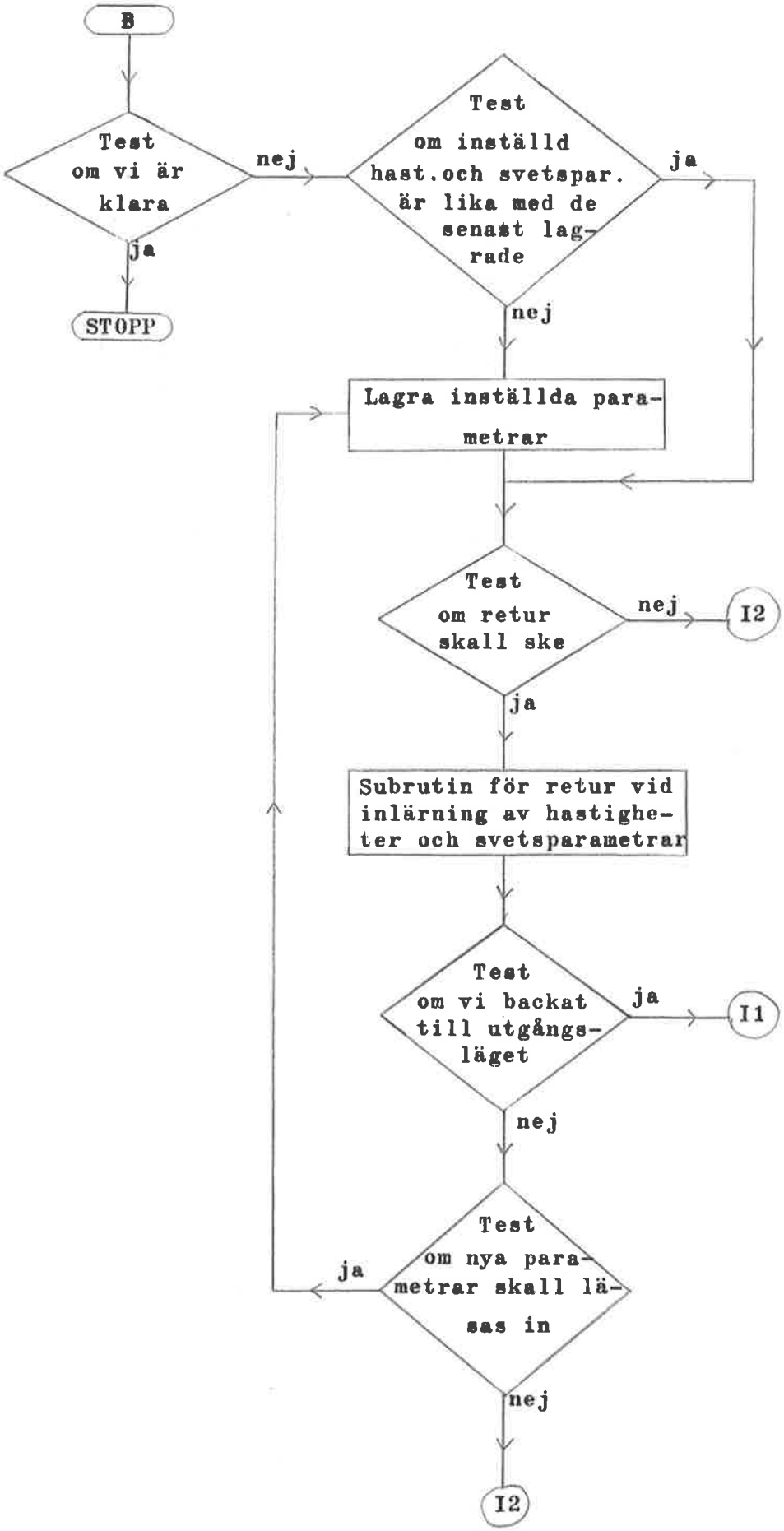
I subrutinen TKORR finns en möjlighet till korrigering vid teach-in genom att backa längs den inlärda kurvan och sedan lära in den på nytt.
Man läser först in koordinaterna för den punkt där svetsmunstycket befinner sig. Hastigheten ställes in till 3 mm/s och koordinatpekaren K minskas med tre.
Koordinaterna för punkten man skall gå från koordinatkorrigeras och lägges,tillsammans med koordinaterna för den punkt man skall gå till,i arbetsceller. Därefter beräknas deltavärden. I subrutinen TESTU testas om uppdelning i delsträckor skall ske. Här sker också beräkning av hastigheterna till motorerna genom ett subrutinanrop. Därefter sänder man ut de till binär offset omvandlade hastigheterna och beräknar korrigeringsvärden för koordinatkorrigering. Detta görs i subrutinen UTSUB.
Till sist testas om korrigeringen vid teach-in skall fortsätta eller om själva teach-in programmet skall återupptagas.

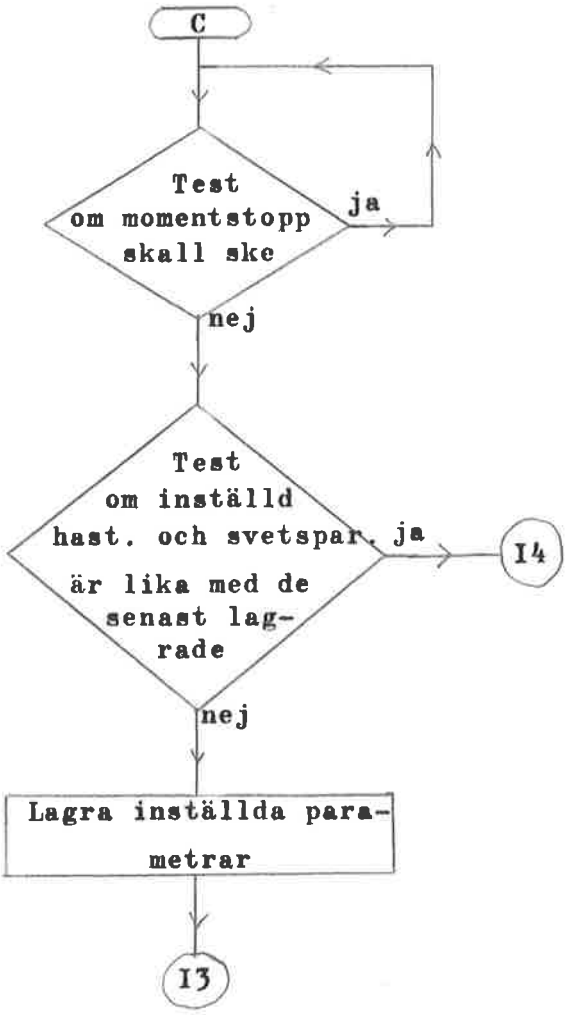
6.7.Subrutinen IAHOS. Inlärning Av Hastighet Och Svetsparametrar.

Subrutinen börjar med att hastighetspekaren nollställes och att koordinatpekaren sättes lika med tre. Detta för att peka på den första punkten dvs begynnelsepunkten.
Koordinaterna för begynnelsepunkten lägges i arbetsceller och räknaren RL nollställes. RL är det tal som senare lagras i LO-vektorn, se kap.3. Hastighet och svetsparametrar

Flödesschema för subrutinen IAHOS.




B
Test om vi är klara
nej
ja
STOPP
Test om inställd hast.och svetspar. är lika med de senast lagrade
ja
nej
Lagra inställda parametrar
Test om retur skall ske
nej
I2
ja
Subrutin för retur vid inlärning av hastigheter och svetsparametrar
Test om vi backat till utgångsläget
ja
I1
nej
Test om nya parametrar skall läsas in
ja
nej
I2


C
Test om momentstopp skall ske
ja
nej
Test om inställd hast. och svetspar. är lika med de senast lagrade
ja
I4
nej
Lagra inställda parametrar
I3

läses in och lagras. Koordinatpekaren K ökas med tre. Koordinaterna för punkten man skall gå från koordinatkorrigeras och lägges,tillsammans med koordinaterna för den punkt man skall gå till, i arbetsceller. Därefter beräknas deltavärden.

I subrutinen TESTU testas om uppdelning i delsträckor skall ske. Här sker också beräkning av hastigheterna till motorerna genom ett subrutinanrop.

En test om man vill göra momentstopp sker. Detta ger en möjlighet att i lugn och ro ställa in en ny hastighet och nya svetsparametrar på styrpanelen. De svetsparametrar som gäller för tillfället sänds ut till svetsaggregatet. Man får då en möjlighet att kontrollera svetsfogen och om den inte är bra göra retur och sedan lära in nya svetsparametrar.

De tidigare beräknade hastigheterna sänds ut till motorerna.

Därefter sker en test om vi är färdiga med parameterinlärningen för detaljen. I så fall görs återhopp till huvudprogrammet.

Om den hastighet och de svetsparametrar som är inställda på panelen är lika med de senast lagrade ökas räknaren RL med ett. I annat fall lagras räknaren RL i LO-vektorn, hastighetspekaren RC inkrementeras och den nya hastigheten och de nya svetsparametrarna lagras. Räknaren RL nollställes.

En test sker om retur vid inlärning av hastighet och svetsparametrar skall ske. Om retur har skett görs en anpassning av räknaren RL beroende på dels vilken hastighet och vilka svetsparametrar som finns inställda på styrpanelen och dels vilken hastighet och vilka svetsparametrar som vi tidigare lärt in vid den punkt vi backat till i returprogrammet.

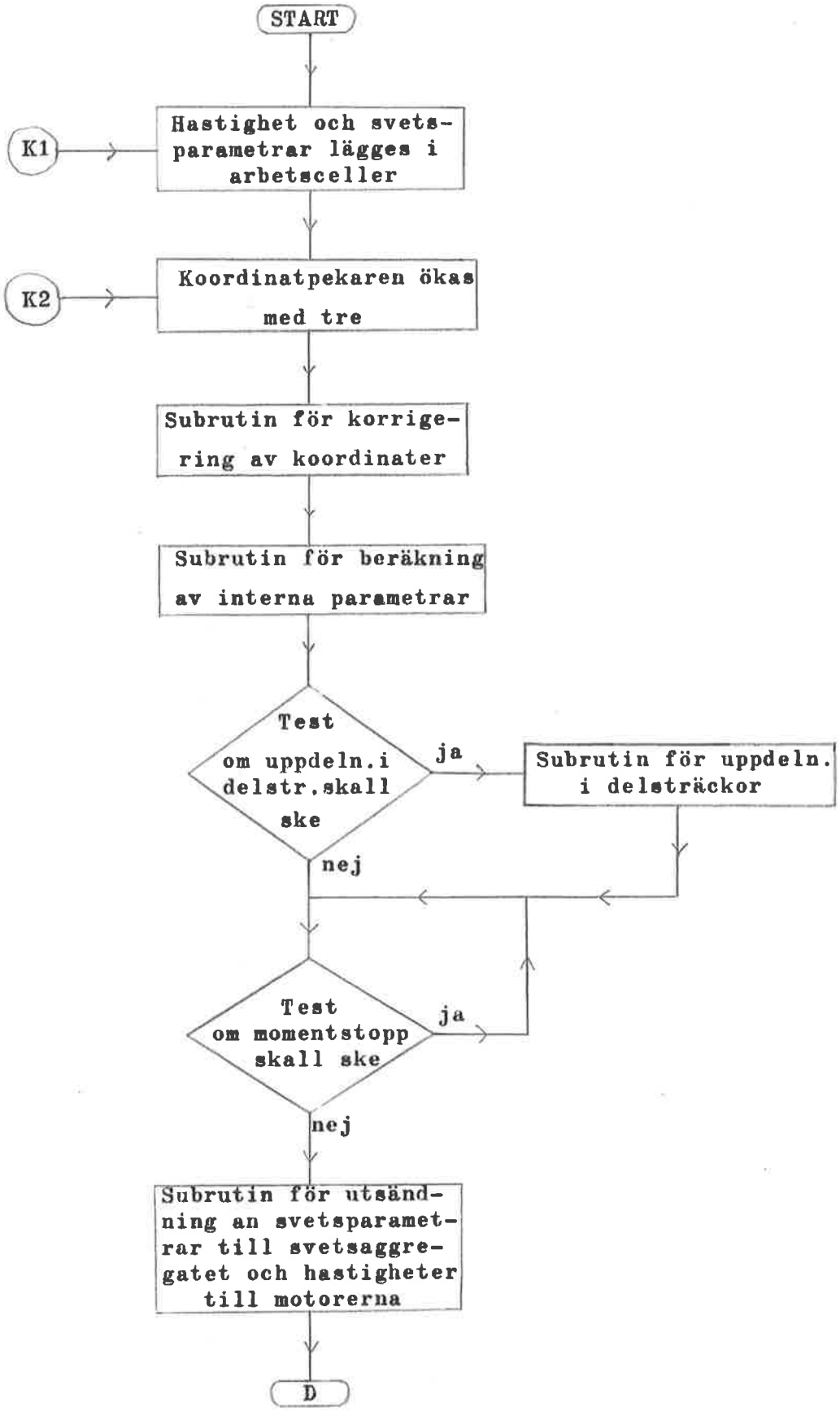
6.8.Subrutinen RETUR.

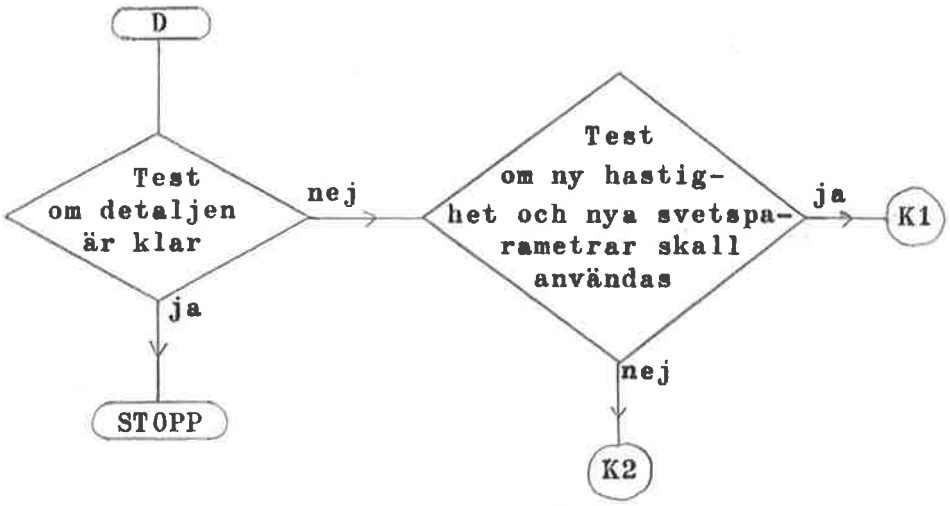
Man börjar med att ställa hastigheten till 3 mm/s och minska koordinatpekaren K med tre.
Koordinaterna för punkten man skall gå ifrån koordinatkorrigeras och lägges, tillsammans med koordinaterna för den punkt man skall gå till, i arbetsceller. Deltavärden beräknas och i subrutinen TESTU testas om uppdelning i delsträckor skall ske. I TESTU beräknas också hastigheterna till motorerna genom ett subrutinanrop.
Subrutinen UTSUB omvandlar hastigheterna till binär offset och sänder ut dem till motorerna. Det görs även en beräkning av korrigeringsvärden för koordinatkorrigering.
I ett par tester testas dels om vi backat till begynnelsepunkten dvs till K=3 och dels om retur fortfarande skall ske.
För att få rätt värde på hastighetspekaren RC vid återgång från returprogrammet testas om räknaren RL är lika med noll. Om så är fallet dekrementeras RC och ett nytt LO-värde lägges i RL, annars dekrementeras RL.

6.9.Subrutinen KOER.

Subrutinen KOER är den programdel som används vid svetsning av detaljer. Här sker dels utstyrning längs den inlärda kurvan och dels sänds svetsparametrar till svetsaggregatet.
Man börjar med att nollställa hastighetspekaren RC och sätta koordinatpekaren K lika med tre. Koordinaterna för begynnelsepunkten lägges i arbetsceller. Aktuellt LO-vär-

Flödesschema för subrutinen KOER.




D
Test
om detaljen
är klar
nej
ja
STOPP
Test
om ny hastig-
het och nya svetspa-
rametrar skall
användas
ja
K1
nej
K2

de lägges i sin arbetscell RL och hastighet och svetsparametrar lägges i sina arbetsceller. Koordinatpekaren K ökas med tre.

Koordinaterna för den punkt man skall gå från koordinatkorrigeras och lägges, tillsammans med koordinaterna för den punkt man skall gå till, i arbetsceller. Därefter beräknas deltavärden.

I subrutinen TESTU testas om uppdelning i delsträckor skall ske. Här beräknas även nya hastigheter till motorerna genom ett subrutinanrop. Genom att slå om en brytare på styrpanelen kan man få programmet att stanna tills brytaren slås om igen. Efter en test om ett sådant stopp skall ske sänds nya svetsparametrar ut till svetsaggregatet. I subrutinen UTSUB omvandlas hastigheterna till binär offset och sänds ut till motorerna. I denna subrutin beräknas också korrigeringsvärden för koordinatkorrigering.

Därefter testas om svetsningen av detaljen är klar. I så fall görs återhopp till huvudprogrammet. I en annan test kontrolleras om en ny hastighet och nya svetsparametrar skall användas. Det sker genom att testa om räknaren RL är lika med noll. Om så är fallet inkrementeras hastighetspekaren RC och ny hastighet och nya svetsparametrar lägges i arbetsceller. Annars dekrementeras räknaren RL.

6.10.Subrutinen LAGUN.

I subrutinen LAGUN koordinatkorrigeras koordinaterna för utgångspunkten vid beräkning av hastigheterna till motorerna mellan denna punkt och den nästföljande punkten. Okorrigerade koordinater lagras också för att användas vid beräkningen av nya korrigeringsvärden i subrutinen UTSUB.

6.11.Subrutinen LDSUB.

Subrutinen LDSUB lägger en punkts koordinater i arbetsceller. Först lägges minnesadressen i HL-registren. Eftersom koordinaterna är lagrade i dubbelord (enl.kap.5.1.), överföres först den minst signifikanta delen av x-koordinaten till E-registret. Adressen inkrementeras och den mest signifikanta delen av x-koordinaten lägges i D-registret. Sedan lagras x-koordinaten i en arbetscell.
Proceduren upprepas på liknande sätt för y- och z-koordinaterna.

6.12.Subrutinerna DELTA och SDEL.

Subrutinen DELTA beräknar deltavärdena Δx, Δy och Δz och i subrutinen SDEL beräknas Δs. Beräkningarna sker enligt kapitel 4.2.

6.13.Subrutinen TESTU.

I denna subrutin testas om uppdelning i delsträckor skall ske. Testen sker på en testparameter beräknad enligt kapitel 4.4. Om uppdelning skall ske anropas subrutinen UDEL. I annat fall beräknas hastigheterna till motorerna. Sedan testas om den punkt man är på väg till under beräkningsgången är uppnådd. Detta sker i subrutinen SUBT.

6.14.Subrutinen UDEL.

I subrutinen UDEL sker uppdelning i delsträckor. Till att börja med beräknas talet N som anger hur många delsträckor uppdelningen skall ske i. Δx, Δy och Δz divideras med N så

att man får delsträckornas storlek i varje koordinatriktning. De värden man då får adderas succesivt till koordinaterna för utgångspunkten vid uppdelning. Man får då koordinaterna för punkterna mellan delsträckorna. En räknare N1 nollställes och inkrementeras.
Koordinaterna för den punkt vi skall gå från koordinatkorrigeras, och koordinaterna för den punkt vi skall gå till bestämmes. Deltavärden beräknas. Genom att kontrollera om N1=N testar vi om uppdelningssträckan är passerad.
De nya hastigheterna till motorerna beräknas. I subrutinen SUBT kontrolleras om den punkt vi var på väg till under beräkningarna är uppnådd. Därefter sänds svetsparametrar ut till svetsaggregatet och hastigheterna till motorerna. Koordinatkorrigeringsvärden beräknas också.

6.15.Subrutinen FREKV.

Här sker beräkningen av de tal som skickas ut till motorerna. Beräkningen sker helt enligt kapitel 4.2. Absoluthastigheten v divideras med 6,3 och resultatet divideras med $\triangle s$. Det värde vi nu får multipliceras med $\triangle x$, $\triangle y$ eller $\triangle z$ för att erhålla x_f, y_f eller z_f.

6.16.Subrutinen SUBT.

Principerna för subrutinen finns angivna i kapitel 4.3. x-koordinaten för den punkt där svetsmunstycket befinner sig läses in och lagras. Absolutbeloppet för skillnaden mellan denna x-koordinat och x-koordinaten för den punkt vi vill komma fram till beräknas. Samma beräkningar utföres för y- och z-koordinaterna. De olika absolutbeloppen summeras.
Man testar sedan om det senaste resultatet är större än det

föregående. Om så är fallet görs återhopp från subrutinen, annars lagras detsenast beräknade resultatet och proceduren upprepas.

6.17.Subrutinen UTSUB.

I denna subrutinen omvandlas hastigheterna till motorerna från representation i tvåkomplement till binär offset. De sänds sedan ut till respektive motor via D/A-omvandling. Sist i surutinen beräknas korrigeringsvärden för koordinatkorrigering. Det finns en möjlighet att hoppa över denna beräkning. Detta för att vid testkörning kunna studera svetsmunstyckets rörelser med eller utan koordinatkorrigering.

6.18.Subrutinen NOLL.

Subrutinen NOLL är väldigt enkel eftersom det enda som sker är att man sänder ut hastigheten noll till alla motorerna.

6.19.Subrutinen FLYT.

I subrutinen FLYT sker en konvertering från 16-bits fixtal till flyttal. Mantissan placeras i ett 16-bits ord och representeras i tvåkomplement. Exponenten består av åtta bitar och representeras i binär offset med teckenbiten i sjunde positionen.

6.20.Subrutinerna FIX16 och FIX8.

I subrutinen FIX16 sker en konvertering från flyttal, i den i 6.19. omnämnda representationen, till 16-bits fixtal och

i FIX8 från flyttal till 8-bits fixtal.

BILAGA.

Exempel på programlistor.

Subrutinerna DELTA, SDEL och UTSUB.

```
PROC DELTA; DELTAVAERDEN BERAEKNAS
BANK 1; INNEHALLER ARBETSCELLER
DOUBLE XKK,YKK,ZKK,XKM,YKM,ZKM,DLX,DLY,DLZ;
DX DLX=XKK-XKM; DELTA X=X[T]-X[T-1]
DX DLY=YKK-YKM; DELTA Y=Y[T]-Y[T-1]
DX DLZ=ZKK-ZKM; DELTA Z=Z[T]-Z[T-1]
ENDPROC;


FINISH;
```

```
PROC SDEL; DELTA S BERAEKNAS
BANK 1; INNEHALLER ARBETSCELLER
DOUBLE DLX↑12,DLY,DLZ,DS↑34,AXM↑62,AYM,AZM,FLM↑78;
SINGLE DSEXP↑103,AXE↑111,AYE,AZE,FLE↑119;
GLOBAL FMUL,FADD,FSQRT,FLYT;
CALL FLYT(DE=DLX); KONV FRAN FIXTAL TILL FLYTTAL
DX AXM=DE;
AXE=B;
CALL FMUL(HL=DE,A=B); DELTA X*DELTA X
DX FLM=DE;
FLE=B;
CALL FLYT(DE=DLY); KONV FRAN FIXTAL TILL FLYTTAL
DX AYM=DE;
AYE=B;
CALL FMUL(HL=DE,A=B); DELTA Y*DELTA Y
CALL FADD(HL=FLM,A=FLE); T=(DELTA X)↑2+(DELTA Y)↑2
DX FLM=DE;
FLE=B;
CALL FLYT(DE=DLZ); KONV FRAN FIXTAL TILL FLYTTAL
DX AZM=DE;
AZE=B;
CALL FMUL(HL=DE,A=B); DELTA Z*DELTA Z
CALL FADD(HL=FLM,A=FLE); (DELTA S)↑2=T+(DELTA Z)↑2
CALL FSQRT; DELTA S=SQRT[(DELTA S)↑2]
DX DS=DE;
DSEXP=B;
ENDPROC;


FINISH;
```

```
PROC UTSUB; FREKVENSER SAENDS UT TILL MOTORERNA
BANK 1; INNEHALLER ARBETSCELLER
DOUBLE XK1↑18,YK1,ZK1,DXKOR,DYKOR,DZKOR,X1↑36,Y1,Z1;
SINGLE XF↑122,YF,ZF;
A=XF; FREKV OMVANDLAS FRAN TVAKOMPL TILL
    ; BINAER OFFSET
A=A.RAL;
.CMC;
A=A.RAR;
OUTPUT(174)=A; FREKV TILL MOTOR X SAENDS UT
A=YF; FREKV OMVANDLAS FRAN TVAKOMPL TILL
    ; BINAER OFFSET
A=A.RAL;
.CMC;
A=A.RAR;
OUTPUT(175)=A; FREKV TILL MOTOR Y SAENDS UT
A=ZF; FREKV OMVANDLAS FRAN TVAKOMPL TILL
    ; BINAER OFFSET
A=A.RAL;
.CMC;
A=A.RAR;
.NOP; PLATS AVSEDD FOER OUTPUT TILL Z MOTOR
.NOP;
A=INPUT(367).AND 1;
IF ZERO TRUE THEN; TEST OM KOORD KORRIG SKALL SKE
  DX DXKOR=X1-XK1; KORRIGERINGSVAERDEN BERAEKNAS
  DX DYKOR=Y1-YK1;
  DX DZKOR=Z1-ZK1;
END;
ENDPROC;


FINISH;
```