CODEN: LUTFD2/(TFRT-5430)/1–104/(1990)

Finjustering av styrsimulator

Frans Hermodsson
Lars Bergenzaun

Institutionen för Reglerteknik
Tekniska Högskolan i Lund
November 1990

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

Document name
Masther Thesis

Date of issue
November 1990

Document Number
CODEN: LUTFD2/(TFRT-5430)/1–104/(1990)

Author(s)
Frans Hermodsson, Lars Bergenzaun

Supervisor
Rolf Johansson, Dan Lindahl (Kockums Marine AB)

Sponsoring organisation
Kockums Marine AB

Title and subtitle
Finjustering av styrsimulator (Improvement of man-machine adaptation of the steering gear in a submarine).

Abstract

The thesis treats the improvement and redesign of man-machine adaptation of the steering gear in a submarine. A second topic is the design of software modules to be used for development of computer interfaces for submarine operation.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
104

Recipient's notes

Security classification

Innehållsförteckning

Förord

Författarna till den här rapporten vill rikta sitt tack till Dan Lindahl och Dan Lindelöf på Kockums AB för all hjälp vi fått. Vi tackar också vår handledare Rolf Johansson, forskningsingenjörerna Rolf Braun och Leif Andersson vid Instutionen för Reglerteknink, för råd och dåd. Vi vill även tacka professor Sven Lindblad och Forskarstuderande Martin Nyström vid Instutionen för Akustik, för deras stöd under examensarbetets akustiska tillämpningar.

Hänvisningar i texten, till referenser ges som (siffra), och till appendix som (bokstav).

1. Inledning

Arbetets syfte

Detta examensarbete består av två deluppgifter. Dels att försöka förbättra tidigare examensarbete, Farkoststyrning och förarmiljö, dels att ta fram ett verktyg i mjukvara för att kunna bygga upp en förarmiljö, ett så kallat MMI (man/machine/interface) för ubåtar med avseende på instrumentering.

Arbetets bakgrund

I dagens ubåtar har man byggt upp en förarmiljö, så att en man ensam skall kunna köra ubåten. Ubåtens manövrering i horisontalled och vertikalled är kopplad till ubåtsförarens ratt på samma sätt som i ett flygplan. Eftersom ratten är mekaniskt skild från roder och trimplan, så förloras en del av förarens fingertoppskänsla för ubåtens uppförande under manövrering. I dag erhålls en motkraft i ratten endast i form av ett par fjädrar.

Kockums vill ta fram en prototyp för en förbättrad förarmiljö till en ubåt. Två tidigare examensarbeten har gått ut på att återkoppla roderkraften till ratten. Första gruppen tog fram hårdvaran, ratt, momentgenererande motor, förstärkare, signalkonditionering och AD/DA-kort. Vidare skrevs en enkel programstruktur för simulering och grafik. Andra gruppen (1) tog fram mjukvaran till simulering, reglering och grafik. Tyvärr visade det sig att rattens roderkänsla innehöll besvärande störningar. Rattens rörelse upplevdes som ryckig, vidare befanns systemet vara alltför instabilt. Ratten var alltför känslig för stora rattrörelser och roderkrafter. Dessutom fanns ett stort glapp kring rodrets nolläge.

2. Förbättring av styrsimulator

Styrsimulatorn - en kort presentation

Styrsimulatorn skall försöka åstadkomma en rattkänsla så att föraren i ratten känner hur ubåten reagerar på roderutslag. Denna har åstadkommits genom att återkoppla simulerad roderkraft och hastighet. Styrsimulatorn är uppbyggd i tre delar.

1 En persondator med ett AD/DA-kort för simulering och grafisk presentation av kurs.

2 Ett signalkonditionerande gränssnitt, som även innehåller hastighetsreglage mellan dator och styrstativ.

3 Styrstativet består av en ratt kopplad till en momentgenerande likströmsmotor via en tandrem med utväxling 1:3. En förstärkare som omvandlar datorns utsignal från spänning till likström och en vinkelgivare kopplad till motoraxeln via en tandrem.

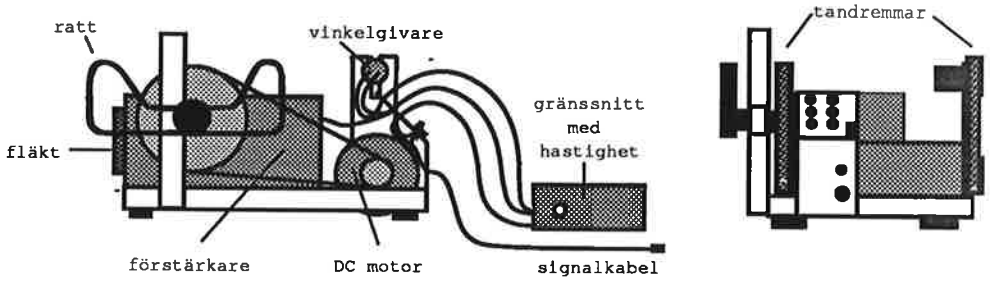
Problem uppställning

Känslan av att verkligen ha kontakt med rodret var kraftigt nedsatt på grund av att ratten hade en snäpplägeskänsla längs rattrörelsen, vibrerade och hade ett alltför stort glapp kring rodrets nollläge. Dessutom fick man vrida ratten försiktigt så man inte överbelastade förstärkaren, vidare var systemet på gränsen till instabilt. En liten stöt mot ratten räckte för att den skulle börja självsvänga. Vissa felorsaker gavs i förra gruppens rapport (1). Vår uppgift var att analysera och åtgärda orsakerna till ovan nämnda problem. Dessutom skulle vi försöka få bort den omständiga kalibreringen av rattvinkel och hastighet, som behövs varje gång man startar simuleringen.

Tillvägagångsätt

Först av allt kunde ett betydande mekaniskt glapp noteras mellan axlar, ratt och tandremshjul, vidare gick rattaxellagret trögt. Fjäderstålsprintarna mellan axlar och tandremshjul byttes mot låsskruvar, och lagret smordes upp. Friktionen minskade och det mekaniska glappet upphörde helt.

Den analoga signalkonditioneringen (A) gjordes om för att få bättre filtering, enklare kretslösning och bättre skärmning. Vidhängande låda med analoga kretsar till hastighet och vinkel slopades helt. Nya kretsar i en skärmad låda



Figur 2.1 Styrsimulatorn i ursprungsskick

sattes direkt på styrstativet. Genom att stabilisera vinkelgivarens och hastighetsreglagets matningspänning kunde kalibreringen av rattvinkel och hastighet slopas. En skyddkrets, som förhindrar att ratten börjar rotera om inte datorn är påslagen, sattes in för att förhindra personskador.

Det ursprungliga AD/DA-kortet visade sig vara för långsamt för "vår" dator, Compaq 386/20E. Kortet var av enkel typ med mycket bristfällig dokumentation. Vi valde att köpa in ett modernare AD/DA-kort (B) avsett för en AT-maskin, även detta kort visade sig vara något långsamt för Compaqen, något som man inte kunnat utläsa av generalagentens något kryptiska och ofullständiga försäljningsbroschyr (B). En inkanal klarar en AD-omvandling på utlovade $12\mu s$, inkanalsmultiplexorn arbetar asynkront och är betydligt långsammare. En fördröjning mellan insampling och AD-omvandling fick kortet att fungera tillfredställande.

3. Förbättring av rattkänla

Det gamla AD/DA-kortet krävde 1ms för en AD och en DA-omvandling tillsammans. Efter byte till ett snabbare AD/DA-kort upplevdes rattrörelsen som något förbättrad. Fortfarande fanns glappet och snäpplägeskänslan kvar i rattrörelsen.

Glappet kring rodrets nolläge fanns kvar efter de mekaniska förbättringarna. Det berodde på att den mekaniska friktionen i systemet var större än motorns moment vid små styrsignaler. En överlagrad likspäningsnivå på en 0.5V, på styrsignalen åtgärdade glappet. Det medförde dock att systemet blev nästintill instabilt. En liten stöt mot ratten fick den att självsvänga.

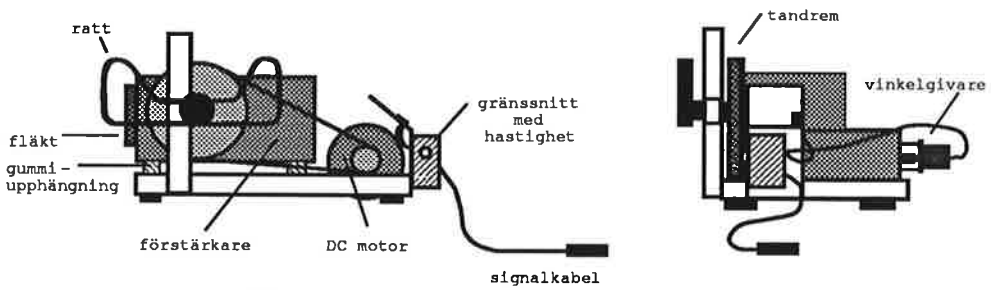
En D-regulator deriverar signalen från vinkelgivaren, vilket medför att små ändringar i insignalen ger stora styrsignaler till förstärkaren. Samplar vi dessutom med 100Hz, tål inte systemet alltför snabba ändringar i insignalen.

Den excitation som utgörs av likspäningsnivån räcker för att utlösa självsvängningen i systemet, om man slår till ratten. Dock fungerar systemet utmärkt under normala rattrörelser. På grund av en brusig insignal och en D-regulator måste insignalen lågpassfiltreras kraftigt vilket gör systemet långsamt, därav självsvängning vid snabba förlopp.

Motorns förstärkare omvandlar styrsignalens spänning till en kraftig likström för att motorn skall kunna genera ett moment. Det innebär en kraftig transformator som vibrerar med 100Hz, en fläkt som generar ett helt spektrum av vibrationer mellan 0 och 1000Hz. Dessa återkopplas akustiskt, dels som vibrationer i ratten, dels som ett överlagrat brus i givarsignalen (B). Genom att gummiupphänga störkällan det vill säga förstärkaren, lyckades vi dämpa vibrationsnivån i ratt och styrstativ ca 10 gånger (C). Efteråt kändes ratten helt fri från akustiskt återkopplade vibrationer. Ytterligare mjukare gång kunde noteras under simuleringen på grund av lägre brusnivå i insignalen.

En ny potentiometer (3) med bättre signalbrus-förhållande, och monterad direkt på motoraxeln gav också märkbara förbättringar med avseende på lägre brusnivå och minskad friktion.

Kvar att åtgärda var snäpplägeskänslan i rattrörelsen, samt ett mindre brus med en frekvens på ca 8-15Hz. De diskreta störfenomenen visade sig härröra från tandremsutväxlingen. Tandremhjulen låg inte i samma plan, vilket gjorde att tandremmen drog snett och gav upphov till någon form av ojämn gång. Efter att ha riktat hjulen och smort upp remmen försvann snäpplägeskänslan så gott som helt.



Figur 3.1 Strysimulatorn i nuvarande skick

4. Utvärdering, slutsatser och rekommendationer

Vi har lyckats komma ganska långt med våra förbättringar. Vibrationsmätningar (D) visar att det finns störningar i ratten vid ca 8 – 15Hz, vilka vi inte lyckats lokalisera. Mätningarna visar också nödvändigheten av kraftig filtrering av givarsignalen och en hög samplingshastighet. Vi tycker oss ha nått så långt vi har kunnat utan att ha gjort alltför stora och dyra ingrepp i projektet. Vill man gå längre är det risk att det kostar mer än det smakar. Vi har dock gjort vissa rekommendationer på saker som kan tänkas förbättras ytterligare.

Till sist har vi infört en ny regulator som skall simulera dagens system med fjädrar som mothåll i ratten (E).

Rekommenderade åtgärder för ytterligare förbättringar

1 Ny starkare motor, så att ratten kan sättas direkt på motoraxeln. Då kan remmen slopas helt, vilket skulle ta bort friktionen helt och eventuella kvarvarande störningar genererade av remmen. Nackdelen är att alla akustiska vibrationer förorsakade av motorn kopplas direkt till ratten.

2 I stället för ny motor kan man byta ut tandremsutväxlingen mot en planetväxel som ger en mjukare gång, har dock samma nackdelar som förslag 1.

3 En digital optisk givare i stället för dagens givare, med släpkontakt mot en bana av konduktiv plast, borde ge ett bättre signalbrus- förhållande hos datorns insignal, vilket borde möjliggöra en lägre filteringsgrad i ingångens lågpassfilter.

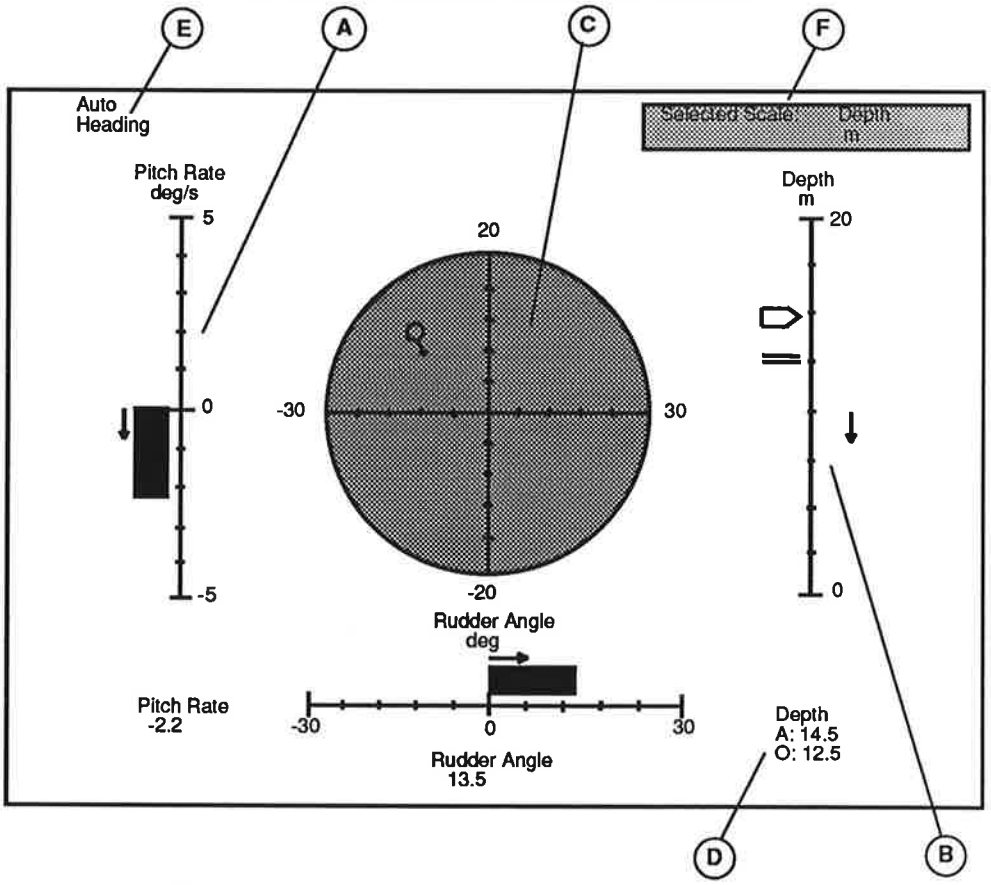
4 En omskrivning av programmet i ett realtidspråk borde ge ett snabbare system, som ger en möjlighet till en högre samplingsfrekvens. Det nuvarande AD/DA-kortet utgör ingen begränsning, då det är snabbare än 1ms.

5. Programspecifikation för MMI

Vår uppgift har varit att skriva ett program, som möjliggör, att man på ett enkelt sätt skall kunna bygga upp en egen förarmiljö för en ubåt på en datorskärm. Förarmiljön skall bestå av ett antal instrument, som på ett snabbt sätt visar på datorskärmen, hur ubåten beter sig. Instrumenten är utformade så att föraren skall få maximal information om ubåtens beteende med hjälp av så få instrument så möjligt. Kockums önskemål om vilka instrument som skall vara möjliga att skapa visas i figur 5.1. Här följer en beskrivning av samtliga typer av instrument.

Skalor

För det första är det möjligt att göra skalor som man kan presentera en godtycklig parameter i. Det finns två typer av skalor. I den första typen, instrument A i figur 5.1, visas ett variabelvärde med hjälp av en färglagd stapel. I den andra typen, instrument B i figur 5.1, kan man visa två variabler, där pilmarkeringen visar ärvärdet och streckmarkeringen visar börvärdet av en parameter. För en godtycklig skala kan man välja följande:



Figur 5.1 Exempel på möjliga instrument till förarmiljön

- Längden av skalstrecket angivet i millimeter
- Färgen på skalan och tillhörande text
- Vilka variabler som skall presenteras i skalan
- Vilken typ av gradering skalan skall ha
- Horisontell eller vertikal skala
- Antalet skalstrecksintervall
- Positionen på skärmen
- Skalans ändvärden
- Tillhörande text till skalan

Har man valt skaltyp enligt instrument A i figur 5.1 kan man välja:

- Staplens färg
- Stapelns bredd

Har man valt skaltyp enligt instrument B i figur 5.1 kan man välja:

- Ärvärdespilens färg
- Börvärdesmarkeringens färg

Man kan välja en godtycklig graderingstyp för skalan. De möjligheter som finns inlagda i programmet är linjär, logaritmisk och eget val. Väljer man eget val måste man föra in en egen vald algoritm i proceduren "F", som finns i modulen "CALC". Den tillhörande texten till skalan består av två rader text à tolv tecken, som placeras direkt ovanför skalan.

Hårkors

Den andra typen av instrument, som man kan skapa är instrument C i figur 5.1, kallad hårkors. Instrumentet består av ett vertikalt och ett horisontellt graderat skalstreck, y- respektive x-axeln, som är placerade i en cirkel. För x-axeln läses en parameter in och för y-axeln läses en annan parameter in. Presentationen av aktuell position i hårkorset sker med hjälp av en markör, som består av en liten färglagd cirkel. I den riktning som markören rör sig i hårkorset markeras med hjälp av en tendenspil enligt figur 5.1. Om markören skulle hamna utanför cirkeln, flyttas markören in precis innanför cirkeln och tendenspilen ritas då inte ut. För ett hårkors kan man välja följande:

- Hårkorsradie angiven i millimeter
- Färgen på cirkeln och skalorna
- Bakgrundsfärgen i hårkorset
- Positionen på skärmen
- Vilka variabler som skall presenteras på x-axeln och y-axeln
- X-axelns och y-axelns ändvärden
- Antalet skalstrecksintervall för x-axeln och y-axeln
- Vilken typ av gradering skalorna skall ha
- Färgen på markören och tendenspilen

Digitalt visande instrument

Det tredje instrumentet som man kan skapa är ett digitalt visande instrument enligt instrument D i figur 5.1. I ett sådant instrument presenteras värdet av maximalt två parametrar. För ett digitalt visande instrument kan man välja följande:

- Tre rader text à tolv tecken
- Positionen på skärmen
- Vilka variabler som skall presenteras (maximalt två stycken)
- Färgen på texten

Text på skärmen

Det finns även möjlighet att skriva ut text på skärmen enligt instrument E i figur 5.1. För respektive textblock kan man välja:

- Tre rader text à tolv tecken
- Färgen på texten

Knapparna F1-F10

För att möjliggöra kommunikation från användarens sida till bildskärmen används tangentbordsknapparna F1-F10. Varje knapp har en "flagga" implementerad, vilken kan sättas med ett nedtryck av respektive knapp. Knapparna F6-F10 är lediga för användaren, som med hjälp av dessa knapparna bland annat kan programmera utskrift på skärmen . Resterande knapparna F1-F5 är redan programmerade och vars funktioner hänvisas till i användarmanualen (I). Vilken av knapparna F1-F5 man valt visas i en ruta i översta högra hörnet enligt instrument F i figur 5.1.

6. Programlösning för MMI

Programmet är uppbyggt med hjälp av fem moduler enligt figur 6.1, där modulnamnet anges i varje modul.

I modulen "CREATE" finns procedurer, som ritar den statiska delen av förarmiljön det vill säga de figurer, som bara behöver ritas en gång. Dessutom finns här procedurer, som sköter omskalningen av instrumenten.

I modulen "PLOT SET" finns procedurer, som ritar den dynamiska delen av förarmiljön det vill säga de figurer, som behöver ritas om varje gång skärmen uppdateras. Dessutom finns här procedurer, som programmerar knapparna F1-F10.

I modulen "CALC" finns funktionerna som utför beräkningar och omvandlingar från millimeter till antalet punkter på skärmen.

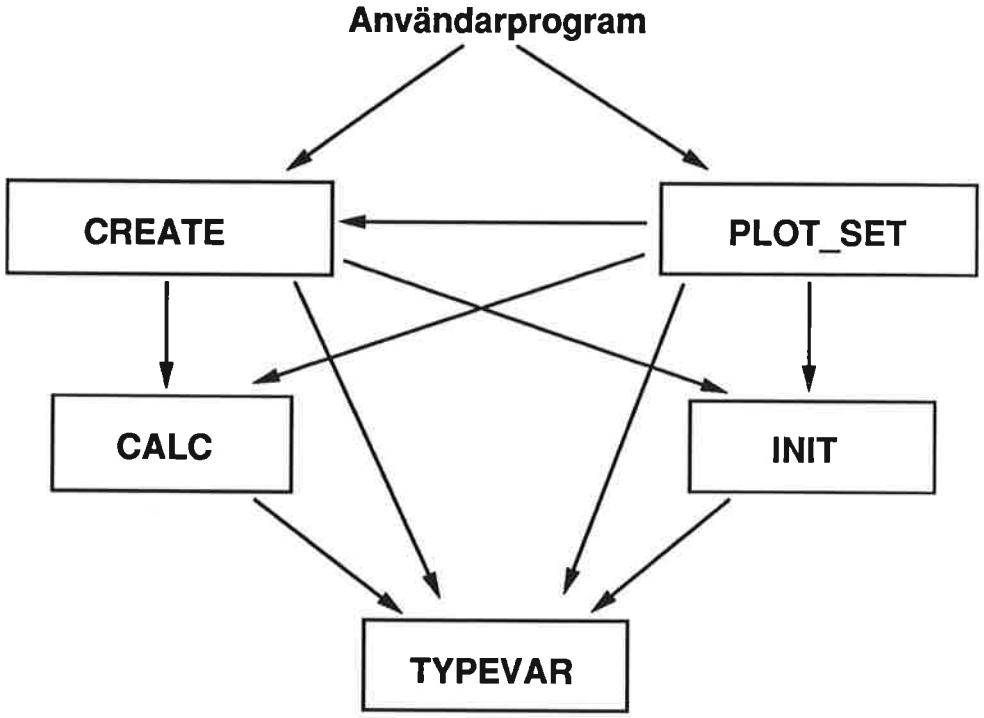
I modulen "INIT" finns alla procedurer, som gör initieringen vid programmets början.

I modulen "TYPEVAR" finns alla deklarationer av variabler, typer och konstanter som modulerna använder.

Programmet är konstruerat på ett sådant sätt, att användaren kan bygga upp sin egen förarmiljö med hjälp av de instrument, som finns implementerade i programmet. Detta krav medför att procedurerna är uppbyggda så oberoende av varandra som möjligt.

I användarprogrammet bygger användaren själv upp den skärm som önskas genom lämpliga proceduranrop till modulerna. (se vidare i användarmanualen (I)).

Varje gång skärmen uppdateras måste procedurerna i "PLOT SET" få tillgång till nya variabelvärden från användarprogrammet. Detta sker genom



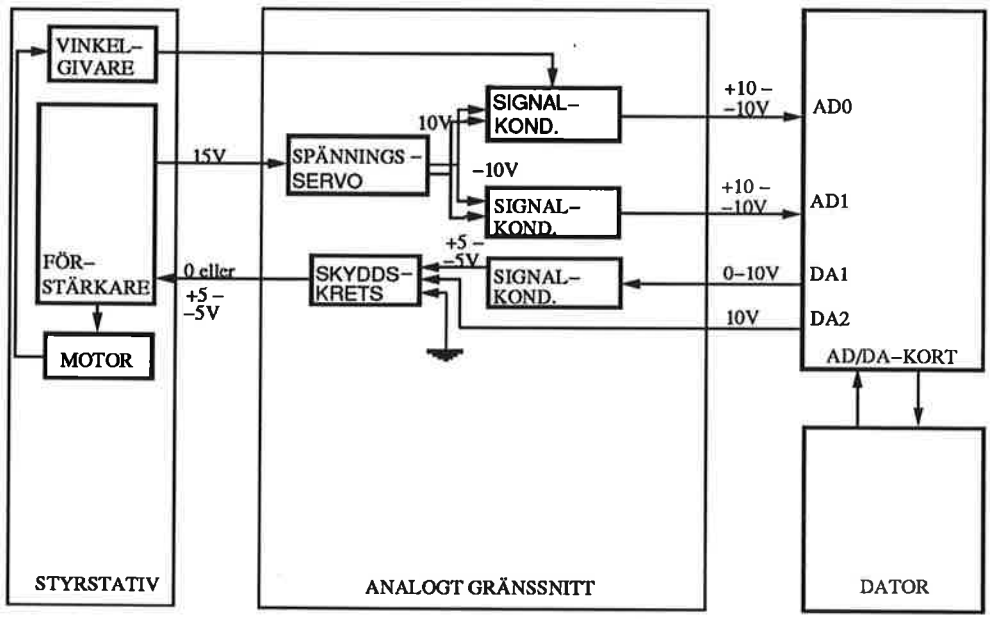
Figur 6.1 Programuppbyggnad

ett gränssnitt, som består av en vektor med alla de variabler, som skall presenteras på skärmen.

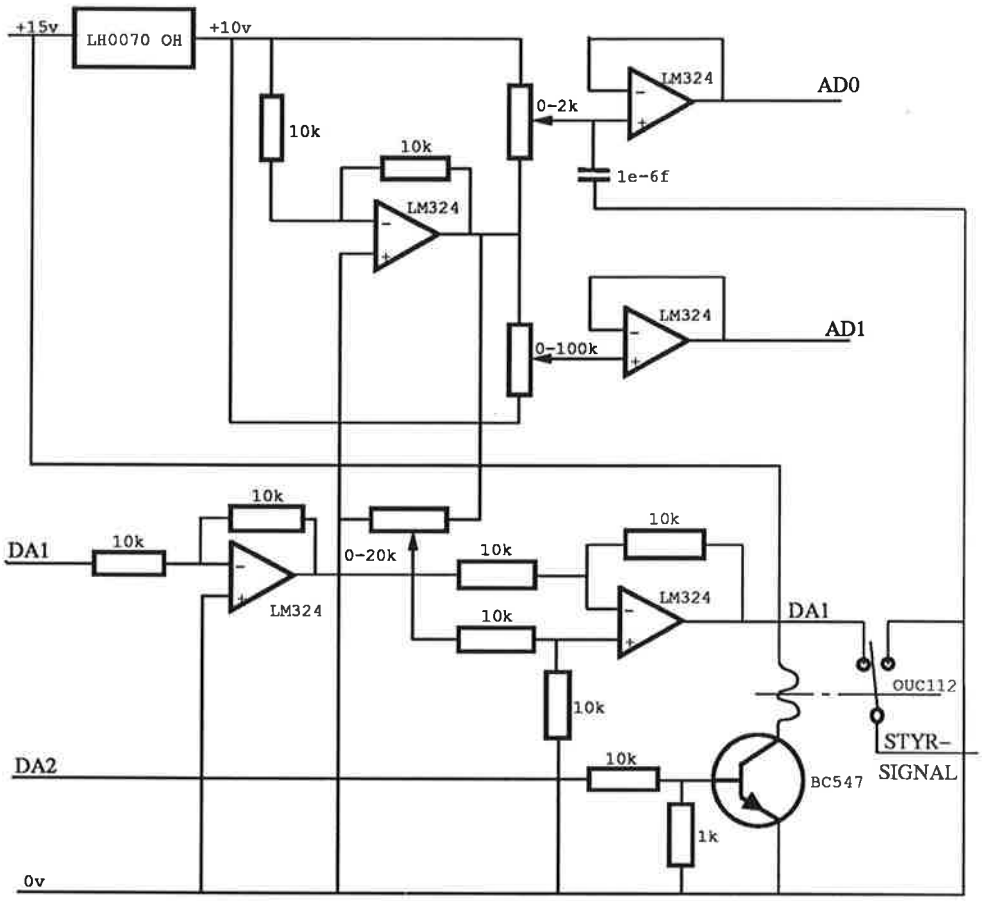
A. Det analoga gränssnittets kopplingsschemor

De analoga kretsar som utgör gränsnittet mellan dator och styrstativ är placerade i en skärmad box fastsatt på styrstativet. Gränsnittet består av signalkonditionerande kretsar och en skyddskrets som förhindrar att ratten börjar rotera okontrollerat om styrstativet startas utan att datorn är i gång.

Vinkelgivaren består av en Midori $0 - 2k\Omega$ vridpotentiometer (3) av hög kvalitet. Ett spänningservo håller spänningen över vinkelgivarpotentiometern och hastighetsreglagepotentiometern exakt till $\pm 10,000V$. Skyddskretsen består av ett relä som jordar förstärkarens ingång när datorn är frånslagen.



Figur A.1 Blockschema över det analoga gränssnittets funktion



Figur A.2 Kopplingsschema över det analoga gränsnittet

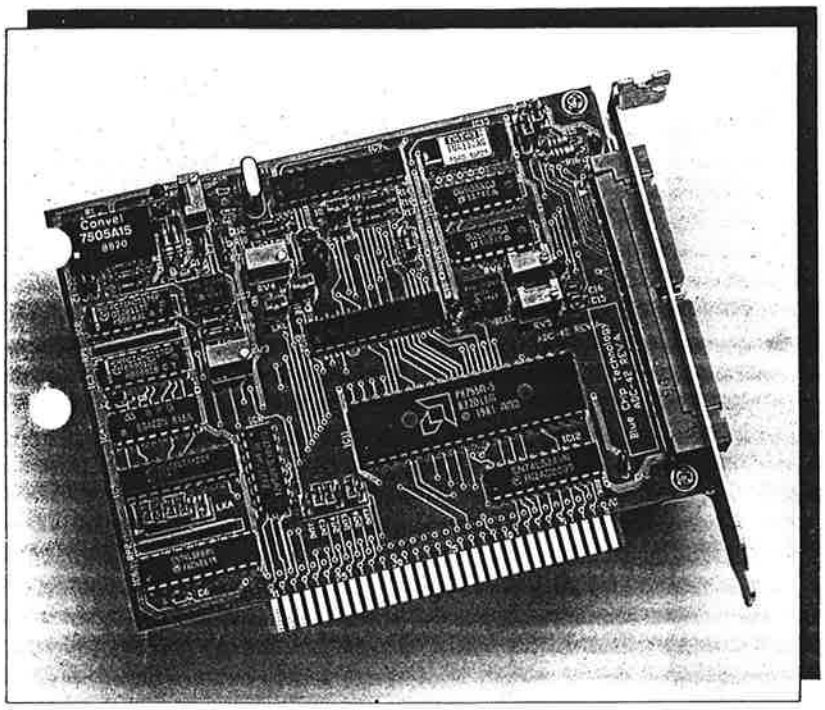
B. AD/DA-kort

AD/DA-kortet är i första hand inköpt, eftersom det förra kortet inte passade Compaqen 386-processor utan var avsett för 1:a generationens PC. Det hade krävts modifieringar i inläsningsrutinen för att sänka datorns inläsningshastighet, vilket skulle medföra onödiga prestandaförsämringar. Omskrivningarna skulle dessutom ta lång tid, eftersom instruktionsboken var av mycket bristfällig karaktär. Med ett modernare kort av högre kvalitet och avpassat för en modernare processor, höjs insamlingtakten av data betydligt, vilket är önskvärt, om man till exempel vill höja insamplingstakten, genom att skriva om programmet i ett realtidsspråk. Flaskhalsen i det nya kortet är inkanalsmultiplexorn som är långsammare än AD-omvandlarerns $12\mu s$. För att kunna styra fler än en kanal med ett högnivåspråk krävs, att man lägger in en liten fördröjningsloop mellan kanalmultiplexningen och AD-omvandlingen, så att multiplexorn hinner skifta kanalkrets innan AD-omvandlingen startar.

BLUE CHIP TECHNOLOGY

ADC-42

Multi—function, 12 Bit Combination Card



The ADC-42 combines the most popular input/output functions all on one low cost card. 12 Bit converters are used to give the highest resolution and accuracy. An on-board timer may be used for fixed frequency sampling.

The analogue input section has 16 single ended or 8 differential channels. The 12 bit Analogue to Digital converter has a fast, 12 microsecond conversion speed. Input ranges can be selected from 10, 5 or 2.5V by on-board links.

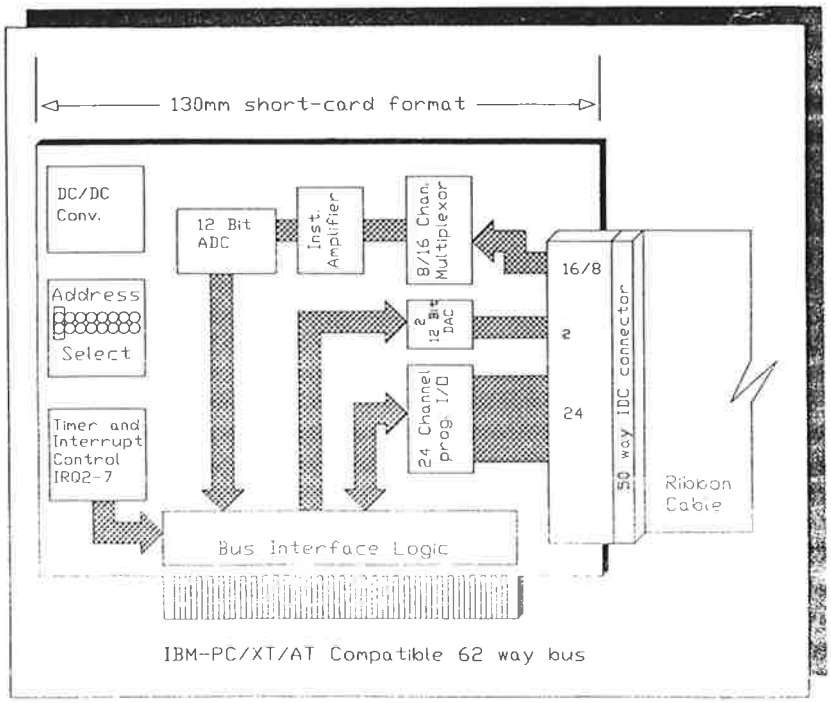
Two 12 bit Digital to Analogue converters are provided for high accuracy control outputs. Full scale may be 10 or 5V, again selected by on-board links.

For versatile digital I/O an 8255 PIO chip is included. This has 24 TTL lines which may be programmed as input or output. Many combinations are possible such as all input, all output or 12 in and 12 out.

The card comes with example programs and full documentation.

It is compatible with the IBM PC, XT, AT, 386, PS/2 Model 30 and any compatible. All signals are terminated on a 50 way IDC ribbon cable connector.

- 16/8 Analogue Inputs
  - 12 Bit, 12 microsecond ADC.
  - 0-10, 5 or 2.5V ranges.
- 2 Analogue Outputs.
  - 12 Bit DAC.
  - 0-10 or 0-5V ranges.
- 24 Programmable I/O.
  - 3 × 8 Bit ports.
  - TTL Levels.
- On-board Programmable Interrupt Source.
- Various Versions Available.
- IBM PC/XT/AT, Model 30 or Compatible.
- Short Card Format.
- 50 Way Ribbon Cable Connector.
- Full Documentation and Example Programs.



Blue Chip Technology, Main Avenue, Hawarden Industrial Park, Deeside, Clwyd, CH5 3PP.
Telephone (0244) 520222 Fax: (0244) 531043 Telex: 61471

A member of the Kemitron group.

C. Akustisk återkoppling

Mätningarna på olika delar av styrstativet gjordes för att konstatera, hur mycket vibrationer styrstativet själv generar, och vad som orsakade dessa. Mätresultaten presenterades av spektrumanalysatorn som ett vibrationsspektrum med hastighet som funktion av frekvens. Störkällan antas generera störsignalen som en funktion av en mängd sinussignaler. Läget ges av

$$x = \frac{\hat{a}}{\sqrt{2}} \sin wt$$

Hastigheten ges av

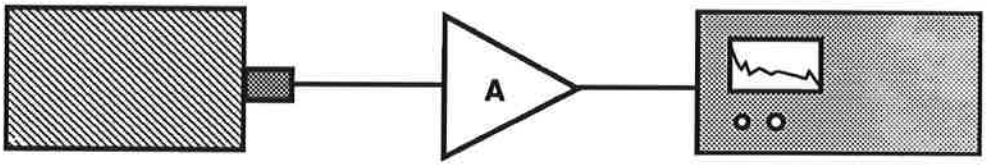
$$\frac{dx}{dt} = \frac{\hat{a}}{\sqrt{2}} w \cos wt$$

Kurvorna visar alltså $\hat{a}w/\sqrt{2}$ fördelat på resonanstoppar och bakgrundsbrus. Om man istället vill ha amplituden på svängningarna skall man alltså dividera hastigheten med w.

Vi lyckades således påvisa, att störkällan sannolikt var fläkten och transformatorn hos förstärkaren. Enklast var att försöka gummiupphänga förstärkaren med mjuka gummiklossar. Systemet med förstärkaren vilande på gummiklossar kan ses som ett andra ordningens lågpassfilter (4). Förstärkarens massa kan liknas vid en spole och gummiklossarna vid en kondensator. Lågpassfiltrets överföringsfunktion ges som

$$H(w) = \frac{1}{(1 - w^2 LC)}$$

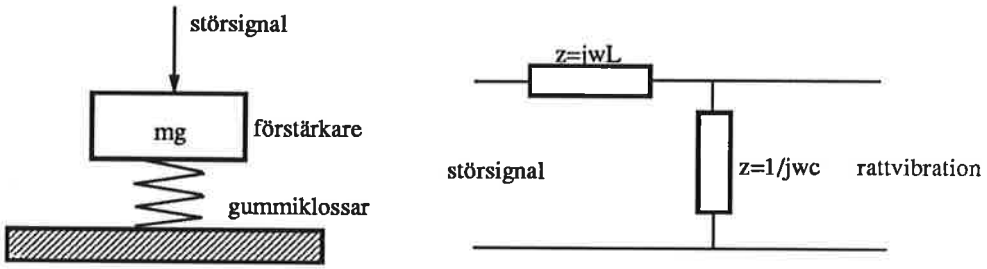
Gränsfrekvensen fås approximativt till $w_0 = \sqrt{k/m}$, $F = kl$, $F = mg$. Efter hyfsning fås $w_0 = \sqrt{9.81/l}$. Gummiklossarna fick monterade en sammanpressning på 15mm, vilket ger $w_0 = 30$Hz. Det stämmer ganska bra med mätningarna.



mätobjekt mätsignalsförstärkare FFT spektrumanalysator

accelerometerpickup

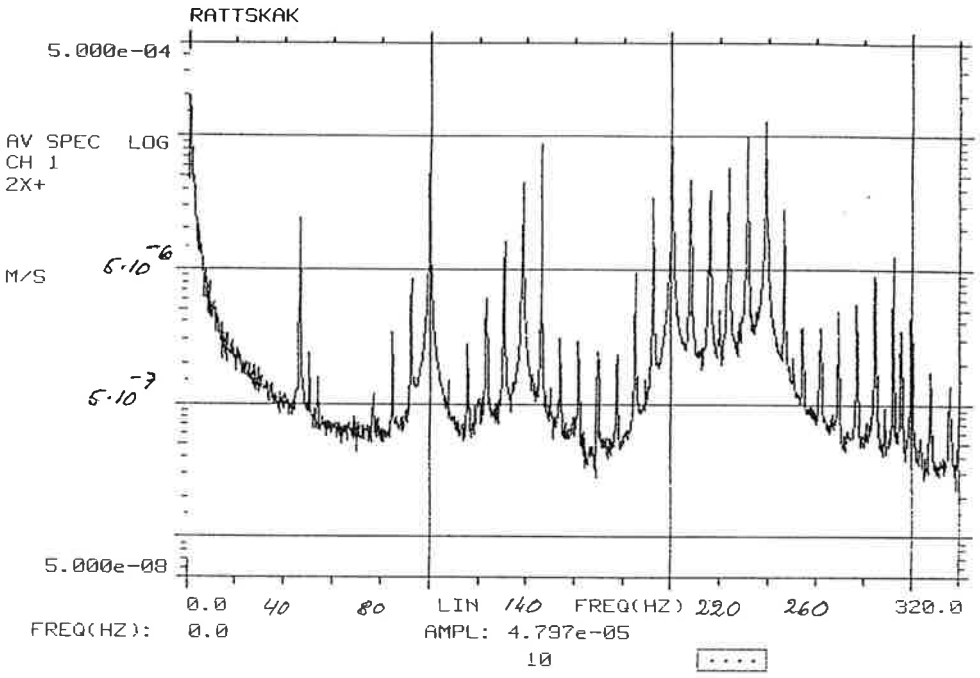
Figur C.1 Mätuppställning



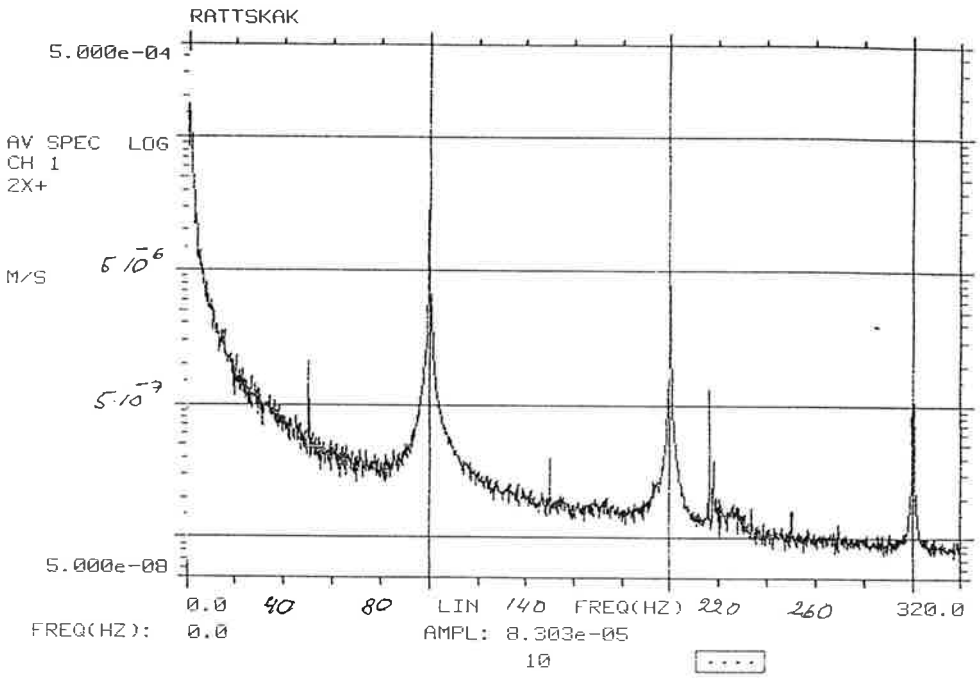
styrstativ inklusive ratt

mekaniskt system elektrisk analogi

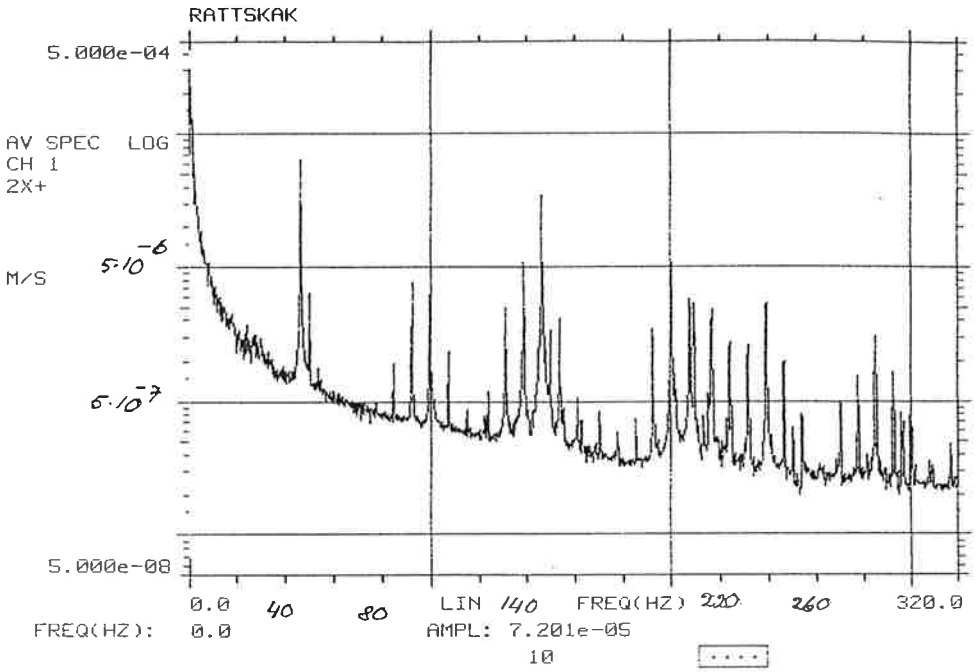
Figur C.2 Gummiupphängningens filterkaraktäristik



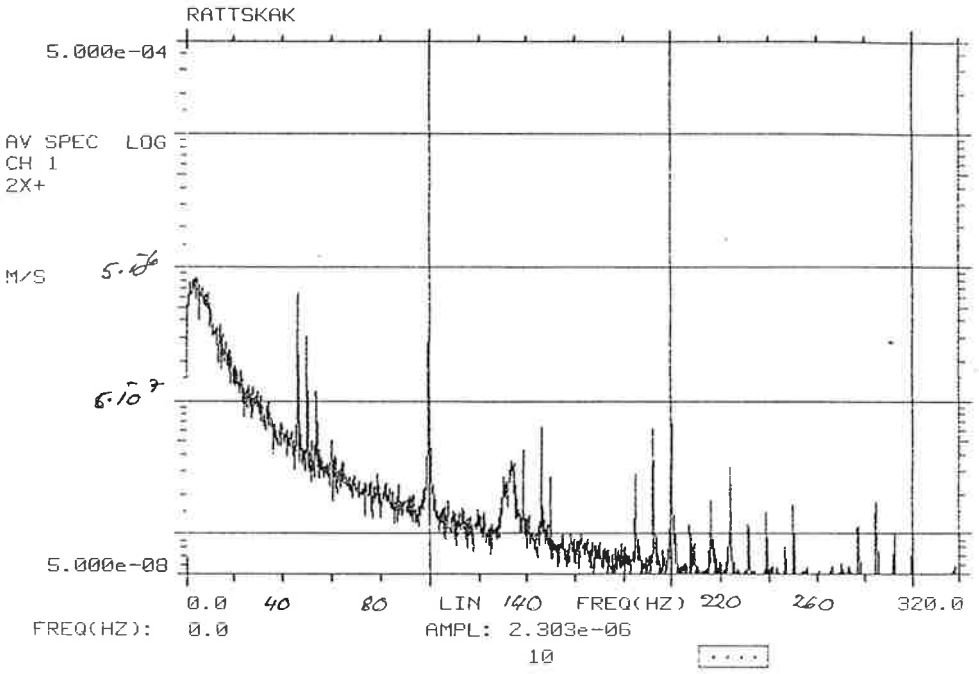
Figur C.3 Förstärkare med fläkt och transformator i gång



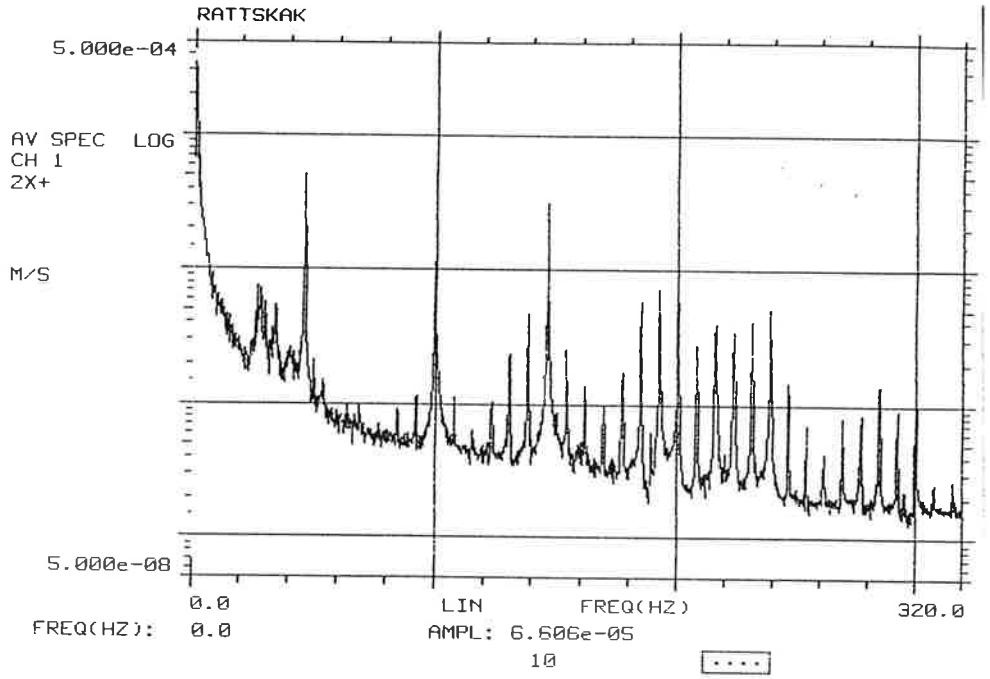
Figur C.4 Förstärkare med transformator i gång



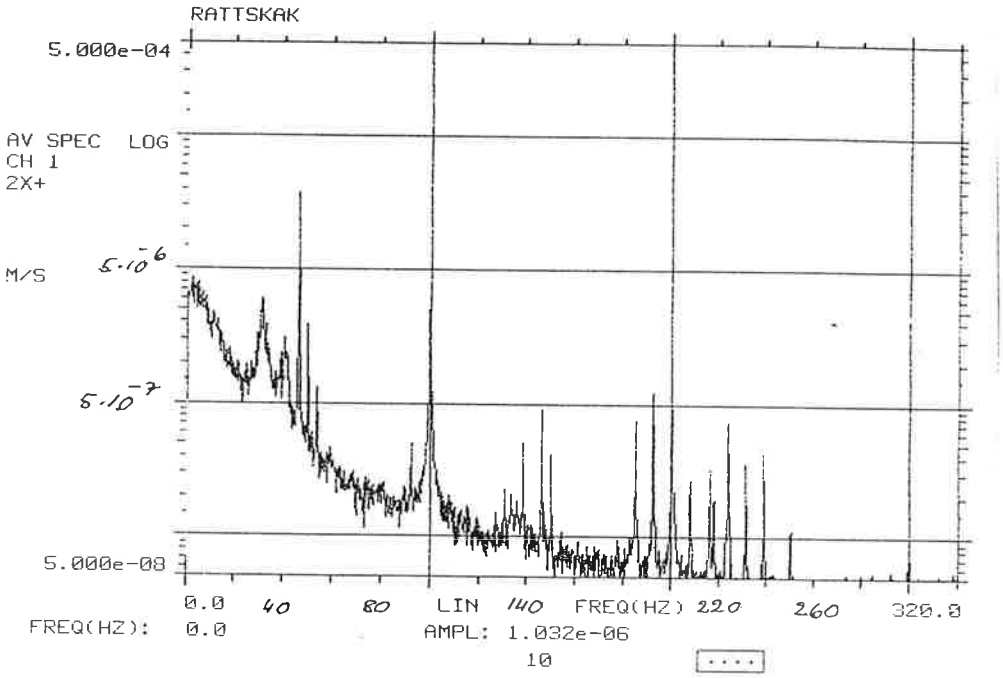
Figur C.5 Rattfäste innan åtgärd



Figur C.6 Rattfäste efter åtgärd



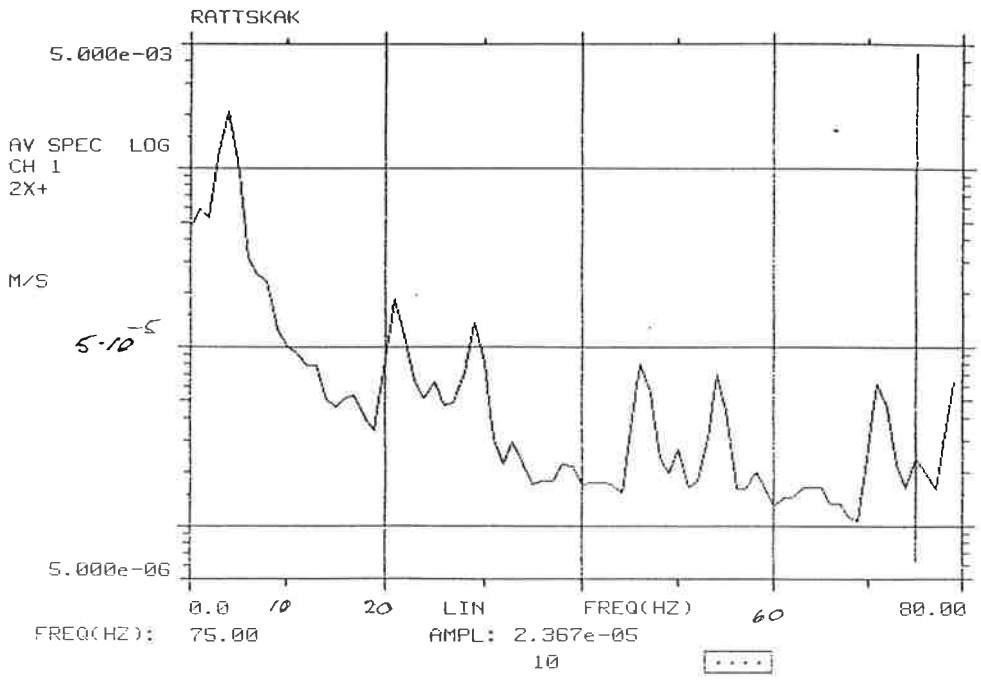
Figur C.7 Vinkelgivare innan åtgärd



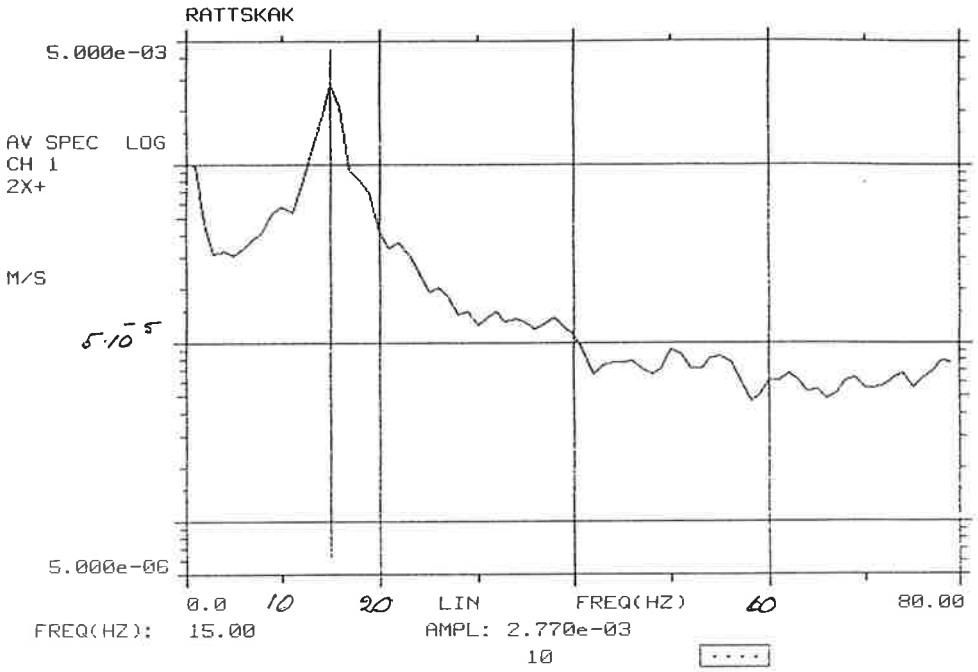
Figur C.8 Vinkelgivare efter åtgärd

D. Utvärderande mätningar

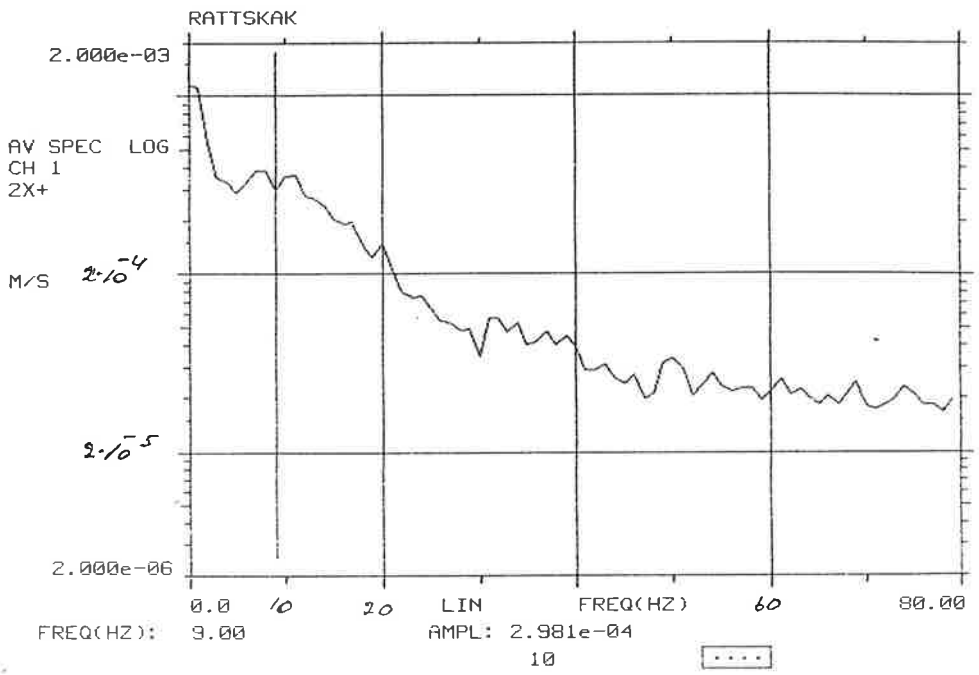
Mätresultaten är likartade för alla regulatorerna, varför endast schablonregulatorns mätresultat är redovisade. Mätuppställningen är samma som i (C). Mätningarna visar inverkan av för låg sampelfrekvens och inverkan av olika filterkonstanter för datorns lågpassfiltrering av vinkelgivarens insignal. En mätning har $f_s = 25$Hz och schablonvärden för övrigt. Resten av mätningarna har $f_s = 100$Hz och ingångens filterkonstant varieras. Filterkonstanten α_{in} som har schablonvärde 0.93 viktar nytt filtrerat värde i förhållande till äldre filtrerat värde. α kan varieras från 0 till 1 Mätningarna är gjorda med $\alpha_{in} = 0.80, 0.93, 0.99$. Datorns lågpassfilter på utgången är av samma typ och har schablonvärde $\alpha_{ut} = 0.2$ genom alla mätningarna. Om man ökar någon av filterkonstanterna ytterligare, får man visserligen en mjukare gång i ratten men också ett betydligt långsammare system, som ger ratten en gummiliknande och oprecis känsla. En lågfrekvent störning finns vid cirka 10 – 15Hz. Övriga resonanser är väl undertryckta. I figur 1 kan man tydligt se inverkan av för låg sampelfrekvens. Grundtonen vid $25Hz$ och dess övertoner framstår som tydliga resonanser. Grundtonen och övertonerna är dessutom frekvensmodulerade, vilket framkommer av att de omges av två toppar. Fenomenet är bekant för radiotekniker som DSBSC (Dubble sideband supressed carrier). En tydlig ganska bredbandig topp $f = 15Hz$ kan ses i figur 2. I figur 3 har toppen i figur 2 nästan försvunnit, och den allmänna brusnivån har minskat cirka 2,5 gånger. Bruset i figur 4 har minskat ytterligare, så att mätbruset vid $50Hz$ börjar framträda.



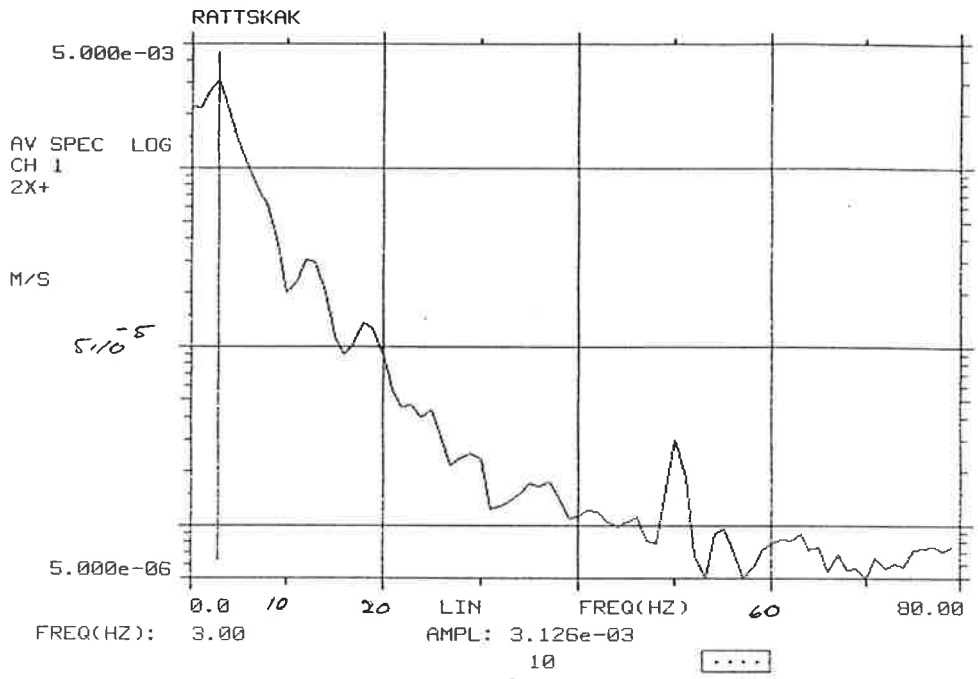
Figur D.1 $f_s = 25Hz, \alpha_{in} = 0.93, \alpha_{ut} = 0.2$



Figur D.2 $f_s = 100Hz, \alpha_{in} = 0.80, \alpha_{ut} = 0.2$



Figur D.3 $f_s = 100Hz, \alpha_{in} = 0.93, \alpha_{ut} = 0.2$



Figur D.4 $f_s = 100Hz, \alpha_{in} = 0.99, \alpha_{ut} = 0.2$

E. Simulering av fjäder

Vi har valt att lägga in ytterligare en regulator i systemet för att försöka simulera dagens system med mothåll i ratten i form av två fjädrar.

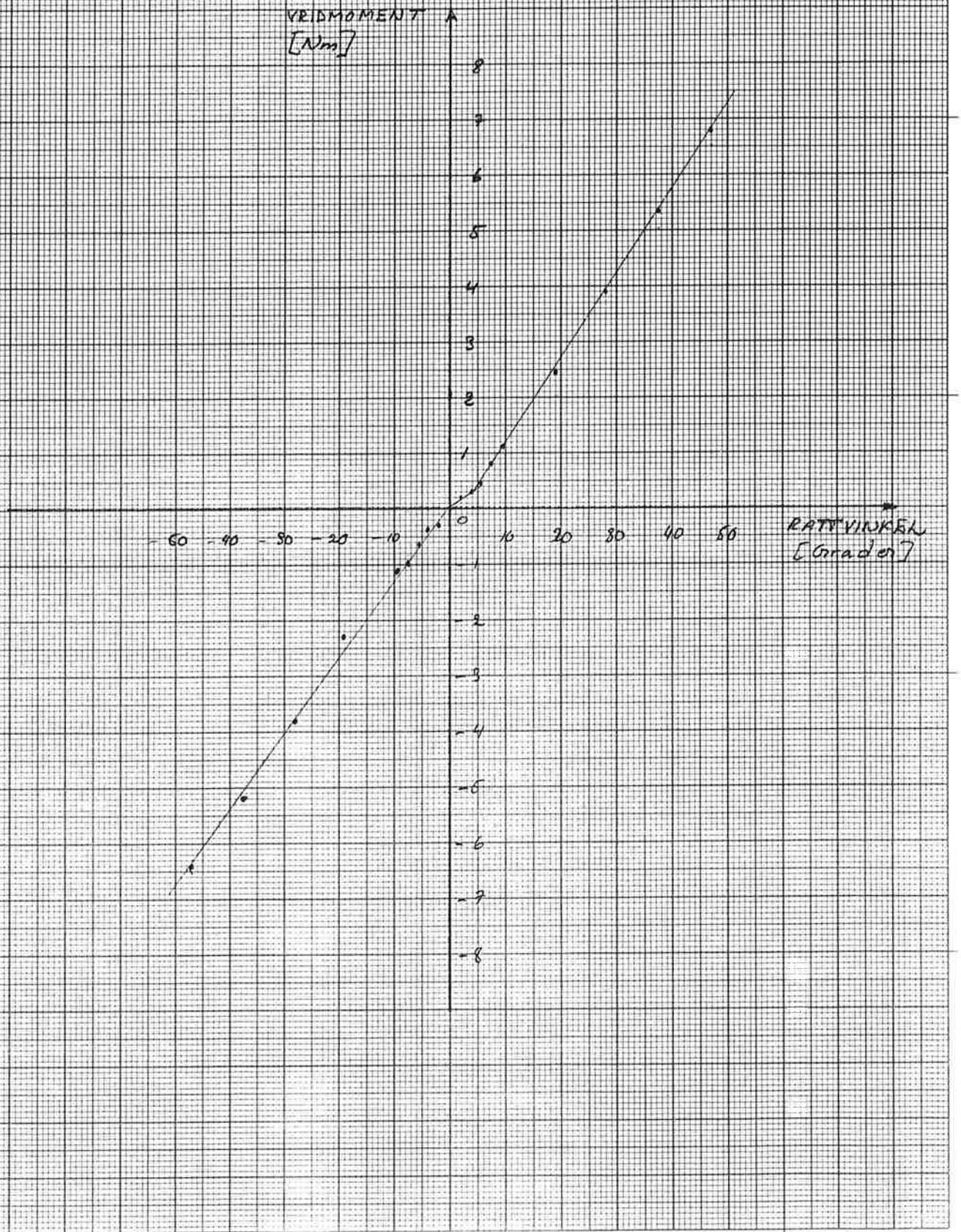
Detta utfördes för att se om vårt simulerade systemet kan efterlikna verkligheten och för att få en referens för jämförelse med dagens system. Mätningarna gick så till, att en dynamometer fästes i ratten och lästes av vid valda rodervinklar.Därefter räknades rodervikel om till mot svarande rattvinkel, och kraften till motsvarande moment.

Första kurvan är presenterad för att kontrollera linjäriteten i systemet. Den andra kurvan innehåller uppmätta värden från HMS Västergötlands ratt och vår simulerade version.

Anledningen till att vi inte kunde åstadkomma samma utseende på momentkurvan kring rattens nolläge är, att ratten pendlade mellan de olika startvärdena på kurvan.

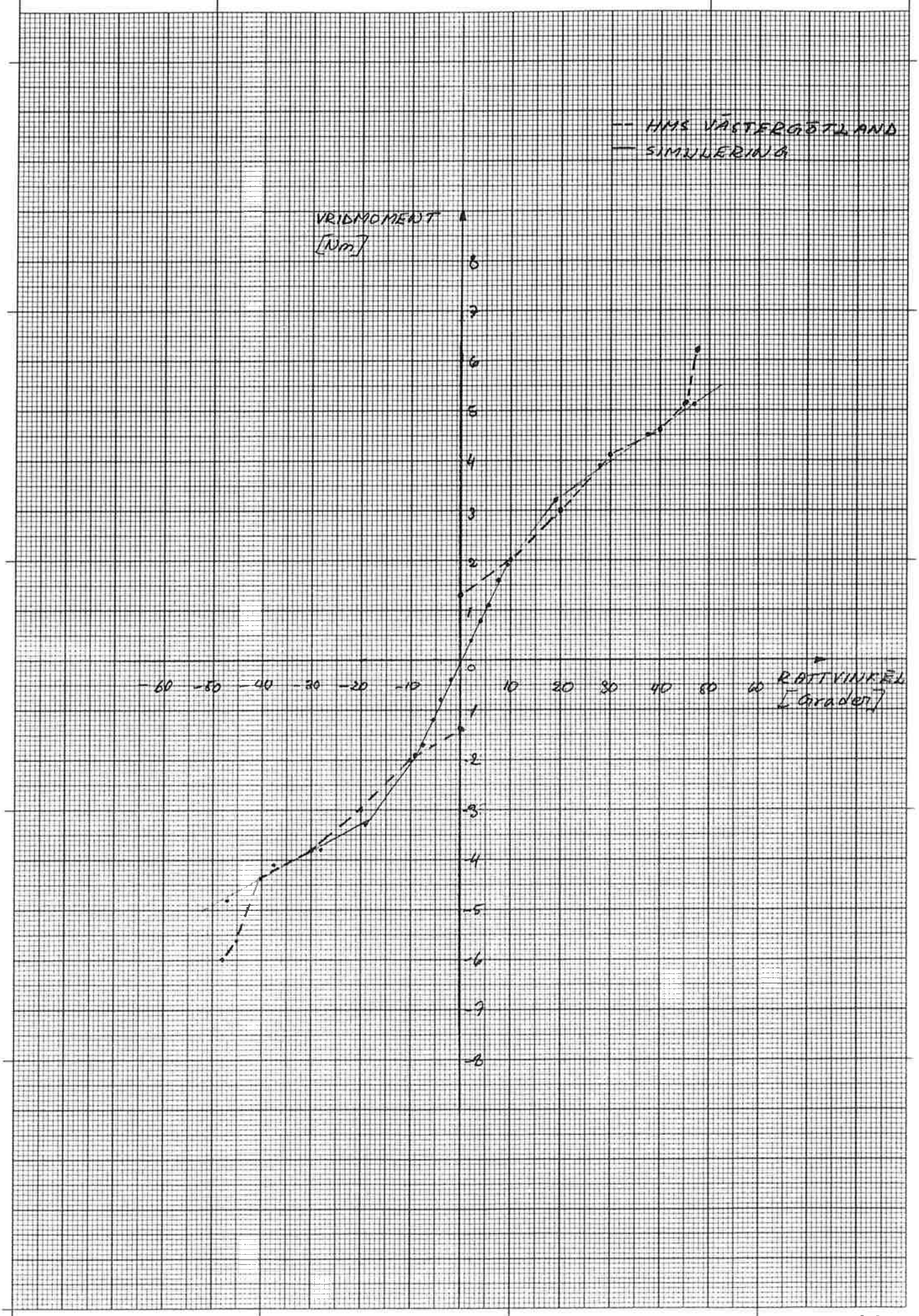
PROPORTIONELL ÅTERKOPPLING
LINJÄRITET

KURVA E:1



MODIFIERAD PROPORTIONELL ÅTERKOPPLING
HMS VÄSTERGÖTLAND

KURVA E:2
LINJÄRITET



F. Programlistning Styrsim

Fullständiga och utförliga programförklaringar finns i Farkoststyrning och förarmiljö. Här förklaras endast våra ändringar i programmet.

Nya AD och DA-procedurer är skrivna för det nya AD/DA-kortet (2). Procedurerna Feedback och Calibrate är förändrade och proceduren Steeringwheelzero är ny.

DA1 Utkanalen för styrsignalen.

DA2 Utkanalen för skyddskretsen. 10V skickas ut till reläet för att det skall slå om förstärkarens ingång från jord till DA1.

Feedback En kompensation har lagts in för att motverka den mekaniska friktionen i systemet.

Calibrate Eftersom man inte längre behöver kalibrera ratt och hastighet, är numera maximala rattutslag och maximal hastighet satta till schablonvärden enligt det nya AD/DA-kortets (2) förusättningar.

Steeringwheelzero Ratten måste horisonteras för att regulatorn inte skall ge alltför kraftig styrsignal. Steeringwheelzero horisonterar ratten innan regleringen startas.

Konstanter med avseende på olika nivåer i styrsignalen är ändrade för att passa det nya AD/DA-kortets förutsättningar. Till exempel 0V är ändrad från 2048 till 2000.

```
PROGRAM StyrSimulering;
(* Observera att programmet räknar med meter/s och vinkeln radianer *)
(* Roderutslaget multipliceras en faktor fyra för att erhålla dr    *)
ä$N+å

USES CRT,GRAPH,DBPLUS;

TYPE
  MatrixArray=ARRAYÄ1..6,0..20Å OF REAL;
  MatrIndTyp =ARRAYÄ1..6Å OF CHAR;

CONST
  Map           : ViewPortType =
                   (x1:  20; y1:  20; x2: 320; y2: 460; Clip : ClipOn);
  Compass       : ViewPortType =
                   (x1: 470; y1: 108; x2: 630; y2: 268; Clip : ClipOn);
  SpeedometerR  : ViewPortType =
                   (x1: 340; y1: 380; x2: 630; y2: 462; Clip : ClipOn);
  AnglometerDR  : ViewPortType =
                   (x1: 340; y1: 283; x2: 630; y2: 365; Clip : ClipOn);
  SpeedometerU  : ViewPortType =
                   (x1: 340; y1: 108; x2: 422; y2: 268; Clip : ClipOn);
  StopDisplay   : ViewPortType =
                   (x1: 440; y1:  20; x2: 530; y2:  80; Clip : ClipOn);

  MatrInd :MatrIndTyp =('a','b','c','d','e','f');
  IntrMask            = $21;
  TimerCom            = $43;
  Kanal0              = $40;
  DrChannel           = 0;
  uChannel            = 1;
  ClFr                = 1.19E6;

VAR
  uKnot,OldUKnot,OldUPl,I,J,OldMask,WeelValue,
  XX,YY,OldRPl,RPl,Choise,XKoord,YKoord,FeedBackChoise,
  LoopCount,GraphDriver,GraphMode,OldPlUKnot,NConstR,
  NConst,s,DRPl,OldDRPl,UPl,SampleFr                    :INTEGER;


  a,b,c,d,e,f,Ts,dr,drOffset,drMultFact,Delta,DConst,
  uMultFact,r,v,v0,psi,x,y,cospsi,LowPassConst,K1,
  sinpsi,Alfa,Beta,LowPassConst2,OldFiltValue,TdConst,
  OldFiltValue2,OldValue,OldValueR,OldHelpVar,K6,K7,
  TdConstR,DConstR,BetaR,OldHelpVarR,Ordn2LowPassConst,
  OldY1Filt,OldY2Filt,K2,K3,K4,K5,K8,K9,K10,K11,K12,K13,
  K14,OldFeedBackValue,dev                          :REAL;

  A_B                                               :MatrixArray;
  SpeedPlot                                         :ARRAYÄ0..20Å OF STRI
NG;
  fi                                                :FILE OF REAL;
  Quit,Stop,Calibrated,Regret,Moved,Break           :BOOLEAN;

(*****************************************************************)



PROCEDURE Ram;
(* Ritar en ram kring alla menyer *)
BEGIN
  ClrScr;
  HighVideo;
  TextColor(14);
  Write('╔═══════════════════════════════════════════════════════════
```

```
═══════════╗');
  Write('╟  EXAMENSARBETE                                          KOCKUMS
MARINE AB ╟');
  Write('╟  STYRKÄNSLA                                            VAPENSEK
TIONEN    ╟');
  Write('╠═══════════════════════════════════════════════════════════════
═══════╣');
  FOR I:=1 TO 19 DO
    Write('╟
          ╟');
  Write('╚═══════════════════════════════════════════════════════════════
═══════╝');
  TextColor(7);
  HighVideo;
END; (* Ram *)

(*************************************************************************)


PROCEDURE Noise ( Fr,Ms :INTEGER);
(* Åstadkommer ljud *)
BEGIN
  Sound(Fr);
  DELAY(Ms);
  NoSound;
END; (* Noise *)

(*************************************************************************)


PROCEDURE FailSound ( Alt :REAL);
(* Ödesljud *)
BEGIN
  Noise(ROUND(440/Alt),300);
  Noise(0,20);
  Noise(ROUND(440/Alt),300);
  Noise(0,20);
  Noise(ROUND(440/Alt),300);
  Noise(0,20);
  Noise(ROUND(350/Alt),900);
END; (* FailSound *)

(*************************************************************************)


PROCEDURE CrashSound;
BEGIN
  FOR I:=1 TO 1000 DO
  BEGIN
    Noise(Round(Random*40),1);
    Noise(Round(Random*45),1);
  END;
END; (* CrashSound *)

(*************************************************************************)


PROCEDURE FunSound;
(* Skapar introduktionsljudet *)
BEGIN
  I:=200;
  J:=6;
  REPEAT
    REPEAT
      Noise(I,J);
      I:=I+30;
```

```
    UNTIL I>4000;
    REPEAT
      Noise(I,J);
      I:=I-30;
    UNTIL I<200;
    J:=J-1;
  UNTIL J=1;
END; (* FunSound *)

(*************************************************************************)


PROCEDURE IntroductionMeny;
(* Ger kort förklaring av programmet *)
BEGIN
  HighVideo;
  ClrScr;
  Ram;
  GoToXY(10,9);
  Write('Detta program simulerar en ubåts manövrar i girled. Simulatorn');
  GoToXY(10,10);
  Write('mäter rattvinkel och hastighet samt beräknar därefter ubåtens');
  GoToXY(10,11);
  Write('rörelse och ritar ut denna på skärmen. Även en kraft-term');
  GoToXY(10,12);
  Write('beräknas och skickas ut till en momentmotor som styr kraften');
  GoToXY(10,13);
  Write('som känns i ratten.');
  GoToXY(10,14);
  Write('För att utprova styrkänsla har ni flera olika alternativa');
  GoToXY(10,15);
  Write('krafttermer att välja mellan. Ni har även möjlighet att ändra ');
  GoToXY(10,16);
  Write('filtreringen av in- och utsignal samt sampelfrekvensen. ');
  GoToXY(33,19);
  SetCursorOff;
  Write('TRYCK RETURN !');
  (*FunSound;*)
  REPEAT
  UNTIL ReadKey=chr(13);
  RestoreCursor;
END; (* IntroductionMeny *)

(*************************************************************************)


PROCEDURE InformationMeny;
( Förklarar använda parametrar och ritar blockschema *)
BEGIN
  SetCursorOff;
  ClrScr;
  Ram;
  GoToXY(34,6);
  WriteLn('ANVÄNDAR GUIDE !');
  GoToXY(6,9);
  WriteLn('Användar guiden är en kort information om de saker man bör känna ')
;
  GoToXY(6,10);
  WriteLn('till om programmet innan man startar en simulering.');
  GoToXY(6,12);
  WriteLn('Simulatorn består av datorn som har en 386-processor, en 387-matema
tik-');
  GoToXY(6,13);
  WriteLn('processor samt ett 12-bitars AD/DA-kort för kommunikation med en ra
tten.');
  GoToXY(6,14);
```

```
WriteLn('Datorn är kopplad till ratten via signalkonditioneringskretsar.');
GoToXY(6,17);
WriteLn('          ┌─────────────┐                     ┌─────────────┐');
GoToXY(6,18);
WriteLn('          │             │«────────u,dr────────│             │');
GoToXY(6,19);
WriteLn('          │    DATOR    │                     │ RATT+MOTOR  │');
GoToXY(6,20);
WriteLn('          │             │────────kraft───────»│             │');
GoToXY(6,21);
WriteLn('          └─────────────┘                     └─────────────┘');
GoToXY(34,23);
WriteLn('TRYCK PÅ RETURN !');
REPEAT
UNTIL ReadKey=chr(13);
ClrScr;
Ram;
GoToXY(6,7);
WriteLn('För att kunna välja återkopplins-alternativ och sätta de konstanter
');
GoToXY(6,8);
WriteLn('som används i signalbehandlingen ges nedan befogade förklaringar.')
;
GoToXY(6,11);
WriteLn('Parametrar:                        Intervall:');
GoToXY(6,14);
WriteLn('u      = hastighet framåt           ( 0 - 10,3 )');
GoToXY(6,15);
WriteLn('v      = hastighet sidled           ( -1.5 - 1.5 )');
GoToXY(6,16);
WriteLn('r      = girhastighet               ( -0.086 - 0.086)');
GoToXY(6,17);
WriteLn('dr     = rattutslag                 ( -2.1 - 2.1)');
GoToXY(6,18);
WriteLn('Delta  = 0.25•dr                    ( -0.5 - 0.5 )');
GoToXY(6,19);
WriteLn('A      = (0.25•dr - v/u)²           ( 0.0 - ± 0.5 )');
GoToXY(6,20);
WriteLn('DregDr = systemtrögheten      d(dr)/dt   = orsakar tröghet i ratten'
);
GoToXY(6,21);
WriteLn('DregR  = giraccelerationen    d(r)/dt');
GoToXY(34,23);
WriteLn('TRYCK PÅ RETURN !');
REPEAT
UNTIL ReadKey=chr(13);
ClrScr;
Ram;
GoToXY(30,6);
WriteLn('BLOCKSCHEMA I DETALJ !');
GoToXY(3,8);
WriteLn('┌──────────────────────────────────┐        ┌──────────────────────────
────────┐');
GoToXY(3,9);
WriteLn('│  ┌───────────┐   ┌──────┐  ┌───┐ │        │ ┌───┐       ┌─────────┐
        │');
GoToXY(3,10);
WriteLn('│  │ SIMULATOR ├<┬─┤FILTER├─┤A/D├─┼<─u,dr─┼─┤ S ├──<──┤  RATT   ├<─
─┐      │');
GoToXY(3,11);
WriteLn('│  └────┬──────┘ │ └──────┘  └───┘ │        │ │ I │       └─────────┘
│       │');
GoToXY(3,12);
WriteLn('│       ├──>──┐  └──────>──────┐   │        │ │ G │
│       │');
GoToXY(3,13);
```

```
  WriteLn('|  ┌──┴──┐  |       ┌──┴──┐    |       | | N.|                ┌──
┴──┐   |');
  GoToXY(3,14);
  WriteLn('|  |d(r)/dt| |       |d(dr)/dt|    |       | |   |                | MO
TOR  |   |');
  GoToXY(3,15);
  WriteLn('|  └──┬──┘  |       └──┬──┘    |       | | K |                └──
┬──┘   |');
  GoToXY(3,16);
  WriteLn('|     ├──<──┴──────<──┘          |       | | O |
|      |');
  GoToXY(3,17);
  WriteLn('|┌────┴──────────┐  ┌──────┐  ┌──┐  |       | | N |  ┌──────────────┐
|      |');
  GoToXY(3,18);
  WriteLn('||BERÄKN.AV KRAFT├>┤FILTER├─┤D/A├─┼─kraft>─┼─| D.├─>─┤ FÖRSTÄRKARE ├
>┘      |');
  GoToXY(3,19);
  WriteLn('|└──────────────┘  └──────┘  └──┘  |       | └──┘   └──────────────┘
      |');
  GoToXY(3,20);
  WriteLn('└──────────────────────────────────┘       └──────────────────────
──────┘');
  GoToXY(34,23);
  WriteLn('TRYCK PÅ RETURN !');
  REPEAT
  UNTIL ReadKey=chr(13);
  ClrScr;
  Ram;
  GoToXY(30,6);
  WriteLn('OBS !!  KOM IHÅG ATT :');
  GoToXY(6,8);
  WriteLn('1. Inte vrida ratten med onödigt stark kraft, dvs TA INTE I !');
  GoToXY(6,10);
  WriteLn('2. Ändringar av parametrar bör ske i små steg. Utgå alltid ifrån');
  GoToXY(6,11);
  WriteLn('   default-värdena, som lätt sätts genom val i huvudmenyn.');
  GoToXY(6,13);
  WriteLn('3. Omkalibrering behövs endast om någon mätare överstiger max-');
  GoToXY(6,14);
  WriteLn('   markeringen under simuleringens gång.   ');
  GoToXY(6,16);
  WriteLn('4. Systemträgheten, kallad DregDR, bör alltid finnas med i den ');
  GoToXY(6,17);
  WriteLn('   återkopplade kraften av stabiltets-skäl. ');
  GoToXY(6,19);
  WriteLn('5. Den återkopplade kraften skall ligga mellan 0 och 4000. Värdet')
;
  GoToXY(6,20);
  WriteLn('   2000 motsvarar således kraften noll. ');
  GoToXY(34,23);
  WriteLn('TRYCK PÅ RETURN !');
  REPEAT
  UNTIL ReadKey=chr(13);
  RestoreCursor;
END; (* InformationMeny *)

(*************************************************************************)


PROCEDURE ShowChosenValues;
(* Visar default-värden och återkopplingsalternativ *)
VAR Ch  :CHAR;
BEGIN
  SetCursorOff;
  ClrScr;
```

```
  Regret:=False;
  Ram;
  GoToXY(10,6);
  Write('I SIMULERINGEN KOMMER FÖLJANDE PARAMETERVÄRDEN ATT ANVÄNDAS :');
  GoToXY(10,9);
  Write('Sampelfrekvens (Hz):                    ',SampleFr:1);
  GoToXY(10,11);
  Write('Ingångens lågpassfilter:                ALFA=',LowPassConst:5:2);
  GoToXY(10,13);
  Write('Utgångens lågpassfilter:                ALFA=',LowPassConst2:5:2);
  GoToXY(10,16);
  Write('Återkopplingsalternativ:                ');
  GoToXY(10,18);
  CASE FeedBackChoise OF
    1: Write('Roderkraft+Giracceleration+Girvinkelhastighet+Systemtröghet');
    2: Write('Giracceleration plus systemtröghet');
    3: Write('Girvinkelhastighet plus systemtröghet');
    4: Write('Roderkraft plus systemtröghet');
    5: Write('Proportionell återkoppling med hastighetsberoende plus systemtrö
ghet');
    6: Write('Rattvinkel');
  END;
  GoToXY(18,21);
  Write('Om ändra tryck på R, annars valfri tangent !');
    IF UpCase(ReadKey)='R' THEN
      Regret:=True;
  RestoreCursor;
END; (* ShowChosenValues *)

(*************************************************************************)


PROCEDURE BreakMeny;
(* Förklarar varför programmet avbrutits *)
BEGIN
  IF Break THEN
  BEGIN
    SetCursorOff;
    ClrScr;
    Ram;
    GoToXY(7,11);
    TextBackground(12);
    TextColor(14);
    Write('╔═══════════════════════════════════════════════════════════════
');
    GoToXY(7,12);
    Write('╟ SIMULERINGEN AVBRÖTS P G A FÖR STORA VARIATIONER I RATTKRAFTEN !╟
');
    GoToXY(7,13);
    Write('╟ KONTROLLERA OM LYSDIODEN PÅ FÖRSTÄRKARKORTET LYSER ! LYSER DEN: ╟
');
    GoToXY(7,14);
    Write('╟ SLÅ AV SPÄNNINGEN TILLS DEN SLOCKNAT, SEN SLÅ PÅ DEN IGEN !      ╟
');
    GoToXY(7,15);
    Write('╚═══════════════════════════════════════════════════════════════
');
    FailSound(0.95);
    FailSound(0.7);
    DELAY(7000);
    Break:=FALSE;
    RestoreCursor;
    NormVideo;
    HighVideo;
  END;
END; (* BreakMeny *)
```

```
(*************************************************************************)

PROCEDURE ReadInputReal (Min,Max :INTEGER; Scale :REAL; Var Input :REAL);
(* Läser inmatat värde och ger felmeddelande om ogiltigt värde matats in *)
VAR
  ErrorPos,CursorX,CursorY    :INTEGER;
  InputNumber                 :STRING;
BEGIN
  CursorX:=WhereX;
  CursorY:=WhereY;
  ReadLn(InputNumber);
  IF LENGTH(InputNumber) <> 0 THEN
  BEGIN
    Val(InputNumber,Input,ErrorPos);
    WHILE (Input < Min) OR (Input > Max) OR (ErrorPos <> 0) DO
    BEGIN
      SetCursorOff;
      TextBackground(12);
      TextColor(14);
      GoToXY(CursorX,CursorY);
      Write('FELAKTIGT VÄRDE !');
      DELAY(2000);
      GoToXY(CursorX,CursorY);
      TextBackground(0);
      TextColor(7);
      HighVideo;
      Write('                  ');
      GoToXY(CursorX,CursorY);
      RestoreCursor;
      ReadLn(InputNumber);
      Val(InputNumber,Input,ErrorPos);
    END;
    Input:=Input*Scale;
  END;
END; (* ReadInputReal *)

(*************************************************************************)

PROCEDURE ReadInputInteger (Min,Max :INTEGER; Var Input :INTEGER);
(* Läser inmatat värde och ger felmeddelande om ogiltigt värde matats in *)
VAR
  ErrorPos,CursorX,CursorY    :INTEGER;
  InputNumber                 :STRING;
BEGIN
  CursorX:=WhereX;
  CursorY:=WhereY;
  ReadLn(InputNumber);
  IF LENGTH(InputNumber) <> 0 THEN
  BEGIN
    Val(InputNumber,Input,ErrorPos);
    WHILE (Input < Min) OR (Input > Max) DO
    BEGIN
      SetCursorOff;
      TextBackground(12);
      TextColor(14);
      GoToXY(CursorX,CursorY);
      Write('FELAKTIGT VÄRDE !');
      DELAY(2000);
      GoToXY(CursorX,CursorY);
      TextBackground(0);
      TextColor(7);
      HighVideo;
      Write('                  ');
```

```
      GoToXY(CursorX,CursorY);
      RestoreCursor;
      ReadLn(InputNumber);
      Val(InputNumber,Input,ErrorPos);
    END;
  END;
END; (* ReadInputInteger *)

(*************************************************************************)


PROCEDURE New_FeedBack_Const;
(* Ändrar värdet på konstanterna som beräknar återkopplings-  *)
(* kraften till Da-omvandlaren                                *)
BEGIN
  ClrScr;
  Ram;
  GoToXY(25,6);
  Write('VÄLJ ÅTERKOPPLINGS-KONSTANTER!');
  GoToXY(7,8);
  CASE FeedBackChoise OF
    1: BEGIN
         Write('Återkopplad signal= u*A*(K1*Sign+K2*A*Delta)-DregR+K3*r+DregDR
+2000');
         GoToXY(10,11);
         Write('K1   (0.0 - 10.0):     ',K1/100:8:0,'    ');
         ReadInputReal(0,10,100,K1);
         GoToXY(10,13);
         Write('K2   (0.0 - 10.0):     ',K2/6000:8:0,'    ');
         ReadInputReal(0,10,6000,K2);
         GoToXY(10,15);
         Write('K3   (0.0 - 10.0):     ',K3/4000:8:0,'    ');
         ReadInputReal(0,10,4000,K3);
       END;
    2: BEGIN
         SetCursorOff;
         GoToXY(10,10);
         Write('     Parametrarna ändras endast i huvudmenyns val nr 6 !');
         GoToXY(34,16);
         Write('TRYCK RETURN !');
         REPEAT
         UNTIL ReadKey=chr(13);
         RestoreCursor;
       END;
    3: BEGIN
         Write('Återkopplad signal= DregDR+K4*r+2000');
         GoToXY(10,11);
         Write('K4   (0.0 - 10.0):     ',K4/4000:8:0,'    ');
         ReadInputReal(0,10,4000,K4);
       END;
    4: BEGIN
         Write('Återkopplad signal= DregDR+u*A*(K5*Sign+K6*Delta*A)+2000');
         GoToXY(10,11);
         Write('K5   (0.0 - 10.0):     ',K5/100:8:0,'    ');
         ReadInputReal(0,10,100,K5);
         GoToXY(10,13);
         Write('K6   (0.0 - 10.0):     ',K6/6000:8:0,'    ');
         ReadInputReal(0,10,6000,K6);
       END;
    5: BEGIN
         Write('Återkopplad signal= Sign*(K7*Ln(K8*abs(dr)+1)*Ln(K9*u+1))+Dreg
DR+2000');
         GoToXY(10,11);
         Write('K7   (0.0 - 10.0):     ',K7/200:8:0,'    ');
         ReadInputReal(0,10,200,K7);
         GoToXY(10,13);
```

```
          Write('K8   (0.0 - 10.0):      ',K8/0.6:8:1,'     ');
          ReadInputReal(0,10,0.6,K8);
          GoToXY(10,15);
          Write('K9   (0.0 - 10.0):      ',K9/0.02:8:1,'     ');
          ReadInputReal(0,10,0.02,K9);
          GoToXY(10,18);
          Write('Systemtrögheten ändras i huvudmenyns val nr 6 !');
        END;
ä   6: BEGIN
          Write('Återkopplad signal= ...HÄR SKRIVER NI IN EGET UTTRYCK OM NI VI
LL...');
          GoToXY(10,11);
          Write('K10  (0.0 - 10.0):      ',K10/SKALFAKTOR SKRIVS HÄR!:8:1,'     '
);
          ReadInputReal(0,10,SKALFAKTOR SKRIVS HÄR!,K10);
          GoToXY(10,13);
          Write('K11  (0.0 - 10.0):      ',K11/SKALFAKTOR SKRIVS HÄR!:8:1,'     '
);
          ReadInputReal(0,10,SKALFAKTOR SKRIVS HÄR!,K11);
          GoToXY(10,15);
          Write('K12  (0.0 - 10.0):      ',K12/SKALFAKTOR SKRIVS HÄR!:8:1,'     '
),
          ReadInputReal(0,10,SKALFAKTOR SKRIVS HÄR!,K12);
          GoToXY(10,15);
          Write('K13  (0.0 - 10.0):      ',K13/SKALFAKTOR SKRIVS HÄR!:8:1,'     '
)
          ReadInputReal(0,10,SKALFAKTOR SKRIVS HÄR!,K13);
          GoToXY(10,15);
          Write('K14  (0.0 - 10.0):      ',K14/SKALFAKTOR SKRIVS HÄR!:8:1,'     '
);
          ReadInputReal(0,10,SKALFAKTOR SKRIVS HÄR!,K14);
        END; å
  END;
END; (* New_FeedBack_Const *)

(*************************************************************************)


PROCEDURE New_FeedBack_Alt;
(* Väljer den sorts återkoppling man vill ha *)
VAR
  CursorX,CursorY,ErrorPos    :INTEGER;
  InputNumber                 :STRING;
BEGIN
  ClrScr;
  Ram;
  GoToXY(15,6);
  WriteLn('VÄLJ ÅTERKOPPLINGSALTERNATIV! FÖREGÅENDE VAL:  ',FeedBackChoise);
  GoToXY(10,9);
  WriteLn('1...Roderkraft+Giracceleration+Girvinkelhastighet+Systemtröghet');
  GoToXY(10,11);
  WriteLn('2...Giracceleration plus systemtröghet');
  GoToXY(10,13);
  WriteLn('3...Girvinkelhastighet plus systemtröghet');
  GoToXY(10,15);
  WriteLn('4...Roderkraft plus systemtröghet');
  GoToXY(10,17);
  WriteLn('5...Proportionell återkoppling med hastighetsberoende');
  GoToXY(10,18);
  WriteLn('    plus systemtröghet');

(****** OM NI SKAPAR ERT EGET UTTRCK FÖR DEN ÅTERKOPPLADE KRAFTEN TAR NI *****
)
(****** BORT KOMMENTARPARENTESERNA HÄREFTER OCH ÄNDRAR ÖVRE FELGRÄNS     *****
)
(****** TILL 6 ISTÄLLET FÖR 5 I WHILE SATSEN NEDAN.                     *****
```

```
)

  GoToXY(10,20);
  WriteLn('6...Rattvinkel');
  GoToXY(10,22);
  Write('Val:  ');
  CursorX:=WhereX;
  CursorY:=WhereY;
  ReadLn(InputNumber);
  IF LENGTH(InputNumber) <> 0 THEN
  BEGIN
    Val(InputNumber,FeedBackChoise,ErrorPos);
    WHILE (FeedBackChoise < 1) OR (FeedBackChoise > 6) DO  (* HÄR ÄNDRAR NI EV
.*)
    BEGIN                                                  (* FELGRÄNS TILL 6
 *)
      SetCursorOff;
      TextBackground(12);
      TextColor(14);
      GoToXY(CursorX,CursorY);
      Write('FELAKTIGT VÄRDE !');
      DELAY(2000);
      GoToXY(CursorX,CursorY);
      TextBackground(0);
      TextColor(7);
      HighVideo;
      Write('                        ');
      GoToXY(CursorX,CursorY);
      RestoreCursor;
      ReadLn(InputNumber);
      Val(InputNumber,FeedBackChoise,ErrorPos);
    END;
  END;
END; (* New_FeedBack_Alt *)

(*************************************************************************)


PROCEDURE New_Filter_And_ControlConst;
(* Ändrar ev. värden på lågpasskonstanten Alfa samt regulatorkonstanterna *)
(* Td,N,K och sampelintervallet.                                          *)
BEGIN
   lrScr;
  Ram;
  GoToXY(19,6);
  Write('VÄLJ FILTER-KONSTANTER SAMT SAMPELFREKVENS!');
  GoToXY(10,9);
   rite('Lågpass-konstant α (insignal)  (0.00 - 1.00)     ',LowPassConst:8:2,'
    ');
  ReadInputReal(0,1,1,LowPassConst);
  GoToXY(10,11);
  Write('Lågpass-konstant α (utsignal)  (0.00 - 1.00)     ',LowPassConst2:8:2,
'    ');
  ReadInputReal(0,1,1,LowPassConst2);
  GoToXY(10,13);
  Write('Sampelfrekvens i Hz (25,50,75,100)               ',SampleFr:8,'    ')
;
  ReadInputInteger(25,100,SampleFr);
  WHILE (SampleFr<>25) AND (SampleFr<>50) AND (SampleFr<>75) AND
        (SampleFr<>100) DO
  BEGIN
    GoToXY(62,13);
    Write('               ');
    GoToXY(62,13);
    ReadInputInteger(25,100,SampleFr);
  END;
```

```
   Beta:=1+NConst/SampleFr/TdConst;
   ClrScr;
   Ram;
   GoToXY(16,6);
   Write('VÄLJ REGLER-KONSTANTER TILL SYSTEMTRÖGHETEN !');
   GoToXY(10,9);
   Write('Derivata konstant Td              (0.0 - 50.0)    ',TdConst:8:1,'     '
);
   ReadInputReal(0,50,1,TdConst);
   Beta:=1+NConst/SampleFr/TdConst;
   GoToXY(10,11);
   Write('Heltals-konstant N                (1.0 - 50.0)    ',NConst:8,'     ');
   ReadInputInteger(1,50,NConst);
   Beta:=1+NConst/SampleFr/TdConst;
   GoToXY(10,13);
   Write('Derivata-förstärkning             (0.0 - 10.0)    ',DConst/11000:8:1,'
     ');
   ReadInputReal(0,10,11000,DConst);
   GoToXY(16,16);
   Write('VÄLJ REGLER-KONSTANTER TILL GIRACCELERATION !');
   GoToXY(10,19);
   Write('Derivata-konstant Td              (0.0 - 50.0)    ',TdConstR/0.6:8:1,'
     ');
   ReadInputReal(0,50,0.6,TdConstR);
   Beta:=1+NConstR/SampleFr/TdConstR;
   GoToXY(10,21);
   Write('Heltals-konstant N                (1.0 - 50.0)    ',NConstR:8,'     ');
   ReadInputInteger(1,50,NConstR);
   Beta:=1+NConstR/SampleFr/TdConstR;
   GoToXY(10,23);
   Write('Derivata-förstärkning             (0.0 - 10.0)    ',DConstR/8E5:8:1,'
     ');
   ReadInputReal(0,10,8E5,DConstR);
END; (* New_Filter_And_ControlConst *)

(*************************************************************************)


PROCEDURE Write_A_and_B_Matrix;
(* Skapar en fil för A och B-matriser som används i beräkningarna *)
VAR
  Ch        :Char;
  I,J       :INTEGER;
  FilNamn   :STRINGÄ3Å;
BEGIN
  ClrScr;
  Writeln;
  Write('Ange sampelfrekvensen i Hz ');Readln(SampleFr);WriteLn;
  Ts:=1/SampleFr;
  Str(SampleFr:2,FilNamn);
  ASSIGN(fi,FilNamn+'.DAT');
  WriteLn('Matriserna är på formen:');
  WriteLn('  a b       e');
  WriteLn('A=        B=' );
  WriteLn('  c d       f');WriteLn;
  FOR I:=0 TO 20 DO
    FOR J:=1 TO 6 DO
      A_BÄJ,IÅ:=0;
  REPEAT
    Write('u=(knop) ');ReadLn(i);
    FOR J:= 1 TO 6 DO
    BEGIN
      Write(MatrIndÄJÅ,'=');ReadLn(A_BÄJ,IÅ);
    END;
    Write('Fler värden J/N ');Readln(Ch);
  UNTIL UpCase(Ch)='N';
```

```
  ReWrite(fi);              (* Skapa filen filnamn.DAT och skriver in   *)
  Write(fi,Ts);             (* Ts och Matriskoefficienterna för u=1..20 *)
  FOR I:=0 TO 20 DO
    FOR J:=1 TO 6 DO
      Write(fi,A_BÄJ,IÅ);
  Close(fi);
  WriteLn;
END; (* Write_A_And_B_Matrix *)

(*************************************************************************)


PROCEDURE NewMatrix;
(* När u ändrats mer än en knop läses nya A- och B matriser in *)
BEGIN
  a:=A_BÄ1,uKnotÅ;b:=A_BÄ2,uKnotÅ;c:=A_BÄ3,uKnotÅ;d:=A_BÄ4,uKnotÅ;
  e:=A_BÄ5,uKnotÅ;f:=A_BÄ6,uKnotÅ;
END; (* NewMatrix *)

(*************************************************************************)


PROCEDURE Read_A_And_B_Matrix;
(* Läser in en tidigare skapad fil för den iterativa formeln     *)
(* v,r(t+1)= A*v,r(t)+b*delta(t). A och B matriserna finns i den *)
V
  ,J      :INTEGER;
  FilNamn :STRINGÄ3Å;
BEGIN
  Str(SampleFr:2,FilNamn);
  ASSIGN (fi,FilNamn+'.DAT');
  RESET (fi);
  Read(fi,Ts);
  FOR I:=0 TO 20 DO
    FOR J:= 1 TO 6 DO
      Read(fi,A_BÄJ,IÅ);
  CLOSE (fi);
END; (* Read_A_And_B_Matrix *)

(*************************************************************************)


F  CTION AD (Channel:INTEGER):INTEGER;
(* Gör analog till digital-omvandling på kanal 'Channel' *)
CONST Baseadr=768;
VAR Highb,Lowb,A,k,i: INTEGER;
BEGIN
  ORTÄBaseadr+12Å:=Channel;              (* Adressering AD-channel (0-15) *)
  For k:= 1 To 15 Do
    i := 0;
  A:=PORTÄBaseadr+2Å;                    (* Start conversion *)
  REPEAT
  UNTIL (PORTÄBASEADR+0Å AND $80)=$80;
  Highb:=PORTÄBaseadr+1Å;                (* Read highbyte *)
  Lowb:=PORTÄBaseadr+2Å;                 (* Read lowbyte *)
  AD:=(Highb*256)+Lowb;
END;  (* AD *)

(*************************************************************************)


PROCEDURE DA1 (Value:INTEGER);
(* Gör digital till analog-omvandling på kanal A *)
CONST Baseadr=768;
VAR Highb,Lowb:INTEGER;
BEGIN
```

```
   Highb:=TRUNC(Value/256);
   Lowb:=Value-Highb*256;
   PORTÄBaseadr+4Å:=Lowb;
   PORTÄBaseadr+5Å:=Highb;
   PORTÄBaseadr+3Å:=0;
END;  (* DA1 *)

(*************************************************************************)


PROCEDURE DA2 (VALUE:INTEGER);
(* Gör digital till analog-omvandling på kanal B *)
CONST BASEADR=768;
VAR HIGHB,LOWB:INTEGER;
BEGIN
   HIGHB:=TRUNC(VALUE/256);
   LOWB:=VALUE-HIGHB*256;
   PORTÄBASEADR+6Å:=LOWB;
   PORTÄBASEADR+7Å:=HIGHB;
   PORTÄBASEADR+3Å:=0;
E' ;  (* DA2 *)

(*************************************************************************)


P' CEDURE InitiateDefault;
(  Initierar defaultvärden på alla återkopplings- och filterparametrar *)
(* samt sampelfrekvens *)
BEGIN
  Quit:=FALSE;
  Calibrated:= FALSE;
  Break:=FALSE;
  LowPassConst:=0.93;
  LowPassConst2:=0.2;
  Ordn2LowPassConst:=0.8;
  DConst:=44E3;
  TdConst:=5;
  NConst:=1;
  DConstR:=3.2E6;
  TdConstR:=3;
  NConstR:=1;
  SampleFr:=100;
   eedBackChoise:=2;
  K1:=500; K2:=30000; K3:=0; K4:=20000; K5:=500; K6:=30000; K7:=1000;
  K8:=3; K9:=0.1; K10:=0; K11:=0; K12:=0; K13:=0; K14:=0;
  Beta:=1+NConst/SampleFr/TdConst;
  BetaR:=1+NConstR/SampleFr/TdConstR;
   ldHelpVar:=0;
  OldValue:=0;
  OldHelpVarR:=0;
  OldValueR:=0;
  S:=0; dev:=0;
  DA1(2000);
  DA2(4000);
END; (* InitiateDefault *)

(*************************************************************************)


PROCEDURE CheckPowerOn;
(* Kontrollerar om spänningen är påslagen *)
BEGIN
  WHILE ((AD(0) < 2050) AND (AD(0) > 1950)) AND
        ((AD(1) < 2050) AND (AD(1) > 1950)) DO
  BEGIN
    SetCursorOff;
```

```
    ClrScr;
    Ram;
    TextBackground(12);
    TextColor(14);
    GoToXY(15,12);
    Write('╔═══════════════════════════════════════════════════╗');
    GoToXY(15,13);
    Write('║ SLÅ PÅ SPÄNNINGEN TILL SIMULATORUTRUSTNINGEN ! ║');
    GoToXY(15,14);
    Write('╚═══════════════════════════════════════════════════╝');
    TextBackground(0);
    TextColor(7);
    HighVideo;
    CrashSound;
    GoToXY(24,20);
    WriteLn('Tryck på Return när du är klar !');
    REPEAT
    UNTIL ReadKey=chr(13);
    RestoreCursor;
  END;
E  ; (* CheckPowerOn *)

(*************************************************************************)


PROCEDURE Calibrate;
(* Kalibrerar ratt och gasreglage *)
CONST
  RadAndSSPAConst  = 0.06981317;   (* 4*Pi/180 *)
  LeftRudderAngle  =-30*RadAndSSPAConst;
  RightRudderAngle = 30*RadAndSSPAConst;
  TotalAngle       = RightRudderAngle-LeftRudderAngle;
  MaxSpeed         = 10.26666667;   (* 20 knop i m/s *)
VAR
  LoValue,HiValue : INTEGER;
BEGIN
  CheckPowerOn;
  LoValue:=4050;
  HiValue:=50;
  drOffset:=LeftRudderAngle-LoValue/(HiValue-LoValue)*TotalAngle;
  drMultFact:=TotalAngle/(HiValue-LoValue);
  HiValue:=4050;
  MultFact:=MaxSpeed/HiValue;
  Calibrated:=TRUE;
  RestoreCursor;
END; (* Calibrate *)

(*************************************************************************)


PROCEDURE Calibrate_Or_Default;
(* Väljer om vi skall kalibrera eller sätta tillbaks defaultvärdena *)
VAR
  CH      :CHAR;
BEGIN
  ClrScr;
  Ram;
  GoToXY(10,12);
  WriteLn('Om ni vill kalibrera eller få tillbaka defaultvärdena tryck  ');
  GoToXY(10,14);
  Write('på K respektive D, annars tryck Return !');
  GoToXY(51,14);
  REPEAT
    CH:=UpCase(ReadKey);
  UNTIL (CH=chr(13)) OR (CH='K') OR (CH='D');
  IF CH='K' THEN
```

```
    Calibrate
  ELSE IF CH='D' THEN
    InitiateDefault;
END; (* Calibrate_Or_Default *)

(*************************************************************************)


FUNCTION LowPass  (Value:INTEGER):REAL;
(* Implementerar ett första ordningens lågpassfilter *)
VAR P  :REAL;
BEGIN
  P:=LowPassConst*OldFiltValue+(1-LowPassConst)*Value;
  OldFiltValue:=P;
  LowPass:=P;
END; (* LowPass *)

(*************************************************************************)

PROCEDURE SteeringWeelZero;
CONST
  KnotConv = 1.94805195; (* m/s till knop *)
Var  Ra,S,I : Integer;
     D : Real;
     Ready : Boolean;
BEGIN
  CheckPowerOn;
  ClrScr;
  Ram;
  SetCursorOff;
  TextColor(14);
    GoToXY(15,12);
    Write('╔═══════════════════════════════════════════════════╗');
    GoToXY(15,13);
    Write('║                                                   ║');
    TextColor(red);
    GotoXY(18,13);
    Write('RÖR EJ RATTEN TILLS SIMULERINGEN STARTAT !!!');
    TextColor(14);
    GoToXY(15,14);
    Write('╚═══════════════════════════════════════════════════╝');
  R:=0;
  S:=0;
  A:=0;
  Delta:=0;
  Break:=FALSE;
  OldFeedBackValue:=2000;
  OldFiltValue:=2048;
  OldFiltValue2:=2000;
  OldY1Filt:=2000;
  OldY2Filt:=2000;
  OldHelpVar:=0;
  OldHelpVarR:=0;
  DA1(2000);
  Ra := AD(drChannel)-2040;
  Ready := False;
  D := 280;
  S := 0;
  While NOT Ready Do
  BEGIN
    If Ra >= 1 Then
      DA1(ROUND(1980-D))
    ELSE If Ra <= -1 Then
      DA1(ROUND(2010+D))
    ELSE
    BEGIN
```

```
      D := D - 5;
      If D = 0 Then
        Ready := True;
    END;
    Ra := AD(drChannel)-2020;
    S := S + 1;
    If S = 20000 Then
      Ready := True;
  END;
  FOR I:=1 TO 50 DO                    (* För att erhålla stabilt dr-värde *)
    dr:=drOffset+LowPass(AD(drChannel))*drMultFact;
  u:=AD(uChannel)*uMultFact;
  uKnot:=ROUND(KnotConv*u);
  NewMatrix; v0:=v;
  v:=v0*a+r*b+dr*e;
  r:=v0*c+r*d+dr*f;
  OldValue:=dr;                  (* För att derivata-delar skall bli noll *)
  OldValueR:=r;                  (* vid simuleringens början *)
  RestoreCursor;
END;  (* SteeringWeelZero *)


(*************************************************************************)


PROCEDURE InstallGraficMode;
(* Introducerar grafisk mod *)
CONST
  PiHalf = 1.5707963;                (* pi/2 *)
BEGIN
  IF NOT Calibrated THEN
    Calibrate;
  SteeringWeelZero;
  x:=375.0; y:=1075;
  u:=0; uKnot:=0;
  OlduKnot:=0; UPl:=129;
  psi:=PiHalf; dr:=0.0;
  XKoord:=80; YKoord:=75;
  RPl:=145;                  (* Ger ett OldRPl som inte suddar vid *)
  DRPl:=145;                  (* Ger ett OldDRPl som inte suddar vid *)
  Read_A_And_B_Matrix;       (* första anrop av PlotDR              *)
  NewMatrix;
  GraphDriver := Detect;     (* Automatisk detektering av grafikhårdvara *)
  InitGraph(GraphDriver,GraphMode,'');
  IF GraphResult <> grOk THEN
    Halt(1);
  SetWriteMode(CopyPut);                    (* Grafikinitiering      *)
  SetFillStyle(SolidFill,LightGray);        (* Färglägger bakgrunden *)
  FloodFill(11,11,LightGray);
  SetColor(Black);
  SetTextJustify(CenterText,TopText);       (* Rita ut text till sjökortet *)
  OutTextXY(170,465,'SJÖKORT');
  Line(250,470,280,470);                    (* x-axel skala *)
  Line(255,468,255,472);                    (* Skalstreck   *)
  Line(275,468,275,472);
  OutTextXY(300,465,'50 m');
  SetColor(White);

  WITH Map DO  (* 300*440 *)
  BEGIN
    Rectangle(Succ(x1),Succ(y1),             (* Märk ut viewport Map *)
                  Pred(x2),Pred(y2));
    SetViewPort(x1, y1, x2, y2, ClipOn);
    SetFillStyle(SolidFill,blue);
    FloodFill(10,10,Blue);
  END;
```

```
SetViewPort(0, 0, GetMaxX, GetMaxY, ClipOn);   (* viewport hela skärmen *)

WITH Compass DO  (* 160*160 *)
BEGIN
  Rectangle(Succ(x1),Succ(y1),              (* Märk ut viewport Compass *)
            Pred(x2),Pred(y2));
  SetViewPort(x1, y1, x2, y2, ClipOn);
  SetFillStyle(SolidFill,Black);
  FloodFill(10,10,GetMaxColor);
  OutTextXY(80,150,'KURS');
  Circle(80,75,50);
  Line(80,20,80,30);        (* Markeringslinje N     *)
  OutTextXY(80,5,'N');
  Line(95,33,99,23);        (* Markeringslinje 20    *)
  OutTextXY(100,13,'20');
  Line(109,41,115,33);      (* Markeringslinje 40    *)
  OutTextXY(122,25,'40');
  Line(119,53,128,48);      (* Markeringslinje 60    *)
  OutTextXY(138,41,'60');
  Line(124,67,134,65);      (* Markeringslinje 80    *)
  OutTextXY(145,59,'80');
  Line(125,75,135,75);      (* Markeringslinje 90 Ö  *)
  OutTextXY(147,71,'Ö');
  Line(124,83,134,85);      (* Markeringslinje 100   *)
  OutTextXY(145,83,'100');
  Line(119,98,128,103);     (* Markeringslinje 120   *)
  OutTextXY(142,100,'120');
  Line(109,109,115,117);    (* Markeringslinje 140   *)
  OutTextXY(131,114,'140');
  Line(95,117,99,127);      (* Markeringslinje 160   *)
  OutTextXY(112,128,'160');
  Line(80,120,80,130);      (* Markeringslinje 180 S *)
  OutTextXY(80,135,'S');
  Line(65,117,61,127);      (* Markeringslinje 200   *)
  OutTextXY(49,130,'200');
  Line(51,109,45,117);      (* Markeringslinje 220   *)
  OutTextXY(28,116,'220');
  Line(41,98,32,102);       (* Markeringslinje 240   *)
  OutTextXY(18,100,'240');
  Line(36,83,26,85);        (* Markeringslinje 260   *)
  OutTextXY(13,83,'260');
  Line(25,75,35,75);        (* Markeringslinje 270 V *)
  OutTextXY(13,71,'V');
  Line(36,67,26,65);        (* Markeringslinje 280   *)
  OutTextXY(13,60,'280');
  Line(41,53,32,48);        (* Markeringslinje 300   *)
  OutTextXY(16,41,'300');
  Line(51,41,45,33);        (* Markeringslinje 320   *)
  OutTextXY(30,25,'320');
  Line(65,33,61,23);        (* Markeringslinje 340   *)
  OutTextXY(55,13,'340');
END;

SetViewPort(0, 0, GetMaxX, GetMaxY, ClipOn);    (* viewport hela skärmen *)

WITH SpeedometerR DO  (* 290*80 *)
BEGIN
  Rectangle(Succ(x1),Succ(y1),            (* Märk ut viewport Speedometer *)
            Pred(x2),Pred(y2));
  SetViewPort(x1, y1, x2, y2, ClipOn);
  SetFillStyle(SolidFill,Black);
  FloodFill(10,10,GetMaxColor);
  SetTextJustify(CenterText,TopText);
  OutTextXY(145,70,'GIRVINKELHASTIGHET (grad/s)');
  Line(15,46,275,46);   (* x-linje *)
```

```
  FOR LoopCount:=-8 TO 8 DO
    Line(145+LoopCount*15,47,145+LoopCount*15,50);
  SetTextJustify(CenterText,TopText);
  OutTextXY(25,55,'8');              (* Rita ut skalmarkeringen på x-axel *)
  OutTextXY(55,55,'6');
  OutTextXY(85,55,'4');
  OutTextXY(115,55,'2');
  OutTextXY(145,55,'0');
  OutTextXY(175,55,'2');
  OutTextXY(205,55,'4');
  OutTextXY(235,55,'6');
  OutTextXY(265,55,'8');
END;

SetViewPort(0, 0, GetMaxX, GetMaxY, ClipOn);    (* viewport hela skärmen *)


WITH AnglometerDR DO  (* 290*80 *)
BEGIN
  Rectangle(Succ(x1),Succ(y1),           (* Märk ut viewport Anglometer *)
            Pred(x2),Pred(y2));
  SetViewPort(x1, y1, x2, y2, ClipOn);
  SetFillStyle(SolidFill,Black);
  FloodFill(10,10,GetMaxColor);
  SetTextJustify(CenterText,TopText);
  OutTextXY(145,70,'RODERVINKEL (grader)');
  Line(15,46,275,46);   (* x-linje *)
  FOR LoopCount:=-6 TO 6 DO
    Line(145+LoopCount*20,47,145+LoopCount*20,50);
  SetTextJustify(CenterText,TopText);
  OutTextXY(25,55,'30');             (* Rita ut skalmarkeringen på x-axel *)
  OutTextXY(45,55,'25');
  OutTextXY(65,55,'20');
  OutTextXY(85,55,'15');
  OutTextXY(105,55,'10');
  OutTextXY(125,55,'5');
  OutTextXY(145,55,'0');
  OutTextXY(165,55,'5');
  OutTextXY(185,55,'10');
  OutTextXY(205,55,'15');
  OutTextXY(225,55,'20');
  OutTextXY(245,55,'25');
  OutTextXY(265,55,'30');
END;


SetViewPort(0, 0, GetMaxX, GetMaxY, ClipOn);    (* viewport hela skärmen *)


WITH SpeedometerU DO  (* 290*80 *)
BEGIN
  Rectangle(Succ(x1),Succ(y1),           (* Märk ut viewport Speedometer *)
            Pred(x2),Pred(y2));
  SetViewPort(x1, y1, x2, y2, ClipOn);
  SetFillStyle(SolidFill,Black);
  FloodFill(10,10,GetMaxColor);
  SetTextJustify(CenterText,TopText);
  OutTextXY(41,140,'HASTIGHET');
  OutTextXY(41,150,'(knop)');
  Line(12,130,60,130);   (* x-linje *)
  Line(62,7,62,130);  (* y-linje *)
  FOR LoopCount:= 0 TO 10 DO
    Line(61,10+LoopCount*12,63,10+LoopCount*12);
  SetTextJustify(CenterText,TopText);
  OutTextXY(70,7,'20');              (* Rita ut skalmarkeringen på y-axel *)
  OutTextXY(70,19,'18');
```

```
    OutTextXY(70,31,'16');
    OutTextXY(70,43,'14');
    OutTextXY(70,55,'12');
    OutTextXY(70,67,'10');
    OutTextXY(70,79,'8');
    OutTextXY(70,91,'6');
    OutTextXY(70,103,'4');
    OutTextXY(70,115,'2');
    OutTextXY(70,127,'0');
  END;

  SetViewPort(0, 0, GetMaxX, GetMaxY, ClipOn);    (* viewport hela skärmen *)

  WITH StopDisplay DO   (* 90*60 *)
  BEGIN
    Rectangle(Succ(x1),Succ(y1),           (* Märk ut viewport StopDisplay *)
              Pred(x2),Pred(y2));
    SetViewPort(x1, y1, x2, y2, ClipOn);
    SetFillStyle(SolidFill,Red);
    FloodFill(10,10,GetMaxColor);
    SetTextJustify(CenterText,Toptext);
    SetTextStyle(Triplexfont,horizDir,4);
    OutTextXY(45,4,'STOP');
    SetTextStyle(DefaultFont,horizDir,1);
    OutTextXY(44,50,'PRESS KEY');
  END;
  LoopCount:=0;
END;  (* InstallGraficMode *)

(*************************************************************************)


PROCEDURE InstallDelay;
(* Installerar avbrott med frekvensen SampleFr mha timer 0 *)
VAR
  CountTo : INTEGER;
BEGIN
  OldMask:=PORTÄIntrMaskÅ;  (* Sparar gammal interruptmask *)
  PORTÄIntrMaskÅ:=$FC;      (* Maskar ut timer0-& tangentbordsinterrupt *)
  CountTo:=ROUND(ClFr/SampleFr);
  PORTÄTimerComÅ:=$36;           (* Öppnar kommandoregistret på timern *)
  PORTÄKanal0Å:=LO( CountTo );   (* Skriver in ny avbrottsfrekvens *)
  PORTÄKanal0Å:=HI( CountTo );
END;   (* InstallDelay *)

(*************************************************************************)


PROCEDURE Back;
(* Återställer interrupt- och skärmtillstånd till normalt läge och *)
(* lägger ut kraften=0 på ratten                                    *)
BEGIN
  PORTÄIntrMaskÅ:=OldMask;
  PORTÄTimerComÅ:=$36;
  PORTÄKanal0Å:=255;
  PORTÄKanal0Å:=255;
  CloseGraph;
  DA1(2000);
END;   (* Back *)

(*************************************************************************)


PROCEDURE PlotR;
(* Plottar girvinkelhastigheten på skärmen *)
CONST
```

```
  RadToGrad =57.29578;
BEGIN
  WITH SpeedometerR DO
  BEGIN
    OldRPl:=RPl;       (* OldRPl används för radering av gammalt r *)
    RPl:=145+ROUND(r*RadToGrad*15);
    SetViewPort(x1, y1, x2, y2, ClipOn);
    SetColor(Black);
    SetFillStyle(SolidFill,Black);
    IF (OldRPl<145) AND (RPl>OldRPl) THEN
      Bar(OldRPl,20,RPl,44)
    ELSE IF (OldRPl>145) AND (OldRPl>RPl) THEN
      Bar(RPl,20,OldRPl,44);
    SetFillStyle(SolidFill,LightMagenta);
    IF (RPl<145) AND (OldRPl>RPl) THEN
      IF OldRPl>145 THEN
        Bar(RPl,20,145,44)
      ELSE
        Bar(RPl,20,OldRPl,44)
    ELSE IF (RPl>145) AND (RPl>OldRPl) THEN
      IF OldRPl<145 THEN
        Bar(145,20,RPl,44)
      ELSE
        Bar(OldRPl,20,RPl,44);
  END; (* SpeedometerR *)
E[illegible]; (* PlotR *)

(*************************************************************************)


PROCEDURE PlotDR;
(* Plottar rattvinkeln på skärmen *)
BEGIN
  WITH AnglometerDR DO
  BEGIN
    OldDRPl:=DRPl;       (* OldRPl används för radering av gammalt r *)
    DRPl:=145-ROUND(dr*56.872);
    SetViewPort(x1, y1, x2, y2, ClipOn);
    SetColor(Black);
    SetFillStyle(SolidFill,Black);
    IF (OldDRPl<145) AND (DRPl>OldDRPl) THEN
      Bar(OldDRPl,20,DRPl,44)
    ELSE IF (OldDRPl>145) AND (OldDRPl>DRPl) THEN
      Bar(DRPl,20,OldDRPl,44);
    SetFillStyle(SolidFill,Yellow);
    IF (DRPl<145) AND (OldDRPl>DRPl) THEN
      IF OldDRPl>145 THEN
        Bar(DRPl,20,145,44)
      ELSE
        Bar(DRPl,20,OldDRPl,44)
    ELSE IF (DRPl>145) AND (DRPl>OldDRPl) THEN
      IF OldDRPl<145 THEN
        Bar(145,20,DRPl,44)
      ELSE
        Bar(OldDRPl,20,DRPl,44);
  END; (* SpeedometerDR *)
END; (* PlotDR *)

(*************************************************************************)


PROCEDURE PlotUKnot;
(* Plottar hastigheten på skärmen *)
BEGIN
  WITH SpeedometerU DO
  BEGIN
```

```
    OldUPl:=UPl;      (* OldUPl används för radering av gammalt UPl *)
    UPl:=129 - UKnot*6;
    SetViewPort(x1, y1, x2, y2, ClipOn);
    SetColor(Black);
    IF UPl>OldUPl THEN
    BEGIN
      SetFillStyle(SolidFill,Black);
      Bar(36,OldUPl,60,UPl);
    END
    ELSE IF UPl<OldUPl THEN
    BEGIN
      SetFillStyle(SolidFill,LightBlue);
      Bar(36,UPl,60,OldUPl);
    END;
  END; (* SpeedometerU *)
END; (* PlotUKnot *)

(*************************************************************************)


PROCEDURE PlotCourse;
(* Plottar kursen på skärmen *)
BEGIN
  WITH Compass DO
  BEGIN
    SetViewPort(x1, y1, x2, y2, ClipOn);
    Line(80,75,XKoord,YKoord); (* Raderar gammal kurs *)
    SetColor(White);
    XKoord:=80-ROUND(45*cospsi);
    YKoord:=75-ROUND(45*sinpsi);
    Line(80,75,XKoord,YKoord);
  END; (* Compass *)
END; (* PlotCourse *)

(*************************************************************************)


PROCEDURE PlotXY;
(* Plottar positionen på skärmen *)
BEGIN
  IF x>747.5 THEN
    x:=2.5
  ELSE IF x<2.5 THEN
    x:=747.5;
  IF y>1100 THEN
    y:=2.5
  ELSE IF y<2.5 THEN
    y:=1100;
  XX:= ROUND(x*0.4);                  (* XX och YY skalas med antal *)
  YY:= ROUND(y*0.4);                  (* meter*0.4                  *)
  WITH Map DO  (* procedure PlotXY *)
  BEGIN
    SetViewPort(x1,y1,x2,y2,ClipOn);
    PutPixel(XX,YY,White);
  END; (* Map *)
END; (* PlotXY *)

(*************************************************************************)


FUNCTION LowPass2  (Value:REAL):REAL;
(* Implementerar ett första ordningens lågpassfilter *)
(* att användas på utsignalen                        *)
VAR P2  :REAL;
BEGIN
  P2:=LowPassConst2*OldFiltValue2+(1-LowPassConst2)*Value;
```

```
  OldFiltValue2:=P2;
  LowPass2:=P2;
END; (* LowPass2 *)

(*************************************************************************)


FUNCTION Sign :INTEGER;
(* Ger det tecken som rattvinkeln har, +1 eller -1 *)
BEGIN
  IF dr<0 THEN
    Sign:=-1
  ELSE IF dr>0 THEN
    Sign:=1
  ELSE
    Sign:=0;
END; (* Sign *)

(*************************************************************************)


FUNCTION DregDR :REAL;
(* Implementerar en D-regulator som reglerar på felet *)
(* mellan nytt och gammalt värde av rattvinkeln       *)
VAR
  Deviation,HelpVar  : REAL;
BEGIN
  Deviation:=OldValue-dr;
  HelpVar:=1/Beta*OldHelpVar+(Beta-1)/Beta*Deviation;
  OldHelpVar:=HelpVar;
  OldValue:=dr;
  DregDR:=DConst*(HelpVar-Deviation);
END; (* DregDR *)


(*************************************************************************)


FUNCTION DregR :REAL;
(* Implementerar en D-regulator som reglerar på felet    *)
(* mellan nytt och gammalt värde av girvinkelhastigheten *)
VAR
  Deviation,HelpVar  : REAL;
BEGIN
  Deviation:=OldValueR-r;
  HelpVar:=1/BetaR*OldHelpVarR+(BetaR-1)/BetaR*Deviation;
  OldHelpVarR:=HelpVar;
  OldValueR:=r;
  DregR:=DConstR*(HelpVar-Deviation);
END; (* DregR *)



(*************************************************************************)
(*  HÄR ÄNDRAR MAN, OM MAN VILL ÄNDRA FORMEL FÖR DEN ÅTERKOPPLADE KRAFTEN  *)
(*************************************************************************)


FUNCTION FeedBack :INTEGER;
(* Ger valt återkopplingsalternativ *)
VAR
  Value,A    :REAL;
CONST
  Bias = 110;
BEGIN
  A:=Alfa*Alfa;
```

```
  CASE FeedBackChoise OF
    1: Value:= u*A*(Sign*K1+K2*Delta*A)-DregR-K3*r+DregDR+1970;
    2: Value:= DregDR-DregR+1970;
    3: Value:= DregDR-K4*r+1970;
    4: Value:= DregDR+(u*A*(K5*Sign+K6*Delta*A))+1970;
    5: Value:= Sign*(K7*Ln(K8*abs(dr)+1)*Ln(K9*u+1))+DregDR+1970;
    6: Value:= Sign*(400*Ln(4*abs(dr)+1))+1970;

(******** HÄR FINNS MÖJLIGHET FÖR ER ATT SJÄLV SKAPA ETT UTTRYCK *******)
(******** FÖR DEN ÅTERKOPPLADE KRAFTEN                          *******)

(*  6: Value:=....använd konstanterna K10 - K14 i ert uttryck... *******)

  END; (* CASE *)
  IF Value < 10 THEN                (* För att undvika olinearitet *)
    Value:=10                       (* hos AD-omvandlaren          *)
  ELSE IF Value > 4085 THEN
    Value:=4085;
  IF ABS(OldFeedBackValue-Value) > 700 THEN   (* För att åstadkomma att   *)
  BEGIN                                       (* simuleringen avbryts om  *)
    Value:=2000;                              (* skillnaden blir för stor *)
    Stop:=TRUE;
    Break:=TRUE;
  END;
  If Value > 2010 Then
    Value := Value + Bias
  ELSE If Value < 1990 Then
    Value := Value - Bias;
  OldFeedBackValue:=Value;
  FeedBack:= ROUND(LowPass2(Value));
END; (* FeedBack *)

(*****************************************************************************)
(*****************************************************************************)


FUNCTION SecondOrderLowPass  (Value:INTEGER):REAL;
(* Implementerar ett andra ordningens lågpassfilter, vilket kan användas *)
(* istället de första ordningens filter som finns på ingång och utgång   *)
VAR Y1Filt,Y2Filt  :REAL;
BEGIN
  Y1Filt:=Ordn2LowPassConst*OldY1Filt+(1-Ordn2LowPassConst)*Value;
  Y2Filt:=Ordn2LowPassConst*OldY2Filt+(1-Ordn2LowPassConst)*Y1Filt;
  OldY1Filt:=Y1Filt;
  OldY2Filt:=Y2Filt;
  SecondOrderLowPass:=Y2Filt;
END; (* SecondOrderLowPass *)

(*****************************************************************************)


PROCEDURE Calculate_And_IO;
(* Hämtar in Rodervinkel och hastighet. Beräknar v och r samt *)
(* skickar ut återkopplingskraften FeedBack till ratten       *)
CONST
  KnotConv = 1.94805195;  (* Omvandlar m/s till knop *)
BEGIN
  OlduKnot:=uKnot;
  dr:=drOffset+LowPass(AD(drChannel))*drMultFact;
  u:=AD(uChannel)*uMultFact;
  uKnot:=ROUND(KnotConv*u);
  IF ABS(uKnot-OlduKnot)>=1 THEN     (* Nya A- och B-matriser om  *)
    NewMatrix;                       (* u ändrats mer än 1 knop   *)
  v0:=v;
  v:= v0*a+r*b+dr*e;
  r:= v0*c+r*d+dr*f;
```

```
   Delta:=0.25*dr;
   Alfa:= Delta-v/u;
   DA1(FeedBack);
END; (* Calculate_And_IO *)

(*************************************************************************)


PROCEDURE Sample;
(* Beräknar ny position och kurs och hanterar IO och plottning *)
BEGIN
   INLINE($F4);  (*    HLT    *)
   Calculate_And_IO;
   psi:=psi+r*Ts;
   sinpsi:=sin(psi);
   cospsi:=cos(psi);
   x:=x-(u*cospsi-v*sinpsi)*Ts;
   y:=y-(u*sinpsi+v*cospsi)*Ts;
   LoopCount:=LoopCount+1;
   CASE LoopCount OF
     1: PlotR;
     2: PlotCourse;
     3: PlotUKnot;
     4: PlotDR;
     5: BEGIN
          PlotXY;
          LoopCount:=0;
        END;
   END; (* CASE *)
   IF KeyPressed THEN
     Stop:=True;
END;   (* Sample *)

(*************************************************************************)


PROCEDURE Simulate;
(* Denna procedur styr hela simuleringen *)
BEGIN
   ShowChosenValues;
   IF NOT Regret THEN
   BEGIN
     Installgraficmode;
     InstallDelay;
     Stop:=FALSE;
     REPEAT
       Sample;
     UNTIL Stop;
     Back;
   END;
END;  (* Simulate *)

(*************************************************************************)


PROCEDURE MainMeny;
(* Ger användaren möjlighet att välja vad programmet skall utföra *)
VAR
   CursorX,CursorY,ErrorPos    :INTEGER;
   InputNumber                 :STRING;
BEGIN
   ClrScr;
   Ram;
   GoToXY(35,6);
   Write('HUVUDMENY !');
   GoToXY(10,8);
```

```
  WriteLn('1... Information om försöksuppställning och parametrar');
  GoToXY(10,10);
  WriteLn('2... Simulera');
  GoToXY(10,12);
  WriteLn('3... Omkalibrera eller återställa till defaultvärden'); äSkapa fil
för A och B matriserna');å
  GoToXY(10,14);
  WriteLn('4... Välj vilka parametrar den återkopplade kraften skall bestå av'
);
  GoToXY(10,16);
  WriteLn('5... Välj konstanter till vald återkopplad kraft');
  GoToXY(10,18);
  WriteLn('6... Välj filter-och reglerkonstanter samt sampelfrekvens');
  GoToXY(10,20);
  WriteLn('7... Avsluta');
  GoToXY(10,22);
  Write('Ange alternativ:  ');
  CursorX:=WhereX;
  CursorY:=WhereY;
  ReadLn(InputNumber);
  Val(InputNumber,Choise,ErrorPos);
  WHILE (Choise < 1) OR (Choise > 7) OR (InputNumber='') DO
  BEGIN
    SetCursorOff;
    TextBackground(12);
    TextColor(14);
    GoToXY(CursorX,CursorY);
    Write('FELAKTIGT VÄRDE !');
    DELAY(2000);
    GoToXY(CursorX,CursorY);
    TextBackground(0);
    TextColor(7);
    HighVideo;
    Write('                        ');
    GoToXY(CursorX,CursorY);
    RestoreCursor;
    ReadLn(InputNumber);
    Val(InputNumber,Choise,ErrorPos);
  END;
END; (* MainMeny *)

(*************************************************************************)

PROCEDURE Loop;
(* Exekverar programmet tills dess att användaren vill avbryta *)
BEGIN
  REPEAT
    BreakMeny;
    MainMeny;
    CASE Choise OF
      1: InformationMeny;
      2: Simulate;
      3: Calibrate_Or_Default;   äWrite_A_And_B_Matrix;å (* TA BORT DETTA  *)
      4: New_FeedBack_Alt;                               (* OM NY SAMPEL-  *)
      5: New_FeedBack_Const;                             (* FREKVENS SKALL *)
      6: New_Filter_And_ControlConst;                    (* SKAPAS         *)
      7: Quit:=TRUE;
    END; (* CASE *)
  UNTIL Quit;
END; (* Loop *)

(*************************************************************************)
(*                          HUVUDPROGRAM                                 *)
(*************************************************************************)
```

```
BEGIN
  IntroductionMeny;
  InitiateDefault;
  Loop;
END. (* Huvudprogram *)

(*****************************************************************************)
(*                                  SLUT                                     *)
(*****************************************************************************)
```

G. Underprogrambeskrivning av MMI

Beskrivning av samtliga procedurer i modulen "PLOT SET":

Plot Ver Scale Arrow: Ritar en vertikal tendenspil till den vertikala skalan. Tendenspilen placeras omedelbart till höger om skalan och visar om skalans variabelvärde minskar eller ökar. Tendenspilens längd hämtas från modulen "TYPEVAR". Inparametrarna är:

- ScaleColor: Färgen på tendenspilen, som är av heltalstyp.
- ArrowXPos: Skärmpositionen på x-axeln, där tendenspilen skall ritas. Är av heltalstyp.
- ArrowYPos: Skärmpositionen på y-axeln, där tendenspilen skall ritas. Är av heltalstyp.
- New: Nya variabelvärdet omvandlat till antalet punkter på skärmen. Är av heltalstyp.
- Old: Gamla variabelvärdet omvandlat till antalet punkter på skärmen. Är av heltalstyp.

Plot Hor Scale Arrow: Ritar en horisontell tendenspil till den horisontella skalan. Tendenspilen placeras ovanför skalan och visar om skalans variabelvärde minskar eller ökar. Tendenspilens längd hämtas från modulen "TYPEVAR". Inparametrarna är:

- ScaleColor: Färgen på tendenspilen. Är av heltalstyp.
- ArrowXPos: Skärmpositionen på x-axeln, där tendenspilen skall ritas. Är av heltalstyp.
- ArrowYPos: Skärmpositionen på y-axeln, där tendenspilen skall ritas. Är av heltalstyp.
- New: Nya variabelvärdet omvandlat till antalet punkter på skärmen. Är av heltalstyp.
- Old: Gamla variabelvärdet omvandlat till antalet punkter på skärmen. Är av heltalstyp.

Plot Bar In Scale: I aktuell skala ritas en färglagd stapel, som motsvarar variabelvärdet till skalan. In- och utparametrarna är:

- Scale: Post som innehåller all information om aktuell skala.

Plot Mark In Scale: Ritar symboler i aktuell skala för är- och börvärdesmarkeringarna. Ärvärdesmarkeringen består av en pilsymbol och börvärdesmarkeringen består av en streckmarkering. Om symbolerna hamnar utanför skalans ändvärden ritas de ej ut. In- och utparametrarna är:

- Scale: Post som innehåller all information om aktuell skala.

Plot Cursor Arrow: Ritar tendenspilen till markören i aktuellt hårkorsinstrument. Tendenspilen anger i vilken riktning markören är på väg i hårkorset. Tendenspilens längd hämtas från modulen "TYPEVAR". In- och utparametrarna är:

- HairCross: Post som innehåller all information om aktuellt instrument.

Inparametrarna är:

- NewX: Nytt variabelvärde för x-axeln omvandlat till antalet punkter på skärmen. Är av heltalstyp.
- NewY: Nytt variabelvärde för y-axeln omvandlat till antalet punkter på skärmen. Är av heltalstyp.
- OldX: Föregående variabelvärdet för x-axeln omvandlat till antalet punkter på skärmen. Är av heltalstyp.
- OldY: Föregående variabelvärdet för y-axeln omvandlat till antalet punkter på skärmen. Är av heltalstyp.
- OrigoX: Positionen för aktuellt hårkors för x-axeln. Är av heltalstyp.
- OrigoY: Positionen för aktuellt hårkors för y-axeln. är av heltalstyp.
- CoursorRad: Radien på markören angivet i millimeter. Är av reell typ.
- Rad: Radien på hela instrument angivet i millimeter. Är av reell typ.

Plot Cursor In Horizon: Ritar ut markören i aktuellt hårkors. Markören består av en färglagd cirkel med radien som hämtas från modulen "TYPEVAR". Om markören skulle hamna utanför instrumentet flyttas markören in i cirkeln och tendenspilen till markören ritas då inte. In- och utparametrar är:

- HairCross: Post som innehåller all information om aktuellt instrument.

Print Digital Instrument Variable: Skriver ut variabelvärdet för en alternativt två variabler i det aktuella instrumentet. In- och utparametrar är:

- DigitalInstr: Post som innehåller all information om aktuellt siffervisande instrument.

Erase Window Text: Suddar den text, som finns för tillfälligt i rutan i översta högra hörnet.

Select Scale F1: Sätter flaggan till knapp F1 och väljer vilken skala man vill behandla med knappen F3.

Select Scale F2: Sätter flaggan till knapp F2 och väljer vilket hårkors man vill behandla med knapp F3.

Move Or Expand Scale Upwards F3: Beroende på val av instrument med knapp F1 och F2 expanderas eller flyttas skalorna uppåt till det aktuella instrumentet.

Move Or Compress Scale Downwards F4: Beroende på val av instrument med knapparna F1 och F2 komprimeras eller flyttas skalorna nedåt till det aktuella instrumentet.

AutoScaling F5: Sätter flaggan till knapp F5 och kopplar in automatisk skalning av instrumenten.

Set Flag F6 till och med Set Flag F10: Sätter flaggan till respektive knapp.

Check If Key Pressed: Kontrollerar om någon av knapparna F1-F10 har blivit nedtryckt. Om så är fallet sker hopp till respektive procedur för vald knapp.

Beskrivning av samtliga procedurerna i modulen "CREATE":

Make Screen Layout: Initierar startvärden för instrumentvariablerna och initierar grafiken, samt ritar upp instrumenten på datorskärmen.

Ver Scale Grad: Graderar en vertikal skala på det sätt som anges med variabeln "grad". Inparametrar är:

- Length: Skalans längd i antalet punkter på skärmen. Är av heltalstyp.
- Res: Antalet skalstrecksintervall i skalan. Är av heltalstyp.
- Max: Skalans översta ändvärde. Är av heltalstyp.
- Min: Skalans understa ändvärde. Är av heltalstyp.
- X: Skärmpositionen på x-axeln där skalan finns. Är av heltalstyp.
- Y: Skärmpositionen på y-axeln där skalan finns. Är av heltalstyp.
- Grad: Anger hur skalan skall graderas. De möjligheter som finns är linjär,logaritmisk och eget val. Är av gradtyp.
- Color: Färg på aktuell skala. Är av heltalstyp.

Hor Scale Grad: Graderar en horisontell skala på det sätt som anges med variabeln "grad". Inparametrar är:

- Length: Skalans längd i antalet punkter på skärmen. Är av heltalstyp.
- Res: Antalet skalstrecksintervall i skalan. Är av heltalstyp.
- Max: Skalans högra ändvärde. Är av heltalstyp.
- Min: Skalans vänstra ändvärde. Är av heltalstyp.
- X: Skärmpositionen på x-axeln där skalan finns. Är av heltalstyp.
- Y: Skärmpositionen på y-axeln där skalan finns. Är av heltalstyp.
- Grad: Anger hur skalan skall graderas. De möjligheter som finns är linjär,logaritmisk och eget val. Är av gradtyp.
- Color: Färg på aktuell skala. Är av heltalstyp.

Make Vertical Scale: Ritar ett vertikalt graderat skalstreck och skriver ut ändvärdena för skalan, samt beräknar nollpositionen för färgstapeln och färgpilen. Skalstrecket graderas med hjälp av proceduren "Ver Scale Grad". In- och utparametrar är:

- Scale: Post som innehåller all information om aktuell skala.

Make Horisontel Scale: Ritar ett horisontellt graderat skalstreck och skriver ut ändvärdena för skalan, samt beräknar nollpositionen för färgstapeln och färgpilen. Skalstrecket graderas med hjälp av proceduren "Hor Scale Grad". In- och utparametrar är:

- Scale: Post som innehåller all information om aktuell skala.

Make Background And Frame: Färglägger bakgrunden med färgen "ScreenBackRoundColor" och ritar en ram runt skärmen med färgen "FrameColor", samt gör en ruta i översta högra hörnet med färgen "WindowColor". Skuggan till rutan görs i färgen "ShaddowColor". Samtliga färger är av heltalstyp.

Make HairCross Scales: Ritar ett vertikalt och ett horisontellt graderat skalstreck i aktuellt hårkorsinstrument, samt skriver ut ändvärdena för skalstrecken. Skalstrecken graderas med procedurerna "Ver Scale Grad" och "Hor Scale Grad". Inparametrar är:

- HairCross: Post som innehåller all information om aktuellt instrument.

Make HairCross: Ritar ett hårkorsinstrument i form av en komplett färglagd cirkel, samt ritar ut en vertikal och en horisontell skala i cirkeln med hjälp av proceduren "Make HairCross Scales". In- och utparametrar:

- HairCross: Post som innehåller all information om aktuellt instrument.

Make Digital Instrument: Skriver ut texten till ett siffervisande instrument. In- och utparametrar:

- Digitalinstr: Post som innehåller all information om aktuellt instrument.

Make Three Text Lines: Skriver ut tre rader text på skärmen. Varje rad text består av tolv rader tecken. In- och utparametrar:

- Text: Post som innehåller all information texten.
- Color: Färgen på texten av heltalstyp.

Expand Scales: Proceduren ändrar ändvärdena på aktuell skala. Om ändvärdena skall ökas eller minskas anges med variablerna "Expand" och "Compress", som är av logisk typ. Om "Expand" är sann och "Compress" är falsk expanderas skalans ändvärden med en faktor angivet av "Magn", som är av heltalstyp. Om "Expand" är falsk och "Compress" är sann komprimeras skalans ändvärden med en faktor angivet av "Magn". In- och utparametrarna är:

- Scale: Post som innehåller all information om aktuell skala.

Utparametrarna är:

- Magn: En faktor som anger hur mycket skalans ändvärden skall ändras. Är av heltalstyp.
- Expand: Anger om skalan skall expanderas. Är av logisk typ.
- Compress: Anger om skalan skall komprimeras. Är av logisk typ.

Expand HairCross: Proceduren ändrar ändvärdena på x-skalan och y-skalan i aktuellt hårkors. Om ändvärdena för skalorna skall ökas eller minskas anges med variablerna "Expand" och "Compress". Om "Expand" är sann och "Compress" är falsk expanderas skalornas ändvärden med en faktor angivet av "Magn", som är av heltalstyp. Om "Expand" är falsk och "Compress" är sann komprimeras skalornas ändvärden med en faktor angivet av "Magn". In- och utparametrarna är:

- HairCross: Post som innehåller information om aktuellt instrument.

Inparametrarna är:

- Magn: En faktor som anger hur mycket skalornas ändvärden skall ändras. Är av heltalstyp.
- Expand: Anger om skalorna skall expanderas. Är av logisk typ.
- Compress: Anger om skalorna skall komprimeras. Är av logisk typ.

Move Scale Up Or Down: Proceduren adderar alternativt subtraherar skalans ändvärden med termen "Intervall". Om "Up" är sann och "Down" är falsk adderas skalans ändvärden med termen "Intervall". Om "Up" är falsk och "Down" är sann subtraheras skalans ändvärden med termen "Intervall". In- och utparametrarna är:

- Scale: Post som innehåller information om aktuellt instrument.
- Intervall: Anger hur mycket skalan flyttas upp eller ned. Är av heltalstyp.

- Up: Anger om skalan skall flyttas uppåt. Är av logisk typ.
- Down: Anger om skalan skall flyttas nedåt. Är av logisk typ.

Auto Scaling: Proceduren kontrollerar om variabelvärdet till aktuell skala befinner sig utanför skalans ändvärden. Om så är fallet skalas instrumentet om enligt följande. Instrumentet skalas om med proceduren "Expand Scale" om variabelpresentationen i skalan är en färglagd stapel. Instrumentet skalas om med proceduren "Move Scale Up Or Down" om variabelpresentationen i skalan är en är- och börvärdesmarkering. In- och utparametrarna är:

- Scale: Post som innehåller information om aktuellt instrument.
- Value: Variabelvärdet för den variabel som presenteras i instrumentet. Är av reell typ.

Beskrivning av samtliga funktioner i modulen "CALC":

Int To Str: Omvandlar en heltalstyp till en sträng med maximalt 10 tecken. Inparameter är:

- I: Talet som skall omvandlas till en sträng. Är av heltalstyp.

Utparameter är:

- Int To Str: Sträng med maximalt 10 tecken.

Real To Str: Omvandlar ett reellt tal till en sträng med maximalt 10 tecken. Inparameter är:

- I: Talet som skall omvandlas till en sträng. Är av reell typ.

Utparameter är:

- Real To Str: Sträng med maximalt 10 tecken.

mm To Pix X: Omvandlar en längd i horisontell led i millimeter till antalet punkter på skärmen. Inparameter är:

- A: Längden i millimeter. Är av reell typ.

Utparameter är:

- mm To Pix X: Längden i antalet punkter på skärmen. Är av heltalstyp.

mm To Pix Y: Omvandlar en längd i vertikal led i millimeter till antalet punkter på skärmen. Inparameter är:

- B: Längden i millimeter. Är av reell typ.

Utparameter är:

- mm To Pix Y: Längden i antalet punkter på skärmen. Är av heltalstyp.

F: Genom att ange en funktionstyp med "Graduation" räknas funktionsvärdet ut för F(Value). Inparametrarna är:

- Value: Ett tal av reell typ.
- Graduation: Anger vilken funktion som skall användas. Är av gradtyp.

Utparameter är:

- F: Det slutliga funktionsvärdet. Är av reell typ.

Variable: Läser ett variabelvärde från variabelvektorn. Inparameter är:

- Nr: Anger den position i vektorn där värdet skall läsas. Är av heltalstyp.

Utparameter är:

- Variable: Variabelvärdet är av reell typ.

Conv Value To Pix: Omvandlar ett variabelvärde till att passa skalans värde på skärmen uttryckt i antalet punkter på skärmen. Graderingstypen av skalan anges med "Grad". Inparametrarna är:

- Value: Variabelvärdet av reell typ.
- Max: Skalans högra ändvärde om skalan är horisontell. Skalans översta ändvärde om skalan är vertikal.
- Min: Skalans vänstra ändvärde om skalan är vertikal. Skalans understa ändvärde om skalan är vertikal.
- ScaleLength: Skallängden uttryckt i antalet punkter på skärmen Är av heltalstyp.
- Grad: Anger hur skalan är graderad. Är av gradtyp.

Utparameter är:

- Conv Value To Pix: Variabelvärdet uttryckt i antalet punkter på skärmen. Är av heltalstyp.

Beskrivning av samtliga procedurer i modulen "INIT":

Parameter Initialization: Initierar startvärdena för ett antal parametrar.

Graph Initialization: Proceduren läser av vilken grafik datorn arbetar med, samt initierar den. De alternativ som finns är VGA- och EGA-grafik.

Instrument Initialization: Här lägger användaren själv in initieringsvärden till samtliga instrument. Se vidare i användarmanualen (I) hur detta skall ske.

Screen Color Initialization: Här väljer användaren själv vilka färger som önskas på skärmen. Se vidare i användarmanualen (I) hur detta skall ske.

Beskrivning av modulen "TYPEVAR":

I denna modul finns alla konstant-, typ- och variabeldeklarationer, som programmet arbetar med.

H. Begränsningar för MMI

Programmet är skrivet i Turbopascal, vilket medför att realtidsprogrammering ej är möjlig. Detta anser vi vara en brist speciellt då man önskar uppdatera instrumenten ofta, eftersom denna uppdatering tar ganska lång tid.

Vi har som ett hjälpmedel för användaren gjort mätningar på hur lång tid det tar att uppdatera respektive instrument. Denna tid sätter en begränsning för användaren för hur man skall uppdatera skärmen. Följande tid för respektive instrument fick vi:

Skalorna: cirka 8 millisekunder

Hårkors: cirka 25 millisekunder

I. Användarmanual och programlistning för MMI

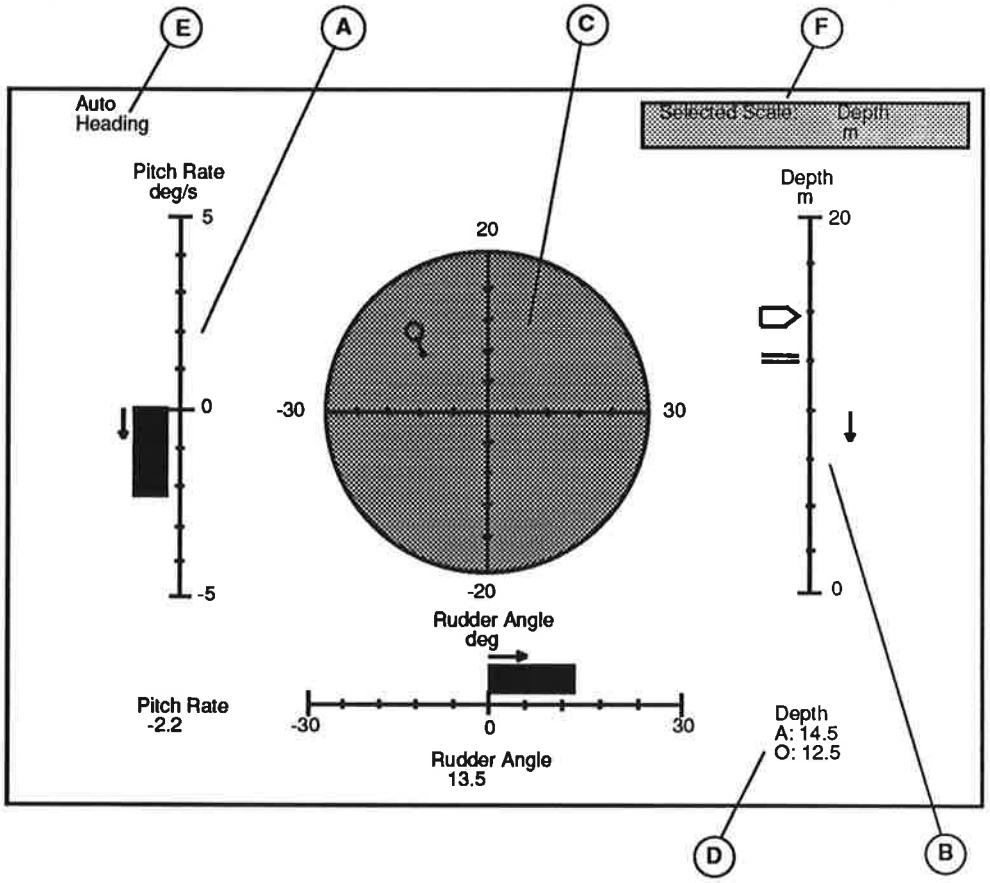
De instrument som är möjliga att skapa återges i figur I.1, där de olika instrumenttyperna är A,B,C,D och E. Alla hänvisningar i användarmanualen görs till denna figur.

När man bestämt sig för en lämplig förarmiljö genom att rita upp en tänkt skärmbild på ett papper går man in i modulerna "INIT","TYPEVAR" och "CREATE" och anger initieringssatser för alla instrumenten .Varje instrument av en viss typ tilldelar man ett tal för att kunna referera till ett visst instrument.

För skärmen gäller att alla positioner räknas från översta vänstra hörnet på skärmen till mittpunten på instrumentet, och alla mått anges i millimeter. Den horisontella axeln kallas x-axeln och den vertikala axeln kallas y-axeln. Programmet räknar om alla mått från millimeter till antalet punkter på skärmen.

De färger som finns tillgängliga för skärmgrafiken är följande: black, blue, green, cyan, red, magenta, brown, lightgray, darkgray, lightblue, lightgreen, lightcyan, lightred, lightmagenta, yellow och white.

Modulen får tillgång till variabelvärden från användarprogrammet med hjälp av en variabelvektor kallad "VariableVector", som är av reell typ. An-



Figur I.1 Exempel på möjliga intrument till förarmiljön

vändaren måste således uppdatera denna vektorn varje gång skärmen skall ritas om.

Här följer ett exempel på hur man initierar de olika typerna av instrument, som visas i figur I.1.

I proceduren "Instrument_Initialization" i modulen "INIT":

Instrument av typen A: Följande variabler måste initieras (värdena är exempel):

```
WITH Scale [1]DO
BEGIN
Length := 80;
ScaleColor := White;
AVarNr := 1;
ScaleType := Log;
ScaleResolution := 10;
ScalePos.X := 60;
ScalePos.Y := 80;
ScaleValue.Max := 5;
ScaleValue.Min := -5;
Text.Line1 := ' Pitch Rate ';
Text.Line2 := ' deg/s ';
ValuePresentation := ColoredBar;
Bar.Color := Green;
Bar.Width := 8;
END;
```

Instrument av typen B: Följande variabler måste initieras (värdena är exempel):

```
WITH Scale [2]DO
BEGIN
Length := 80;
ScaleColor := White;
AVarNr := 2;
OVarNr := 3;
ScaleType := Lin;
ScaleResolution := 10;
ScalePos.X := 180;
ScalePos.Y := 80;
ScaleValue.Max := 20;
ScaleValue.Min := 0;
Text.Line1 := ' Depth ';
Text.Line2 := ' m ';
ValuePresentation := ColoredMark;
Mark.AColor := Yellow;
```

```
Mark.OColor := LightCyan;
END;
```

För instrumenten av typen A och B gäller:

- Length: Skalans totala längd angivet i millimeter.
- ScaleColor: Färgen på skalan och texten.
- AVarNr: Anger positionen i variabelvektorn där ärvärdet finns.
- OVarNr: Anger positionen i variabelvektorn där börvärdet finns.
- ScaleType: Anger skalans graderingstyp. Lin, log eller eget val.
- ScaleResolution: Antalet graderingsintervall i skalan.
- ScalePos.X: Skalans position på skärmen i x-led angivet i mm.
- ScalePos.Y: Skalans position på skärmen i y-led angivet i mm.
- ScaleValue.Max: Skalans översta ändvärde om skalan är vertikal. Skalans högra ändvärde om skalan är horisontell.
- ScaleValue.Min: Skalans understa ändvärde om skalan är vertikal. Skalans vänstra ändvärde om skalan är horisontell.
- Text.Line1: Översta raden text à tolv tecken.
- Text.Line2: Understa raden text à tolv tecken.
- ValuePresentation: Val av skaltyp A eller B, således "ColoredBar" eller "ColoredMark".
- Bar.Color: Färgen på stapeln för skaltyp A.
- Bar.Width: Bredden på stapeln för skaltyp A angivet i mm.
- Mark.AColor: Ärvärdesmarkeringens färg för skaltyp B.
- Mark.OColor: Börvärdesmarkeringens färg för skaltyp B.

Instrument av typen C: Följande variabler måste initieras (värdena är exempel):

```
WITH HairCross [1]DO
BEGIN
Radius := 35;
ScaleColor := White;
BackRoundColor := Black;
Origo.X := 120;
Origo.Y := 80;
WITH XScale DO
BEGIN
VarNr := 4;
ScaleValue.Max := 20;
ScaleValue.Min := -20;
ScaleResolution := 10;
ScaleType := Lin;
END;
WITH YScale DO
BEGIN
VarNr := 5;
```

ScaleValue.Max := 20;

ScaleValue.Min := -20;

ScaleResolution := 10;

ScaleType := Lin;

END;

CoursorColor := LightCyan;

END;

För instrument av typen C gäller:

- Radius: Radien på hårkorset angivet i millimeter.
- ScaleColor: Färgen på axlarna och cirkeln till instrumentet.
- BackRoundColor: Bakgrundsfärgen i hårkorset.
- Origo.X: Hårkorsets position på skärmen i x-led angivet i mm.
- Origo.Y: Hårkorsets position på skärmen i y-led angivet i mm.
- XScale.VarNr: Anger positionen i variabelvektorn där x-skalans ärvärde finns.
- XScale.ScaleValue.Max: X-skalans högra ändvärde.
- XScale.ScaleValue.Min: X-skalans vänstra ändvärde.
- XScale.ScaleResolution: Antalet graderingintervall i x-skalan.
- XScale.ScaleType: Anger x-skalans graderingstyp. Lin, log eller eget val.
- YScale.VarNr: Anger positionen i variabelvektorn, där y-skalans ärvärde finns.
- YScale.ScaleValue.Max: Y-skalans översta ändvärde.
- YScale.ScaleValue.Min: Y-skalans understa ändvärde.
- YScale.ScaleResolution: Antalet graderingsintervall i y-skalan.
- YScale.ScaleType: Anger y-skalans graderingstyp. Lin,log eller eget val.

Instrument av typen D: Följande variabler måste initieras (värdena är exempel):

WITH DigitalInstr [1]DO

BEGIN

VarNr [1]:= 2;

VarNr [2]:= 3;

TextPos.X := 166;

TextPos.Y := 160;

NrOfVar := 2;

Color := White;

Line1 := ' Depth ';

Line2 := ' A: ';

Line3 := ' O: ';

END;

För instrument av typen D gäller:

- VarNr[1]: Anger positionen i variabelvektorn där den första variabeln finns.

- VarNr[2]: Anger positionen i variabelvektorn där den andra variabeln finns. Om endast en variabel önskas utelämnas denna rad.
- TextPos.X: Instrumentets position på skärmen i x-led angivet i mm.
- TextPos.Y: Instrumentets position på skärmen i y-led angivet i mm.
- NrOfVar: Antalet variabler som presenteras i instrumentet, max två.
- Color: Färgen på texten till hela instrumentet.
- Line1: Första raden text à 12 tecken.
- Line2: Andra raden text à 12 tecken.
- Line3: Tredje raden text à 12 tecken

Text av typen E: Följande måste initieras (värdena är exempel):

```
WITH Text[1]DO
BEGIN
TextPos.X := 80;
TextPos.Y := 3;
Line1 := ' Auto ';
Line2 := ' Heading ';
Line3 := ' ';
TextColor := White;
END;
```

För text av typen E gäller:

- TextPos.X: Textens position på skärmen i x-led angivet i mm.
- TextPos.Y: Textens position på skärmen i y-led angivet i mm.
- Line1: Första raden text à tolv tecken.
- Line2: Andra raden text à tolv tecken.
- Line3: Tredje raden text à tolv tecken.
- TextColor: Färgen på texten.

I proceduren "Screen_Color_Initialization" i modulen "INIT":

När initieringen i proceduren "Instrument_Initialization" är klar går man över till proceduren "Screen Color Initialization". I denna proceduren anger man vilka färger skärmen skall ha enligt följande (värdena är exempel):

```
ScreenBackRoundColor := Red;
FrameColor := LightGray;
WindowColor := DarkGray;
WindowShaddowColor := Black;
```

För färgerna gäller följande:

- ScreenBackRoundColor: Skärmens bakgrundsfärg.
- FrameColor: Färgen på ramen runt skärmen.
- WindowColor: Färgen på rutan i översta högra hörnet.
- WindowShaddowColor: Färgen på rutans skugga i översta högra hörnet.

I modulen "TYPEVAR":

Nästa steg är att ange ett antal konstanter i modulen "TYPEVAR". Följande konstanter måste initieras (värdena är exempel):

CursorRadius = 2.0;

CursorArrowLength = 3.0;

ScaleArrowLength = 5.0;

För konstanterna gäller följande:

- CursorRadius: Radien på markören till samtliga hårkorsinstrument på skärmen angivet i millimeter.
- CursorArrowLength: Tendenspilens längd till markören för samtliga hårkors på skärmen angivet i millimeter.
- ScaleArrowLength: Tendenspilens längd till samtliga skalor på skärmen angivet i millimeter.

I proceduren "Make_Screen_Layout" i modulen "CREATE".

Slutligen måste man gå in i modulen "CREATE", där man i proceduren "Make Screen Layout" anger proceduranrop för procedurer, som ritar upp den statiska delen av förarmiljön på skärmen.

Inparametrarna till procedurerna är de variabelposterna som man tidigare angav i modulen "INIT".

Förutom de fem första raderna i "Make_Screen_Layout" skall följande skrivas till (raderna är exempel för att åstadkomma skärmen enligt figur I.1):

```
Make_Vertical_Scale (Scale[1]);
Make_Vertical_Scale (Scale[2]);
Make_HairCross (HairCross[1]);
Make_Digital_Instrument (DigitalInstr[1]);
Make_Three_Text_Lines (Text[1],Text[1].TextColor);
```

Om man till exempel vill ha skala 1 horisontell istället för vertikal ändrar man den översta raden ovan till:

```
Make_Horisontel_Scale (Scale[1]);
```

I användarprogrammet:

När samtliga moduler är initierade kompileras alla modulerna var för sig och användaren kan sedan skriva sitt eget användarprogram. I användarprogrammet måste följande finnas med för att möjliggöra kommunikation mellan användaren och modulerna (enkelt exempel på om man vill uppdatera alla instrument varje gång man beräknar nya variabelvärden):

```
PROGRAM Användarprogram;
USES Crt,CREATE,PLOT_SET,TYPEVAR;
BEGIN
Make_Screen_Layout;
REPEAT
(**** Beräkningar som användaren gör ****);
VariableVector [1]:= (**** en variabel ****);
VariableVector [2]:= (**** en variabel ****);
VariableVector [3]:= (**** en variabel ****);
VariableVector [4]:= (**** en variabel ****);
VariableVector [5]:= (**** en variabel ****);
```

```
Plot_Bar_In_Scale (Scale[1]);
Plot_Mark_In_Scale (Scale[2]);
Plot_Coursor_In_HairCross (HairCross[1]);
Print_Digital_Instrument_Variable (DigitalInstr[1]);
Check_If_Key_Pressed (Stop);
UNTIL Stop;
END;
```

Knapparna F6-F10:

Knapparna F6-F10 är lediga för användarens egna tillämpningar. Här kan man till exempel föra in lämplig programtext som ger utskrift på skärmen vid nedtryckning av önskad knapp. För detta ändamålet finns proceduren "Make Three Text Lines" att tillgå.

Användaren kan också utnyttja de flaggor som finns implementerade med hjälp av vektorn "FLAG", som består av tio element, som är av logisk typ. Flaggorna sätts vid knappnedtryck och användaren kan använda flaggorna efter eget önskemål i användarprogrammet.

Knapparna F1-F5:

Resterande knapparna F1-F5 är redan programmerade och har följande funktioner:

F1: Val av den skala, som man vill behandla.

F2: Val av det hårkors, som man vill behandla.

F3: Om man valt en skala enligt instrument A i figur I.1 med knappen F1 expanderas skalans ändvärden. Om man valt en skala enligt instrument B med knappen F1 flyttas skalintervallet uppåt. Om man istället valt ett hårkors enligt instrument C med knappen F2, expanderas x- och y-skalans ändvärden.

F4: Om man valt en skala enligt instrument A i figur I.1 med knappen F1, komprimeras skalans ändvärden. Om man valt en skala enligt instrument B med knappen F1, flyttas skalintervallet nedåt. Om man istället valt ett hårkors enligt instrument C, komprimeras x- och y-skalans ändvärden.

F5: Automatisk skalning av samtliga skalor och hårkors enligt samma princip som den manuella skalningen med knapparna F3 och F4.

Vid val av manuell skalning, kopplas den automatiska skalningen ur och på samma sätt om man valt automatisk skalning kopplas den manuella skalningen ur. I en ruta i översta högra hörnet på skärmen anges vilken av knapparna F1,F2 eller F5 man valt.

```
UNIT CREATE;

INTERFACE

USES Graph,CALC,TYPEVAR,INIT;

Procedure Make_Screen_Layout;
Procedure Ver_Scale_Grad (Length,Res,Max,Min,X,Y : Integer;
                          Grad : GradType; Color : Integer);
Procedure Hor_Scale_Grad (Length,Res,Max,Min,X,Y : Integer;
                          Grad : GradType; Color : Integer);
Procedure Make_Vertical_Scale (Var Scale : ScalePost);
Procedure Make_Horisontel_Scale (Var Scale : ScalePost);
Procedure Make_Background_And_Frame;
Procedure Make_HairCross_Scales (HairCross : HairCrossPost);
Procedure Make_HairCross (Var HairCross : HairCrossPost);
Procedure Make_Digital_Instrument (Var DigitalInstr : DigitalInstrPost);
Procedure Make_Three_Text_Lines (Text : TextPost; Color : Integer);
Procedure Expand_Scale (Var Scale : ScalePost; Magn : Integer;
                        Expand,Compress : Boolean);
Procedure Expand_HairCross (Var HairCross : HairCrossPost; Magn : Integer;
                            expand,compress : Boolean);
Procedure Move_Scale_Up_Or_Down (Var Scale : ScalePost; Interval : Integer;
                                 Up,Down : Boolean);
Procedure Auto_Scaling (Var Scale:ScalePost; Value : Real);

IMPLEMENTATION

Procedure Make_Screen_Layout;
BEGIN
   Parameter_Initialization;
   Graph_Initialization;
   Screen_Color_Initialization;
   Instrument_Initialization;
   Make_Background_And_Frame;
   Make_Vertical_Scale (ScaleÄ1Å);
   Make_Horisontel_Scale (ScaleÄ2Å);
   Make_Vertical_Scale (ScaleÄ3Å);
   Make_HairCross (HairCrossÄ1Å);
   Make_Digital_Instrument (DigitalInstrÄ1Å);
   Make_Digital_Instrument (DigitalInstrÄ2Å);
   Make_Digital_Instrument (DigitalInstrÄ3Å);
END;

Procedure Ver_Scale_Grad (Length,Res,Max,Min,X,Y : Integer;
                          Grad : GradType; Color : Integer);
(* Graderar en vertikal skala på det sätt som anges i 'grad' *)
VAR Yvalue,HalfRes,ZeroPos,B,Interval : Integer;
    V : Real;
BEGIN
  SetColor (Color);
  If ((Min < 0) AND (Max > 0)) OR ((Min > 0) AND (Max < 0)) Then
  BEGIN
    HalfRes := Res Div 2;
    ZeroPos := Y;
    YValue := ZeroPos;
    For B := 1 To (HalfRes - 1) Do
    BEGIN
      Interval := ROUND (Length / 2 * F (ABS(Max)*b/HalfRes,Grad) /
                                       F (ABS(Max),Grad));
      YValue := ZeroPos - Interval;
      MoveTo (X - 2,YValue);
      LineTo (X + 2,YValue);
    END;
    YValue := ZeroPos;
    For B:= 1 To (HalfRes - 1) Do
```

```
    BEGIN
      Interval := ROUND (Length / 2 * F (ABS(Min)*b/HalfRes,Grad) /
                                       F (ABS(Min),Grad));
      YValue := ZeroPos + Interval;
      MoveTo (X - 2,YValue);
      LineTo (X + 2,YValue);
    END;
  END
  ELSE If ABS (Max) > ABS (Min) Then
  BEGIN
    ZeroPos := ROUND (Y + Length / 2);
    YValue := ZeroPos;
    For B:= 1 To (Res - 1) Do
    BEGIN
      V := b / Res *(ABS(Max)-ABS(Min))+ABS(Min);
      Interval := ROUND (Length*(F(V,Grad) - F(ABS(Min),Grad)) /
                                (F(ABS(Max),Grad) - F(ABS(Min),Grad)));
      YValue := ZeroPos - Interval;
      MoveTo (X - 2,YValue);
      LineTo (X + 2,YValue);
    END;
  END
  ELSE If ABS (Max) < ABS (Min) Then
  BEGIN
    ZeroPos := ROUND ( Y - Length / 2);
    Yvalue := ZeroPos;
    For B:= 1 To (Res - 1) Do
    BEGIN
      V := b / Res *(ABS(Min)-ABS(Max))+ABS(Max);
      Interval := ROUND (Length*(F(V,Grad) - F(ABS(Max),Grad)) /
                                (F(ABS(Min),Grad) - F(ABS(Max),Grad)));
      YValue := ZeroPos + Interval;
      MoveTo (X - 2,YValue);
      LineTo (X + 2,YValue);
    END;
  END;
END;

Procedure Hor_Scale_Grad (Length,Res,Max,Min,X,Y : Integer;
                          Grad : GradType; Color : Integer);
(* Graderar en horisontell skala på det sätt som anges i 'grad'. *)
Var XValue,HalfRes,ZeroPos,B,Interval : Integer;
BEGIN
  SetColor (Color);
  If ((Min < 0) AND (Max > 0)) OR ((Min > 0) AND (Max < 0)) Then
  BEGIN
    HalfRes := Res Div 2;
    ZeroPos := X;
    XValue := ZeroPos;
    For B := 1 To (HalfRes - 1) Do
    BEGIN
      Interval := ROUND (Length / 2 * F (ABS(Max)*b/HalfRes,Grad) /
                                       F (ABS(Max),Grad));
      XValue := ZeroPos + Interval;
      MoveTo (XValue,Y - 2);
      LineTo (XValue,Y + 2);
    END;
    XValue := ZeroPos;
    For B:= 1 To (HalfRes - 1) Do
    BEGIN
      Interval := ROUND (Length / 2 * F (ABS(Min)*b/HalfRes,Grad) /
                                       F (ABS(Min),Grad));
      XValue := ZeroPos - Interval;
      MoveTo (XValue,Y - 2);
      LineTo (XValue,Y + 2);
    END;
```

```
    END
    ELSE If ABS (Max) > ABS (Min) Then
    BEGIN
      ZeroPos := ROUND (X - Length / 2);
      XValue := ZeroPos;
      For B:= 1 To (Res - 1) Do
      BEGIN
        Interval := ROUND (Length*(F(ABS(Max)*b/Res,Grad) - F(ABS(Min),Grad)) /
                                   (F(ABS(Max),Grad) - F(ABS(Min),Grad)));
        XValue := ZeroPos + Interval;
        MoveTo (XValue,Y - 2);
        LineTo (XValue,Y + 2);
      END;
    END
    ELSE If ABS (Max) < ABS (Min) Then
    BEGIN
      ZeroPos := ROUND (X + Length / 2);
      Xvalue := ZeroPos;
      For B:= 1 To (Res - 1) Do
      BEGIN
        Interval := ROUND (Length*(F(ABS(Min)*b/Res,Grad) - F(ABS(Max),Grad)) /
                                   (F(ABS(Min),Grad) - F(ABS(Max),Grad)));
        XValue := ZeroPos - Interval;
        MoveTo (XValue,Y - 2);
        LineTo (XValue,Y + 2);
      END;
    END;
  END;

Procedure Make_Vertical_Scale (Var Scale : ScalePost);
(* Ritar ett vertikalt graderat skalstreck och skriver ut max- och  *)
(* minvärdena för skalan, samt anger nollpositionen för färgstapeln *)
Var
  Length,ZeroPos,LowPos,HighPos : Integer;
  Max,Min,Res,X,Y : Integer;
BEGIN
  X := mm_To_Pix_X (Scale.ScalePos.X);
  Y := mm_To_Pix_Y (Scale.ScalePos.Y);
  Length := mm_To_Pix_Y (Scale.Length);
  HighPos := ROUND (Y - Length / 2);
  LowPos := ROUND (Y + Length / 2);
  Max := Scale.ScaleValue.Max;
  Min := Scale.ScaleValue.Min;
  Res := Scale.ScaleResolution;
  SetColor (Scale.ScaleColor);                          (* Ritar skalstrecket *)
  MoveTo (X - 5,HighPos);
  LineTo (X + 5,HighPos);
  MoveTo (X,HighPos);
  LineTo (X,Lowpos);
  MoveTo (X - 5,LowPos);
  LineTo (X + 5,LowPos);
  OutTextXY (X - 45,Y - 25 - ROUND (Length / 2),       (* Skriver ut texten *)
             Scale.Text.Line1);
  OutTextXY (X - 45,Y - 15 - ROUND (Length / 2),
             Scale.Text.Line2);
  OutTextXY (X + 10,HighPos - 3,Int_To_Str (Max));
  OutTextXY (X + 10,LowPos - 3,Int_To_Str (Min));
  If ((Min < 0) AND (Max > 0)) OR                   (* Beräknar nollpositionen *)
     ((Min > 0) AND (Max < 0)) Then                 (* för färgstapeln         *)
  BEGIN
    ZeroPos := Y;
    MoveTo (X - 5,Y);
    LineTo (X + 5,Y);
    OutTextXY (X + 10,Y - 3,Int_To_Str (0));
  END
  ELSE If ABS (Min) < ABS (Max) Then
```

```
    ZeroPos := LowPos
  ELSE If ABS (Min) > ABS (Max) Then
    ZeroPos := HighPos;
  Ver_Scale_Grad(Length,Res,Max,Min,X,Y,Scale.ScaleType,Scale.ScaleColor);
  Scale.Bar.XPos := X - 5;
  Scale.Bar.YPos := ZeroPos;
  Scale.Mark.XPos := X - 5;
  Scale.Mark.YPos := ZeroPos;
  Scale.Placement := Ver;
  Scale.OldBarValue := 0;
  Scale.ScaleExist := True;
END;

Procedure Make_Horisontel_Scale (Var Scale : ScalePost);
(* Ritar ett horisontellt graderat skalstreck och skriver ut max och min- *)
(* värdena, samt anger nollpositionen för färgstapeln                      *)
Var Length,ZeroPos,LeftPos,RightPos,Max,Min,Res,X,Y : Integer;
BEGIN
  X := mm_To_Pix_X (Scale.ScalePos.X);
  Y := mm_To_Pix_Y (Scale.ScalePos.Y);
  Length := mm_To_Pix_X (Scale.Length);
  LeftPos := ROUND (X - Length / 2);
  RightPos := ROUND (X + Length / 2);
  Res := Scale.ScaleResolution;
  Max:=Scale.ScaleValue.Max;
  Min:=Scale.ScaleValue.Min;
  SetColor (Scale.ScaleColor);                    (* Ritar skalstrecket *)
  MoveTo (LeftPos,Y - 5);
  LineTo (LeftPos,Y + 5);
  MoveTo (LeftPos,Y);
  LineTo (RightPos,Y);
  MoveTo (RightPos,Y - 5);
  LineTo (RightPos,Y + 5);
  OutTextXY (X - 47,Y - 50,Scale.Text.Line1);(* Skriver ut texten *)
  OutTextXY (X - 47,Y - 50 + 10,Scale.Text.Line2);
  OutTextXY(LeftPos - 5,Y + 10,Int_To_Str (Min));
  OutTextXY(RightPos - 5,Y + 10,Int_To_Str (Max));
  If ((Min < 0) AND (Max > 0)) OR                 (* Beräknar nollpositionen *)
     ((Min> 0) AND (Max< 0)) then                 (* för färgstapeln         *)
  BEGIN
    ZeroPos := X;
    MoveTo (X,Y - 5);
    LineTo (X,Y + 5);
    OutTextXY (X - 3,Y + 10,InT_to_Str (0));
  END
  ELSE If ABS (Min) < ABS (Max) Then
    ZeroPos := LeftPos
  ELSE If ABS (Min) > ABS (Max) Then
    ZeroPos := RightPos;
  Hor_Scale_Grad (Length,Res,Max,Min,X,Y,Scale.ScaleType,Scale.ScaleColor);
  Scale.Bar.XPos := ZeroPos;
  Scale.Bar.YPos := Y - 5;
  Scale.Mark.XPos := ZeroPos;
  Scale.Mark.YPos := Y - 5;
  Scale.Placement := Hor;
  Scale.OldBarValue := 0;
  Scale.ScaleExist := True;
END;

Procedure Make_BackGround_And_Frame;
(* Färglägger bakgrunden och ritar ut ram *)
Var K : Integer;
BEGIN
   SetFillStyle (SolidFill,ScreenBackGroundColor);(* Färglägger Bakgrunden *)
   FloodFill (5,5,ScreenBackGroundColor);
   SetColor (FrameColor);
```

```
    Rectangle (0,0,GetMaxX,GetMaxY);                    (* Ram Runt Skärmen *)
    SetColor (black);
    Rectangle (1,1,GetMaxX - 1,GetMaxY - 1);
    SetColor (FrameColor);
    Rectangle (2,2,GetMaxX - 2,GetMaxY - 2);
    SetFillStyle (SolidFill,WindowColor);(* Gör ruta i översta högra hörnet *)
    Bar (360,11,627,32);
    SetColor (WindowShaddowColor);
    MoveTo (364,10); LineTo (630,10);
    MoveTo (364,9); LineTo (630,9);
    MoveTo (364,8); LineTo (630,8);
    MoveTo (628,11); LineTo (628,28);
    MoveTo (629,11); LineTo (629,28);
    MoveTo (630,11); LineTo (630,28);
END;


Procedure Make_HairCross_Scales (HairCross : HairCrossPost);
(* Ritar horisontella och vertikala graderade skalor i cirkeln *)
Var XRad,YRad,XPos,YPos,XValue,YValue,A,Interval :Integer;
BEGIN
  With HairCross Do
  BEGIN
    XRad := mm_To_Pix_X (Radius);
    YRad := mm_to_Pix_Y (Radius);
    XPos := mm_To_Pix_X (Origo.X);
    YPos := mm_To_Pix_Y (Origo.Y);
    SetColor (ScaleColor);
    With XScale Do
    BEGIN
      MoveTo (XPos - XRad,YPos);
      LineTo (XPos + XRad,YPos);
      OutTextXY (XPos - XRad - 35,YPos - 3,
                 Int_To_Str (ScaleValue.Min));
      OutTextXY (XPos + XRad + 5,YPos - 3,
                 Int_To_Str (ScaleValue.Max));
      Hor_Scale_Grad (2 * XRad,ScaleResolution,ScaleValue.Max,ScaleValue.Min,
                      XPos,YPos,ScaleType,HairCross.ScaleColor);
    END;
    With YScale Do
    BEGIN
      MoveTo (XPos,Ypos - YRad);
      LineTo (XPos,YPos + YRad);
      OutTextXY (XPos - 10,YPos - YRad - 10,
                 Int_To_Str (ScaleValue.Max));
      OutTextXY (XPos - 15,YPos + YRad + 5,
                 Int_To_Str (ScaleValue.Min));
      Ver_Scale_Grad (2 * YRad,ScaleResolution,ScaleValue.Max,ScaleValue.Min,
                      XPos,YPos,ScaleType,HairCross.ScaleColor);
    END;
  END;
END;

Procedure Make_HairCross (Var HairCross : HairCrossPost);
(* Ritar en färglagd cirkel med en horisontell och en vertikal skala *)
(* i cirkeln.                                                         *)
Var XRad,YRad,XPos,YPos : integer;
BEGIN
  XRad := mm_To_Pix_X (HairCross.Radius);
  YRad := mm_To_Pix_Y (HairCross.Radius);
  XPos := mm_To_Pix_X (HairCross.Origo.X);
  YPos := mm_To_Pix_Y (HairCross.Origo.Y);
  SetFillStyle (SolidFill,HairCross.BackGroundColor);
  FillEllipse (XPos,YPos,XRad,YRad);
  Make_HairCross_Scales (HairCross);
  HairCross.OldCursorXValue := 0;
```

```
  HairCross.OldCursorYValue := 0;
  HairCross.OldArrowBaseX := 0;
  HairCross.OldArrowBaseY := 0;
  HairCross.OldArrowHeadX := 0;
  HairCross.OldArrowHeadY := 0;
  HairCross.HairCrossExist :=True;
END;

Procedure Make_Digital_Instrument (Var DigitalInstr : DigitalInstrPost);
(* Ritar texten till ett digitalt visande instrument med maximalt *)
(* två visande variabler                                          *)
Var XPos,YPos,K : Integer;
BEGIN
  With DigitalInstr Do
  BEGIN
    XPos := mm_To_Pix_X (TextPos.X);
    YPos := mm_To_Pix_Y (TextPos.Y);
    SetColor (Color);
    OutTextXY (XPos,YPos,Line1);
    OutTextXY (XPos,YPos + 10,Line2);
    OutTextXY (XPos,YPos + 20,Line3);
    For K :=1 To NrOfVar Do
      OldVarValueÄKÅ := 0;
  END;
END;

Procedure Make_Three_Text_Lines (Text : TextPost; Color : Integer);
(* Skriver ut tre rader text på skärmen *)
Var X,Y : Integer;
BEGIN
  With Text Do
  BEGIN
    X := mm_To_Pix_X (TextPos.X);
    Y := mm_To_Pix_Y (TextPos.Y);
    SetColor (Color);
    OutTextXY (X,Y,Line1);
    OutTextXY (X,Y + 10,Line2);
    OutTextXY (X,Y + 20,Line3);
  END;
END;


Procedure Expand_Scale (Var Scale : ScalePost; Magn : Integer;
                        Expand,Compress : Boolean);
Var ScaleLength,XPos,YPos,HighPos,LowPos,ZeroPos,LeftPos,RightPos : Integer;
BEGIN
  With Scale Do
  BEGIN
    MaxGuard := Scalevalue.Max Mod Magn;
    MinGuard := Scalevalue.Min Mod Magn;
    if ((Scalevalue.Max>0) and (Scalevalue.Min<=0)) or
       ((Scalevalue.Max<=0) and (Scalevalue.Min>0)) or
       ((Scalevalue.Max<0) and (Scalevalue.Min=0)) or
       ((Scalevalue.Max=0) and (Scalevalue.Min<0)) Then
    Begin
      If Placement = Ver Then
      BEGIN
        ScaleLength := mm_To_Pix_Y (Length);
        XPos := mm_To_Pix_X (ScalePos.X);
        YPos := mm_To_Pix_Y (ScalePos.Y);
        HighPos := ROUND (YPos - ScaleLength / 2);
        LowPos := ROUND (YPos + ScaleLength / 2);
        Ver_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                        ScaleValue.Min,XPos,Ypos,ScaleType,
                        ScreenBackGroundColor);
        SetColor (ScreenBackGroundColor);
```

```
        OutTextXY (XPos + 10,HighPos - 3,Int_To_Str (ScaleValue.Max));
        OutTextXY (XPos + 10,LowPos - 3,Int_To_Str (ScaleValue.Min));
        if expand then
        begin
          Scalevalue.Max := Scalevalue.Max*Magn;
          Scalevalue.Min := Scalevalue.Min*Magn;
          SetColor (ScaleColor);
          OutTextXY (XPos-45,YPos-25-ROUND(ScaleLength/2),Text.Line1);
          OutTextXY (XPos-45,YPos-15-ROUND(ScaleLength/2),Text.Line2);
        END
        Else if compress  And (MaxGuard=0) And (MinGuard=0) Then
        Begin
          Scalevalue.Max := Scalevalue.Max Div Magn;
          Scalevalue.Min := Scalevalue.Min Div Magn;
        END;
        Ver_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                        ScaleValue.Min,XPos,Ypos,ScaleType,ScaleColor);
        SetColor (ScaleColor);
        MoveTo (XPos,HighPos);
        LineTo (XPos,LowPos);
        OutTextXY (XPos + 10,HighPos - 3,Int_To_Str (ScaleValue.Max));
        OutTextXY (XPos + 10,LowPos - 3,Int_To_Str (ScaleValue.Min));
      End
      ELSE If Placement = Hor Then
      BEGIN
        ScaleLength := mm_To_Pix_X (Length);
        XPos := mm_To_Pix_X (ScalePos.X);
        YPos := mm_To_Pix_Y (ScalePos.Y);
        RightPos := ROUND (XPos + ScaleLength / 2);
        LeftPos := ROUND (XPos - ScaleLength / 2);
        Hor_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                        ScaleValue.Min,XPos,Ypos,ScaleType,
                        ScreenBackGroundColor);
        SetColor (ScreenBackGroundColor);
        OutTextXY (LeftPos - 5,YPos + 10,Int_To_Str (ScaleValue.Min));
        OutTextXY (RightPos - 5,YPos + 10,Int_To_Str (ScaleValue.Max));
        if expand then
        begin
          Scalevalue.Max := Scalevalue.Max*Magn;
          Scalevalue.Min := Scalevalue.Min*Magn;
        END
        Else if compress And (MaxGuard=0) And (MinGuard=0) Then
        begin
          Scalevalue.Max := Scalevalue.Max Div Magn;
          Scalevalue.Min := Scalevalue.Min Div Magn;
        END;
        Hor_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                        ScaleValue.Min,XPos,Ypos,ScaleType,ScaleColor);
        SetColor (ScaleColor);
        MoveTo (LeftPos,YPos);
        LineTo (RightPos,YPos);
        OutTextXY (LeftPos - 5,YPos + 10,Int_To_Str (ScaleValue.Min));
        OutTextXY (RightPos - 5,YPos + 10,Int_To_Str (ScaleValue.Max));
      END;
    END;
    MaxGuard := Scalevalue.Max Mod SQR(Magn);
    MinGuard := Scalevalue.min Mod SQR(Magn);
  end;
END;

Procedure Expand_HairCross (Var HairCross : HairCrossPost; Magn : Integer;
                            Expand,Compress : Boolean);
Var XRad,YRad,XPos,YPos,XValue,YValue,A,Interval :Integer;

BEGIN
  With HairCross Do
```

```
BEGIN
  if XScale.Scalevalue.Max < YScale.Scalevalue.Max then
    MaxGuard := XScale.Scalevalue.Max Mod Magn
  Else
    MaxGuard := YScale.Scalevalue.Max Mod Magn;
  if ((XScale.Scalevalue.Max>0) or (YScale.Scalevalue.Max<=0)) or
     ((XScale.Scalevalue.Max<=0) or (YScale.Scalevalue.Max>0)) or
     ((XScale.Scalevalue.Max<0) or (YScale.Scalevalue.Max=0)) or
     ((XScale.Scalevalue.Max=0) or (YScale.Scalevalue.Max<0)) Then
  Begin
    XRad := mm_To_Pix_X (Radius);
    YRad := mm_to_Pix_Y (Radius);
    XPos := mm_To_Pix_X (Origo.X);
    YPos := mm_To_Pix_Y (Origo.Y);
    SetColor (ScreenBackGroundcolor);
    With XScale Do
    BEGIN
      MoveTo (XPos - XRad,YPos);
      LineTo (XPos + YRad,YPos);
      OutTextXY (XPos - XRad - 35,YPos - 3,
                 Int_To_Str (ScaleValue.Min));
      OutTextXY (XPos + XRad + 5,YPos - 3,
                 Int_To_Str (ScaleValue.Max));
      Hor_Scale_Grad (2 * XRad,ScaleResolution,ScaleValue.Max,
                      ScaleValue.Min,XPos,YPos,ScaleType,BackGroundcolor);
    END;
    SetColor (ScreenBackGroundColor);
    With YScale Do
    BEGIN
      MoveTo (XPos,Ypos - YRad);
      LineTo (XPos,YPos + YRad);
      OutTextXY (XPos - 10,YPos - YRad - 10,
                 Int_To_Str (ScaleValue.Max));
      OutTextXY (XPos - 15,YPos + YRad + 5,
                 Int_To_Str (ScaleValue.Min));
      Ver_Scale_Grad (2 * YRad,ScaleResolution,ScaleValue.Max,
                      ScaleValue.Min,XPos,YPos,ScaleType,BackGroundColor);
    END;
    If Expand Then
    Begin
      With Xscale do
      Begin
        Scalevalue.Max := Scalevalue.Max*Magn;
        Scalevalue.Min := Scalevalue.Min*Magn;
      End;
      With Yscale do
      Begin
        Scalevalue.Max := Scalevalue.Max*Magn;
        Scalevalue.Min := Scalevalue.Min*Magn;
      End;
    End
    Else if Compress And (MaxGuard=0) Then
    Begin
      WIth Xscale do
      Begin
        Scalevalue.Max := Scalevalue.Max Div Magn;
        Scalevalue.Min := Scalevalue.Min Div Magn;
      End;
      WIth Yscale do
      Begin
        Scalevalue.Max := Scalevalue.Max Div Magn;
        Scalevalue.Min := Scalevalue.Min Div Magn;
      End;
    End;
    SetColor (Scalecolor);
    With XScale Do
```

```
      BEGIN
        MoveTo (XPos - XRad,YPos);
        LineTo (XPos + YRad,YPos);
        OutTextXY (XPos - XRad - 35,YPos - 3,
                   Int_To_Str (ScaleValue.Min));
        OutTextXY (XPos + XRad + 5,YPos - 3,
                   Int_To_Str (ScaleValue.Max));
        Hor_Scale_Grad (2 * XRad,ScaleResolution,ScaleValue.Max,ScaleValue.Min
,
                        XPos,YPos,ScaleType,HairCross.Scalecolor);
      END;
      With YScale Do
      BEGIN
        MoveTo (XPos,Ypos - YRad);
        LineTo (XPos,YPos + YRad);
        OutTextXY (XPos - 10,YPos - YRad - 10,
                   Int_To_Str (ScaleValue.Max));
        OutTextXY (XPos - 15,YPos + YRad + 5,
                   Int_To_Str (ScaleValue.Min));
        Ver_Scale_Grad (2 * YRad,ScaleResolution,ScaleValue.Max,ScaleValue.Min
,
                        XPos,YPos,ScaleType,HairCross.ScaleColor);
      END;
    END;
    if XScale.Scalevalue.Max < YScale.Scalevalue.Max then
      MaxGuard := XScale.Scalevalue.Max Mod SQR(Magn)
    Else
      MaxGuard := YScale.Scalevalue.Max Mod SQR(Magn);
  End;
End;

Procedure Move_Scale_Up_Or_Down (Var Scale : ScalePost; Interval : Integer;
                                 Up,Down : Boolean);
Var ScaleLength,XPos,YPos,HighPos,LowPos,ZeroPos,LeftPos,RightPos : Integer;
BEGIN
  With Scale Do
  BEGIN
    If Placement = Ver Then
    BEGIN
      ScaleLength := mm_To_Pix_Y (Length);
      XPos := mm_To_Pix_X (ScalePos.X);
      YPos := mm_To_Pix_Y (ScalePos.Y);
      HighPos := ROUND (YPos - ScaleLength / 2);
      LowPos := ROUND (YPos + ScaleLength / 2);
      Ver_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                      ScaleValue.Min,XPos,Ypos,ScaleType,
                      ScreenBackGroundColor);
      SetColor (ScreenBackGroundColor);
      OutTextXY (XPos + 10,HighPos - 3,Int_To_Str (ScaleValue.Max));
      OutTextXY (XPos + 10,LowPos - 3,Int_To_Str (ScaleValue.Min));
      If Up Then
      BEGIN
        If (ScaleValue.Max > ScaleValue.Min) AND (ScaleValue.Max<>0) Then
        BEGIN
          ScaleValue.Max := ScaleValue.Max + Interval;
          ScaleValue.Min := ScaleValue.Min + Interval;
        END
        ELSE If (ScaleValue.Max<ScaleValue.Min) AND (ScaleValue.Max<>0) Then
        BEGIN
          ScaleValue.Max := ScaleValue.Max - Interval;
          ScaleValue.Min := ScaleValue.Min - Interval;
        END;
      END
      ELSE If Down Then
      BEGIN
        If (ScaleValue.Max > ScaleValue.Min) AND (ScaleValue.Min<>0) Then
```

```
      BEGIN
        ScaleValue.Max := ScaleValue.Max - Interval;
        ScaleValue.Min := ScaleValue.Min - Interval;
      END
      ELSE If (ScaleValue.Max<ScaleValue.Min) AND (ScaleValue.Min<>0) Then
      BEGIN
        ScaleValue.Max := ScaleValue.Max + Interval;
        ScaleValue.Min := ScaleValue.Min + Interval;
      END;
    END;
    Ver_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                    ScaleValue.Min,XPos,Ypos,ScaleType,ScaleColor);
    SetColor (ScaleColor);
    MoveTo (XPos,HighPos);
    LineTo (XPos,LowPos);
    OutTextXY (XPos + 10,HighPos - 3,Int_To_Str (ScaleValue.Max));
    OutTextXY (XPos + 10,LowPos - 3,Int_To_Str (ScaleValue.Min));
  END
  ELSE If Placement = Hor Then
  BEGIN
    ScaleLength := mm_To_Pix_X (Length);
    XPos := mm_To_Pix_X (ScalePos.X);
    YPos := mm_To_Pix_Y (ScalePos.Y);
    RightPos := ROUND (XPos + ScaleLength / 2);
    LeftPos := ROUND (XPos - ScaleLength / 2);
    Hor_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                    ScaleValue.Min,XPos,Ypos,ScaleType,
                    ScreenBackGroundColor);
    SetColor (ScreenBackGroundColor);
    OutTextXY (LeftPos - 5,YPos + 10,Int_To_Str (ScaleValue.Min));
    OutTextXY (RightPos - 5,YPos + 10,Int_To_Str (ScaleValue.Max));
    If Up Then
    BEGIN
      If (ScaleValue.Max > ScaleValue.Min) AND (ScaleValue.Max<>0) Then
      BEGIN
        ScaleValue.Max := ScaleValue.Max + Interval;
        ScaleValue.Min := ScaleValue.Min + Interval;
      END
      ELSE If (ScaleValue.Max<ScaleValue.Min) AND (ScaleValue.Max<>0) Then
      BEGIN
        ScaleValue.Max := ScaleValue.Max - Interval;
        ScaleValue.Min := ScaleValue.Min - Interval;
      END;
    END
    ELSE If Down Then
    BEGIN
      If (ScaleValue.Max > ScaleValue.Min) AND (ScaleValue.Min<>0) Then
      BEGIN
        ScaleValue.Max := ScaleValue.Max - Interval;
        ScaleValue.Min := ScaleValue.Min - Interval;
      END
      ELSE If (ScaleValue.Max<ScaleValue.Min) AND (ScaleValue.Min<>0) Then
      BEGIN
        ScaleValue.Max := ScaleValue.Max + Interval;
        ScaleValue.Min := ScaleValue.Min + Interval;
      END;
    END;
    Hor_Scale_Grad (ScaleLength,ScaleResolution,ScaleValue.Max,
                    ScaleValue.Min,XPos,Ypos,ScaleType,ScaleColor);
    SetColor (ScaleColor);
    MoveTo (LeftPos,YPos);
    LineTo (RightPos,YPos);
    OutTextXY (LeftPos - 5,YPos + 10,Int_To_Str (ScaleValue.Min));
    OutTextXY (RightPos - 5,YPos + 10,Int_To_Str (ScaleValue.Max));
  END;
END;
```

```
END;

Procedure Auto_Scaling (Var Scale : ScalePost; Value : Real);
(* Ändrar max- och minvärdena automatiskt i aktuell skala *)
Var Max,Min : Integer;
BEGIN
  Max := Scale.ScaleValue.Max;
  Min := Scale.ScaleValue.Min;
  If Scale.ValuePresentation = ColoredBar Then
  BEGIN
    If (Max > 0) AND (Min < 0) Then
    BEGIN
      If (Value > Max) OR (Value < Min) Then
        Expand_Scale (Scale,2,True,False);
    END
    ELSE If (Max < 0) AND (Min > 0) Then
    BEGIN
      If (Value < Max) OR (Value > Min) Then
        Expand_Scale (Scale,2,True,False);
    END
    ELSE If Max > Min Then
    BEGIN
      If (Value > Max) OR (Value < Min) Then
        Expand_Scale (Scale,2,True,False);
    END
    ELSE If Max < Min Then
    BEGIN
      If (Value < Max) OR (Value > Min) Then
        Expand_Scale (Scale,2,True,False);
    END;
  END
  ELSE If Scale.ValuePresentation = ColoredMark Then
  BEGIN
    If Max > Min Then
    BEGIN
      If Value > Max Then
        Move_Scale_Up_Or_Down (Scale,ABS((Max-Min) DIV 2),True,False)
      ELSE If Value < Min Then
        Move_Scale_Up_Or_Down (Scale,ABS((Max-Min) DIV 2),False,True);
    END
    ELSE If Max < Min Then
    BEGIN
      If Value < Max Then
        Move_Scale_Up_Or_Down (Scale,ABS((Max-Min) DIV 2),True,False)
      ELSE If Value > Min Then
        Move_Scale_Up_Or_Down (Scale,ABS((Max-Min) DIV 2),False,True);
    END;
  ND;
END;
END.
```

```
UNIT Plot_Set;

INTERFACE

USES Graph,Crt,CREATE,CALC,TYPEVAR,INIT;

Procedure Plot_Ver_Scale_Arrow (ScaleColor,ArrowXPos,ArrowYPos,New,Old :
                                Integer);
Procedure Plot_Hor_Scale_Arrow (ScaleColor,ArrowXPos,ArrowYPos,New,Old :
                                Integer);
Procedure Plot_Bar_In_Scale (Var Scale: ScalePost);
Procedure Plot_Mark_In_Scale (Var Scale : ScalePost);
Procedure Plot_Cursor_Arrow (Var HairCross : HairCrossPost; NewX,NewY,OldX,
                             OldY,OrigoX,OrigoY,CursorRad,Rad : Integer);
Procedure Plot_Cursor_In_HairCross (Var HairCross : HairCrossPost);
Procedure Print_Digital_Instrument_Variable ( Var DigitalInstr :
                                              DigitalInstrPost);
Procedure Erase_Window_Text;
Procedure Select_Scale_F1;
Procedure Select_HairCross_F2;
Procedure Move_Or_Expand_Scale_Upwards_F3;
Procedure Move_Or_Compress_Scale_Downwards_F4;
Procedure Auto_Scaling_F5;
Procedure Set_Flag_F6;
Procedure Set_Flag_F7;
Procedure Set_Flag_F8;
Procedure Set_Flag_F9;
Procedure Set_Flag_F10;
Procedure Check_If_Key_Pressed (Var Stop : Boolean);

IMPLEMENTATION

Procedure Plot_Ver_Scale_Arrow (ScaleColor,ArrowXPos,ArrowYPos,New,Old :
                                Integer);
(* Ritar vertikal tendenspil till respektive skala *)
Var
  K,ArrLength : Integer;
BEGIN
  ArrLength := mm_to_pix_Y (ScaleArrowLength);
  MoveTo (ArrowXPos,ArrowYPos);
  If New > Old Then
  BEGIN
    SetColor (ScreenBackGroundColor);
    LineTo (ArrowXPos,ArrowYPos + ArrLength);
    For k:=1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos - K,ArrowYPos + ArrLength - K);
      LineTo (ArrowXPos + K,ArrowYPos + ArrLength - K);
    END;
    MoveTo (ArrowXPos,ArrowYPos);
    SetColor (ScaleColor);
    LineTo (ArrowXPos,ArrowYPos - ArrLength);
    For K := 1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos - K,ArrowYPos - ArrLength + K);
      LineTo (ArrowXPos + K,ArrowYPos - ArrLength + K);
    END;
  END;
  If New < Old Then
  BEGIN
    SetColor (ScreenBackGroundColor);
    LineTo (ArrowXPos,ArrowYPos - ArrLength);
    For K := 1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos - K,ArrowYPos - ArrLength + K);
      LineTo (ArrowXPos + K,ArrowYPos - ArrLength + K);
```

```
    END;
    MoveTo (ArrowXPos,ArrowYPos);
    SetColor (ScaleColor);
    LineTo (ArrowXPos,ArrowYPos + ArrLength);
    For K := 1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos - K,ArrowYPos + ArrLength - K);
      LineTo (ArrowXPos + K,ArrowYPos + ArrLength - K);
    END;
  END;
End;

Procedure Plot_Hor_Scale_Arrow (ScaleColor,ArrowXPos,ArrowYPos,New,Old :
                                Integer);
(* Ritar horisontell tendenspil i respektive skala *)
Var
  K,ArrLength :integer;
BEGIN
  ArrLength := mm_to_pix_X (ScaleArrowLength);
  MoveTo (ArrowXPos,ArrowYPos);
  If New > Old Then
  BEGIN
    SetColor (ScreenBackGroundColor);
    LineTo (ArrowXPos - ArrLength,ArrowYPos);
    For K :=1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos - ArrLength + K,ArrowYPos - K);
      LineTo (ArrowXPos - ArrLength + K,ArrowYPos + K);
    END;
    MoveTo (ArrowXPos,ArrowYPos);
    SetColor (ScaleColor);
    LineTo (ArrowXPos + ArrLength,ArrowYPos);
    For K := 1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos + ArrLength - K,ArrowYPos - K);
      LineTo (ArrowXPos + ArrLength - K,ArrowYPos + K);
    END;
  END;
  If New < Old Then
  BEGIN
    SetColor (ScreenBackGroundColor);
    LineTo (ArrowXPos + ArrLength,ArrowYPos);
    For K := 1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos + ArrLength - K,ArrowYPos - K);
      LineTo (ArrowXPos + ArrLength - K,ArrowYPos + K);
    END;
    MoveTo (ArrowXPos,ArrowYPos);
    SetColor (ScaleColor);
    LineTo (ArrowXPos - ArrLength,ArrowYPos);
    For K := 1 To 2 Do
    BEGIN
      MoveTo (ArrowXPos - ArrLength + K,ArrowYPos - K);
      LineTo (ArrowXPos - ArrLength + K,ArrowYPos + K);
    END;
  END;
END;



Procedure Plot_Bar_In_Scale (Var Scale: ScalePost);
(* Ritar färgstapeln till aktuell skala *)
Var
  BarWidth,BarLength,ScaleLength,Old,New,ArrowXPos,ArrowYPos : Integer;
  Value : Real;
BEGIN
```

```
Value := Variable (Scale.AVarNr);
Old := Scale.OldBarValue;
If Flag Ä5Å = True Then
  Auto_Scaling (Scale,Value);
With Scale.Bar Do
BEGIN
  If Scale.Placement = Ver Then
  BEGIN
    ScaleLength := mm_To_Pix_Y (Scale.Length);
    BarWidth := mm_To_Pix_Y (Width);
    With Scale Do
    BEGIN
      New := Conv_Value_To_Pix (Value,ScaleValue.Max,
                                ScaleValue.Min,ScaleLength,ScaleType);
    End;
    If Old < 0 Then
    BEGIN
      If New < Old Then
      BEGIN
        SetFillStyle (SolidFill,Color);
        Bar (XPos - BarWidth,YPos - New,XPos,YPos - Old);
      END;
      If (New > Old) AND (New <= 0) Then
      BEGIN
        SetFillStyle (SolidFill,ScreenBackGroundColor);
        Bar (XPos - BarWidth,YPos - Old,XPos,YPos - New);
      END;
      If New > 0 Then
      BEGIN
        SetFillStyle (SolidFill,ScreenBackGroundColor);
        Bar (XPos - BarWidth,YPos - Old,XPos,YPos);
        SetFillStyle (SolidFill,Color);
        Bar (XPos - BarWidth,YPos,XPos,YPos - New);
      END;
    END
    ELSE If Old >= 0 Then
    BEGIN
      If New > Old Then
      BEGIN
        SetFillStyle (SolidFill,Color);
        Bar (XPos - BarWidth,YPos - Old,XPos,YPos - New);
      END;
      If (New < Old) AND (New >= 0) Then
      BEGIN
        SetFillStyle (SolidFill,ScreenBackGroundColor);
        Bar (XPos - BarWidth,YPos - New,XPos,YPos - Old);
      END;
      If New < 0 Then
      BEGIN
        SetFillStyle (SolidFill,ScreenBackGroundColor);
        Bar (XPos - BarWidth,YPos,XPos,YPos - Old);
        SetFillStyle (SolidFill,Color);
        Bar (XPos - BarWidth,YPos - New,XPos,YPos);
      END;
    END;
    ArrowXPos := XPos - BarWidth - 3;
    ArrowYPos := Ypos;
    Plot_Ver_Scale_Arrow (Scale.ScaleColor,ArrowXPos,ArrowYPos,New,Old)
  END;
  If Scale.Placement = Hor Then
  BEGIN
    ScaleLength := mm_To_Pix_X (Scale.Length);
    BarWidth := mm_To_Pix_X (Width);
    With Scale Do
    BEGIN
      New := Conv_Value_To_Pix (Value,ScaleValue.Max,
```

```
                                        ScaleValue.Min,ScaleLength,ScaleType);
      END;
      If Old < 0 Then
      BEGIN
        If New < Old Then
        BEGIN
          SetFillStyle (SolidFill,Color);
          Bar (XPos + New,YPos,XPos + Old,YPos - BarWidth);
        END;
        If (New > Old) AND (New <= 0) Then
        BEGIN
          SetFillStyle (SolidFill,ScreenBackGroundColor);
          Bar (XPos + Old,YPos,XPos + New,YPos - BarWidth);
        END;
        If New > 0 Then
        BEGIN
          SetFillStyle (SolidFill,ScreenBackGroundColor);
          Bar (XPos + Old,YPos,XPos,YPos - BarWidth);
          SetFillStyle (SolidFill,Color);
          Bar (XPos,YPos,XPos + New,YPos - BarWidth);
        END;
      END
      ELSE If Old >= 0 Then
      BEGIN
        If New > Old Then
        BEGIN
          SetFillStyle (SolidFill,Color);
          Bar (XPos + Old,YPos,XPos + New,YPos - BarWidth);
        END;
        If (New < Old) AND (New >= 0) Then
        BEGIN
          SetFillStyle (SolidFill,ScreenBackGroundColor);
          Bar (XPos + New,YPos,XPos + Old,YPos - BarWidth);
        END;
        If New < 0 Then
        BEGIN
          SetFillStyle (SolidFill,ScreenBackGroundColor);
          Bar (XPos,YPos,XPos + Old,YPos - BarWidth);
          SetFillStyle (SolidFill,Color);
          Bar (XPos + New,YPos,XPos,YPos - BarWidth);
        END;
      END;
      ArrowXpos := Xpos;
      ArrowYPos := YPos - BarWidth - 3;
      Plot_Hor_Scale_Arrow (Scale.ScaleColor,ArrowXPos,ArrowYPos,New,Old);
    END;
  END;
  Scale.OldBarValue := New;
END;


Procedure Plot_Mark_In_Scale (Var Scale : ScalePost);
(* Ritar symboler vid skalan för är- och börvärdesmarkeringarna *)
Var
  ScaleLength,OldA,NewA,OldO,NewO,ArrowXPos,ArrowYPos,
  X,Y,HighPos,LowPos,LeftPos,RightPos : Integer;
  AValue,OValue : Real;
BEGIN
  With Scale Do
  BEGIN
    If Placement = Ver Then
      ScaleLength := mm_To_Pix_Y (Length)
    ELSE If Placement = Hor Then
      ScaleLength := mm_To_Pix_X (Length);
    X := mm_to_pix_X (Scale.ScalePos.X);
    Y := mm_to_pix_Y (Scale.ScalePos.Y);
```

```
  AValue := Variable (AVarNr);
  OValue := Variable (OVarNr);
  If FlagÄ5Å = True Then
    Auto_Scaling (Scale,AValue);
  NewA := Conv_Value_To_Pix (AValue,ScaleValue.Max,ScaleValue.Min,
                             ScaleLength,ScaleType);
  NewO := Conv_Value_To_Pix (OValue,ScaleValue.Max,ScaleValue.Min,
                             ScaleLength,ScaleType);
END;
OldA := Scale.OldAValue;
OldO := Scale.OldOValue;
With Scale.Mark Do
BEGIN
  If Scale.Placement = Ver Then
  BEGIN                                        (* Gamla markörerna ritas över*)
    HighPos := ROUND (Y - ScaleLength / 2);
    LowPos  := ROUND (Y + ScaleLength / 2);
    If ((Ypos - OldO) >= HighPos) AND ((YPos - OldO) <= LowPos) Then
    BEGIN
      SetColor (ScreenBackGroundColor);
      MoveTo (XPos - 2,YPos - 1 - OldO);
      LineTo (XPos - 27,YPos - 1 - OldO);
      MoveTo (XPos - 2,YPos + 1 - OldO);
      LineTo (XPos - 27,YPos + 1 - OldO);
    END;
    If ((YPos - OldA) >= HighPos) AND ((YPos - OldA) <= LowPos) Then
    BEGIN
      SetColor (ScreenBackGroundColor);
      MoveTo (XPos - 2,YPos - OldA);
      LineTo (XPos - 12,YPos - OldA - 5);
      LineTo (XPos - 27,YPos - OldA - 5);
      LineTo (XPos - 27,YPos - OldA + 5);
      LineTo (XPos - 12,YPos - OldA + 5);
      LineTo (XPos - 2,YPos - OldA);
    END;                                       (* Nya markörena ritas *)
    If ((YPos - NewO) > HighPos) AND ((YPos - NewO) < LowPos) Then
    BEGIN
      SetColor (OColor);
      MoveTo (XPos - 2,YPos - 1 - NewO);
      LineTo (XPos - 27,YPos - 1 - NewO);
      MoveTo (XPos - 2,YPos + 1 - NewO);
      LineTo (XPos - 27,YPos + 1 - NewO);
    END;
    If ((YPos - NewA) > HighPos) AND ((YPos - NewA) < LowPos) Then
    BEGIN
      SetColor (AColor);
      MoveTo (XPos - 2,YPos - NewA);
      LineTo (XPos - 12,YPos - NewA - 5);
      LineTo (XPos - 27,YPos - NewA - 5);
      LineTo (XPos - 27,YPos - NewA + 5);
      LineTo (XPos - 12,YPos - NewA + 5);
      LineTo (XPos - 2,YPos - NewA);
    END;
    ArrowXPos := X + 20;
    ArrowYPos := Y;
    Plot_Ver_Scale_Arrow (Scale.ScaleColor,ArrowXPos,ArrowYPos,NewA,OldA);
  END
  ELSE if Scale.Placement = Hor then
  BEGIN
    LeftPos := ROUND (X - ScaleLength / 2);
    RightPos := ROUND (X + ScaleLength / 2);
    If ((XPos + OldO) <= RightPos) AND ((XPos + OldO) >= LeftPos) Then
    BEGIN
      SetColor (ScreenBackGroundColor);
      MoveTo (XPos + OldO - 1,YPos - 2);
      LineTo (XPos + OldO - 1,YPos - 24);
```

```
        MoveTo (XPos + OldO + 1,YPos - 2);
        LineTo (XPos + OldO + 1,YPos - 24);
      END;
      If ((XPos + OldA) <= RightPos) AND ((XPos + OldA) >= LeftPos) THEN
      BEGIN
        SetColor (ScreenBackGroundColor);
        MoveTo (XPos + OldA,YPos - 2);
        LineTo (XPos + OldA + 5,YPos - 12);
        LineTo (XPos + OldA + 5,YPos - 24);
        LineTo (XPos + OldA - 5,YPos - 24);
        LineTo (XPos + OldA - 5,YPos - 12);
        LineTo (XPos + OldA,YPos - 2);
      END;                                          (* Nya markörena ritas *)
      If ((XPos + NewO) < RightPos) AND ((XPos + NewO) > LeftPos) Then
      BEGIN
        SetColor (OColor);
        MoveTo (XPos + NewO - 1,YPos - 2);
        LineTo (XPos + NewO - 1,YPos - 24);
        MoveTo (XPos + NewO + 1,YPos - 2);
        LineTo (XPos + NewO + 1,YPos - 24);
      END;
      If ((XPos + NewA) < RightPos) AND ((XPos + NewA) > LeftPos) Then
      BEGIN
        SetColor (AColor);
        MoveTo (XPos + NewA,YPos - 2);
        LineTo (XPos + NewA + 5,YPos - 12);
        LineTo (XPos + NewA + 5,YPos - 24);
        LineTo (XPos + NewA - 5,YPos - 24);
        LineTo (XPos + NewA - 5,YPos - 12);
        LineTo (XPos + NewA,YPos - 2);
      END;
      ArrowXPos := X;
      ArrowYPos := Y + 20;
      Plot_Hor_Scale_Arrow (Scale.ScaleColor,ArrowXPos,ArrowYPos,NewA,OldA);
    END;
  END;
  Scale.OldOValue := NewO;
  Scale.OldAValue := NewA;
END;

Procedure Plot_Cursor_Arrow (Var HairCross : HairCrossPost; NewX,NewY,OldX,
                             OldY,OrigoX,OrigoY,CursorRad,Rad : Integer);
(* Ritar tendenspilen för coursen i cirkeln *)
Var
  CursorArrLength,ArrowBaseX,ArrowBaseY,ArrowHeadX,
  ArrowHeadY : Integer;
  Y,X : real;
BEGIN
  CursorArrLength := mm_To_Pix_X (CursorArrowLength);
  With HairCross Do
  BEGIN
    SetColor (BackGroundColor);                 (* Suddar gamla tendenspilen *)
    If  SQRT ( SQR (OldArrowHeadX-origoX) + SQR (OldArrowHeadY-origoY)) <
        (Rad-2) Then
    BEGIN
      MoveTo (OldArrowBaseX,OldArrowBaseY);
      LineTo (OldArrowHeadX,OldArrowHeadY);
      Circle (OldArrowHeadX,OldArrowHeadY,1);
    END;
    Sample := Sample + 1;
    If Sample = 2 Then
    BEGIN
      Sample := 0;
      Y := NewY - OldY;                         (* Ritar nya tendenspilen *)
      X := NewX - OldX;
      If X = 0 Then
```

```
      BEGIN
        If Y > 0 Then
          CursorAlfa := Pi/2
        ELSE If Y < 0 Then
          CursorAlfa := -Pi/2
        ELSE If Y = 0 Then
          CursorAlfa := Maxint;
      END
      ELSE If X > 0 Then
        CursorAlfa := ARCTAN (Y / X)
      ELSE If X < 0 Then
        CursorAlfa := ARCTAN (Y / X) - Pi;
    END;
    If CursorAlfa <> Maxint Then
    BEGIN
      ArrowBaseX := ROUND (OrigoX + NewX + CursorRad * Cos (CursorAlfa));
      ArrowBaseY := ROUND (OrigoY - NewY - CursorRad * Sin (CursorAlfa));
      ArrowHeadX := ROUND (ArrowBaseX + CursorArrLength *Cos (CursorAlfa));
      ArrowHeadY := ROUND (ArrowBaseY - CursorArrLength *Sin (CursorAlfa));
    END
    ELSE If CursorAlfa = Maxint Then
    BEGIN
      ArrowBaseX := OrigoX + NewX;
      ArrowBaseY := OrigoY - NewY;
      ArrowHeadX := ArrowBaseX;
      ArrowHeadY := ArrowBaseY;
    END;
    If  SQRT (SQR (ArrowHeadX-OrigoX)+SQR (ArrowHeadY-OrigoY)) < (Rad-2) Then
    BEGIN
      SetColor (CursorColor);
      MoveTo (ArrowBaseX,ArrowBaseY);
      LineTo (ArrowHeadX,ArrowHeadY);
      Circle (ArrowHeadX,ArrowHeadY,1);
    END;
    OldArrowBaseX := ArrowBaseX;
    OldArrowBaseY := ArrowBaseY;
    OldArrowHeadX := ArrowHeadX;
    OldArrowHeadY := ArrowHeadY;
  END;
END;

Procedure Plot_Cursor_In_HairCross (Var HairCross : HairCrossPost);
(* Ritar coursen i aktuell cirkel *)
Var
  XScaleLength,YScalelength,NewX,NewY,OldX,OldY,XRad,Yrad,CursorXRad,
  CursorYRad,NewHyp,OldHyp,MaxHyp,OutX,OutY,OrigoX,OrigoY  : Integer;
  XValue,YValue : Real;
BEGIN
  With HairCross Do
  BEGIN
    OrigoX := mm_To_Pix_X (Origo.X);
    OrigoY := mm_To_Pix_Y (Origo.Y);
    XScaleLength := mm_To_Pix_X (Radius*2);
    YScaleLength := mm_To_Pix_Y (Radius*2);
    XRad := mm_To_Pix_X (Radius);
    YRad := mm_To_Pix_Y (radius);
    XValue := Variable (XScale.VarNr);
    YValue := Variable (YScale.VarNr);
    NewX := Conv_Value_To_Pix (XValue,XScale.ScaleValue.Max,XScale.
                               ScaleValue.Min,XScaleLength,XScale.ScaleType);
    NewY := Conv_Value_To_pix (YValue,YScale.ScaleValue.Max,YScale.
                               ScaleValue.Min,YScaleLength,YScale.ScaleType);
    OldX := OldCursorXValue;
    OldY := OldCursorYValue;
    CursorXRad := mm_To_Pix_X (CursorRadius);
    CursorYRad := mm_To_Pix_Y (Cursorradius);
```

```
    MaxHyp := ROUND (XRad - CursorXRad - 2);
    NewHyp := ROUND (SQRT (SQR (NewX) + SQR (NewY)));
    If (SQR (OldX) + SQR (OldY)) < SQR (MaxHyp) Then
    BEGIN
      SetColor (BackGroundColor);
      Ellipse (OrigoX+OldX,OrigoY-OldY,0,360,CursorXRad,CursorYRad);
      Plot_Cursor_Arrow (HairCross,NewX,NewY,OldX,OldY,OrigoX,OrigoY,
                         CursorXRad,XRad);
    END
    ELSE
    BEGIN
      SetColor (BackGroundColor);
      OldHyp := ROUND (SQRT (SQR (OldX) + SQR (OldY)));
      OutX := ROUND (OldX * (MaxHyp / OldHyp));
      OutY := ROUND (OldY * (MaxHyp / OldHyp));
      Ellipse (OrigoX+OutX,OrigoY-OutY,0,360,CursorXRad,CursorYRad);
    END;
    If NewHyp < MaxHyp Then
    BEGIN
      SetColor (CursorColor);
      Ellipse (OrigoX+NewX,OrigoY-NewY,0,360,CursorXRad,CursorYRad);
    END
    ELSE
    BEGIN
      SetColor (CursorColor);
      OutX := ROUND (NewX * (MaxHyp / NewHyp));
      OutY := ROUND (NewY * (MaxHyp / NewHyp));
      Ellipse (OrigoX+OutX,OrigoY-OutY,0,360,CursorXRad,CursorYRad);
    END;
    Make_HairCross_Scales (HairCross);
    OldCursorXValue := NewX;
    OldCursorYValue := NewY;
    If Flag Ä5Å = True Then
    BEGIN
      If NewHyp > MaxHyp Then
      BEGIN
        Expand_HairCross (HairCross,2,True,False);
        NewX := Conv_Value_To_Pix (XValue,XScale.ScaleValue.Max,XScale.
                                   ScaleValue.Min,XScaleLength,XScale.ScaleType);
        NewY := Conv_Value_To_pix (YValue,YScale.ScaleValue.Max,YScale.
                                   ScaleValue.Min,YScaleLength,YScale.ScaleType);
      END;
    END;
  END;
END;

Procedure Print_Digital_Instrument_Variable ( Var DigitalInstr :
                                              DigitalInstrPost);
(* Skriver ut variabelvärdet i det aktuella instrumentet *)
Var
  Xpos,YPos,K : Integer;
BEGIN
  With DigitalInstr Do
  BEGIN
    XPos := mm_To_Pix_X (TextPos.X);
    YPos := mm_To_Pix_Y (TextPos.Y);
    For K := 1 To NrOfVar Do
    BEGIN
      SetColor (ScreenBackGroundColor);
      OutTextXY (XPos+30,YPos+10+(K-1)*10,Real_To_Str (OldVarValueÄkÅ));
      SetColor (Color);
      OutTextXY (XPos+30,YPos+10+(K-1)*10,Real_To_Str (Variable(VarNrÄkÅ)));
      OldVarValueÄKÅ := Variable (VarNrÄKÅ);
    END;
  END;
END;
```

```
Procedure Erase_Window_Text;
BEGIN
  SetColor (WindowColor);
  If Flag Ä1Å = True Then
  BEGIN
    OutTextXY (370,13,'Selected Scale:');
    OutTextXY (518,13,ScaleÄOldScaleNumberÅ.Text.Line1);
    OutTextXY (518,23,ScaleÄOldScaleNumberÅ.Text.Line2);
  END
  ELSE If Flag Ä2Å = True Then
  BEGIN
    OutTextXY (370,13,'Selected HairCross:');
    OutTextXY (530,13,Int_To_Str(OldHairCrossNumber));
  END
  ELSE If Flag Ä5Å = True Then
  BEGIN
    OutTextXY (387,18,'A U T O S C A L I N G  ! !');
  END;
END;

Procedure Select_Scale_F1;
(* Val av skalnummer *)
BEGIN
  Erase_Window_Text;
  Repeat
    ScaleNumber := ScaleNumber + 1;
  Until (ScaleNumber=NrOfScales) Or (ScaleÄScaleNumberÅ.ScaleExist = True);
  If ScaleÄScaleNumberÅ.ScaleExist = True Then
  BEGIN
    SetColor(White);
    OutTextXY (370,13,'Selected Scale:');
    OutTextXY (518,13,ScaleÄScaleNumberÅ.Text.Line1);
    OutTextXY (518,23,ScaleÄScaleNumberÅ.Text.Line2);
    OldScaleNumber := ScaleNumber;
    FlagÄ1Å:=True;
    FlagÄ2Å:=False;
    FlagÄ5Å:=False;
  END;
  If ScaleNumber = NrOfScales Then
    ScaleNumber := 0;
END;

Procedure Select_HairCross_F2;
(* Val av hårkorsnummer *)
BEGIN
  Erase_Window_Text;
  Repeat
    HairCrossNumber := HairCrossNumber + 1;
  Until (HairCrossNumber = NrOfHairCross) OR
        (HairCrossÄHairCrossNumberÅ.HairCrossExist = True);
  If HairCrossÄHairCrossNumberÅ.HairCrossExist = True Then
  BEGIN
    SetColor(White);
    OutTextXY (370,13,'Selected HairCross:');
    OutTextXY (530,13,Int_To_Str(OldHairCrossNumber));
    OldHairCrossNumber := HairCrossNumber;
    FlagÄ1Å:=False;
    FlagÄ2Å:=True;
    FlagÄ5Å:=False;
  END;
  If HairCrossNumber = NrOfHairCross Then
    HairCrossNumber := 0;
END;

Procedure Move_Or_Expand_Scale_Upwards_F3;
```

```
(* Manuell skalning uppåt *)
BEGIN
  If Flag Ä1Å = True then
  BEGIN
    If ScaleÄScaleNumberÅ.ValuePresentation = ColoredMark Then
      Move_Scale_Up_Or_Down (ScaleÄScaleNumberÅ,2,True,False)
    ELSE If ScaleÄScaleNumberÅ.ValuePresentation = ColoredBar Then
      Expand_Scale (ScaleÄScaleNumberÅ,2,True,False);
  END
  ELSE If Flag Ä2Å = True Then
    Expand_HairCross (HairCrossÄHairCrossNumberÅ,2,true,false);
END;

Procedure Move_Or_Compress_Scale_Downwards_F4;
(* Manuell skalning neråt *)
BEGIN
  If Flag Ä1Å = True Then
  BEGIN
    If ScaleÄScaleNumberÅ.ValuePresentation = ColoredMark Then
      Move_Scale_Up_Or_Down (ScaleÄScaleNumberÅ,2,False,True)
    ELSE If ScaleÄScaleNumberÅ.ValuePresentation = ColoredBar Then
      Expand_Scale (ScaleÄScaleNumberÅ,2,False,True);
  END
  ELSE If Flag Ä2Å = True Then
    Expand_HairCross (HairCrossÄHairCrossNumberÅ,2,false,true);
EN

Procedure Auto_Scaling_F5;
(* Auto Scaling *)
BEGIN
  Erase_Window_Text;
  If FlagÄ5Å = False then
  BEGIN
    SetColor (White);
    OutTextXY (387,18,'A U T O S C A L I N G  ! !');
    FlagÄ1Å:=False;
    FlagÄ2Å:=False;
    FlagÄ5Å := True;
  END
  ELSE if FlagÄ5Å = True then
  BEGIN
    FlagÄ5Å := False;
  ND;
END;

Procedure Set_Flag_F6;
(* AutoDepth *)
Be n
   If FlagÄ6Å = False then
   BEGIN
     Make_Three_Text_Lines (TextÄ1Å,TextÄ1Å.TextColor);
     FlagÄ6Å :=True
   END
   ELSE if FlagÄ6Å = True then
   BEGIN
     Make_Three_Text_Lines (TextÄ1Å,ScreenBackGroundColor);
     FlagÄ6Å := False;
   END;
END;

Procedure Set_Flag_F7;
(* AutoHeading *)
Begin
   If FlagÄ7Å = False then
   BEGIN
     Make_Three_Text_Lines (TextÄ2Å,TextÄ2Å.TextColor);
```

```
      FlagÄ7Å :=True
    END
    ELSE if FlagÄ7Å = True then
    BEGIN
      Make_Three_Text_Lines (TextÄ2Å,ScreenBackGroundColor);
      FlagÄ7Å := False;
    END;
END;

Procedure Set_Flag_F8;
BEGIN
    If FlagÄ8Å = False then
    BEGIN
      FlagÄ8Å :=True
    END
    ELSE if FlagÄ8Å = True then
    BEGIN
      FlagÄ8Å := False;
    END;
END;

Procedure Set_Flag_F9;
BEGIN
    If FlagÄ9Å = False then
    BEGIN
      FlagÄ9Å :=True
    END
    ELSE if FlagÄ9Å = True then
    BEGIN
      FlagÄ9Å := False;
    END;
END;


Procedure Set_Flag_F10;
BEGIN
    If FlagÄ10Å = False then
    BEGIN
      FlagÄ10Å :=True
    END
    ELSE if FlagÄ10Å = True then
    BEGIN
      FlagÄ10Å := False;
    END;
END;


Procedure Check_If_Key_Pressed (Var Stop : Boolean);
BEGIN
    If KeyPressed Then
    BEGIN
      Key := ReadKey;
      If Key = Chr (32) then
      BEGIN
        Stop := True;
        CloseGraph;
      END
      ELSE If Key = Chr (59) Then
        Select_Scale_F1
      ELSE If Key = Chr (60) Then
        Select_HairCross_F2
      ELSE If Key = Chr (61) Then
        Move_Or_Expand_Scale_Upwards_F3
      ELSE If Key = Chr (62) Then
        Move_Or_Compress_Scale_Downwards_F4
      ELSE If Key = Chr (63) Then
```

```
      Auto_Scaling_F5
    ELSE If Key = Chr (64) Then
      Set_Flag_F6
    ELSE If Key = Chr (65) Then
      Set_flag_F7
    ELSE If Key = Chr (66) Then
      Set_Flag_F8
    ELSE If Key = Chr (67) Then
      Set_Flag_F9
    ELSE If Key = Chr (68) Then
      Set_Flag_F10;
  END;
END;
END.
```

```
UNIT CALC;

INTERFACE

USES Graph,INIT,TYPEVAR;

Function Int_To_Str (i : longint) : String;
Function Real_To_Str (i : Real ) : String;
Function mm_To_Pix_X (A : Real) : Integer;
Function mm_To_Pix_Y (b : real) : Integer;
Function F ( Value : Real; Graduation : GradType) : Real;
Function Variable (Nr : Integer) : Real;
Function Conv_Value_To_Pix (Value:Real; Max,Min,ScaleLength : Integer;
                            Grad:GradType) : Integer;

IMPLEMENTATION

Function Int_To_Str (i : longint) : String;
(* Omvandlar en heltalstyp till en sträng med maximalt 10 tecken *)
var
  s : stringÄ10Å;
BEGIN
  Str(i,s);
  Int_To_Str := s;
END;

Function Real_To_Str (i : Real ) : String;
(* Omvandlar ett reellt tal till en sträng *)
Var
  s : stringÄ10Å;
BEGIN
  Str(i:1:1,s);
  Real_To_Str := s;
END;

Function mm_To_Pix_X (A : Real) : Integer;
(* Omvandlar en längd i X-led i mm till antalet punkter på skärmen *)
BEGIN
  mm_To_Pix_X := ROUND (A*(ScaleFactorX));
END;

Function mm_To_Pix_Y (b : real) : Integer;
(* Omvandlar en längd i Y-led i mm till antalet punkter på skärmen *)
Begin
  mm_to_pix_y := ROUND (B*(ScalefactorY));
End;

Function F ( Value : Real; Graduation : GradType) : Real;
(* F tilldelas värdet av F(value) där 'graduation' anger funktionen *)
BEGIN
  If Graduation = Lin Then                                  (* Linjär   y=x *)
    F := Value
  ELSE If Graduation = Log Then                  (* logaritmisk  y=ln(x+1) *)
    F := LN (Value+1)
  ELSE If Graduation = x2 Then                          (* kvadratisk  y=x2 *)
    F:= SQR (Value)
  ELSE If Graduation = ArcTn Then                    (* arctan  y=arctan(x) *)
    F:= ARCTAN (value)
  ELSE If Graduation = Own Then
                    ;          (* Här kan egen godtycklig funktion väljas *)
END;


Function Variable (Nr : Integer) : Real;
(* Läser in ett variabelvärde från variabelvektorn *)
BEGIN
```

```
  Variable := VariableVectorÄNrÅ;
END;

Function Conv_Value_To_Pix (Value:Real; Max,Min,ScaleLength : Integer;
                            Grad:GradType) : Integer;
(* Omvandlar ett variabelvärde till att passa skalans värde på skärmen *)
(* Graderingen på skalan kan vara linjär eller logaritmisk             *)
Var
  New : Integer;
  PosFactor,NegFactor : Real;
BEGIN
  PosFactor := 0;
  NegFactor := 0;
  If (Max > 0) AND (Min < 0) Then
  BEGIN
    If Value >= 0 Then
      PosFactor := F (Value,Grad) / F (Max,Grad)
    ELSE If Value < 0 Then
      NegFactor := F (ABS (Value),Grad) / F (ABS (Min),Grad);
  END
  ELSE If (Max < 0) AND (Min > 0) Then
  BEGIN
    If Value >= 0 Then
      NegFactor := F (Value,Grad) / F (Min,Grad)
    ELSE If Value < 0 Then
      PosFactor := F (ABS(Value),Grad) / F (ABS(Max),Grad);
  END
  ELSE If ABS (Max) > ABS (Min) Then
  BEGIN
    If Max > 0 Then
    BEGIN
      If Value >= 0 Then
        PosFactor := (F (Value,Grad) - F (Min,Grad)) /
                     (F (Max,Grad) - F (Min,Grad));
    END
    ELSE If Max < 0 Then
    BEGIN
      If Value <= 0 Then
        PosFactor := (F (ABS(Value),Grad) - F (ABS(Min),Grad)) /
                     (F (ABS(Max),Grad) - F (ABS(Min),Grad));
    END;
  END
  ELSE If ABS (Max) < ABS (Min) Then
  BEGIN
    If Min > 0 Then
    BEGIN
      If Value >= 0 Then
        NegFactor := (F (Value,Grad) - F (Max,Grad)) /
                     (F (Min,Grad) - F (Max,Grad));
    END
    ELSE If Min < 0 Then
    BEGIN
      If Value <= 0 Then
        NegFactor := (F (ABS(Value),Grad) - F (ABS(Max),Grad)) /
                     (F (ABS(Min),Grad) - F (ABS(Max),Grad));
    END;
  END;
  If (Max > 0) AND (Min < 0) Then
  BEGIN
    If Value >= 0 Then
      New := ROUND (PosFactor * ScaleLength / 2)
    ELSE If Value < 0 Then
      New := -ROUND (NegFactor * ScaleLength / 2);
  END
  ELSE If (Max < 0) AND (Min > 0) Then
  BEGIN
```

```
    If Value >= 0 Then
      New := -ROUND (NegFactor * ScaleLength / 2)
    ELSE If Value < 0 Then
      New := ROUND (PosFactor * ScaleLength / 2);
  END
  ELSE If ABS (Max) > ABS (Min) Then
    New := ROUND (PosFactor * ScaleLength)
  ELSE If ABS (Max) < ABS (Min) Then
    New := -ROUND (NegFactor * ScaleLength);
  Conv_Value_To_Pix := New;
END;
END.
```

```
UNIT INIT;

INTERFACE

USES Crt,Graph,TYPEVAR;

Procedure Parameter_Initialization;
Procedure Graph_Initialization;
Procedure Instrument_Initialization;
Procedure Screen_Color_Initialization;

IMPLEMENTATION

Procedure Parameter_Initialization;
(* Initierar startvärden för ett antal parametrar *)
Var K : Integer;
BEGIN
  For K := 1 To NrOfScales Do
    Scale ÄKÅ.ScaleExist := False;
  For K := 1 To NrOfHairCross Do
    HairCross ÄKÅ.HairCrossExist := False;
  ScaleNumber := 0;
  OldScaleNumber := 1;
  HairCrossNumber := 0;
  OldHairCrossNumber :=1;
  Sample := 0;
  CursorAlfa := Maxint;
  For K := 1 To 10 Do
    Flag ÄKÅ := False;
  Stop := False;
END;

Procedure Graph_Initialization;
(* Initierar grafiken *)
Var  Mode : Integer;
Begin
   GraphDriver := Detect;                              (* GrafikInitiering *)
   InitGraph (GraphDriver,GraphMode,' ');
   ErrorCode := GraphResult;
   IF GraphResult <> GrOk THEN
   BEGIN
      Writeln ('Graphics Error:', GraphErrorMsg(ErrorCode));
      Writeln ('Program Aborted...');
      Halt (1);
   END;
   Mode := GetGraphMode;
   If GraphDriver = VGA Then                  (* Skalfaktor till 'mmToPix' *)
   BEGIN                                      (* för VGA-grafik            *)
     Black        := 0;
     Blue         := 1;
     Green        := 2;
     Cyan         := 3;
     Red          := 4;
     Magenta      := 5;
     Brown        := 6;
     LightGray    := 7;
     DarkGray     := 8;
     LightBlue    := 9;
     LightGreen   := 10;
     LightCyan    := 11;
     LightRed     := 12;
     LightMagenta := 13;
     Yellow       := 14;
     White        := 15;
     if Mode = VGALo Then
     BEGIN
```

```
      ScaleFactorX := 640 / 240;
      ScaleFactorY := 200 / 180;
    END
    ELSE If Mode = VGAMed Then
    BEGIN
      ScaleFactorX := 640 / 240;
      ScaleFactorY := 350 / 180;
    END
    ELSE If Mode = VGAHi Then
    BEGIN
      ScaleFactorX := 640 / 240;
      ScaleFactorY := 480 / 180;
    END;
  END
  ELSE If GraphDriver = EGA Then              (* Skalfaktor till 'mmToPix' *)
  BEGIN                                       (* för EGA-grafik            *)
    Black        := 0;
    Blue         := 1;
    Green        := 2;
    Cyan         := 3;
    Red          := 4;
    Magenta      := 5;
    Brown        := 20;
    LightGray    := 7;
    DarkGray     := 56;
    LightBlue    := 57;
    LightGreen   := 58;
    LightCyan    := 59;
    LightRed     := 60;
    LightMagenta := 61;
    Yellow       := 62;
    White        := 63;
    If Mode = EGALo Then
    BEGIN
      ScaleFactorX := 640 / 240;
      ScaleFactorY := 200 / 180;
    END
    ELSE If Mode = EGAHi Then
    BEGIN
      ScaleFactorX := 640 / 240;
      ScaleFactorY := 350 / 180;
    END;
  END
  ELSE If GraphDriver = HERCMONO Then
    If Mode = HercMonoHi Then
    BEGIN
      ScalefactorX := 720 / 240;
      ScaleFactorY := 348 / 180;
    END;
  SetWriteMode (CopyPut);
END;

Procedure Instrument_Initialization;
(* Initierar värdena till skalorna, cirklarna, och texterna *)
BEGIN
  With Scale Ä1Å Do                           (* Initiering Av Skala 1 *)
  BEGIN
    Length := 80;
    ScaleColor := White;
    AVarNr := 1;
    OVarNr := 7;
    ScaleType := Lin;
    ScaleResolution := 20;
    ScalePos.X := 60;
    ScalePos.Y := 80;
    ScaleValue.Max := 20;
```

```
    ScaleValue.Min := 0;
    Text.Line1 := '   Speed    ';
    Text.Line2 := '   knot     ';
    ValuePresentation := ColoredMark;
    Mark.AColor := Yellow;
    Mark.OColor := white;
  END;
  With Scale Ä2Å Do                              (*Initiering av skala 2 *)
  BEGIN
    Length := 80;
    ScaleColor := White;
    AVarNr := 2;
    ScaleType := Lin;
    ScaleResolution := 20;
    ScalePos.X := 120;
    ScalePos.Y := 145;
    ScaleValue.Max := 30;
    ScaleValue.Min := -30;
    Text.Line1 := 'Rudder Angle';
    Text.Line2 := '    deg     ';
    ValuePresentation := ColoredBar;
    Bar.Color := green;
    Bar.Width := 8;
  END;
  With Scale Ä3Å Do                             (* Initiering av skala 3 *)
  BEGIN
    Length := 80;
    ScaleColor := White;
    AVarNr := 3;
    ScaleType := Log;
    ScaleResolution := 10;
    ScalePos.X := 180;
    ScalePos.Y := 80;
    ScaleValue.Max := 5;
    ScaleValue.Min := -5;
    Text.Line1 := ' Turn Rate  ';
    Text.Line2 := '   deg/s    ';
    ValuePresentation := ColoredBar;
    Bar.Color := Blue;
    Bar.Width := 8;
  END;
  With HairCross Ä1Å Do                          (* Initiering av cirkel 1 *
)
  BEGIN
    Radius := 35;
    ScaleColor := White;
    BackGroundColor := Black;
    Origo.X := 120;
    Origo.Y := 80;
    With XScale Do
    BEGIN
      VarNr := 4;
      ScaleValue.Max := 20;
      ScaleValue.Min := -20;
      ScaleResolution := 10;
      ScaleType := log;
    END;
    With YScale Do
    BEGIN
      VarNr := 5;
      ScaleValue.Max := 40;
      ScaleValue.Min := -40;
      ScaleResolution := 10;
      ScaleType := Log;
    END;
    CursorColor := lightcyan;
```

```
    END;
    With DigitalInstrÄ1Å Do          (* Initierar det digitala instrumentet 1 *)
    BEGIN
      VarNrÄ1Å := 1;
      TextPos.X := 45;
      TextPos.Y := 160;
      NrOfVar := 1;
      Color := White;
      Line1 := '  Speed      ';
      Line2 := '             ';
      Line3 := '             ';
    END;
    With DigitalInstrÄ2Å Do          (* Initierar det digitala instrumentet 2 *)
    BEGIN
      VarNrÄ1Å := 2;
      TextPos.X := 90;
      TextPos.Y := 160;
      NrOfVar := 1;
      Color := White;
      Line1 := 'Rudder Angle';
      Line2 := '             ';
      Line3 := '             ';
    END;
    With DigitalInstrÄ3Å Do          (* Initierar det digitala instrumentet 3 *)
    BEGIN
      VarNrÄ1Å := 3;
      TextPos.X := 166;
      TextPos.Y := 160;
      NrOfVar := 1;
      Color := White;
      Line1 := ' Turn Rate   ';
      Line2 := '             ';
      Line3 := '             ';
    END;
END;

Procedure Screen_Color_Initialization;
BEGIN
  ScreenBackGroundColor := Red;
  FrameColor := LightGray;
  WindowColor := DarkGray;
  WindowShaddowColor := black;
E );
END.
```

```
UNIT TYPEVAR;

INTERFACE

CONST
  (* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
  (*        Följande sexton färger finns tillgängliga:                 *)
  (*                                                                   *)
  (*        Black, Blue, Green, Cyan, Red, Magenta, Brown, LightGray,  *)
  (*        DarkGray, LightBlue, LightGreen, LightCyan, LightRed,      *)
  (*        LightMagenta, Yellow, White                                *)
  (* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)

                              (* * * * * * * * * * * * * * * * * * * * *)
  CursorRadius = 2.0;         (*  Val av radie på coursen               *)
  CursorArrowLength = 3.0;    (*  Val av tendenspilens längd för coursorn *)
  ScaleArrowLength = 5.0;     (*  Val av tendenspilens längd för skalorna *)
                              (*  Alla mått anges i millimeter           *)
                              (*                                         *)
  NrOfScales = 10;            (*  Maximala antalet skalor på skärmen     *)
  NrOfHairCross = 3;          (*  Maximala antalet cirklar på skärmen    *)
  NrOfTexts = 20;             (*  Maximala antalet textposter på skärmen *)
  NrOfInstr = 20;             (*  Maximala antalet dig.instr. på skärmen *)
  NrOfVariables = 40;         (*  Maximala antalet variabler i var.vektorn *)
                              (* * * * * * * * * * * * * * * * * * * * *)

TYPE
  GradType = (Lin,Log,x2,ArcTn,Own);
  PlacementType = (Ver,Hor);
  PlotType = (ColoredBar,ColoredMark);
  LineType = Packed Array Ä1..12Å Of Char;
  ScalePost = Record
                Length : Integer;
                ScaleColor : Integer;
                AVarNr : Integer;
                OVarNr : Integer;
                ScaleType : GradType;
                Placement : PlacementType;
                ScaleResolution : Integer;
                ScalePos : Record
                             X,Y : Integer;
                           End;
                ScaleValue : Record
                               Max,Min : Integer;
                             End;
                Text : Record
                         Line1,Line2 : Linetype;
                       End;
                ValuePresentation : PlotType;
                Bar : Record
                        Color : Integer;
                        Width : Integer;
                        XPos,YPos : Integer;
                      End;
                Mark : Record
                         AColor : Integer;
                         OColor : Integer;
                         XPos,YPos : Integer;
                       End;
                OldBarValue,OldAValue,OldOValue : Integer;
                ScaleExist : Boolean;
              End;

  HairCrossPost = Record
                    Radius : Integer;
                    ScaleColor : Integer;
```

```
                BackGroundColor : Integer;
                Origo : Record
                          X,Y :Integer;
                        End;
                XScale : Record
                           VarNr : Integer;
                           ScaleValue : Record
                                          Max,Min : Integer;
                                        End;
                           ScaleResolution : Integer;
                           ScaleType : GradType;
                         End;
                YScale : Record
                           VarNr : Integer;
                           ScaleValue : Record
                                          Max,Min : Integer;
                                        End;
                           ScaleResolution : Integer;
                           ScaleType : GradTYpe;
                         End;
                CursorColor : Integer;
              OldCursorXValue, OldCursorYValue, OldArrowBaseX,
              OldArrowBaseY, OldArrowHeadX, OldArrowHeadY : Integer;
              HairCrossExist : Boolean;
            End;
  DigitalInstrPost = Record
                       VarNr : Array Ä1..2Å Of Integer;
                       TextPos : Record
                                   X,Y : Integer;
                                 End;
                       NrOfVar : Integer;
                       OldVarValue : ArrayÄ1..2Å Of Real;
                       Color : Integer;
                       Line1,Line2,Line3 : LineType;
                     End;
  TextPost = Record
               TextPos : Record
                           X,Y : Integer;
                         End;
               Line1,Line2,Line3 : LineType;
               TextColor : Integer;
             End;

  ScaleVectorType = Array Ä1..NrOfScalesÅ Of ScalePost;
  HairCrossVectorType = Array Ä1..NrOfHairCrossÅ Of HairCrossPost;
  TextVectorType = Array Ä1..NrOfTextsÅ Of TextPost;
  DigitalInstrVectorType = Array Ä1..NrOfInstrÅ Of DigitalInstrPost;
  VariableVectorType = Array Ä1..NrOfVariablesÅ Of Real;

VAR
  Scale : ScaleVectorType;
  HairCross : HairCrossVectorType;
  DigitalInstr : DigitalInstrVectorType;
  Text : TextVectorType;
  VariableVector : VariableVectorType;
  GraphDriver,GraphMode,ErrorCode  : Integer;
  d,ScaleFactorX,ScaleFactorY,CursorAlfa : Real;
  oeka,minska,stop : boolean;
  HairCrossExist : Array Ä1..NrOfHairCrossÅ Of Boolean;
  ScaleNumber,OldScaleNumber,HairCrossNumber,OldHairCrossNumber : Integer;
  MaxGuard,MinGuard,Sample : Integer;
  Flag : Array Ä1..10Å of boolean;
  Key : Char;
  ScreenBackGroundColor,FrameColor,WindowColor,WindowShaddowColor : Integer;
  black,blue,green,cyan,red,magenta,brown,lightgray,darkgray,lightblue,
  lightgreen,lightcyan,lightred,lightmagenta,yellow,white : Integer;
```

```
IMPLEMENTATION
END.
```

Referenser

1 Micael Edler och Fredrik Holst: Farkoststyrning och förarmiljö, LTH Augusti 1989

2 Blue Ship Technology, Data Acquisition and Control Cards 1990

3 Visitron Green Pot Midori Precisions CO., LTD 1990

4 Leo L. Beranek: Acoustics The American Institute of Physics Inc 1986

5 Borland International: Turbo Pascal 1989