CODEN:LUTFD2/(TFRT-5329)/01-081/(1985)

LOGGNINGSPROGRAM FÖR μMAC-5000

PER NILSSON

INSTITUTIONEN FÖR REGLERTEKNIK
LUNDS TEKNISKA HÖGSKOLA
SEPTEMBER 1985

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 118
S 221 00 Lund Sweden

Document name
Master thesis

Date of issue
September 1985

Document number
CODEN:LUTFD2/(TFRT-5329)/01-081/(1985)

Author(s)
Per Nilsson

Supervisor
Ivar Gustavsson ASEA. Björn Wittenmark

Sponsoring organization

Title and subtitle
Loggningsprogram för µMac-5000
(A program for data logging on µ-Mac 5000.)

Abstract

This program can collect analog values, with a constant sampling period, from 24 different input channels. µ-Mac 5000 can communicate with a VT100 compatible terminal or an IBM-PC. Collected signals, after conversion to engineering units are stored in the RAMarea of µ-Mac 5000. They can also be transfered (directly or later) to a disk on IBM-PC. During logging it is possible to plot signals and show values of the signals on a VT100 terminal.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
81

Recipient's notes

Security classification

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 118
S 221 00 Lund Sweden

Document name
Master thesis

Date of issue
September 1985

Document number
CODEN:LUTFD2/(TFRT-5329)/01-081/(1985)

Author(s)
Per Nilsson

Supervisor
Ivar Gustavsson ASEA. Björn Wittenmark

Sponsoring organization

Title and subtitle
Loggningsprogram för µMac-5000
(A program for data logging on µ-Mac 5000.)

Abstract

This program can collect analog values, with a constant sampling period, from 24 different input channels. µ-Mac 5000 can communicate with a VT100 compatible terminal or an IBM-PC. Collected signals, after conversion to engineering units are stored in the RAMarea of µ-Mac 5000. They can also be transfered (directly or later) to a disk on IBM-PC. During logging it is possible to plot signals and show values of the signals on a VT100 terminal.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
81

Recipient's notes

Security classification

Examensarbete

Loggningsprogram för µMac-5000

Examensarbetet har utförts hos ASEA Generation

Handledare: På ASEA - Ivar Gustavsson

På Skolan - Björn Wittenmark,
Institutionen för Reglerteknik

Utfört av: Per Nilson E-80

LTH den 4.9.85

Innehållsförteckning.

1. Introduktion

Ett Realtids-loggningsprogram har utvecklats för μ -Mac 5000. μ -Mac 5000 är en dator som har möjlighet att samla in och ställa ut både analoga och digitala signaler. Datorn kan kommunicera med en VT-100 kompatibel terminal och IBM-PC.

Detta loggningsprogram samlar in analoga mätvärden via de analoga ingångarna med konstant samplingsintervall. Antalet analoga ingångar är begränsat till 24 st. De insamlade värdena konverteras till sk. ingenjörsstorheter innan de lagras i μ -Macens RAMarea. De kan även lagras på diskett. Operatören bestämmer vid uppstart av loggningsprogrammet om man efter varje sampel direkt skall överföra de konverterade mätvärdena till en fil på IBM-PC, eller om man vid ett senare tillfälle skall överföra alla värdena i RAMet till en fil på disk (dvs. även gamla sampelvärden skall överföras till disk).

För att få en uppfattning om hur en signal varierar med tiden är det möjligt att få en grafisk presentation av signalens konverterade mätvärden i en sk. x-y graf. Det är dessutom möjligt att kontinuerligt efter varje nytt sampel titta på det uppmätta värdet för varje signal. Den bild där detta presenteras visar också de konverterade värdena för signalerna.

2. Syfte med examensarbetet

Huvudidén bakom exjobbet är att skapa ett realtidsprogram för μ-Mac 5000, som kan samla in analoga värden. Det skall vara möjligt att med minimal arbetsinsats och minimala förkunskaper kunna använda loggningsprogrammet.

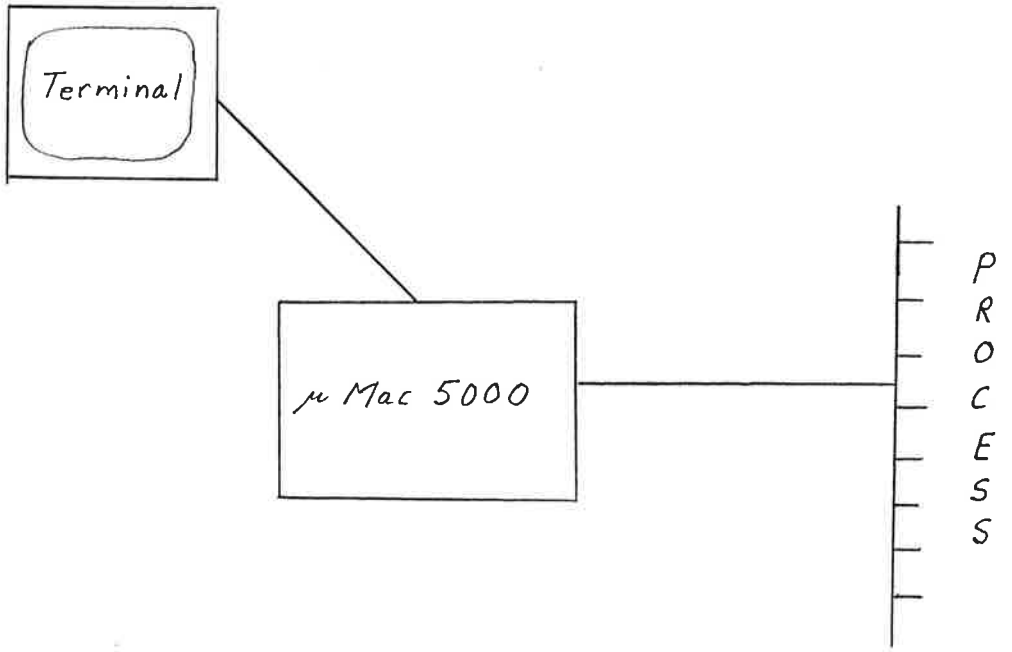
Programmet skall bestå av en initieringsdel som skall vara menystyrd. I initieringsdelen skall man ange alla parametrar som programmet behöver. Därefter kan loggningen starta.

De insamlade värdena skall kunna lagras i primärminnet för senare överflyttning till en IBM-PC. Det skall också vara möjligt att vara direkt ansluten till en IBM-PC för kontinuerlig överföring av insamlade data. Då μ -Macen körs självständigt skall det vara möjligt att plotta insamlade data eller visa senast samplade värden på en VT100 kompatibel terminal. Data som överförs till filer på IBM-PCn skall dessutom kunna överföras till en VAX-780 för senare analys av data. På nästa sida visas hur de olika enheterna skall kunna kommunicera.

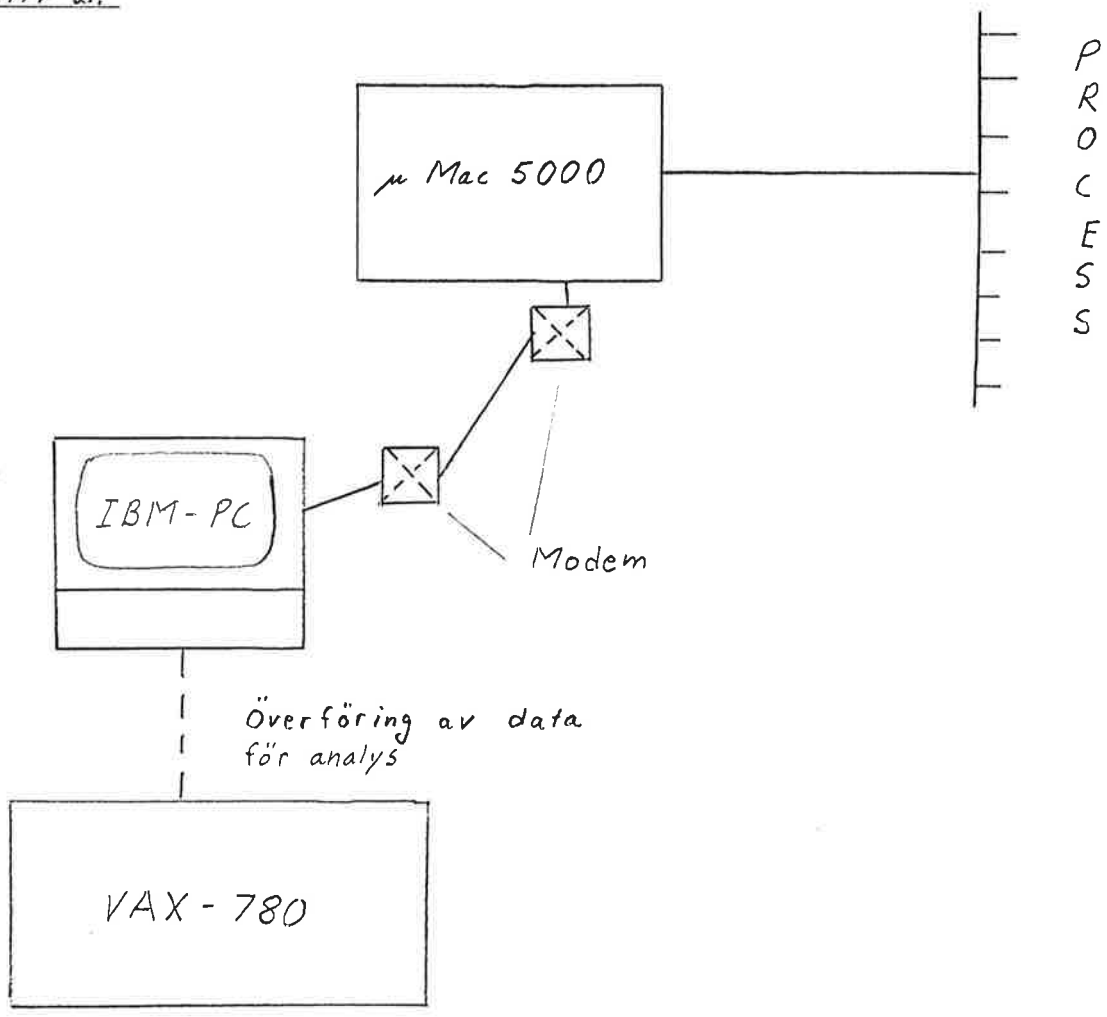
Det är tänkt att användaren endast skall behöva läsa igenom denna manual för att kunna använda loggningsprogrammet.

Hur µ-Mac 5000 skall kunna kommunicera.

Alternativ 1.



Alternativ 2.



3. Beskrivning av μ-Mac 5000

3.1. Presentation

μ -Mac 5000 är en kort-dator utvecklad av Analog Devices. Den är tänkt att användas i industriell miljö för övervakning och styrning. Den är avsedd att kunna köras självständigt från en vanlig terminal med bildskärm.

På kortet finns möjlighet att ansluta både analoga och digitala in/ut signaler. Datorn utnyttjar en 16-bitars μ -processor (Intel 8088), som kan programmeras från en terminal eller från en IBM-PC i μ -MACBASIC. μ -MACBASIC liknar mycket vanlig BASIC men är i några avseenden utvidgad för att passa till μ -Mac 5000.

Till de analoga signalernas in/ut-gångar är sk. signalkonditioneringsenheter anslutna. Dessa enheter möjliggör mätning av olika signaltyper såsom temperatur, mV, Volt samt 4-20mA. Det finns fyra typer av signalkonditioneringsenheter att välja mellan (QMX1-QMX4). I enheterna QMX3 och QMX4 ingår dessutom ± 1000V isolationsskydd och en viss analog filtrering. Varje enhet består av fyra in- eller ut-gångar.

Omvandling av de analoga signalerna till digitala sker via en 14-bitars A/D-omvandlare (13 bitar + tecken).

På μ -Mac 5000-kortet finns det 12 kanaler för analoga in eller ut, 8 digitala in (varav två kan användas som pulsräknar- och frekvens-ingångar), 8 digitala ut, två kommunikationsportar, 24 Volts spänningsmatning eller 220 Volts AC matning. Till μ -Mac 5000 är det möjligt att ansluta expanderkort för att utöka antalet in-ut signaler.

De digitala I/O enheterna är anpassade för TTL-nivåer och för vanliga kontaktswitchar. De två ingångarna för pulsräkning och frekvens innehåller en 32 bits-räknare och kan operera över ett område från 0 till 20 kHz.

Båda kommunikationsportarna är av seriell typ. Port P5 används vid program-utveckling. Den andra porten P6 ("remote") kan t ex. vara kopplad till en skrivare. Båda portarna kan kommunicera enligt signalsnitten RS-232C, RS-422 eller RS-423. Överföringshastigheten för portarna kan sättas från 150 baud till 19200 baud.

På μ -Mac 5000-kortet finns 80k bytes ROM, som innehåller bla. BASIC-språket och kompilatorn. Basic-kompilatorn behöver dessutom 12k bytes RAM som användes vid kompilering av varje nyinskriven programrad. Totalt finns 56k bytes RAM varav användaren kan använda ca. 44k bytes. RAMet är "battery backed up". Användaren har dessutom möjlighet att lagra 32k bytes

program i EPROM. På kortet finns en ledig sockelplats (Z205) för ett ROM. Denna plats utnyttjas för de första 16k byten. Vill man lagra mer än dessa 16k byte är man tvungen att lagra hela programmet, inklusive huvudprogrammet. Detta EPROM skall då placeras på den plats där kompilatorn nu sitter (Z204).Det rekommenderas emellertid ej att mer än 16k bytes program lagras i EPROM. Anledningen till detta är att det annars ej går att styra programmet från en terminal, utan man måste utnyttja de digitala signalerna för att styra programmet. Dessutom blir det problem med kommunikationen om denna sker via en IBM-PC.

Med μ -MacBasicen finns det möjlighet att skriva, editera och exekvera Basic-programmen. Genom att utnyttja systemets avbrott kan realtidsprogram utvecklas. Datorn kan hantera 8 olika avbrott, se kap 3.5.

3.2. Programuppbyggnad

De program som utvecklas på μ -Macen skrivs i μ -MacBasic. Detta är ett högnivåspråk som använder många av de kommandon som finns i standard Basic. Basic är ett interpreterande språk. Språket är utvidgat med en del instruktioner för att passa till μ -Mac 5000. Många av dessa instruktioner kan man finna i programspråk som Pascal och Ada. Dessutom finns vissa programfunktioner för att samla in data eller ställa ut styrsignaler via de analoga/digitala in/ut-gångarna.

Nedan presenteras ett antal viktiga programinstruktioner som ej finns i vanlig standard Basic. Dessa är speciellt viktiga om man vill ha välstrukturerade program.

I vanlig standard Basic används instruktionen Gosub X när man anropar ett underprogram. I μ -MacBasic finns även denna instruktion, men där finns ytterligare två sätt att skapa underprogram. Det är med instruktionerna "Procedure name" och "type Function name". Dessa två motsvarar Pascals underprogram. Hur dessa subprogram skall skapas diskuteras i kap. 3.2.2.

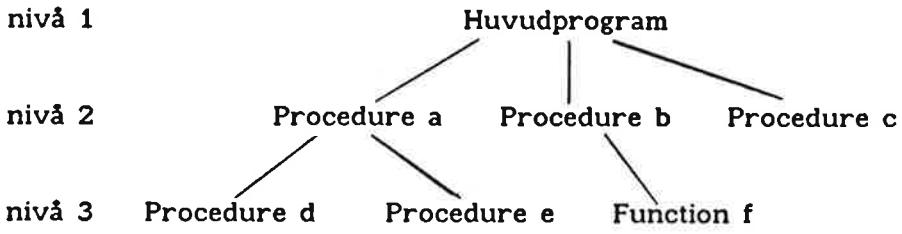
3.2.1. Variabeldeklarationer

För att få välstrukturerade program är det lämpligt att använda meningsfulla variabelnamn, så att även andra än programmeraren kan sätta sig in i programmen. Detta är möjligt i μ -MacBasic ty variabler ,konstanter, procedurer och funktioners namn kan bestå av max 251 tecken. Med kommandot AUTODEF ON/OFF bestämmer man om variablerna skall automatiskt typ-deklareas eller ej. Med AUTODEF ON anges variablerna enligt standard Basic dvs. något av tecknen $,% eller ingenting läggs till efter varje variabel. Då AUTODEF OFF används måste varje ny variabel vara deklarerad innan den får användas.
Exempel på variabeldeklarationer då AUTODEF OFF :

```
INTEGER X
REAL Y
STRING Z
INTEGER ARRAY A(12)
```

3.2.2. Skapa underprogram

Vid utveckling av program utnyttjar μ -Macen så kallade arbetsareor. Skapas en funktion eller en procedur får den en egen arbetsarea som endast innehåller underprogrammets deklarationer och programrader. Huvudprogrammet har sin arbetsarea och måste ligga på översta nivån. Programmen utvecklas alltså hierarkiskt, se fig. nedan.



Vanligtvis befinner man sig i huvudprogrammets arbetsarea (på nivå 1, högsta nivån). Detta visas på bild-skärmen med symbolen "." i början av varje rad. Vid byte av arbetsarea till någon annan nivå kommer symbolen ":" att visas i början av varje ny rad.

För att skapa proceduren a måste man befinna sig i huvudprogrammets arbetsarea. Sedan är det bara att skriva "Procedure a". Proceduren a skapas och man hamnar automatiskt i dess arbetsarea. Procedurerna b och c skapas på liknande sätt. De måste även skapas från huvudprogrammets arbetsarea, så att de hamnar på nivå 2. Proceduren a innehåller två procedurer dvs. procedure d och procedure e. Dessa procedurer skapas från proceduren a:s arbetsarea genom att ange "Procedure d" resp. "Procedure e". Det är alltså viktigt att hålla reda på vilken nivå man befinner sig då nya procedurer eller funktioner skall skapas, annars kan de hamna på fel nivå.

När man har skrivit in en procedurs eller funktions programrader och vill byta arbetsarea till en högre nivå ges kommandot EXIT X. Detta medför ett hopp till nivå X, där X-anger en siffra enligt ovanstående fig. För att komma till nivå 1 kan MAIN kommandot även ges. Hur man byter arbetsarea till en lägre nivå när proceduren/funktionen på denna nivå är skapad beskrivs i kap. 3.3.2.

Funktioner skiljer sig lite från funktioner i programspråket Pascal. En funktion skapas med följande programrad.

typ Function name,
där typ=string,real eller integer
name=funktionens namn

Då detta kommando ges kommer man automatiskt in i funktionens

arbetsarea. Funktioner kan endast returnera ett värde. Den variabel som returnerar värdet heter alltid RESULT, "typ" anger datatypen för RESULT. Innan funktionen lämnas måste alltså RESULT få ett värde.

3.2.3. Variabeldeklarationer för underprogram

En Procedurs eller en Funktions argument anges enligt ex. nedan.

INTEGER ARG Z	-In parameter
REAL ARG S	-In parameter
INTEGER ARG B/VAR	-Både in och ut parameter

Övriga variabler deklareras som vanligt.

3.2.4 Anropa underprogram

Anrop av procedurer och funktioner ser ut precis som i Pascal. Vanligtvis kan inte Proceduren a anropa Proceduren b (se figuren i kap. 3.2.2), men om Proceduren b deklareras såsom EXTERNAL i a:s deklarationsdel är det möjligt att få tillgång till den.

Det har betydelse i vilken ordning procedurer/funktioner skapas. Om en procedur eller en funktion a skall anropa en annan procedur/funktion b på samma nivå, är det viktigt att proceduren/funktionen b har skapats innan proceduren/funktionen a skapas, se figuren i kap. 3.2.2. Anledningen till detta är att om procedurerna/funktionerna lagras på IBM-PC och man vid ett senare tillfälle vill få tillgång till dem i μ -Macen uppstår kompileringsfel när de överförs. Kompileringsfel uppstår pga. att proceduren a anropar proceduren b som ej har skapats.

Ibland vill man i en procedur eller funktion få tillgång till vissa variabler som finns deklarerade på nivån ovanför. Detta uppnår man genom att deklarera variablerna såsom EXTERNAL i underprogrammets deklarationsdel.

3.2.5. Blockstrukturer

μ -MacBasic har programfunktioner för blockstrukturer som ej finns i standard Basic. Liknande blockstrukturer finns i programspråket Pascal. Nedan visas exempel på dem. För dokumentation av samtliga blockstrukturer, se μ -MacBasic programming manualen sid 12-9.

```
Do                  Do If "villkor"     Do
  (satser)            (satser)            (satser)
End Do              End Do              Repeat

While "villkor" Do
  (satser)
Repeat
```

3.2.6. Kommandon för att komprimera program

Vid utveckling av stora progam kan det hända att μ -Macens RAMarea ej räcker till. Det finns emellertid kommandon som kan användas för att komprimera programmet. Dessa kommandon är STORE MODULE och COMPRESS. När något av dessa kommandon ges försvinner källkoden. De utnyttjas främst för att komprimera procedurer och funktioner. STORE MODULE kan även användas för att komprimera huvudprogram. Nackdelen med att ge dessa kommandon är att det inte är möjligt att editera dessa programrader igen. Det är inte heller möjligt att titta på koden igen. Det är därför lämpligt att först lagra de inskrivna programraderna till en fil på IBM-PC innan något av dessa kommandon ges. Hur lagringen av program på IBM-PC sker behandlas senare, se kapitel 3.6.2.

COMPRESS och STORE MODULE skiljer sig lite åt. Befinner man sig i huvudprogrammet och ger COMPRESS kommer alla procedurer och funktioner att komprimeras. Ges STORE MODULE kommer även huvudprogrammet att komprimeras. Vill man att bara en procedur eller funktion skall komprimeras är man tvungen att gå in i dess arbetsarea och ge kommandot där. Då kommandot COMPRESS ges i en procedurs arbetsarea sker en övergång till nivån ovanför och därefter svarar systemet att proceduren har komprimerats. Om kommandot STORE MODULE ges svarar systemet med ett nummer på den modul som just skapats. Detta nummer ligger mellan 2-8, dvs. max 7st. moduler kan skapas. Moduler kan även lagras till EPROM. Det är då viktigt att känna till modulnumret så att rätt modul överförs via port P6 till en EPROM-programmerare. Det är inte möjligt att lagra komprimerade underprogram, som skapats av kommandot COMPRESS.

Komprimerade funktioner och procedurer anropas som vanligt med sina respektive namn.

STORE MODULE bör göras på underprogram som utnyttjas ofta. Fördelen är att andra procedurer eller funktioner som vill anropa de komprimerade underprogrammen inte behöver deklarera dessa såsom EXTERNAL. Om kommandot COMPRESS används måste emellertid EXTERNAL-deklarationer finnas med.

3.3. Programeditering

Det finns några kommandon som är viktiga att känna till vid programutveckling på μ -Mac 5000. Dessa kommer att beskrivas nedan. Det gäller STATUS, MAIN, NEW, ERASE, Reset-knappen, CTRL CX, EXIT X.

Innan man börjar skriva in sina program på μ -Macen är det viktigt att de är välstrukturerade, annars kan svårigheter uppstå vid editering av programmen. I kapitel 3.2.2 beskrevs hur procedurer och funktioner skapas. Har en procedur/funktion en gång skapats, går den ej att ta bort om man inte vill sudda ut hela sitt program. Det är inte heller möjligt att editera eller att ta bort enstaka deklarerade variabler.

Nackdelen med att det ej går att plocka bort deklarerade variabler och skapade procedurer/funktioner kan lösas om programmet lagras till en fil på IBM-PC. Programmet lagras som ASCII-tecken. Genom att utnyttja editeringsmöjligheter på IBMen kan programmen struktureras om, och onödiga variabel-deklarationer tas bort. Hur lagringen av program på IBM-PCn går till behandlas i kapitel 3.6.2.

3.3.1. Kommandon som raderar flera programrader

Med kommandot NEW är det möjligt att radera alla deklarerade variabler och programrader i en arbetsarea, men procedurens/funktionens namn försvinner ej, dvs. arbetsarean kommer att vara tom med endast procedurens/funktionens namn. Kommandot kan även radera flera arbetsareor. Befinner man sig i ett underprogram på nivå X (X är en siffra), och ger kommandot NEW så kommer underprogrammets deklarationsdel och programrader att raderas samt alla arbetsareor under denna nivå. Ges NEW i huvudprogrammet raderas alltså hela RAMarean.

ERASE är snarlikt kommandot NEW. Skillnaden är att ERASE raderar alltid hela RAMarean oberoende vilken arbetsarea man befinner sig i.

3.3.2. Editering av programrader och underprogram

Vanliga programrader kan editeras. Detta görs med kommandot EDIT X, där X anger den programrad som skall editeras.

När man vill editera en procedur/funktion skriver man EDIT namn. "namn" är procedurens/funktionens namn. Det är viktigt att "EDIT namn" ges från den arbetsarea som ligger på nivån över proceduren/funktionen "namn", annars svarar systemet med att underprogrammet inte finns.

3.3.3. Status-kommandot

I μ -MacBasicen finns ett STATUS-kommando som anger hur variabler skall deklareras, om avbrottsflaggan är satt eller ej och vilken precision som används. Då kommandot STATUS ges kan systemet svara enligt:

Autodef: On...Precision: 8 ... Interrupt: On

Autodef beskrevs i kapitlet 3.2.1.
Precision anger noggrannheten för reella tal. Den kan väljas i intervallet 4-24. Det är endast möjligt att ändra precision efter något av kommandona NEW och ERASE.

Då interruptflaggan är "ON" lägger systemet till en interruptflagga (som ej går att se) efter varje nyinskriven programrad. Om ett interrupt inträffar då en programrad med interruptflagga exekveras behandlas avbrottet av en interruptrutin, då raden är färdigexekverad. Det sker då ett subrutinhopp till denna rutin. Interrupt-flaggan kan ändras med något av kommandona NO INTERRUPT och ON INTERRUPT.

3.3.4. Byta arbetsarea

Kommandona MAIN och EXIT X används vid övergång från en arbetsarea till en annan på högre nivå. Dessa beskrevs i kapitlet 3.2.2.

3.3.5. Avbryta pågående exekvering

Ibland vill man bryta exekveringen av ett program. Det finns två sätt att göra detta. Antingen skriver man CTRL CX eller trycker man på Reset- knappen på μ -Mac 5000 kortet. Denna Reset-knapp finns även på ben A9, port P5. Innan man trycker på Reset-knappen måste switchläget för varmstart vara inställt (S3-2), se kapitel 3.4. Annars kan programmet gå förlorat.

3.4. Switchlägen

På μ -Mac 5000 kortet finns 3 st. DIP-switchar (Dual Inline page) benämnda S2,S3 och S6. Varje switch har ett antal positioner. Switchpositionernas lägen är märkta som "on" eller "off". Markeringen anges inom parantes nedan, efter förklaringen av switchpositionens läge.

Dipswitch S2 används för att ge rätt dataformat till kommunikationsporten P5, man sätter bla. antal databitar, antal stoppbitar och om paritet skall användas (se μ -Mac Operation Manual sid 1-17). Position 6 (S2-6) anger den terminaltyp som μ -Macen skall kommunicera med. Antingen används en VT100 terminal ("on") eller en Lear Siegler ("off").

Dipswitch S3 har tre positioner, som anger vilken mode μ -Macen skall operera i. Den första positionen (S3-1) anger WOS/Terminal.

WOS-	ett program som upprättar kommunikation mellan μ -Macen och IBM-PC, ("on")
Terminal-	ansluten till en vanlig terminal, ("off")

Vid framtagning av loggningsprogrammet användes WOS-programmet ofta för lagring av μ -Mac-program till filer på IBM-PC. Programmen överfördes sedan tillbaka till μ -Macen. När dessa överföringar gjordes hade det ingen betydelse i vilket läge denna DIPswitch-position hade.

Den andra positionen (S3-2) anger om Kall- eller Varm-start skall ske, då man trycker på Reset-knappen.

Kallstart-	innebär att hela RAMarean raderas, och initiering av systemets I/O-enheter, ("off")
Varmstart-	gör samma sak förutom att RAMarean förblir oförändrad, ("on")

Den tredje positionen (S3-3) anger Restart/Resume.

Resume-	innebär att programmet fortsätter där det slutade sist efter ett spänningsavbrott, ("on")
Restart-	medför att alla moduler i systemet exekveras efter varje ny "power up", ("off")

De övriga positionerna hos denna Dipswitch används ej. Användaren kan utnyttja dessa i sitt program, för att t ex. ange olika exekveringsvägar. För att veta i vilket läge en switch-position befinner sig måste dess RAMadress läsas av i programmet (se sid 1-18 μ-Mac Operation Manual).

DIP-switch S6 används för att ställa in rätt överföringshastighet för kommunikationsport P5. Hastigheten kan variera från 150 baud till 19200 baud, se μMac-5000 Operation manualen sid 1-11.

3.5. Avbrottssystemet

μ -Mac 5000 har 8 stycken avbrott.

1.) Bruten kommunikationslinje för port P6, (brott på kabeln)
2.) Spänningsavbrott
3.) Tidsavbrott 1
4.) Kommunikationsavbrott för port P6
5.) Uppräkningsavbrott 1
6.) Tidsavbrott 2
7.) Kommunikationsavbrott för port P5
8.) Uppräkningsavbrott 2

De är rangordnade så att det översta interruptet har högst prioritet och den nedersta lägst prioritet. Tidigare har nämnts att efter varje programrad måste det finnas en interrupt-flagga som testar om ett interrupt har inträffat när raden har exekverats. Systemet lägger själv till denna flagga när programraden skapas om Status var Interrupt ON .Om ett interrupt har kommit och interruptflaggan finns, sker ett subrutinhopp till en interrupt-rutin. Denna avbrotts-rutin tar hand om alla de ovan angivna avbrotten.

Avbrottsrutinen måste ha ett speciellt utseende. Den skapas med kommandot Interrupt Procedure name där name är procedurens namn. För att veta vilket avbrott som inträffat, så finns en instruktion,On INTR, som används för att ta reda på vilket avbrott som inträffat.
Nedan visas ett exempel på detta.

```
10 Rem avbrottsrutin
20 On INTR GOTO 100,200,300,400,500,600,700,800
100 Rem Bruten kommunikationslinje för port P6.
```

```
110 Rem Här behandlas detta avbrott
120 EXIT      :Rem motsvarar uthopp från rutinen
200 Rem Spänningsavbrott
  osv.
```

Innan ett interrupt kan inträffa måste det initieras (sättas på). Detta görs med instruktionen Setint(nr,arg), som måste finnas i programmet.

nr- anger vilket interrupt

arg- beror på vilket interrupt som används, se μ -Mac Programming Manual sid 15-6 - 15-7.

Systemet måste även veta vad avbrottsrutinen heter då ett interrupt inträffar. Därför skall följande sats finnas i programmet: On Interrupt name

name- Interruptprocedurens namn

Satserna Setint(nr,arg) och On Interrupt name måste alltså exekveras innan programmet kan behandla avbrott. Ett interrupt stängs av med instruktionen Clearint(nr).

Beskrivning av de vanligaste avbrotten:

Tidsavbrott: Hur ofta ett tidsavbrott skall ske anges med Setint(nr,arg) instruktionen. Arg skall anges i sekunder kan variera från 0 - +9.99*10^253.

Kommunikations-avbrott: Detta avbrott genereras när ett speciellt ASCII-tecken uppträder på kommunikationsporten. Vilket ASCII-tecken som systemet skall reagera på anges med Setint(nr,arg) instruktionen. Vill man att flera olika typer av avbrott skall kunna uppträda skriver man vissa lämpliga tecken till kommunikationsporten innan det speciella ASCII-tecknet ges. I interruptrutinen undersöker man sedan vilka tecken som ligger i bufferten. Är tecknen de önskade skall avbrottet behandlas på ett speciellt sätt.

Uppräkningsavbrott: Med Setint(nr,arg) instruktionen anger man ett tal

mellan 1-65534. Då räknaren når detta tal generas ett avbrott. Räknaren reagerar på den nedåtgående flanken hos den inkommande pulsen. Räknaren måste aktiveras innan den kan börja räkna. Detta görs med instruktionen EVSTART.

3.6. Kommunikation och Programlagring på disk

För att μ -Macen skall kunna kommunicera med en IBM-PC krävs en speciell programkod som upprättar kommunikationen. WOS-programmet är speciellt utvecklat för att passa till μ -Macen. Programmet startas upp i IBMen. Ett annat program som kan kommunicera med μ -Macen är VTERM- programmet. Detta program är utvecklat för att få IBM-PCn att uppföra sig som en VT100-terminal. Dessutom kan VTERM-programmet kommunicera med en VAX-780 dator, samt överföra filer från IBM-PCn till VAXen. Detta beskrivs i kapitel 4.5.

3.6.1. WOS-programmet

Nedan sammanfattas vad WOS-programmet kan klara av.

1.) Terminalkommunikation. Asynkron kommunikation mellan IBM-PC och μ -Macen.
2.) Utvecklade program på μ -Macen kan överföras för lagring till IBMens diskett.
3.) Redigering av filer, t ex. skapa och ta bort filer.
4.) Överföra data under exekvering från μ -Macen till en diskett på IBM-PCn. Det är även möjligt att läsa filer från IBM-PCn.
5.) Utnyttja en skrivare som är ansluten till IBMens parallella utgång.

WOS-programmet upprättar själv rätt överföringshastighet och dataformat med μ -Macen, nämligen 9600 baud, respektive 8 databitar, 2 stoppbitar och ingen paritet.

Upprättande av kommunikation mellan μ -Macen och IBM-PCn med hjälp av WOS-programmet:

1.) Anslut kommunikationsport P5 på μ -Macen till IBMens asynkrona in/ut-gång.
2.) Starta upp IBM-PCn med DOS 2.1.
3.) Använd den diskett där filen WOS.EXE finns.
4.) Gå till den "disk drive" där den ovannämnda filen finns.
5.) Ge kommandot WOS, följt av CR. På skärmen visas nu WOS med stora bokstäver.
6.) Kommunikationen med μ -Macen är nu klar. Vid svårigheter med kommunikationen tryck på Reset knappen. Använd Varm-start så att inte

RAMet raderas.

CTRL Z används när man vill gå ur WOS-programmet.

3.6.2. Programlagring på disk med hjälp av WOS-programmet

För att överföra program från μ -Macens RAMarea till en disk på IBM-PCn utnyttjas kommandot LIST.

LIST "filnamn" -	Överför alla programrader i en arbetsarea till filen filnamn.
LIST ALL "filnamn" -	Överför alla programrader i aktuell arbetsarea och i alla underordnade arbetsareor till filen filnamn.

"filnamn" är ett standard IBM filnamn.

För att hämta lagrade program på IBM-PCn till μ -Macen används kommandot ENTER "filnamn". Filen filnamn innehåller de program som skall överföras till den arbetsarea man befinner sig i eller till någon lägre nivå i μ -Macen. Det är viktigt att man befinner sig på rätt nivå då detta kommando ges, annars kan procedurer och funktioner hamna på fel nivå. Om en procedur har kommit på fel nivå så finns det inga μ -Mac-kommandon som kan placera den på rätt nivå igen.

3.6.3. VTERM-programmet

WOS-programmet upprättar själv rätt överföringshastighet och dataformat. Däremot måste detta anges när VTERM-programmet används. Kommunikationsparametrarna för VTERM-programmet kan lagras på disk, så att man slipper ställa in dem varje gång kommunikation skall ske med μ -Macen. De finns lagrade i filen VTERM.SET. Parametrarna skall vara de samma som WOS-programmet använder, se kapitel 3.6.1. På μ -Mac-kortet skall även switcharna S2 och S6 vara inställda för denna typ av kommunikation, se kapitel 3.4.

Upprättande av kommunikation mellan μ -Macen och IBM-PCn med hjälp av VTERM-programmet:

1.) Anslut kommunikationsport P5 på μ -Macen till IBMens asynkrona in/ut-gång.
2.) Starta upp IBM-PCn med DOS 2.1.
3.) Använd den diskett där filerna VTERM.EXE och VTERM.SET finns.

4.) Gå till den "disk drive" där de ovannämnda filerna finns.

5.) Ge kommandot VTERM, ev. VTERM xxxx
xxxx är namn på en fil med setup-parametrar, som motsvarar VTERM.SET, ifall ej standard setup-parametrar kan användas.

6.) Tryck på F5 tills "set up"-menyn för VTERM kommer upp. Ändra eventuellt kommunikationsparametrarna så att de överensstämmer med de i kap. 3.6.1. (se Hjälpmenyn för VTERM hur man gör detta).

7.) Spara eventuellt ovanstående kommunikationsparametrar, med kommandot Alt S, se Manual för VTERM.

8.) Tryck på F5 tills den tomma menyn kommer upp. Kommunikationen med μ -Macen är nu klar.

Med CTRL BREAK går man ur VTERM-programmet. Vid problem med kommunikationen se kapitel 3 i VTERM-manualen.

3.6.4. Skillnaden mellan VTERM- och WOS- programmen

WOS-programmet klarar inte av kommandon av typen att placera ut Cursorn i lämplig position på bildskärmen på en VT100-terminal. Detta klarar däremot VTERM-programmet. Detta program är utvecklat för att få IBM-PCn att uppföra sig som en VT100-terminal. Däremot klarar inte VTERM-programmet av någon kommunikation med IBMens disketter när den kommunicerar med μ -Macen.

I kapitel 3.4 diskuterades Switchlägena. Där fanns en speciell switch just för WOS (S3-position 1). Detta läge har ej använts, ty WOS-programmet kunde kommunicera med μ -Macen även då denna switch-position var i läget Terminal. VTERM-programmet kan endast kommunicera med μ -Macen då Terminal-läget är inställt. Låt därför denna switch-position vara i Terminal-läget, så slipper man växla switchens läge vid byte av kommunikationsprogram.

3.7. Lagring av program i EPROM

I kapitel 3.1 angavs att man inte bör lagra mer än 16k bytes program i EPROM. Det är endast program som ligger som moduler i μ -Macen som kan överföras till EPROM. Programmen bör bestå endast av procedurer och funktioner.

Hur en programmering av ett EPROM går till beskrivs nedan.
Den modul som skall programmeras skickas via porten P6 till "PROM-brännaren". Alla data som skickas via denna port är av Intel standard-format, i hexadecimal

form, för fullständig specifikation se μ -Mac 5000 Operation Manual Appendix D-2.

De olika stegen för att överföra data till EPROMet.

1.) Innan funktioner och procedurer skapas är det viktigt att μ -Macen har Kall-startats, och att switch-läget Restart används.

2.) När alla procedurer och funktioner som skall promprogrammeras finns inne i μ -Macens RAMarea ges kommandot STORE MODULE, se kapitel 3.1.6. Systemet svarar med ett nummer på modulen.

3.) Ändra switch-lägena till Varm-start och Resume.

4.) Anslut en RS-232C kabel mellan μ -Macens port P6 och prom-programmeringsapparaten. Hur anslutningen skall göras visas i μ -Mac 5000 Operation Manual sid 3-7.

5.) Ställ in prom-programmeringsapparaten för rätt överföringshastighet samt dataformat. Lämpliga kommunikationsparametrar är: överföringshastighet 1200 baud, 7 databitar, 1 stoppbit och jämn paritet.

6.) Upprätta kommunikationsport P6 genom att ge följande kommandon till μ -Macen COM(1,1200,"on","even",7,1) och CAL("off").

7.) Ge nödvändiga kommandon till prom-programmeringsapparaten så att den kan ta emot data från μ -Macen till sin RAMarea.

8.) Modulen i μ -Macens RAMarea kan nu överföras till prom-programmerinsapparatens RAMarea. Ge följande kommando till μ -Macen:
PROGRAM MODULE X AT YYYY
X=modulens nummer.
YYYY=En adress som anger var lagringen av data skall börja i prom-programmeringsapparatens RAMarea.

9.) Då överföringen av data till prom-programmeringsapparaten är klar svarar μ -Mac-systemet med följande sats:
Program Next Module at ZZZZ
ZZZZ=Start-adressen för nästa modul som skall prom-programmeras.

10.) Överför nu data från prom-programmeringsapparatens RAMarea till ett EPROM.

11.) Lagringen av data till EPROMet är nu klart.

4. Operatörsmanual

4.1. Uppkoppling och start av programmet

Loggningsprogrammet finns lagrat på en disk och vissa delar ligger dessutom lagrat i ett EPROM. EPROMet är placerat på μ -Mac-kortet (på platsen Z205). Programmet som ligger lagrat i EPROM finns även på disk i en fil som heter MODULE.ROM. I Appendix E finns flödesschema på huvudprogrammet och de större underprogrammen. Innan man kan starta loggningsprogrammet måste några programfiler överföras till μ -Macens RAMarea. De filer som skall användas skickas från en IBM-PC till μ -Macen. Datakommunikationen dem emellan sköts av WOS-programmet. Uppstarten av WOS-programmet beskrevs i kapitlet 3.6.1.

4.1.1. Överföring av loggningsprogrammet till μ-Macen från en IBM-PC

Det finns två olika loggningsprogram, det vanliga och en reducerad variant där plottningsprogrammet saknas. Med plottningsprogrammet kan insamlade mätvärden presenteras grafiskt. Då det reducerade loggningsprogrammet används tjänar man ungefär 2k byte av RAMarean. Denna plats utnyttjas i stället för lagring av fler mätvärden.

Innan man överför filerna från IBMen till μ -Macen måste man ha bestämt sig för vilket loggningsprogram som skall användas.
Nedan visas vilka filer som skall överföras till μ -Macen. De två filerna mainupl.ram och dekupl.ram skall användas då plottningsprogrammet ej utnyttjas. Då får filen plotta.ram inte användas, upl-står för utan plottningsprogram.

Filerna skall överföras i den ordning de står nedan till μ -Macen.

1.)	Dek.ram/	-Huvudprogrammets deklarationsdel.
	Dekupl.ram	-Huvudprogrammets deklarationsdel utan variabler för plottningsprogram.
2.)	Initiera.ram	-Initieringsprogram.
3.)	Plotta.ram	-Plottningsprogram.
4.)	Diskov.ram	-Program som överför insamlade data till disk.
5.)	Interr.ram	-Interruptrutinen.
6.)	Main.ram/	-Huvudprogrammet.
	Mainupl.ram	-Huvudprogrammet utan satser för att utföra plottning.

μ -Macens Ramarea räcker inte till om alla filerna överförs direkt efter varandra till μ -Macen. Därför måste kommandot COMPRESS ges efter varje fil-överföring mellan punkterna 2-5. Detta gör att programmets plats i RAMarean komprimeras med ungefär 50%.
Nedan visas ett exempel på hur loggningsprogrammet skapas då programmet överförs från IBM-PCn till μ -Macen. Varianten med plottningsprogrammet utnyttjas därvid.

1.) Se till att μ -Macens switchar är inställda på Varm-start och Restart.
2.) Upprätta kommunikationen mellan IBM-PCn och μ -Mac 5000 med hjälp av WOS-programmet, se kapitel 3.6.1.
3.) Ge kommandot ERASE, RAMarean raderas.
4.) Ge kommandot STATUS för att se att interruptflaggan är på. Om den är avstängd ge kommandot ON INTERRUPT.
5.) Ge kommandot ENTER "Dek.ram"
6.) Vänta på att "." visas på skärmen. Första filen är nu överförd till μ -Macen.
7.) Ge kommandot ENTER "Initiera.ram".
8.) Vänta på att "." visas på skärmen, skriv därefter kommandot COMPRESS.
9.) Ge kommandot ENTER "Plotta.ram"
10.) Vänta på att "." visas på skärmen. Ge kommandot COMPRESS.
11.) Ge kommandot ENTER "Diskov.ram"
12.) Vänta på att "." visas på skärmen. Ge kommandot COMPRESS.
13.) Ge kommandot NO INTERRUPT.
14.) Ge kommandot ENTER "Interr.ram".
15.) Vänta på att "." visas på skärmen. Ge kommandot COMPRESS.
16.) Ge kommandot ON INTERRUPT.
17.) Ge kommandot ENTER "Main.ram".
18.) Vänta på att "." visas på skärmen. Hela loggningsprogrammet finns nu i μ -Macen.
19) Ge kommandot CTRL Z, (går ur WOS-programmet).

När filen Interr.ram skulle överföras till μ -Macen stängdes avbrottsflaggan av pga. att denna fil innehåller en interruptrutin. I en interrupt-rutin får inga avbrott ske.

4.1.2 Starta loggningsprogrammet

Nedan anges hur loggningsprogrammet startas från en IBM-PC då programmet finns i μ -Macens RAMarea.

1.) Upprätta kommunikationen mellan IBM-PCn och μ -Macen med hjälp av VTERM-programmet (se kapitel 3.6.3).
2.) Nu är det bara att ge kommandot RUN så exekveras programmet.

För att kunna starta loggningsprogrammet måste kommunikationen ha upprättats med VTERM. Det kan ej startas från WOS.

Om en vanlig VT100 kompatibel terminal är ansluten till μ -Macen är det bara att ställa in rätta kommunikationsparametrar för den. Dessa parametrar skall vara de samma som WOS-programmet utnyttjar, se kapitel 3.6.1. Därefter ger man kommandot RUN för att starta programmet.

4.2. Beskrivning av loggningsprogrammet

Loggningsprogrammet är till för att samla in mätvärden från max 24 analoga ingångar. 12 av de analoga ingångarna sitter på μ -Mac 5000 kortet medan de övriga sitter på expanderkortet 4010. Mätdata samlas in då ett tidsavbrott inträffar. När ett tidsavbrott skall inträffa bestäms av det konstanta samplingsintervallet som anges i initieringsdelen innan loggningen startar. Insamlade värden från de analoga signalerna konverteras till "ingenjörsstorheter" innan de lagras i en ringbuffert i μ Macens RAMarea, se Appendix A.

De konverterade mätvärdena kan överföras till disk efter varje nytt sampel. Detta överföringssättet kommer i fortsättningen att kallas Direkt överföring. Det är inte nödvändigt att utnyttja "direkt överföring". Istället kan man vänta tills RAMarean blir nästan full och då överföra dessa värden till disk. Detta sätt att överföra data till disk kommer att kallas Mjölkning.

Om RAMarean blir full innan en Mjölkning hinner ske kommer gamla sampelvärden att skrivas över av nya pga. att en ringbuffert utnyttjas. När en Mjölkning utförs kommer tiden för de saknade sampelvärdena att anges i den fil som skapas.

De filer som lagras på disk är av två typer. Den ena filen heter namn.DAT och innehåller endast signalernas konverterade värden. Den andra, kallad namn.DOK, innehåller all information om de loggbara signalerna samt information

om samplingsintervall, start-loggningstidpunkt och överskriden tid i RAMarean. Filen namn.DOK innehåller all information som anges i initieringsdelen av loggningsprogrammet och som därför fullständigt beskriver hur mätningen utförts.

Det är viktigt att samplingsintervallet ej är för kort ty då kommer vissa sampelvärden att gå förlorade. Hur fort det går att sampla kommer att behandlas i kapitlet Prestanda. Ovan nämndes att data i μ -Macen kunde överföras till en disk på två olika sätt. Det är även möjligt att byta överföringssätt dem emellan under en loggning om samplingsintervallet ej är för kort. I initieringsdelen anges hur överföringssättet skall ske då loggningen startas. När loggningen har startat kan speciella kommandon ges för att byta överföringssätt, se kapitel 4.3.

4.2.1. Val av kommunikationsprogram

Nedan anges vad som ingår i loggningsprogrammet.

a.) Initieringsdel. Det är möjligt att visa initieringsdelen under pågående loggning.

b.) Meny som visar de senast samplade mätvärdena.

c.) Plottningsprogram. Kan plotta max 2 signalers konverterade mätvärden samtidigt.

d.) Byte av överföringssätt. Från Direkt överföring till Mjölkning eller vice versa.

e.) Avsluta loggningen eller utföra en Mjölkning.

För att klara alla ovanstående punkter måste växling ske mellan WOS-prog. och VTERM-prog. Nedan anges när VTERM-prog. resp. WOS-prog. kan användas vid de olika överföringssätten.

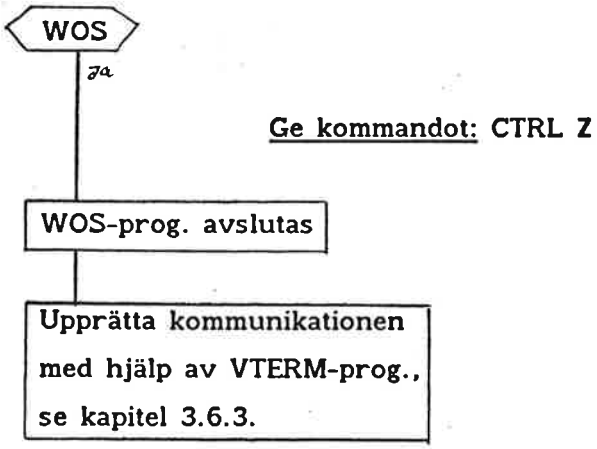
Mjölkning- Om VTERM-prog. används får endast punkterna a.) till c.) utföras. Vill man utföra punkterna d.) och e.) måste kommunikationsprogrammet bytas till WOS.

Direkt överföring- VTERM-prog. får bara användas för punkt a.). Programmet får endast användas vid uppstart då alla variabler skall initieras.
WOS-programmet skall användas när initieringen är klar, dvs. då loggningen skall starta (se schema kapitel 4.2.3.). WOS-prog. får endast användas för punkterna d.) och e.).

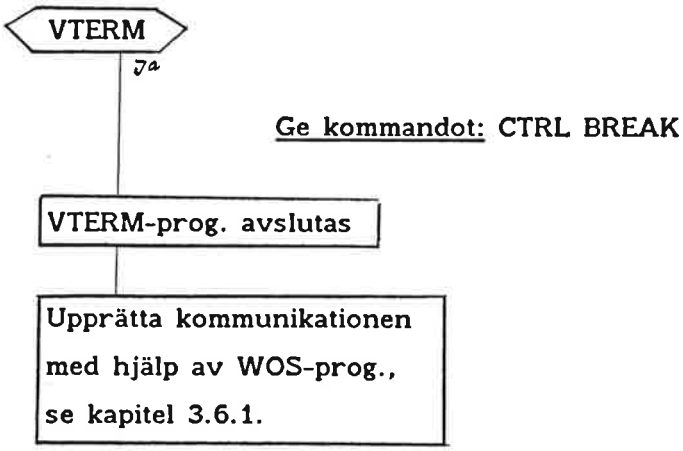
Om en VT100-terminal är ansluten måste överföringssättet vara Mjölkning. Endast punkterna a.)- c.) kan utföras.

4.2.2. Byta kommunikationsprogram

Byte från WOS till VTERM.

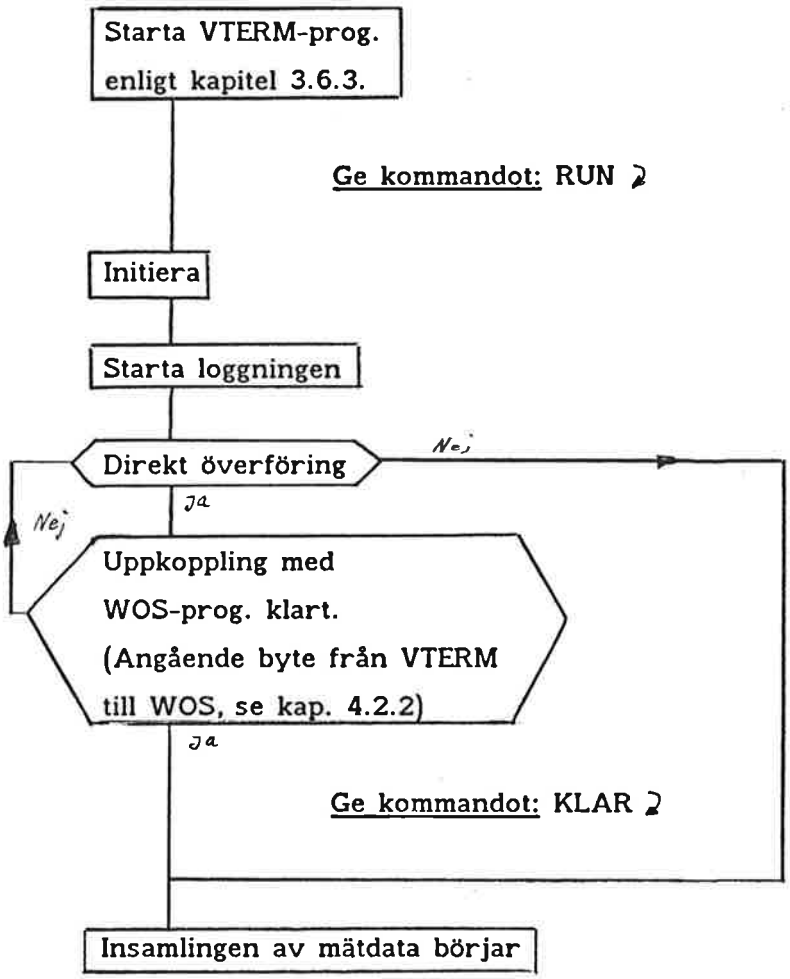


Byte från VTERM till WOS



4.2.3. Loggningsprogrammets initieringsdel

Då loggningsprogrammet startas med kommandot RUN får man upp en Huvudmeny för initieringsdelen, se Appendix D-2. Från denna meny kan subrutinhopp ske till 3 initieringsmenyer (se Appendix D-3 - D-5) eller så kan loggningen startas om alla initieringar redan gjorts. Nedan visas vad som händer vid de två olika överföringssätten då man anger att loggningen skall starta. Där visas även hur uppstarten sker av loggningsprogrammet.



↵ motsvarar RETURN-tangenten.

Vid Direkt överföring kan inte insamling av mätdata börja förrän WOS-programmet sköter kommunikationen mellan IBM-PCn och μ -Macen.

Det som skall anges för resp. Meny i initieringsdelen anges nedan.

Meny 1: Dagens datum och tidpunkt, se Appendix D-3.

Meny 2: För varje Signal som skall deklareras anges:
Signalnummer (OBS! mellan 0-23, ej 1-24, ty signalnummer=kanalnr.

på kortet), signalbeskrivning, mätområde (en kod anges), konverteringssätt (omvandling till "ingenjörsstorheter", linjärt y=k*x+m eller ett rotuttryck y=k*sqr(x)), k- och eventuellt m-konstanten till de föregående ekvationerna samt enhet på konverterad signal. Se Appendix D-4.

Meny 3: Filnamn, Konfiguration (Mjölkning eller Direkt överföring), samplingsintervall och vilka signaler som skall loggas. Se Appendix D-5.

Första gången man går in i någon initieringsmeny måste samtliga värden anges innan det är möjligt att gå ur menyn. Har en meny initierats och man går in i den igen visas alla tidigare inmatade värden. Dessa värden kan editeras om man flyttar sig till aktuell rad med ↑ eller ↓ tangenterna, och därefter trycker på tangenten F1. I meny 3 skriver programmet ut hur länge loggningen kan pågå vid Mjölkning, (dvs. tiden anger när RAMarean blir full).

4.2.4. Kommandon

Nedan anges de kommandon som finns tillgängliga då loggningen startat.

Om man befinner sig i WOS:

CTRLB — Vid Mjölkning avslutas loggningen.
Vid Direkt överföring avslutas loggningen om samplingsintervallet < 1.5 sek., annars sker övergång till Mjölkning.

CTRLB CTRLB — Övergång från Direkt överföring till Mjölkning. Därefter avslutas loggningen.

M CTRLB — En Mjölkning utförs. Alla mätvärdena i Ringbufferten överförs till en fil parallellt med pågående loggning.

*md CTRLB — Byte från Mjölkning till Direkt överföring.

Om man befinner sig i VTERM eller är ansluten till en VT100 terminal:

S CTRLB — Då detta kommando ges kommer en meny upp som visar vilka signaler som loggas, se Appendix D-6. Av dessa signaler kan max 5 st. väljas ut åt gången för visning av

senast samplade värde. Efter att ha valt vilka signaler som skall visas, kommer bilden upp där de visas, se Appendix D-7. Då ett nytt sampelvärde kommit in, visas värdet direkt på bildskärmen. Menyn visar både det konverterade mätvärdet och det verkliga.

E CTRLB

För att avbryta visningen av de senast samplade värdena ges kommandot E CTRLB. Återhopp sker direkt till Hjälpmenyn.

I CTRLB

Visar samtliga initieringsvärden, samt tidpunkten då loggningen startade. De olika menyerna når man från huvudmenyn genom att ange en siffra på den meny man vill komma till. Huvudmenyn kommer upp då I CTRLB ges. Dessa menyer förutom Meny 1 (tidsmenyn) är de samma som initieringsmenyerna vid uppstart, se Appendix D-2, D-4 - D-5. Meny 1 innehåller nu all tidigare information samt tidpunkten och datumet då loggningen startade.

Då överföringssättet är Mjölkning kan man räkna ut datum och klockslag då RAMarean blir full. I Meny 3 visas en tid som anger hur länge loggningen kan pågå innan RAMarean blir full. Tiden anges på formen dygn:timmar:min:sek. Genom att addera denna tid till start-loggningstidpunkt fås datum och klockslag då RAMarean blir full. En Mjölkning bör naturligtvis ske innan RAMarean blir full.

För att kunna se en viss signals parametrar måste man ta in Meny 2 och ange numret på signalen.

P CTRLB

Plottningsprogrammet kan grafiskt presentera insamlade data i en x-y graf. Max 2 signaler kan visas samtidigt. Innan en plottning kan ske skall följande ha definerats.

- Vilken eller vilka signaler som skall plottas.
- Antal skalmarkeringar på x-axeln (tidsaxeln).
- Om mätvärdena skall visas inom valda max- och min-gränser eller om signalernas max resp. min skall utgöra maximum resp. minimum av signalens/signalernas värden som skall plottas.
- Om automatisk plottning skall ske av de 71 senast samplade värdena.

- Om plottningen skall starta vid ett speciellt sampelnummer.
- Om avståndet till nästa sampelnr. skall vara större än 1, dvs. man kan plotta värdena med ett annat sampelintervall.

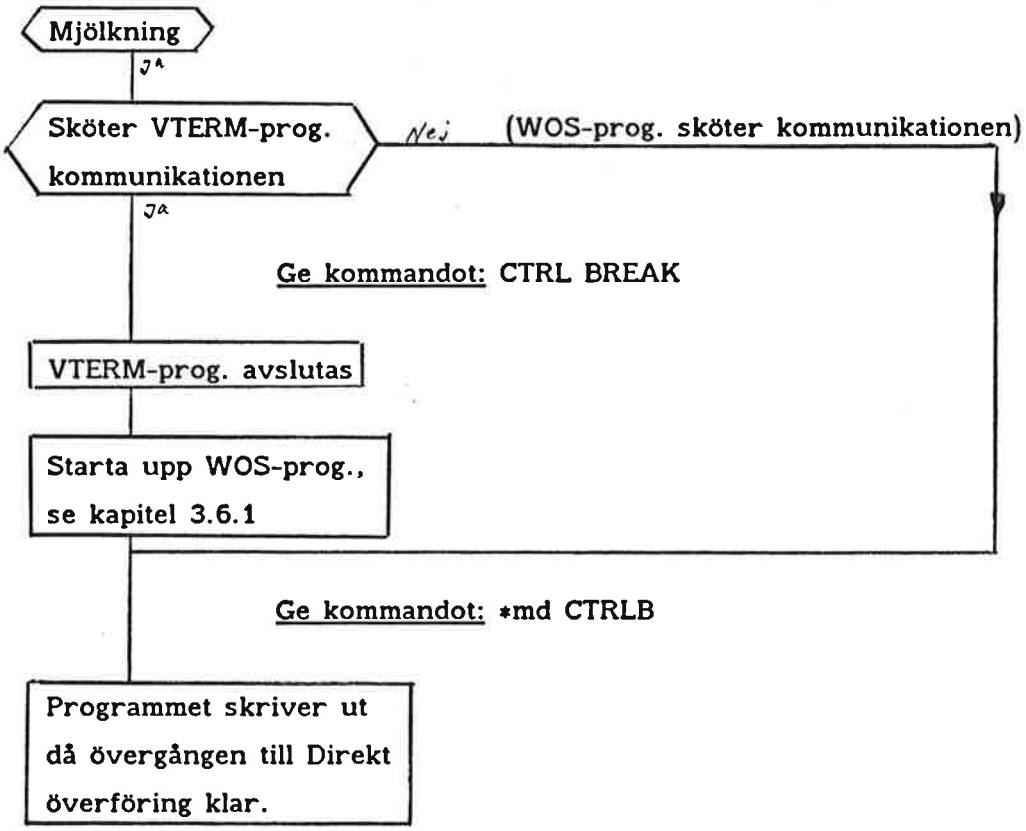
Då automatisk plottning väljs kommer en ny plottning att ske automatiskt om ett nytt sampelvärde kommer in. Den automatiska plottningen avbryts med kommandot A CTRLB.

A CTRLB — Då samplingsintervallet är < 6 sekunder startar inte den automatiska plottningen direkt efter varje nytt sampelvärde. Programmet frågar då istället om en ny automatisk plottning skall ske. Anledningen till detta är att operatören behöver tid på sig för att analysera plottningsbilden.

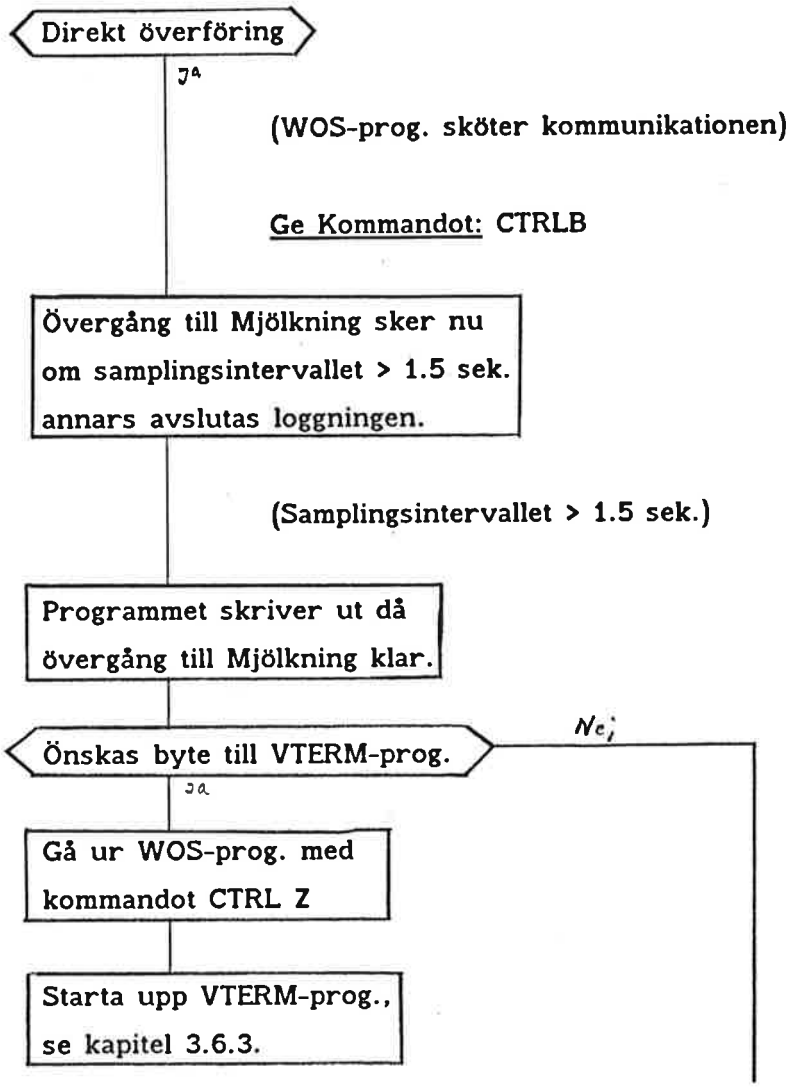
H CTRLB — Visar Hjälpmenyn, se Appendix D-8.

4.3. Övergång från Mjölkning till Direkt överföring och vice versa

Övergång från Mjölkning till Direkt överföring.



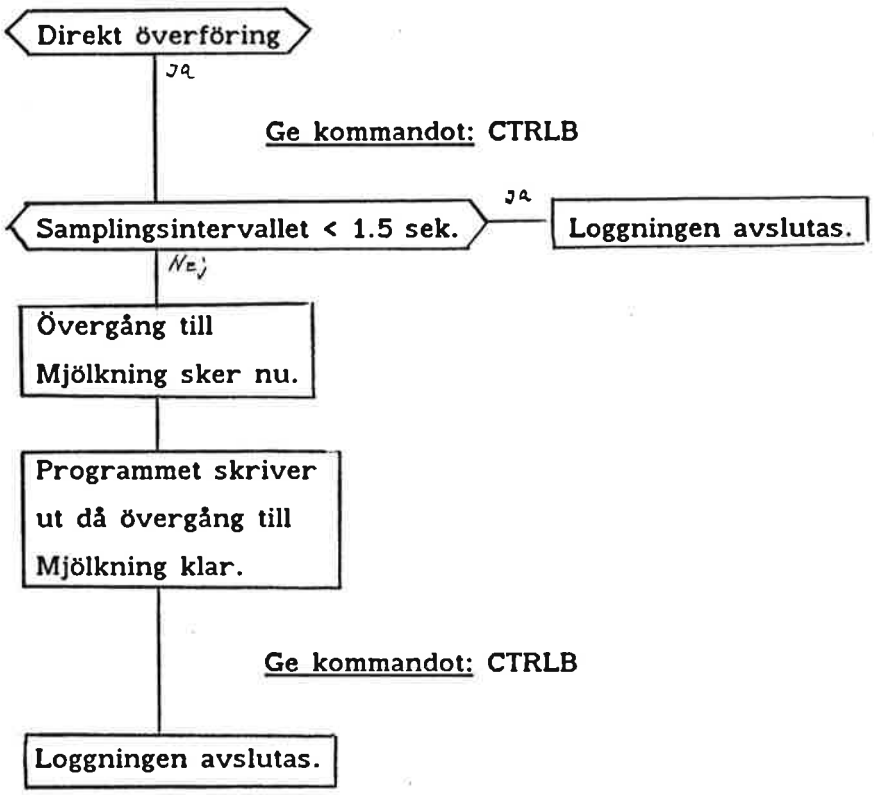
Övergång från Direkt överföring till Mjölkning.



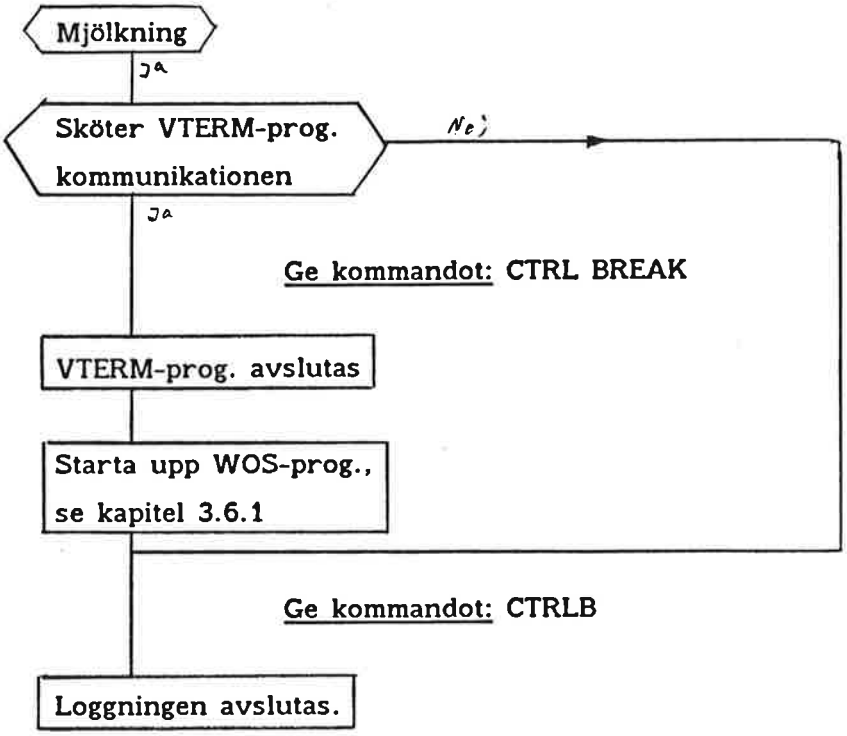
4.4. Avbrytning av programmet.

Nedan visas hur en Mjölkning går till och hur en loggning avslutas vid de två olika överföringssätten. Dessutom anges vilka kommandon som skall ges.
Det krävs att WOS-programmet sköter kommunikationen mellan μ -Macen och IBM-PCn.

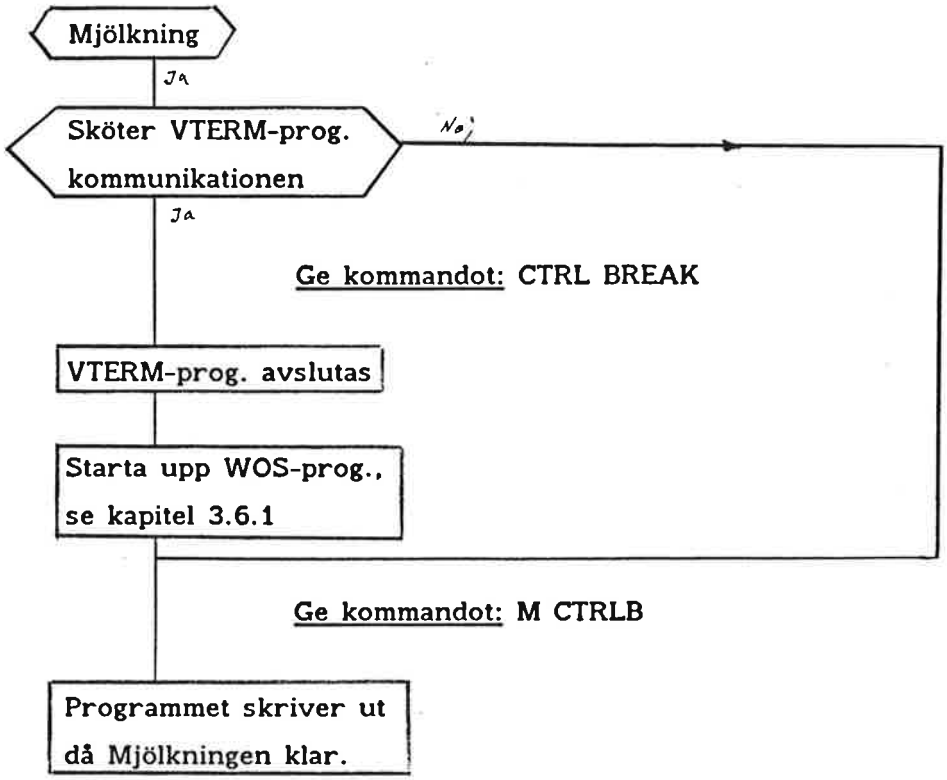
Loggningen skall avslutas vid Direkt överföring.



Loggningen skall avslutas vid Mjölkning.



Utför en Mjölkning.



Vid en Mjölkning skapas alltid två nya filer, en dokumentationsfil och en datafil. Dessa båda filer skiljs åt genom att .DOK eller .DAT läggs till filnamnet. Om det sker två Mjölkningar i rad kommer det alltså att skapas 4 olika filer. För att kunna skilja de två senaste filerna från de två föregående läggs en siffra till filnamnet. Första filen som lagras har då numret 0 och den därpå 1 osv.

4.5. Omvandling av filer för VAX

Då en loggning har avslutats finns enligt ovan ofta ett antal filer från en och samma mätning. De är av typen namnXX.DOK och namnXX.DAT.

namn-	Namnet på filen som angavs i initieringsdelen av loggningsprogrammet.
XX-	Ett nummer som är till för att skilja olika Mjölkningar åt.
.DOK-	Anger att det är en dokumentationsfil. Den innehåller samtliga initieringsparametrar.
.DAT-	Anger att det är en data-fil. Den innehåller konverterade sampelvärden.

I Appendix B visas hur de filer som loggningsprogrammet har skapat ser ut. Den första sidan visar hur dokumentationsfilen ser ut och den andra hur datafilen ser ut. Där förklaras även vad värdena betyder.

De filer som har samma "namn" men olika nummer, dvs. filer från en och samma mätning, vill man överföra till en enda fil efter mätningens slut. Den resulterande filen skall sedan överföras till en VAX-780 dator, där data kan analyseras av identifieringsprogrammet IDPAC. Ett program för IBM-PC har skrivits i BASIC för att sammanfoga flera filer med samma "namn" till en. Detta program heter KONVERT.BAS. Då programmet startas frågar det efter "namn" (filnamnet) och högsta numret för filerna (XX). Därefter skapas det en ny fil som kallas namn.T, där namn motsvarar "namn" (filnamnet). Hur denna fil ser ut framgår av Appendix B. Den skapade filen namn.T har ett sådant dataformat som IDPAC-programmet på VAX-780 datorn accepterar. Överföringen av den skapade filen till VAX-780 datorn kan sedan ske med hjälp av VTERM-programmet. Detta visas nedan.

1.) Anslut IBMens asynkrona in/ut- gång till VAXen.

2.) Starta upp IBM-PCn med DOS 2.1.

3.) Använd den diskett där VTERM.SET och VTERM.EXE finns.

4.) Gå till den "disk drive" där de ovannämnda filerna finns.

5.) Ge kommandot VTERM ev. VTERM xxxx,
xxxx- är namn på en fil med setup-parametrar. Den motsvarar VTERM.SET.

6.) Tryck på F5 tills "setup-menyn" kommer upp. Ändra eventuellt kommunikationsparametrarna så att de överensstämmer med VAXens, se Hjälpmenyn för VTERM hur man gör detta. Spara eventuellt dessa kommunikationsparametrar med kommandot Alt S.

7.) Tryck på F5 tills den tomma menyn kommer upp.
Tryck på RETURN tangenten.
Nu skall IBMen kunna kommunicera med VAXen. På bildskärmen frågar VAXen nu efter USERNAME och PASSWORD. Ange dessa.

8.) Tryck på F5 tills menyn där ett filnamn skall anges kommer upp. Ange filnamn. Filnamnet skall vara namnet på den fil man vill överföra från en disk till VAXen. Se till så att det står SEND ej RECIEVE vid filnamnet (filen skall skickas till VAXen). Se Hjälpmenyn vid problem.

9.) Tryck på F5 tills bilden som kan kommunicera med VAXen kommer upp.

10.) Se till att den fil som skall överföras till VAXen finns i rätt "disk drive".

11.) Genom att skriva create "filnamn" och därefter ge kommandot Alt T skapas och överförs den önskade filen till VAXen, "filnamn" måste vara samma filnamn enligt punkt 8.).

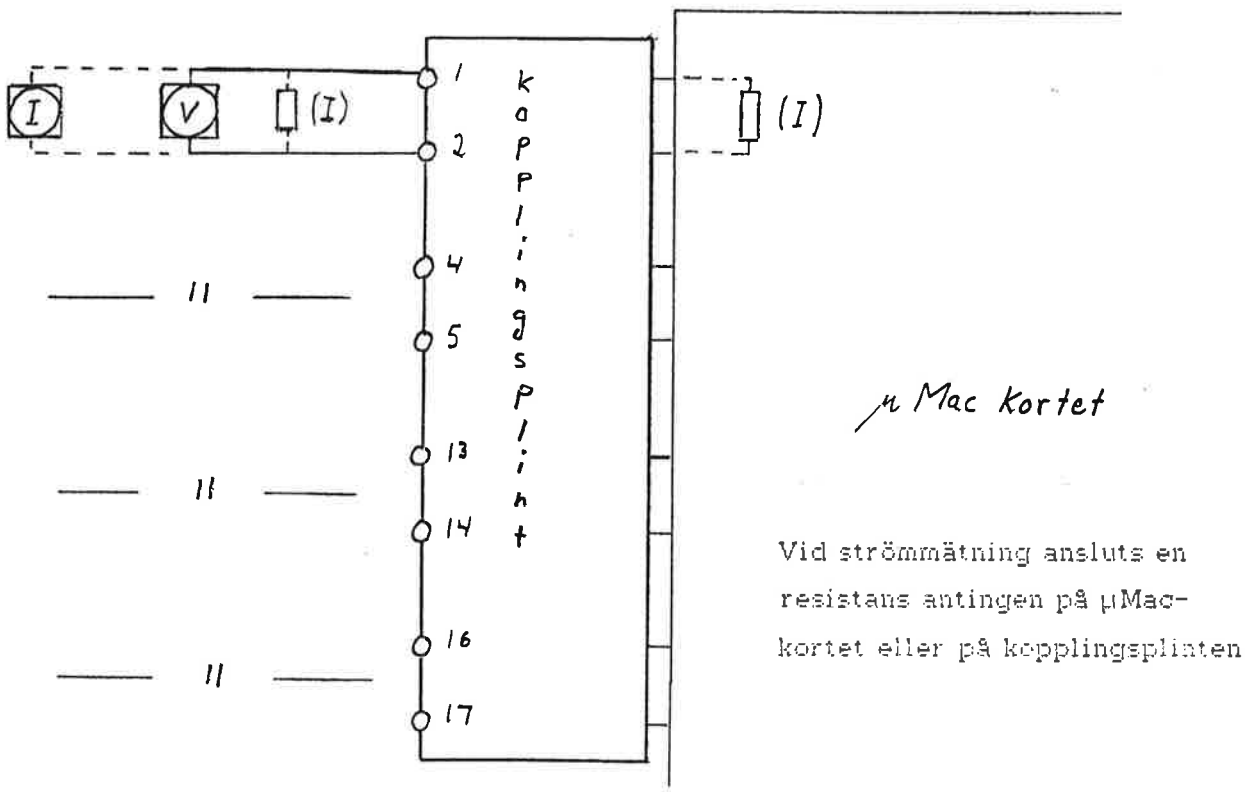
Med CTRL BREAK går man ur VTERM-programmet. Vid problem med kommunikationen se VTERM-manualen, kapitel 2.

4.6. Hårdvara som behövs för att kunna samla in Analoga signaler.

μ -Macen kan mäta spänningar och strömmar upp till 10V resp. 20mA. De analoga signalerna ansluts till μ -Macen via kopplingsplintar. På en kopplingsplint kan 4 signaler anslutas. Kopplingsplint AC 1800 används vanligtvis men om man skall mäta spänningar upp till ± 10V bör kopplingsplint AC 1814 användas. Annars får signalen spänningsdelas innan den kopplas till AC 1800.

μ -Macen måste veta vilket mätområde signalen ligger inom. I loggningsprogrammets initieringsdel anger man detta med ett kod-nummer för varje signal, se kapitel 4.2.3. Detta kod-numret använder även datorn internt.
För att kunna utföra ström-mätning måste vissa Resistanser anslutas, antingen på kopplingsplinten eller på μ -Mac-kortet.

Nedan visas kopplingsschema för att utföra en ström eller spänningsmätning.



Resistanser som skall anslutas vid en strömmätning:

Mätområde	Resistans	
± 20 mA	4.0 Ω	0.5%
± 1 mA	80.6 Ω	0.1%
4-20 mA	4.0 Ω	0.5%

Följande spänningsdelning krävs då något av nedanstående spänningsområde utnyttjas. Kopplingsplint AC 1800 skall användas.

Mätområde	Spänningsdelningsfaktor
1 V	50:1
5 V	50:1
10 V	100:1

μ -Macen kan även användas för att mäta temperatur då termoelement är anslutet till den. Då loggningsprogrammet ej utnyttjar detta tas det ej upp här. Se μ -Mac Operation Manual sid 1-33 - 1-35 för mer information.

5. Prestanda

5.1. Tillgänglig data-area.

I μ -Macens RAMarea lagras de konverterade insamlade mätvärdena i en endimensionell vektor. Storleken på denna anges nedan. Den varierar beroende på vilket loggningsprogram som används, dvs om plottning är möjlig eller ej.

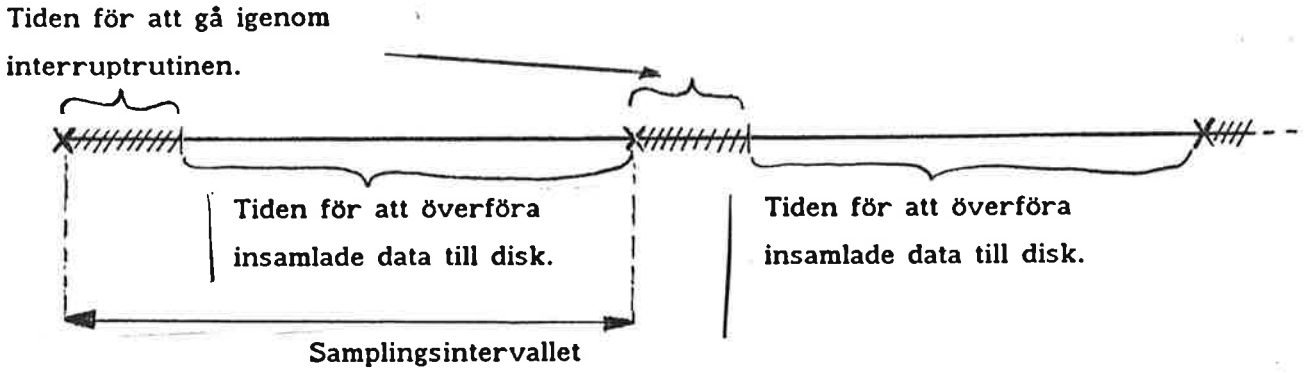
a.) 1800 platser då plottningsprogrammet används.
b.) 3800 platser då plottningsprogrammet ej används.

En plats motsvarar ett mätvärde.

5.2. Tider för lagring av data på disk.

Att lagra ett signalvärde till disk då man gör append på filen,(dvs då filen redan är öppen), tar ungefär 40ms. Vid val av samplingsintervall är det viktigt att man beaktar denna tid om data skall överföras till disk under loggningen (mjölkning eller direkt överföring).
Om man har tänkt att "Mjölka" flera gånger är det viktigt att det blir gott om tid över för att överföra insamlade data till en fil, dvs. samplingsintervallet får ej vara för kort. Annars kommer kanske denna överföring att ta extremt lång tid eller aldrig bli klar, i vilket fall sampelvärden går förlorade. Vid Mjölkning skall dessutom en fil öppnas. Den längsta tiden för dessa instruktioner är 1.5 sekund. Detta innebär att samplingsintervallet ej får vara mindre än denna tid om man skall göra flera Mjölkningar. Hur snabbt man kan sampla bestäms också av hur lång tid det tar att exekvera interruptrutinen. Loggningsprogrammet exekverar denna rutin varje gång ett tidsavbrott inträffar (vid varje nytt sampel). I rutinen ingår bla. insamling av de analoga värdena samt konvertering av dessa.

Nedan visas hur de olika tiderna står i relation till varandra.



X= Nytt sampel.

Viktiga max-tider för en vanlig diskett.

Öppna en fil	Stänga en fil	Skriva in ett värde då append gjorts
1.5 sek.	1.5 sek.	40 ms.

Viktiga tider för en Winchester.

Öppna en fil	Stänga en fil	Skriva in ett värde då append gjorts
0.8 sek.	0.7 sek.	38 ms.

5.3. Minsta möjliga samplingsintervall

Den tid det tar att exekvera interruptrutinen bestämmer i huvudsak hur fort man kan sampla. Det som tar tid i denna rutin är AIN kommandot. Väljer man en integrationstid på 0.02 sek. för att eliminera 50 Hz störningar tar det lång tid att samla in mätvärden från många signaler, eftersom sampling sker signal för signal.

Nedanstående formler anger hur fort man kan sampla om man ej tillåter flera Mjölkningar. Hur man skall välja samplingsintervall, då flera Mjölkningar skall utföras, illustreras med ett exempel. Det tar olika lång tid att exekvera interruptrutinen beroende på vilket konverteringssätt som valts (omvandlingen till sk. ingenjörsstorheter).

Tiden att exekvera interruptrutinen vid linjär konvertering.

0.20+ 0.04*(antal loggbara signaler) (Vid Mjölkning)
0.20+ 0.08*(antal loggbara signaler) (Vid Direkt överföring)

Tiden att exekvera interruptrutinen vid rot-konvertering.

0.20+ 0.075*(antal loggbara signaler) (Vid Mjölkning)
0.20+ 0.115*(antal loggbara signaler) (Vid Direkt överföring)

Vill man sampla ännu fortare än ovanstående formler måste interruptrutinen modifieras, se Appendix C.

Exempel.) Flera Mjölkningar skall utföras.

Antag att 10 signaler skall samplas och att konverteringssättet är linjärt för samtliga signaler.
Tiden för att gå igenom interruptrutinen = 0.20 +0.04*10= 0.6 sek.
Tiden att överföra 10 värden till disk = 0.04*10 =0.4 sek.
Tiden för att öppna eller stänga en fil = 1.5 sek.
Alltså måste samplingsintervallet väljas så att det är större än 1.5 sek., annars kan sampelvärden gå förlorade då en fil öppnas eller stängs. Om ett samplingsintervall på 1.5 sek. väljs, är tillgänglig tid för överföring av sampelvärden till disk mellan två efterföljande sampel 0.9 sek (1.5-0.6). Under denna tid hinner man alltså överföra mätvärden för två sampel. Om det finns 1800 RAM platser skulle det alltså ta ungefär 7 minuter innan Mjölkningen är klar.
Innan kommandot M CTRLB ges bör man ha räknat ut hur lång tid överföringen till disk kommer att ta. Det kan kanske vara idé att stänga av loggningen och starta upp igen.

6. Diskussion av programmet.

Det skapade loggningsprogrammet är uppbyggt av procedurer och funktioner, främst för att underlätta förståelsen av programmet. Alla procedurer och funktioner ligger på nivå 2, se kapitel 3.2.2. 16 k bytes av programmet ligger lagrat i EPROM. Många av de procedurer och funktioner som utnyttjas ofta ligger där, t ex. att placera ut markören på bildskärmen, att radera bildskärmen, att skriva ut inmatad text till bufferten för bildskärmen, etc. Det resterande programmet ligger lagrat i RAMarean. Programmet fyller ungefär 55 k bytes av RAMarean, då är även den dataarea som alla variabler behöver inkluderad.

μ-MacBasic är i många avseenden bättre än standard Basic. Det är lättare att strukturera program i μ-MacBasic, främst genom att det finns blockstrukturer (se kapitel 3.2.5), procedurer och funktioner. Dessutom är det möjligt att deklarera variabler. I μ-MacBasic finns även en del specialinstruktioner för att hantera avbrott och behandla in-/ut-enheter.

μ-Macen har en del nackdelar. Den största nackdelen är att det ej går att ta bort enstaka deklarerade variabler. För att ta bort dessa variabler måste programmet överföras till en IBM-PC. Sedan får man utnyttja IBMens editeringsmöjligheter. Genom att man ibland måste använda en IBM-PC vid vissa editeringar förlorar man mycket tid. Den mesta tiden åtgår för att överföra programmet mellan datorerna. En annan nackdel med μ-Macen är att RAMarean är ganska liten. När stora program skall skapas räcker inte RAMarean till (dock kan den nu utvidgas till 128 k bytes). Då är man tvungen att komprimera några procedurer/funktioner. Upptäcker man vid testning att det är något fel i någon komprimerad procedur/funktion, kan den ej editeras. Det är viktigt att kopior av original procedurerna/funktionerna finns lagrade på en diskett, annars är det ej möjligt att editera dem. Har man ingen kopia lagrad får procedurena/funktionerna skrivas in från början igen till μ-Macen. Vid utveckling av stora program kan mycket tid gå till spillo.

7. Framtida utvidgningar.

7.1. Starta loggningen via digital in.

För att starta loggningsprogrammet via en digital insignal måste vissa ändringar göras i programmet.

Hela initieringsbiten kan man låta vara oförändrad, se Appendix D. Då man anger att loggningen skall starta i initieringsmenyn kommer detta ej att ske förrän ett yttre pulsavbrott genererats. Detta kommer från en digital pulsingång. Pulsräknaren måste initieras i huvudprogrammet. Den skall räkna till ett, dvs. det räcker med en puls på ingången för att utlösa ett interrupt.

Det finns två stycken puls-ingångar på μ -Mac-kortet. De heter kanal 0 och 1. Om kanal 0 används måste interrupt nummer 5 i interruptrutinen få en hopp-adress.

De instruktioner som skall utföras i interruptrutinen vid ett pulsavbrott är följande:

1.) Pulsavbrottet skall stängas av ,görs med instruktionen Clearint(5).

2.) Tidsavbrottet skall sättas på. Rad 290 och rad 300 i Huvudprogrammet skall överföras till denna rutin, se Appendix F sid F-4. Raderna skall ej finnas kvar i Huvudprogrammet.

3.) I tidsavbrottsrutinen skall samtliga satser som gäller då First_in_RAM= 1 även finnas i denna rutin, se Appendix F sid F-5 och F-6.

7.2. Ändra samplingsintervall

Här skall diskuteras en programstruktur där samplingsintervallet kan ändras. Programmet skall startas av operatören genom ett kommando, dvs. enligt det befintliga loggningsprogrammet. Sedan börjar insamlingen av mätdata med ett konstant samplingsintervall. Om en puls kommer in på pulsingången ändras samplingsintervallet till ett fördefinerat värde. Detta värde måste ha definerats i en tabell i initieringsmenyn, se schemat på sid 46 Tabellen består av ett antal samplingsintervall och hur lång tid sampling skall ske med dessa samplingsintervall. Då en puls kommer in på pulsingången börjar man alltid med det översta samplingsintervallet i tabellen. Detta samplingsintervall används nu i X1 sek. (X1 är ett tal) därefter byts samplingsintervallet till nästa värde i tabellen.

Detta samplingsintervall används i X2 sek, osv. Då man klarat av det sista värdet i tabellen skall övergång ske till sampelvärdet som användes vid uppstart. Nederst på sid 46 visas ett tidsschema hur samplingsintervallet ändras.

Då detta program skall skapas måste man använda två stycken tidsavbrott (tidsavbrott 1 och tidsavbrott 2). Tidsavbrott 1 används för att ge rätt samplingsintervall. Tidsavbrott 2 anger hur länge man skall använda ett samplingsintervall. Nedan anges vilka avbrott som skall behandlas i interruptrutinen och vad som skall göras.

Tidsavbrott 1.

Avläsning av de analoga ingångarna och avläsning av klockan. Dessa värden lagras i RAMet.
Ändra Ringbuffertens pekare.
Vid direkt överföring , överför insamlade värden till IBM-PCn.

Tidsavbrott 2.

Här sker byte av samplingsintervall, dvs. tidsavbrott 1 och 2 initieras med nya värden.

Pulsavbrott

Byte av samplingsintervall. Första värdet i den fördefinerade tabellen skall väljas (tidsavbrott 1 och 2 initieras).

Kommunikationsavbrott

Här kan loggningen avslutas med ett kommando från operatören. Dessutom kan kommandon ges för att t ex. utföra en Mjölkning.

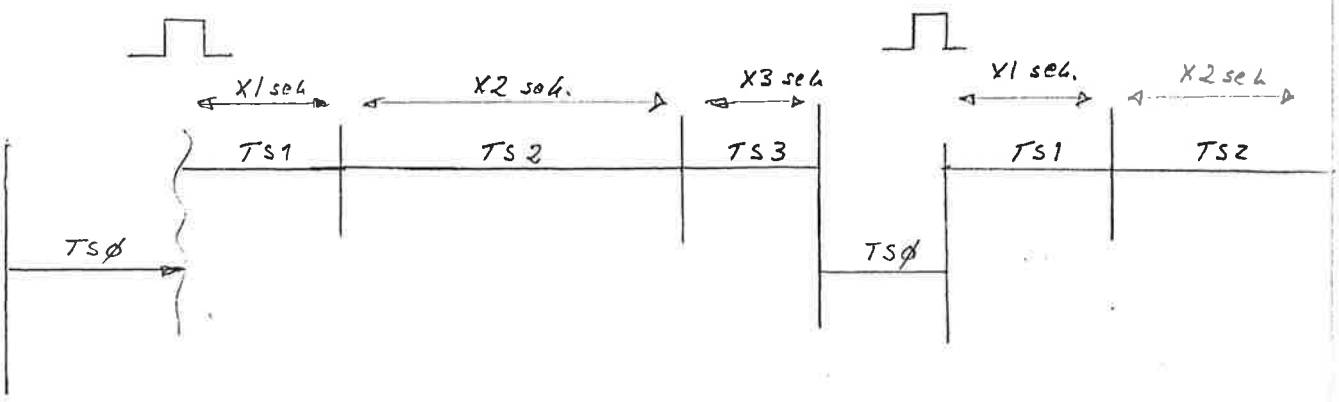
Då samplingsintervallet kan ändras är det lämpligt att vid varje nytt sampel lagra både signalens värde och tidpunkten för avläsningen i μ -Macens RAMarea. Detta skall göras i interrupt-procedurens tidsavbrottsdel, för tidsavbrott 1. Värdena lagras i två stycken endimensionella vektorer.

Vill man utnyttja det befintliga loggningsprogrammet då detta program skall skapas är man tvungen att göra ganska omfattande modifieringar. Det man kan använda är: Nästan hela initieringsdelen, bitar av huvudprogrammet, stora delar av interruptrutinen och menyn för visning av senast samplade värden.
Tidsavbrott 1 skall se ut som det befintliga loggningsprogrammets tidsavbrott, förutom att tidpunkten för avläsningen också måste lagras i μ-Macens RAMarea.
Meny 3 (filmenyn) i initieringsdelen måste ändras, se Appendix D sid D-5. I menyn skall raden som anger när RAMarean blir full tagas bort.
En extra meny måste tillfogas programmet. Där skall anges vilka samplingsintervall som skall genomgås. Den kan se ut enligt sid 46.
Plottningsprogrammet kan man ej använda.
Underprogrammen Dokumentfil, Mjolka, Stang_av_loggningen, Mjolka_till_direkt och Direkt_till_mjolka överför insamlade data till disk. Dessa underprogram måste även överföra tidpunkten för varje sampel, annars kan de ej användas.

Meny för de samplingsintervall som skall genomgås.

	Samplingsintervall (i sek.)	Den tid samplingen skall pågå (i sek.)
1	TS1	X1
2	TS2	X2
3	TS3	X3
4		
5		

En digital insignal gör att samplingsintervallet kommer att ändras efter ett speciellt mönster, se ovan



7.3. μ -Mac 5000 ansluten till en kassettstation.

Vid Direkt överföring överförs data kontinuerligt från μ-Macen till en fil på en IBM-PC. I stället för att ockupera en IBM-PC kan det vara önskvärt att överföra dessa data till en kassettstation. Frontec-Sern i Göteborg har tagit fram den programkod som behövs för att bla. lagra data på en liten TEAC-typ kassett.

Port P6 på μ-Macen används för att kommunicera med kassetten, via signalsnittet RS-232.

Förutom att lagra data på kassetten är det möjligt att:

1.) Redigera filer, t ex. skapa och ta bort filer.
2.) Läsa filer från kassetten.
3.) Lagra utvecklade program.

Alla de drivrutiner som behövs för att klara av det ovanstående finns lagrat i EPROM. Dess minnesbehov är 7 k byte.
Datautrymmet per sida på ett kassettband är 240 k byte.
Filbiblioteket kan maximalt bestå av 64 st. filer.

En nackdel med kassetten är att då data överförs till kassetten, ger den ej ifrån sig någon signal om bandet skulle bli fullt.
Lagrade data på kassettbandet kan överföras till en IBM-PC men kommunikationen måste ske via μ-Macen. Den programkod som behövs i μ-Macen får man själv skriva.

7.4. Anslutning till skrivare.

För att loggningsprogrammet skall kunna kommunicera med en skrivare krävs att vissa rutiner läggs till programmet.

Tillägg till interruptrutinen (kommunikationsavbrott), se Appendix F sid F-7:

```
Do If Len(Kommando$)>=3
        If Left(Kommando$,2)=PR Then Printerflag=1
End Do
```

Tillägg till Huvudprogrammets loop, se Appendix F sid F-4 (rad 330-410):

```
If Printerflag=1 Then Printerflag=0 : Utskrift
```

Printerflag är en flagga som blir satt då kommunikationsavbrottet PR CTRLB ges från tangentbordet. Utskrift är en procedur som skriver ut de konverterade värdena i RAMet till en skrivare.

Utskriftsrutinen bör innehålla:

1.) Val av vilka signaler som skall skrivas ut.
2.) Från vilket sampel utskriften skall ske.
3.) Om ett begränsat antal värden skall skrivas ut.

I rutinen skall det även ingå instruktioner som upprättar kommunikationen med skrivaren. Det sker med instruktinen COM . Skrivaren skall vara ansluten till port P6 på μ -Macen. Kommunikationen sker seriellt.

8. Referenser.

μMacBasic Command Reference
1983 Analog Device

μMac-5000 Operation
1984 Analog Device

μMacBasic Concepts
1983 Analog Device

μMacBasic Programming
1983 Analog Device

μMac-4010 Analog Input Expander Board
1983 Analog Device

μMac-5000 WOS IBM-PC Version
1984 Analog Device

IBM-PC Disk Operating System
Version 2.1
1983 Microsoft Corp.

IBM-PC Basic
1983 Microsoft

VTERM
1984 Coefficient Systems Corporation

9. Appendix

9.1. Appendix A. Programmets Ringbuffert.

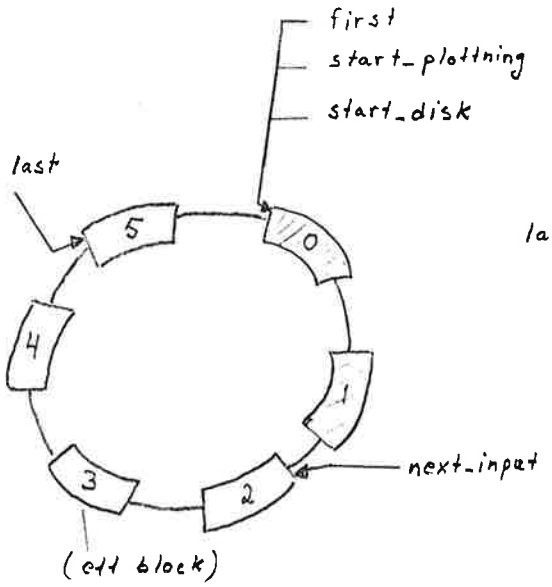
Ringbufferten i programmet är en endimensionell vektor som används för lagring av de konverterade insamlade mätvärdena. Ringbufferten innehåller ett antal block, där ett block innehåller mätvärdena från ett sampel. Antalet mätvärde i ett block bestäms alltså av hur många signaler som loggas.

Innan en loggning får starta räknas det ut hur stor Ringbufferten skall vara, dvs. hur många block Ringbufferten skall innehålla. Då t ex. en endimensionell vektor på 1800 platser används och man loggar 7 signaler, kommer Ringbufferten att innehålla 257 st. block eller 1799 platser. Antalet signaler som loggas måste alltså vara jämt delbart med antalet platser i Ringbufferten.

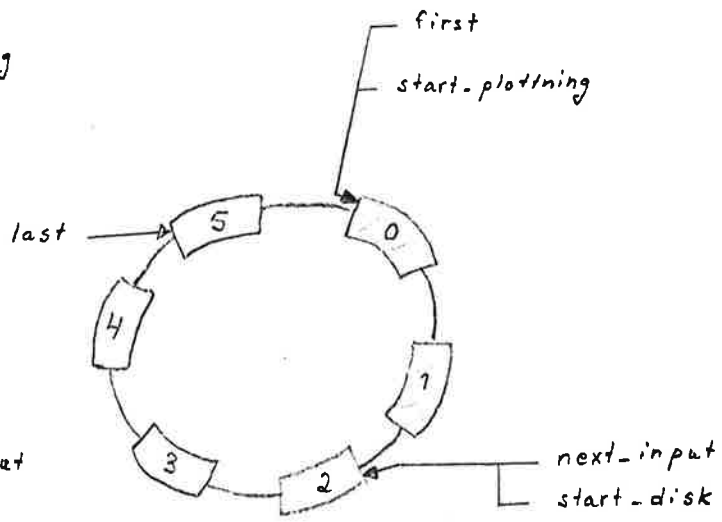
Det första blocket i Ringbufferten har numret 0. Nedan visas en Ringbuffert med 6 block och de pekare som loggningsprogrammet utnyttjar.

Exempel 1 visar pekarnas läge efter 2 sampel (Mjölkning).
Exempel 2 visar pekarnas läge efter 2 sampel (Direkt överföring).

Exempel 1



Exempel 2



first-	Anger första sampelplatsen i Ringbufferten.
last-	Anger sista sampelplatsen i Ringbufferten.
next_input-	Data-platsen för nästa sampel.
start_disk-	Anger från vilket sampel data skall överföras till disk.
start_plottning-	Anger det sampel där plottningen kan börja.

9.2. Appendix B. Datastrukturen för lagrade filer.

KALLE.DOK

```
   15     2     0     0     0     0     0     0     0
*PEREXJ.*         1
test        volt              0       1                    0                  3
test2       volt              0       1                    0                  3
 5                    6/ 12/ 85         12:22:42.89         10
```

Ovan visas hur en Dokumentationsfil ser ut. Den är överförd från μ -Macen till en fil på IBM-PCn. Alla rader är skrivna i ASCII enligt de FORTRAN-format som anges nedan.

Rad 1 är skriven på formatet 10I5.

pos 1 -	Antalet tidpunkter (antal sampel).
pos 2 -	Antal signaler som loggas.
pos 3- pos 9 -	Utnyttjas ej, skall vara 0.
pos 10 -	Antalet textrader, den översta räknas ej.

Rad 2 är skriven på formatet A80. Anger vem som skapat loggningsprogrammet.

Rad 3 och Rad 4 är skrivna på formatet A80. Anger initieringsparametrarna för varje signal.

pos 1 -	Signalbeskrivning, max 8 tecken.
pos 2 -	Enhet på konverterad signal, max 8 tecken.
pos 3 -	Linjärt eller Sqr.(0=linjärt, 1= sqr.).
pos 4 -	k-värdet.
pos 5 -	m-värdet.
pos 6 -	Mätområdet för signalen (kod).

Rad 5 är skriven på formatet A80.

pos 1 -	Samplingsintervall i sek.
pos 2 -	Datum då loggningen startade (månad/ dag/ år).
pos 3 -	Tidpunkt då loggningen startade.
pos 4 -	Överskriden tid i RAMarean (Mjölkningen har ej skett i tid). Tiden anges i sek.

KALLE.DAT

```
1.34037E-01
1.34087E-01
1.34083E-01
1.33945E-01
1.34048E-01
1.33930E-01
1.33945E-01
1.33938E-01
1.33938E-01
1.33930E-01
1.34068E-01
1.34056E-01
1.33923E-01
1.33915E-01
1.34056E-01
1.34052E-01
1.33911E-01
1.33911E-01
1.33945E-01
1.34037E-01
1.33900E-01
1.34007E-01
1.33896E-01
1.33884E-01
1.33892E-01
1.33884E-01
1.33892E-01
1.33881E-01
1.34022E-01
1.34010E-01
```

Ovan visas Datafilen som hör till Dokumentationsfilen på föregående sida. Även denna fil har överförts från μ-Macen till IBM-PCn. Mätvärdena lagras under varandra enligt FORTRAN formatet E16.5 . De två första värdena i Datafilen ovan tillhör sampeltidpunkt 1 och de efterföljande två tillhör sampeltidpunkt 2., osv.
Det översta värdet hör alltså till signalen "test" och värdet under hör till signalen "test2".

KALLE.T

```
   21    2    0    0    0    0    0    0    0    5
*PEREXJ.*        1
test       volt              0       1                    0                3
test2      volt              0       1                    0                3
 5                    6/ 12/ 85        12:22:42.89
Sampeltidpunkt 1- 2 saknas. (data for 10 sek.)
      1.34037E-01      1.34087E-01
      1.34083E-01      1.33945E-01
      1.34048E-01      1.33930E-01
      1.33945E-01      1.33938E-01
      1.33938E-01      1.33930E-01
      1.34068E-01      1.34056E-01
      1.33923E-01      1.33915E-01
      1.34056E-01      1.34052E-01
      1.33911E-01      1.33911E-01
      1.33945E-01      1.34037E-01
      1.33900E-01      1.34007E-01
      1.33896E-01      1.33884E-01
      1.33892E-01      1.33884E-01
      1.33892E-01      1.33881E-01
      1.34022E-01      1.34010E-01
      1.34007E-01      1.34007E-01
      1.34014E-01      1.33930E-01
      1.34007E-01      1.34007E-01
      1.33865E-01      1.33858E-01
      1.34007E-01      1.33991E-01
      1.33858E-01      1.33846E-01
```

Basic programmet KONVERT som skall exekveras på IBM-PCn överför samtliga Datafiler och Dokumentationsfiler med samma namn till en fil, benämnd namn.T (namn=filnamnet). Denna fil skickas sedan över till en VAX-780 dator med hjälp av VTERM programmet. Filen namn.T har det rätta dataformatet för IDPAC, som sedan kan användas.

Filhuvudet på den skapade filen namn.T skiljer sig lite åt från den tidigare Dokumentationsfilen. På Rad 5 anges inte överskriden tid i RAMaren, istället anges det nu på raden under. Där anges även vilka sampeltidpunkter som saknas.

Datafilen är nu skriven på formatet 5E16.5. Filen ovan har på varje rad två sampelvärden. Dessa kommer från samma sampel, där det första värdet tillhör signalen "test" och den andra tillhör signalen "test2".

Det är möjligt att editera in ytterligare ASCII-information till den skapade filen på VAXen. Dessa rader skall tillfogas innan uppräkningen av mätvärdena börjar, dvs. efter uppräkningen av de sampelvärden som saknas. Antalet inledande textrader i filen måste ändras, talet som finns i pos.10 på rad 1. Antalet tillfogade rader skall adderas till detta tal.

9.3. Appendix C. Uppsnabbning av interruptrutinen.

Det kan vara önskvärt att samla in data till μ -Macen så fort som möjligt och därefter avsluta loggningen innan RAMarean blir full. När detta görs får inga andra programfunktioner utnyttjas (t ex. plottning). Om ett kommunikationsavbrott inträffar måste dess interruptrutin genomgås. Under denna tid kan sampelvärden gå förlorade. Kommunikationsavbrottet bör därför endast användas då loggningen skall avslutas. Interruptrutinen för tidsavbrott skall vara så snabb som möjligt och innehålla så få satser som möjligt.
Nedan anges de modifieringar som behöver göras i programmet. Allt som börjar med en asterix och en siffra refererar till programmet på näst-kommande sida.

*1.) Här räknas starttidpunkten för loggningen ut. Dessa satser bör placeras i Huvudprogrammet, där tidsavbrottet initieras (se Appendix F sid F-4). För att få rätt starttidpunkt måste man lägga till den tid det tar innan första tidsavbrottet skall inträffa (ungefär lika med samplingsintervallet).

*2.) De insamlade mätvärdena skall ej konverteras.

*3.) De inlästa värdena skall direkt överföras till Ringbufferten. Inläsningen av värdena bör ej ligga i en loop utan enligt följande:

```
Omformad_Insignal(Next_Input*Gen_Integ_Par(5)) = AIN(1)
Omformad_Insignal(Next_Input*Gen_Integ_Par(5)+1) = AIN(2)
osv.
```

Detta innebär att de signaler som skall loggas måste användaren själv ange både i interruptrutinen (enligt ovan) och i initieringsdelen då programmet startas.

Integrationstiden i AIN instruktionen bestämmer väsentligen hur fort det går att sampla. Används en integrationstid på 0.001 sek. och en signal loggas går det att samla in data ungefär var 35te ms ,då konfigurationen är Mjölkning.
Vill man snabba upp interruptrutinen ännu mer måste man gå över till Assembler-programmering. Ett Basic program får bestå av anrop till assembler-rutiner. Dessa rutiner måste lagras i EPROM. Det är viktigt att veta var (adressen) assemblerprogrammen lagras i EPROM, annars anropar man fel assemblerkod från Basicprogrammet. Anrop av Assemblerrutiner skall ske med kommandot CALL adr., adr. anger adressen till Assemblerkoden i EPROMet, se μ-Mac Command Reference.

```
Interrupt Procedure Inlasning
EXTERNAL: First
EXTERNAL: Last
EXTERNAL: Start_plottning
EXTERNAL: Start_disk_uthamt
EXTERNAL: Next_input
EXTERNAL: Gen_integ_par
EXTERNAL: Gen_real_par
EXTERNAL: New_sampel_flag
EXTERNAL: First_in_RAM
EXTERNAL: Insignal
EXTERNAL: Converted_signal
EXTERNAL: Omformad_insignal
EXTERNAL: Par_real
EXTERNAL: Par_integ
INTEGER: I%,Place_in_block,Hour,Min
STRING: Kommando$Ä1GÅ
EXTERNAL: Logg_disp_flag,Logg_Flag,Init_flag,Autoplot71,Plot_flag,
          New_plot_flag,Mjolk_flag,Stang_av_flag
EXTERNAL: Just_start_up
EXTERNAL: Start_date_logg
EXTERNAL: Start_time_logg
REAL: Sek
EXTERNAL: Help_flag,Mjolk_direkt_flag,Direkt_mjolk_flag
INTEGER: Nr%
EXTERNAL: Start_disk,Direkt_flag,Num,Start_tomning,Antal_CTRLB
INTEGER: CTRLB

   10 Rem Procedure Som Tar Hand Om Avbrotten
   20 On Intr Goto 880,880,30,880,880,880,580,880
   30 Rem Tidsavbrott
   40 Do If First_In_RAM=1      :Rem ny loggning eller då byte från Direkt
                                     överföring till Mjölkning har skett.
   50   GTime(Hour,Min,Sek)
   60   Start_Time_Logg=Time$
*1 70   Start_Date_Logg=Date$  :Rem Här Lagras Start-Tidpunkten för Loggningen
   80   Gen_Real_Par(5)=Hour*3600+Min*60+Sek
   90   If Just_Start_Up=1 Then Gen_Real_Par(4)=Gen_Real_Par(5) :
        Gen_Real_Par(2)=Gen_Real_Par(5) :Just_Start_Up=0    :Rem ny loggning
  100 End Do
  110 For I%=0 To 23            :Rem Läser In Signalerna
  120   If Par_Integ(3,I%)=1 Then Insignal(I%)=AIN(I%)      :Rem signalen loggas
  130 Next
  140 Place_In_Block=0
  150 For I%=0 To 23      :Rem Lägger De Inlästa Signalerna I RAMMET
  160   Do If Par_Integ(3,I%)=1
  170     Rem Först Konverteras Signalen
  180     If Par_Integ(1,I%)=0 Then Converted_Signal(I%)=Insignal(I%)*
*2        Par_Real(0,I%)+Par_Real(1,I%)
  190     If Par_Integ(1,I%)=1 Then Converted_Signal(I%)=Par_Real(0,I%)*
          SQR(Insignal(I%))
  200     Rem Nu Lagras Signalen I RAMMET
*3210     Omformad_Insignal((Next_Input Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+
          Place_In_Block)=Converted_Signal(I%)
  220     Place_In_Block=Place_In_Block+1
  230   End Do
  240 Next
  250 Do If First_In_RAM<>1
  260   Do If Gen_Integ_Par(2)=1                :Rem konfigurationen=Mjolkning
  270     Do If Next_Input Mod Gen_Integ_Par(6)=First Mod Gen_Integ_Par(6)
  280       First=First+1
```

9.4. Appendix D. Menyer i loggningsprogrammet

Innehåll

****
********** MÄTINSAMLINGSSYSTEM **********
****

1.Ange nytt datum och klockslag

2.Skapa signaler eller ändra skapade signaler

3.Skapa eller ändra : Filnamn, Konfiguration med omvärlden, Samplingsintervall
Loggbara signaler
Då konfigurationen=mjölkning anges dumpningstidpunkt

4.Startar loggningen

Vid uppstart skall valda siffror anges i ordningen (1-4)
Vid ny loggning kan start ske från punkt 3

Meny 1 (Datemenyn)

Aktuellt datum (månad/dag/år):

Aktuell tidpunkt (tim:min:sek):

Alla ändringar klara (Y/N) :

Signal Nr: 1 Inmatningsmode
Signalbeskrivning: kalle

Möjliga mätområden för signalen

Kod	Mätområde	Installation	Kod	Mätområde	Installation
01	25 mV		06	10 V	100:1 Resistor
02	50 mV		24	4-20mA (0-100%)	Resistor
03	100 mV		25	4-20mA (1-5V)	Resistor
04	1V	50:1 Resistor	26	+-20mA (+-100%)	Resistor
05	5V	100:1 Resistor	27	+-1mA (+-100%)	Resistor

Typ av mätområde för insignalen, kod: 03

Hur skall den uppmätta signalen konverteras
(0=linjärt, y=k*x+m : 1=sqr., y=k*sqr(x): 0
k-värdet: 1
m-värdet: 0

Enhet på konverterad insignal: Volt

Önskas fler signalmenyer (Y/N):

Meny 3 (Filmenyn)

Filnamn: sten

Konfigurationen med omvärlden
(0=Ansluten direkt till disk, 1=Mjölkning): 1

Samplingsintervall i sekunder: 7

De deklarerade signalerna

Signal	Beskrivning	Signal	Beskrivning	Signal	Beskrivning
1	kalle				

Signaler som skall lagras (signal 1, signal 2, ...)
: 1
Loggningen kan pågå, vid mjolkning (dygn:tim:min:sek): 0:0:23:20
Alla ndringar klara (Y/N) :

Val av signaler för visning av de senast samplade värdena

De signaler som utnyttjas är

Signal	Beskrivning	Mätområde	Enhet	Signal	Beskrivning	Mätområde	Enhet
1	kalle	3	Volt				

Ange de signaler som skall visas kontinuerligt. Ange dem på formen
(signal 1,signal 2,...) max 5st. : 1

Visning av de senast samplade värdena

De senaste samplade värdena

Signal nr	1
Beskrivning	kalle
Mätområde	3
Enhet	Volt
Samplat värde Nr: 23	1.336E-01
Konv. värde	
Sampel nr	
21	1.334E-01
22	1.334E-01
23	1.336E-01

E-CTRLB -Avbryter visningen och återhopp sker till Hjälp-menyn

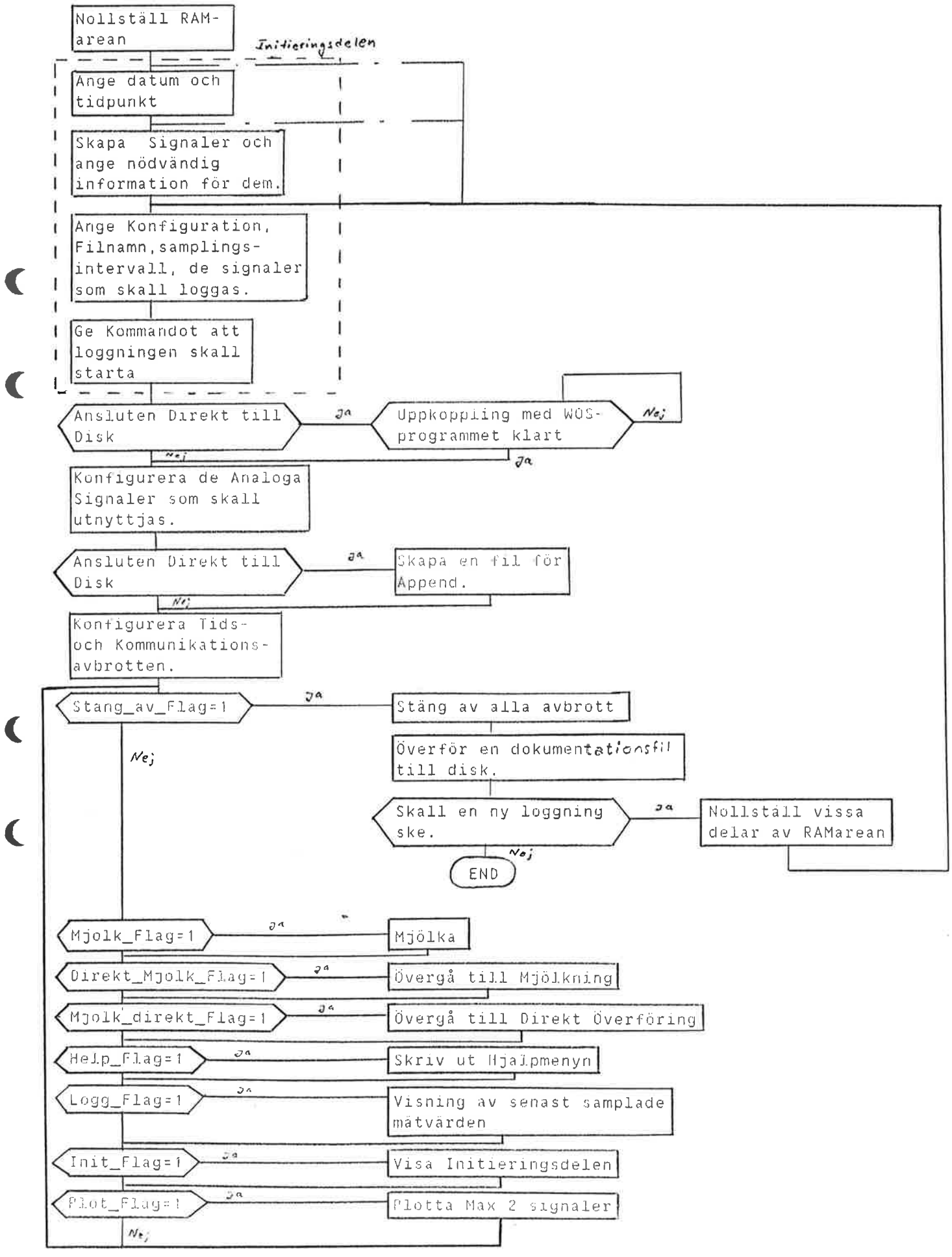
HJÄLP_MENY

CTRLB	:Vid Mjölkning avslutas loggningen. Vid Direkt överföring avslutas loggningen om samplings-intervallet är kort annars övergång till Mjölkning. Använd WOS-programmet.
M-CTRLB	:Överför insamlad data i RAMarean till en fil på disk. Används vid Mjölkning. Använd WOS prog.
S-CTRLB	:Visar de senast samplade värdena. Visar verklig signal och konverterad signal.
I-CTRLB	:Visar Initieringsdelen.
P-CTRLB	:Möjliggör Plottning av max 2 signaler.
H-CTRLB	:Hjälp-menyn
*md-CTRLB	:Byte från Mjölkning till Direkt överföring . WOS-programmet måste användas.
CTRLB--CTRLB	:Övergång från Direkt överföring till Mjölkning, därefter avbryts loggningen. Det andra CTRLB får först komma då den första övergången är klar. Andvänd WOS-programmet.

9.5. Appendix E. Flödesschema för loggningsprogrammet

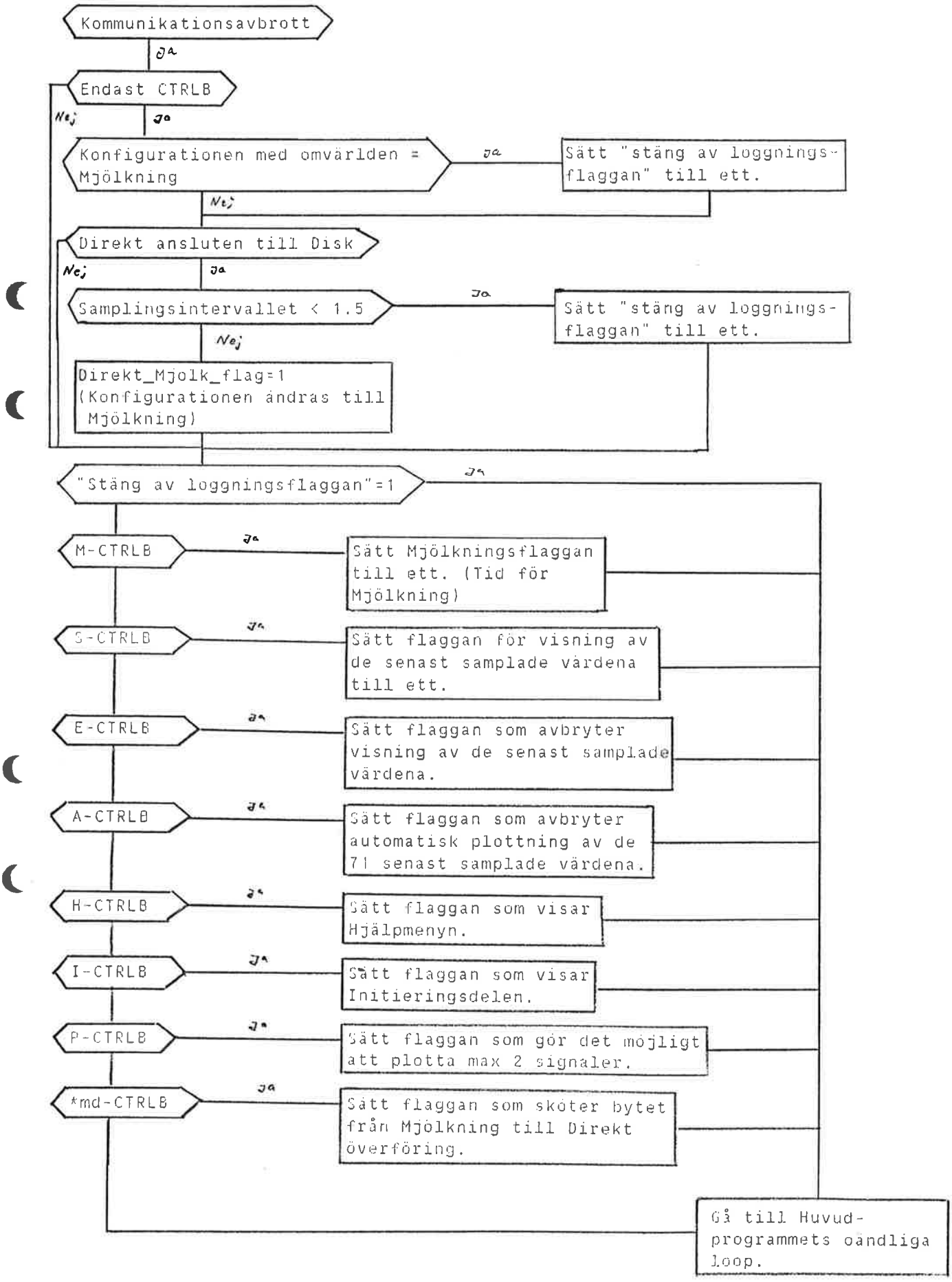
Innehåll

Flodes-schema för Huvudprogrammet.

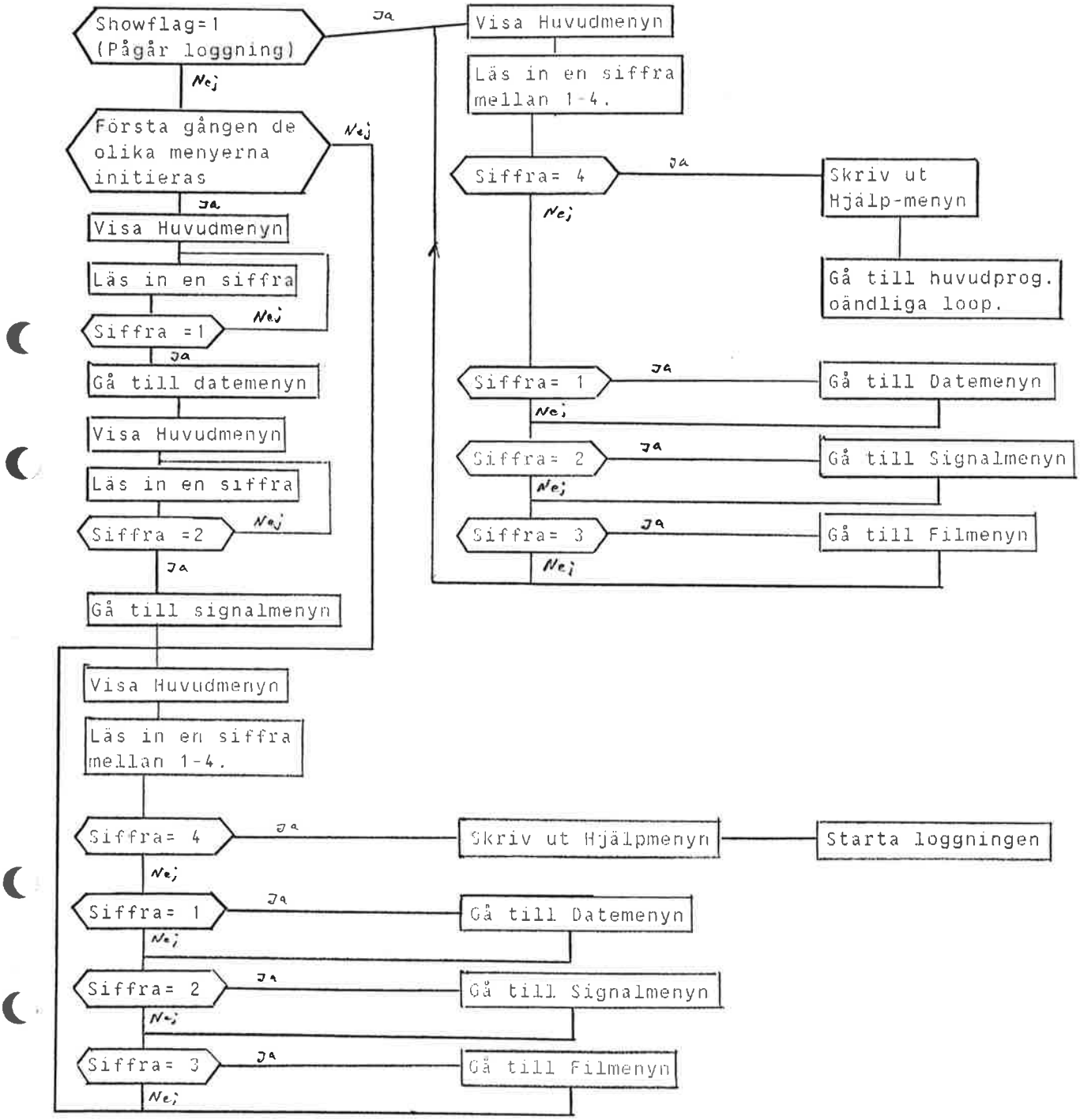


Flödes-schema för Interrupt-rutinen.

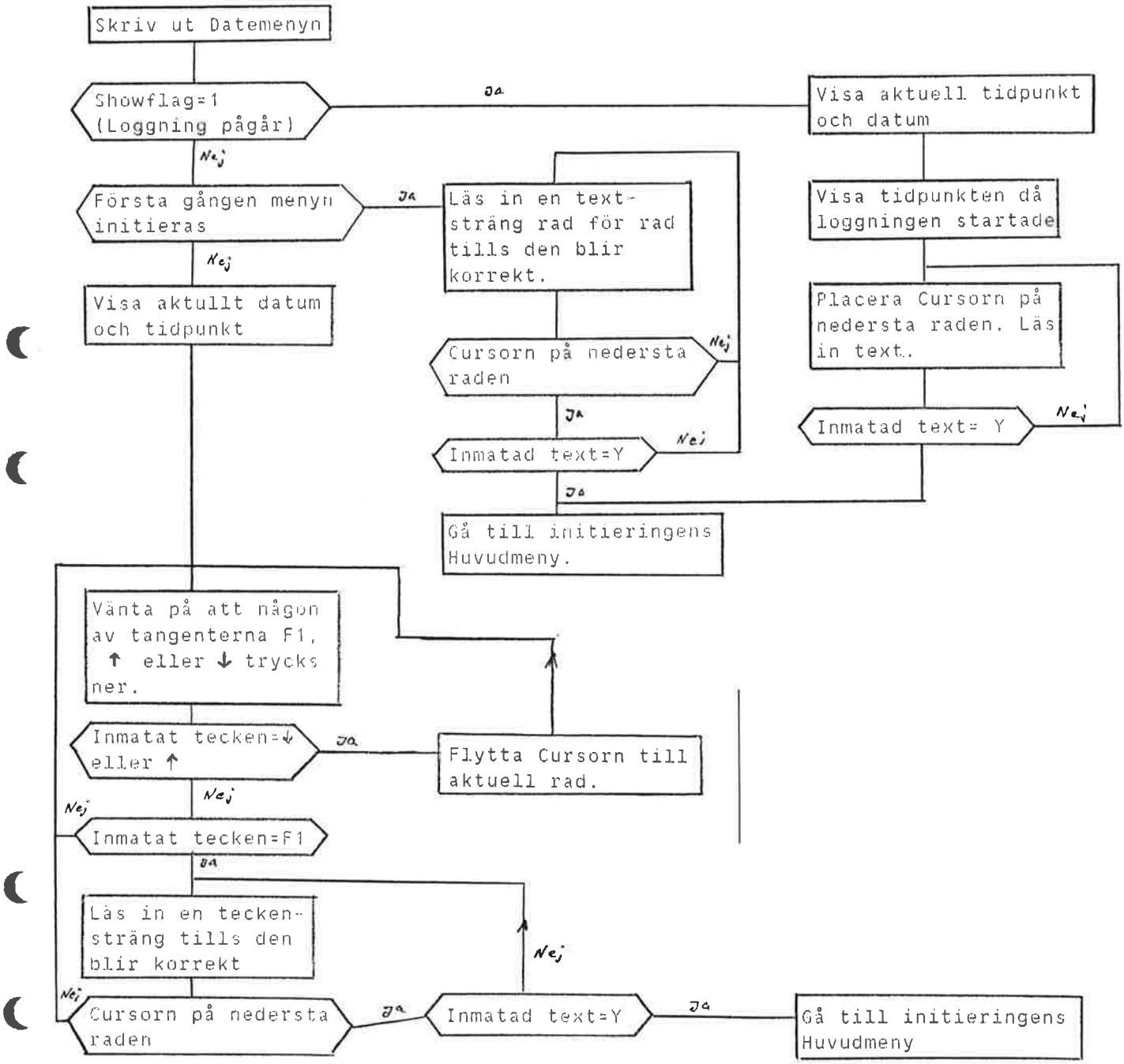
Flödes-schema för interrupt-rutinen



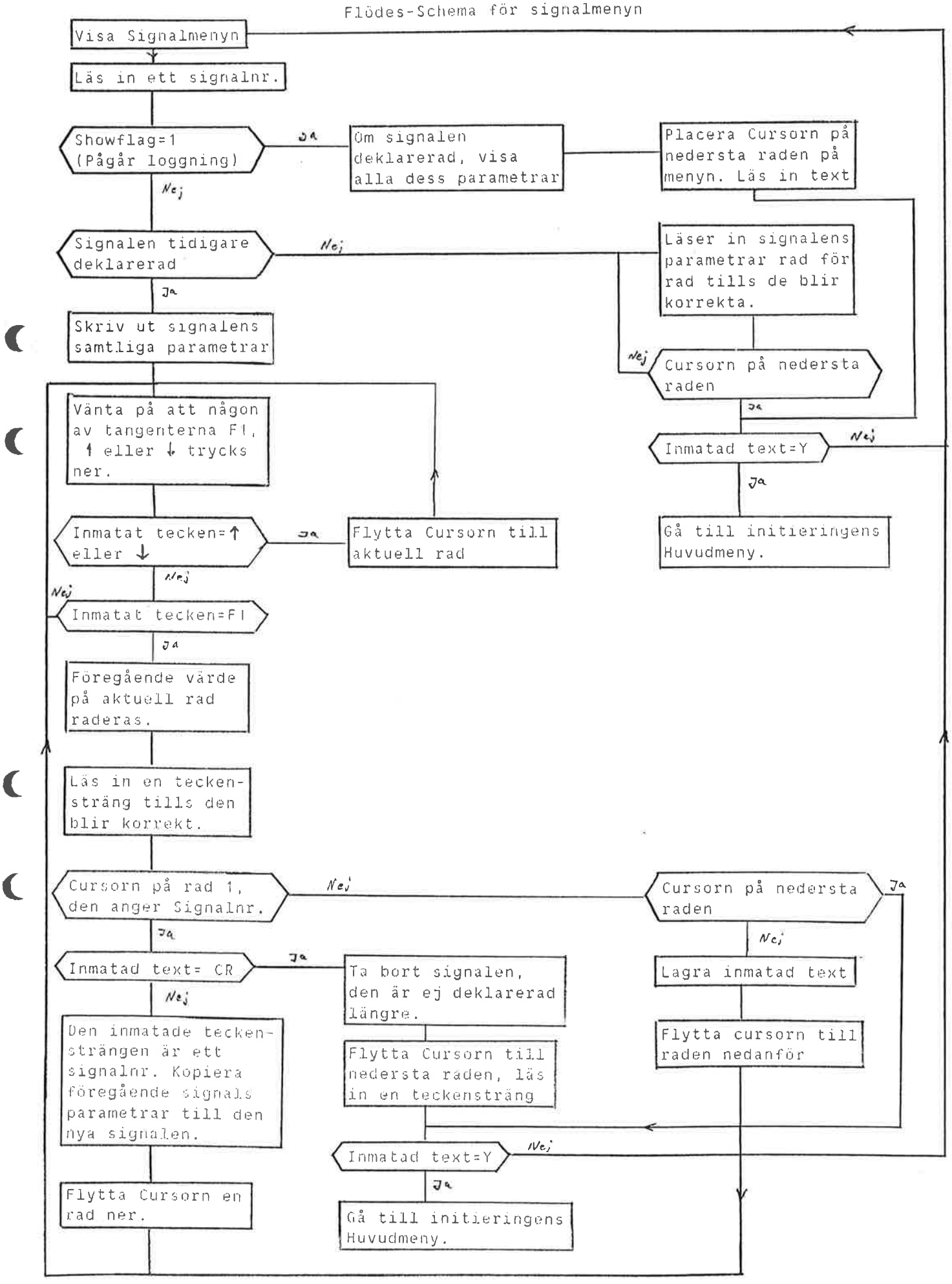
Flödes-schema för initieringsmenyn.



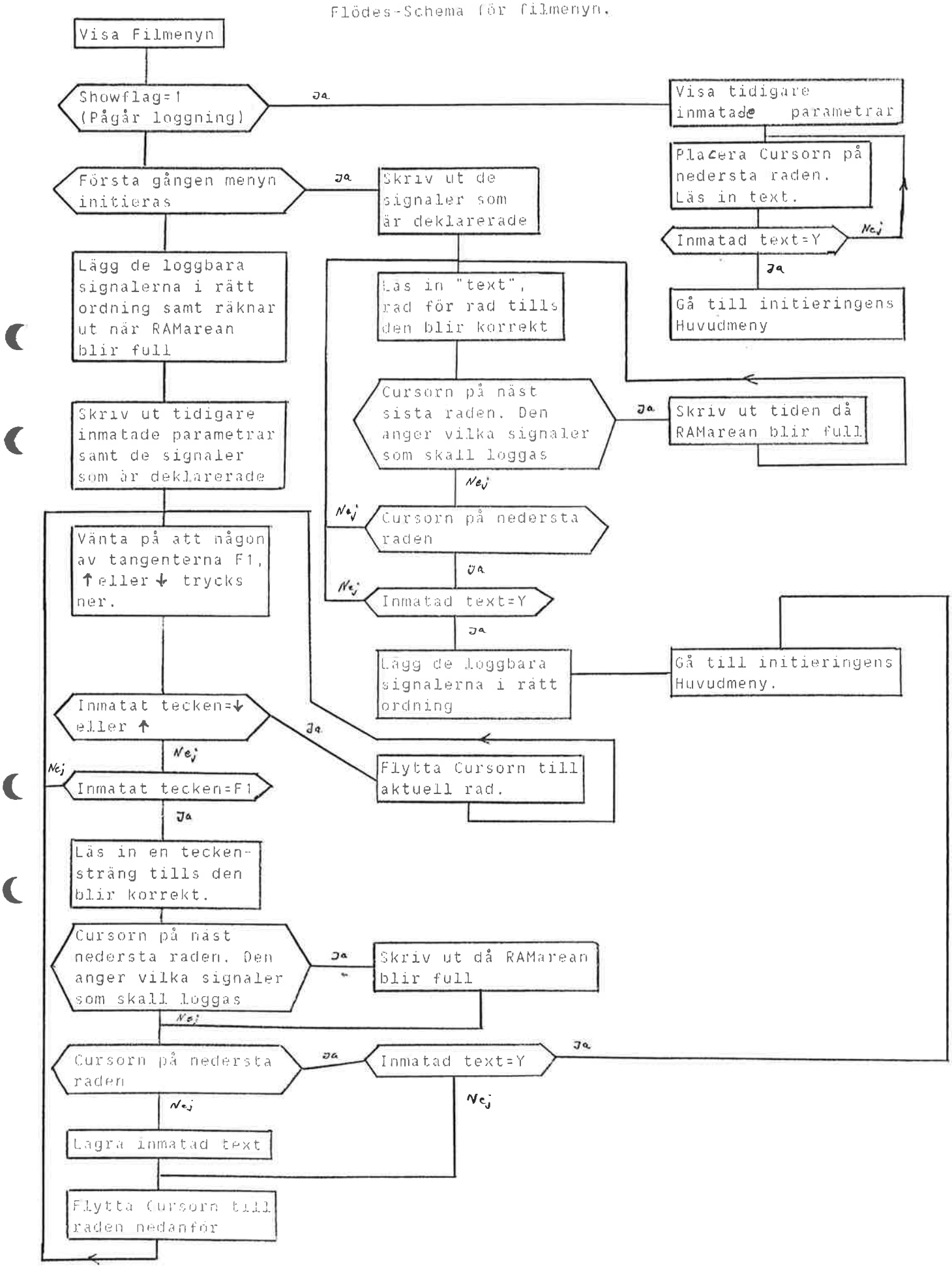
Flödes-schema för Datemenyn.



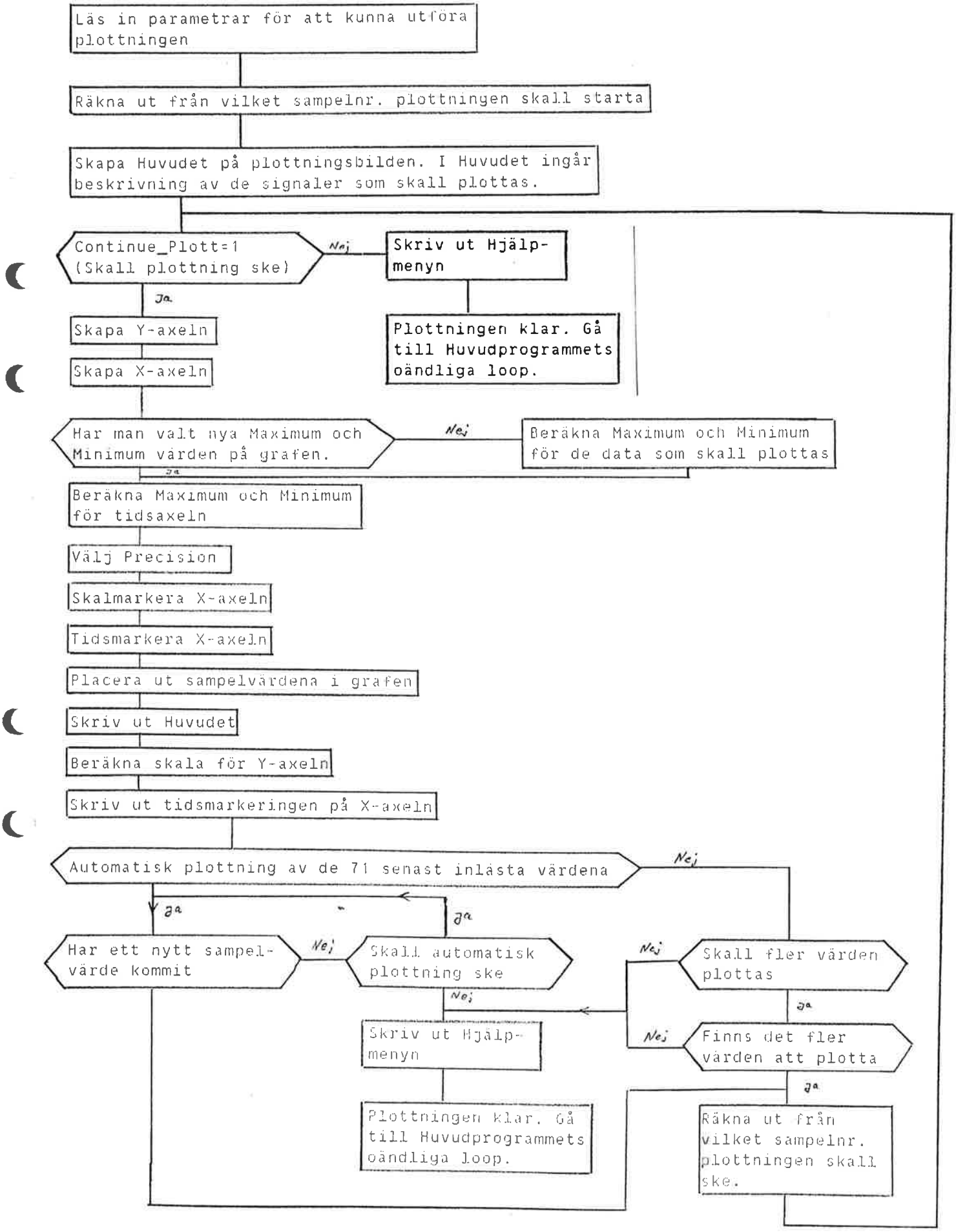
Flödes-Schema för signalmenyn



Flödes-Schema för filmenyn.



Flödes-schema för plottningsprogrammet.



9.6. Appendix F. Programlistningarför Huvudprogrammet och Interruptrutinen

Innehåll

Specifikation över vad vissa variabler betyder i huvudprogrammet.
==========

Par_real(x,y)
==========
Reell variabel som innehåller konverterings-konstanterna för formlerna
a=k*b+m resp. a=k*sqr(b) . (b=Den mätbara signalen, a=Konverterad signal)
y-anger vilken insignal (mellan 0 och 23)
x=0- k-värdet
x=1- m-värdet

Par_str(x,y)
==========
Sträng variabel som är max 8 tecken lång
y-anger vilken insignal (mellan 0 och 23)
x=0- mätenhet på konverterad signal
x=1- signalbeskrivning , upplysningar om signalen

Par_integ(x,y)
==========
Heltals variabel
y-anger vilken insignal (mellan 0 och 23)
x=0- kod ,anger vilket mätområde signalen ligger inom. De tillgängliga mätområdena är angivna i signalmenyn där val av kod skall ske.
x=1- anger hur konverteringen skall ske (0=linjärt, 1=sqr).
x=2- anger om signalen är deklarerad (1=deklarerad,0=ej deklarerad).
x=3- anger om signalen loggas (1=loggas, 0=loggas ej).

Gen_real_par(x)
==========
Reell variabel
x=0- samplingsintervallets längd
x=1- integrationstid, utnyttjas i AIN-kommandot. Är nu vald till 0.02s
x=2- tidpunkten i sek. då loggningen första gången startade. Dvs. ej tidpunkten efter någon mjölkning.
x=3- anger överskriden tid i RAMarean, anges i sek. Kan ske vid mjölkning om man har överskridit den tid då RAMarean blir full.
x=4- anger den tidpunkt då plottning kan starta
x=5- den senaste start-tidpunkten (tex. efter en mjölkning)

Gen_integ_par(x)
==========
Heltals variabel
x=0- anger hur många sampel som hittills gjorts.
x=1- anger numret på den fil som skall lagras nästa gång på disk. Numret ligger mellan 0-99.
x=2- mode. Anger om insamlade data lagras direkt till en fil på disk (Direkt överföring) eller om insamlade data skall lagras vid en senare tidpunkt tex. strax före RAMarean blir full (Mjölkning) Mjölkning=1, Direkt överföring=0
x=3- Används ej
x=4- Anger hur många platser som utnyttjas för lagring av insamlade mätvärden i RAMet.
x=5- antal loggbara signaler
x=6- anger hur många block som får plats i RAMarean .Längden på ett block=antalet loggbara signaler.

Omformad_insignal(x)
====================
Reell variabel
Innehåller samtliga konverterade mätvärden, *dvs. de* som får plats i RAMarean.

Insignal(x)
===========
Reell variabel
Innehåller de senast samplade mätvärdena.

Converted_signal(x)
===================
Reell variabel
Innehåller de senast samplade konverterade mätvärdena.

Huvudprogrammet

```
 10 Clear                          :Rem Nollställer RAMet
 20 File_Mode=0                    :Rem Inget har matats in i Filmenyn
 30 Showflag=0                     :Rem Loggning PåGåR Ej
 40 First_start=1  :Rem forsta gången som startas upp,dvs. ny loggning
 60 Nr_RAM_Places=1800
 70 Main_Declare                   :Rem Initieringsdelen
 80 First_In_RAM=1                 :Rem Ny loggning
 90 Just_Start_Up=1                :Rem Ny loggning
100 Gen_Integ_Par(6)=Gen_Integ_Par(4)/Gen_Integ_Par(5)  :Rem Antal Block
110 Gen_Real_Par(1)=0.02           :Rem integrationstiden
120 Last=Gen_Integ_Par(6)-1        :Rem Pekar På Det Sista Blocket I RAMet
122 Framforhallning=0              :Rem används vid plottning,anger hur många
    sampelvärden som man måste ha samlat in innan plottning kan ske
124 If Gen_real_par(0)>=1 And Gen_real_par(0)<5 Then Framforhallning=2
126 If Gen_real_par(0)<1 Then Framforhallning=5
130 Showflag=1
135 If Gen_integ_par(2)=1 Then Antal_CTRLB=1 Else Antal_CTRLB=0  :Rem Då
    konfigurationen=Mjölkning sätts Antal_CTRLB=1 så att loggningen kan
    upphöra när man begär det
140 Do If Gen_Integ_Par(2)=0  :Rem Direkt ansluten till disk
150   Input "Om uppkopplingen med WOS-programmet klart: skriv KLAR: "Text$
160 Repeat If Text$<>"KLAR"
170 For I%=0 To 23                :Rem Konfigurerar De Analoga Insignalerna
180   Do If Par_Integ(3,I%)=1          :Rem Signalen Loggbar
190     Intype(I%,Par_Integ(0,I%),Gen_Real_Par(1))
200   End Do
210 Next
220 Num=2
230 Do If Gen_Integ_Par(2)=0  :Rem öppnar Filen Om Direkt Ansluten Till Disk
240   Open("o",Filename_Dat,Num)
250   Close(Num)
260   Open("a",Filename_Dat,Num)
270 End Do
280 First_Time_Open=1
290 Setint(3,Gen_Real_Par(0))     :Rem Tidsavbrott
300 Setint(7,2)                   :Rem Komunikationsavbrott
310 On Interrupt Inlasning
320 Do
330   If Stang_Av_Flag=1 Then Stang_Av_Flag=0 : Stang_Av_Loggningen :Exit
340   If Mjolk_Flag=1 Then Mjolk_Flag=0 : Mjolka
350   If Direkt_Mjolk_Flag=1 Then Direkt_Mjolk_Flag=0 :Direkt_Till_Mjolka
360   If Mjolk_Direkt_Flag=1 Then Mjolk_Direkt_Flag=0 :Mjolka_Till_Direkt
370   If Help_Flag=1 Then Help_Flag=0 : Help_Meny_v
380   If Logg_Flag=1 Then Logg_Flag=0 : Logg_Display
390   If Init_Flag=1 Then Init_Flag=0 : Main_Declare
400   If Plot_Flag=1 Then Plot_Flag=0 : Plottning
410 Repeat
420 Do If Ny_Start=1
422   Siffra=Right$(Str$(Gen_Integ_Par(1)),Len(Str$(Gen_Integ_Par(1)))-1)
424   Lage=Instr(1,Filename_Dat,Siffra)
426   Filename_Dat=Left$(Filename_Dat,Lage-1)+"0"+".DAT"
428   Filename_Dok=Left$(Filename_Dok,Lage-1)+"0"+".DOK"
430   Clear(Omformad_Insignal,Converted_Signal,Insignal,First,Last,
      Start_Disk,Next_Input,Start_Plottning,New_Sampel_Flag,Gen_Integ_Par(0),
      Gen_Integ_Par(1),Gen_Real_Par(2),Gen_Real_Par(3),Gen_Real_Par(4),
      Gen_Real_Par(5),Start_Date_Logg)
440   Clear(Start_Time_Logg,Showflag)
450 End Do
460 If Ny_Start=1 Then Ny_start=0 : Goto 60
```

Interrupt Procedure Inlasning

```
Interrupt Procedure Inlasning
EXTERNAL: First
EXTERNAL: Last
EXTERNAL: Start_plottning
EXTERNAL: Start_disk_uthamt
EXTERNAL: Next_input
EXTERNAL: Gen_integ_par
EXTERNAL: Gen_real_par
EXTERNAL: New_sampel_flag
EXTERNAL: First_in_RAM
EXTERNAL: Insignal
EXTERNAL: Converted_signal
EXTERNAL: Omformad_insignal
EXTERNAL: Par_real
EXTERNAL: Par_integ
INTEGER: I%,Place_in_block,Hour,Min
STRING: Kommando$Ä16Å
EXTERNAL: Logg_disp_flag,Logg_Flag,Init_flag,Autoplot71,Plot_flag,
          New_plot_flag,Mjolk_flag,Stang_av_flag
EXTERNAL: Just_start_up
EXTERNAL: Start_date_logg
EXTERNAL: Start_time_logg
REAL: Sek
EXTERNAL: Help_flag,Mjolk_direkt_flag,Direkt_mjolk_flag
INTEGER: Nr%
EXTERNAL: Start_disk,Direkt_flag,Num,Start_tomning,Antal_CTRLB
INTEGER: CTRLB

   10 Rem Procedure Som Tar Hand Om Avbrotten
   20 On Intr Goto 880,880,30,880,880,880,580,880
   30 Rem Tidsavbrott
   40 Do If First_In_RAM=1      :Rem ny loggning eller då byte från Direkt
                                     överföring till Mjölkning har skett.
   50   GTime(Hour,Min,Sek)
   60   Start_Time_Logg=Time$
   70   Start_Date_Logg=Date$  :Rem Här Lagras Start-Tidpunkten för Loggningen
   80   Gen_Real_Par(5)=Hour*3600+Min*60+Sek
   90   If Just_Start_Up=1 Then Gen_Real_Par(4)=Gen_Real_Par(5) :
        Gen_Real_Par(2)=Gen_Real_Par(5) :Just_Start_Up=0    :Rem ny loggning
  100 End Do
  110 For I%=0 To 23            :Rem Läser In Signalerna
  120   If Par_Integ(3,I%)=1 Then Insignal(I%)=AIN(I%)      :Rem signalen loggas
  130 Next
  140 Place_In_Block=0
  150 For I%=0 To 23      :Rem Lägger De Inlästa Signalerna I RAMMET
  160   Do If Par_Integ(3,I%)=1
  170     Rem Först Konverteras Signalen
  180     If Par_Integ(1,I%)=0 Then Converted_Signal(I%)=Insignal(I%)*
          Par_Real(0,I%)+Par_Real(1,I%)
  190     If Par_Integ(1,I%)=1 Then Converted_Signal(I%)=Par_Real(0,I%)*
          SQR(Insignal(I%))
  200     Rem Nu Lagras Signalen I RAMMET
  210     Omformad_Insignal((Next_Input Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+
          Place_In_Block)=Converted_Signal(I%)
  220     Place_In_Block=Place_In_Block+1
  230   End Do
  240 Next
```

```
250 Do If First_In_RAM<>1
260   Do If Gen_Integ_Par(2)=1                    :Rem konfigurationen=Mjölkning
270     Do If Next_Input Mod Gen_Integ_Par(6)=First Mod Gen_Integ_Par(6)
280       First=First+1
290       Last=Last+1
300       Gen_Real_Par(3)=Gen_Real_Par(3)+Gen_Real_Par(0)    :Rem överskriden
                                                               Tid i RAMarean
310     End Do
320     Do If Next_Input Mod Gen_Integ_Par(6)=Start_Disk Mod Gen_Integ_Par(6)
330       Start_Disk=Start_Disk+1                 :Rem överskriden tid i RAMarean
340       Start_Tomning=Start_Tomning+1
350     End Do
360   End Do
370 End Do
380 Do If First_In_RAM<>1
390   Do If Next_Input Mod Gen_Integ_Par(6)=Start_Plottning Mod
      Gen_Integ_Par(6)
400     Start_Plottning=Start_Plottning+1
410     Gen_Real_Par(4)=Gen_Real_Par(4)+Gen_Real_Par(0)
420   End Do
430 End Do
440 Next_Input=Next_Input+1
450 Gen_Integ_Par(0)=Gen_Integ_Par(0)+1    :Rem Totalt Antal Samplade Värden
460 First_In_RAM=0
470 New_Sampel_Flag=1                 :Rem Anger Att Ett Nytt Sampel Har Kommit
480 New_Plot_Flag=1                   :Rem Anger att ett Nytt Sampel har kommit
                                          används vid Autoplottning.
490 Do If Gen_Integ_Par(2)=0          :Rem Direkt överföring till disk
500   For I%=0 To Gen_Integ_Par(5)-1
510     Nr%=(Start_Disk Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+I%
520     Print #Num Using "#.#####^^^^"Omformad_Insignal(Nr%)
530   Next
540   Start_Disk=Start_Disk+1
550   Last=Last+1 :First=First+1
560 End Do
570 Exit
580 On Error Goto 880                 :Rem Kommunikations-avbrott
590 Kommando$=""
600 Readchr Kommando$,Loc(0)
610 CTRLB=0
620 Do If Len(Kommando$)=1            :Rem CTRLB
630   If Asc(Left$(Kommando$,1))=2 Then CTRLB=1
640 End Do
650 Do If Len(Kommando$)=0 Or CTRLB=1
660   If Antal_CTRLB=1 Then Stang_Av_Flag=1  :Exit
670   Do If Antal_CTRLB=0
680     If Gen_Real_Par(0)<1.5 Then Stang_Av_Flag=1 Else Antal_CTRLB=1 :
        Direkt_Mjolk_Flag=1: Print "Övergång från Direkt överföring till
        Mjölkning sker nu."
690   End Do
700 End Do
```

```
  710 If Stang_Av_Flag=1 Then Print "Loggningen avslutas." :Exit
  720 Do If Len(Kommando$)>=1
  730   Do If Gen_Integ_Par(2)=1
  740     If Left$(Kommando$,1)="M" Then Mjolk_Flag=1 :Print "Mjölkning pågår.
          " :Exit                    :Rem MjöLkning
  750     If Left$(Kommando$,1)="S" Then Logg_Flag=1 :Exit   :Rem Visning Av
          de Loggbara Signaler
  760     If Left$(Kommando$,1)="E" Then Logg_Disp_Flag=0 :Exit   :Rem Avbryter
          Visningen Av De Loggbara Signalerna
  770     If Left$(Kommando$,1)="A" Then Autoplot71=0 :Exit  :Rem Avbryter
          Autoplottning
  780     If Left$(Kommando$,1)="H" Then Help_Flag=1 :Exit   :Rem Help-Menyn
  790     If Left$(Kommando$,1)="I" Then Init_Flag=1: Exit  :Rem  Visar
          Initieringsdelen
  800     If Left$(Kommando$,1)="P" Then Plot_Flag=1 : Exit   :Rem MöJligöR
          Plottning
  810     Do If Len(Kommando$)>=3
  820       Do If Left$(Kommando$,3)="*md"  :Rem Byte Fran Mjolka Till Direkt
  830         Antal_CTRLB=0 :Mjolk_Direkt_Flag=1:Print "Övergång från
              Mjölkning till Direkt överföring sker nu."
  840       End Do
  850     End Do
  860   End Do
  870 End Do
  880 Exit
EXIT 1
```

9.6. Appendix F. Programlistningar

Innehåll

Program som skall överföras till RAMarean

KONVERT.BAS

Ett program som sammanfogar flera filer (Datafiler och Dokumentationsfiler) med sammma "namn" till en fil, dvs. filer skapade av loggningsprogrammet från en och samma mätning. Den skapade filen kallas namn.T, "namn" är samma filnamn som ovan. Den skapade filen kan överföras till en VAX-780 dator, där data kan analyseras av identifieringsprogrammet IDPAC.

```
 10 DIM RAD$(30)
 20 DIM TOVERRUN(100)
 30 DIM ANTALSAMPEL%(100)
 40 INPUT "Ange filmamn (max 6 tecken) ";FILENAME$
 50 IF LEN(FILENAME$)>6 THEN GOTO 40
 60 INPUT "Ange filnumret pa den sista filen ";FILENR%
 70 EMPTY$=""
 72 DELSTR1$="Sampeltidpunkt"
 74 DELSTR2$=" saknas. (data for"
 76 DELSTR3$=" sek.)"
 80 FOR S%=1 TO 80
 90     EMPTY$=EMPTY$+CHR$(32)
100 NEXT S%
110 FIRST%=1
120 TOTSAMPEL%=0
130 FOR I%=0 TO FILENR%
140     FILNAMN$=FILENAME$+RIGHT$(STR$(I%),LEN(STR$(I%))-1)+".DOK"
150     OPEN FILNAMN$ FOR INPUT AS #2
160     LINE INPUT #2,A$
170     ANTALSAMPEL%(I%)=VAL(LEFT$(A$,5))
180     TOTSAMPEL%=TOTSAMPEL%+ANTALSAMPEL%(I%)
190     IF FIRST%=1 THEN GOSUB 1000:FIRST%=0 :GOTO 230
200     FOR K%=1 TO ANTALSIGNALER%+2
210         LINE INPUT #2,A$
220     NEXT K%
230     TOVERRUN(I%)=VAL(MID$(A$,53,14))
240     CLOSE #2
250 NEXT I%
260 REM
270 SUM%=0
280 ANTALTEXTRADER%=0
290 FOR I%=0 TO FILENR%
300     SUM%=SUM%+ANTALSAMPEL%(I%)
310     IF TOVERRUN(I%)<>0 THEN ANTALTEXTRADER%=ANTALTEXTRADER%+1
320 NEXT I%
330 ANTALTEXTRADER%=ANTALTEXTRADER%+ANTALSIGNALER%+2
340 REM oversta raden i dok-filen
350 RAD$(0)=LEFT$(EMPTY$,50)
360 FOR I%=15 TO 45 STEP 5
370     MID$(RAD$(0),I%,1)="0"
380 NEXT I%
390 LANGD%=LEN(STR$(TOTSAMPEL%))-1
400 MID$(RAD$(0),6-LANGD%,LANGD%)=RIGHT$(STR$(TOTSAMPEL%),LANGD%)
410 LANGD%=LEN(STR$(ANTALSIGNALER%))-1
420 MID$(RAD$(0),11-LANGD%,LANGD%)=RIGHT$(STR$(ANTALSIGNALER%),LANGD%)
430 LANGD%=LEN(STR$(ANTALETEXTRADER%))-1
440 MID$(RAD$(0),51-LANGD%,LANGD%)=RIGHT$(STR$(ANTALTEXTRADER%),LANGD%)
450 REM
460 OPEN FILENAME$+".T" FOR OUTPUT AS #3
470 CLOSE #3
480 OPEN FILENAME$+".T" FOR APPEND AS #3
485 PRINT #3,RAD$(0)
490 FOR I%=1 TO ANTALSIGNALER%+2
500     PRINT #3,RAD$(I%)
510 NEXT I%
```

```
520 LAST%=1
540 FOR I%=0 TO FILENR%
550     PLACE%=I%
560     ANTAL%=CINT(TOVERRUN(I%)/SAMPELTIME)
570     IF ANTAL%>=1 THEN GOSUB 810
580     LAST%=LAST%+ANTALSAMPEL%(I%)+ANTAL%
590 NEXT I%
610 TOM$="     "
650 SIGNALSTR$=""
660 FOR I%=0 TO FILENR%
670     FILNAMN$=FILENAME$+RIGHT$(STR$(I%),LEN(STR$(I%))-1)+".DAT"
680     OPEN FILNAMN$ FOR INPUT AS #2
690     FOR K%=1 TO ANTALSAMPEL%(I%)
700         ANT%=1
705         FOR L%=1 TO ANTALSIGNALER%
710            LINE INPUT #2,SIGNAL$
720            SIGNALSTR$=SIGNALSTR$+TOM$+SIGNAL$
730            IF ANT% MOD 5 =0 THEN PRINT #3,SIGNALSTR$ : SIGNALSTR$=""
740            ANT%=ANT%+1
745         NEXT L%
747         IF ANT% MOD 5 <>0 THEN PRINT #3,SIGNALSTR$
748         SIGNALSTR$=""
750     NEXT K%
780     CLOSE #2
790 NEXT I%
795 CLOSE #3
796 PRINT "konverteringen ar klar"
797 END
800 REM slut pa huvudprogrammet
810 IF ANTAL%=1 THEN GOSUB 860 ELSE GOSUB 880
830 FELRAD$=FELSTR$+LEFT$(EMPTY$,80-LEN(FELSTR$))
840 PRINT #3,FELRAD$
850 RETURN
860 FELSTR$=DELSTR1$+STR$(LAST%)+DELSTR2$+STR$(TOVERRUN(PLACE%))+DELSTR3$
870 RETURN
880 FELSTR$=DELSTR1$+STR$(LAST%)+"-"+STR$(LAST%+ANTAL%-1)+DELSTR2$+STR$(TOVERRUN
(PLACE%))+DELSTR3$
890 RETURN
1000 REM
1010 ANTALSIGNALER%=VAL(MID$(A$,6,5))
1020 FOR P%=1 TO ANTALSIGNALER%+1
1030     LINE INPUT #2,A$
1040     RAD$(P%)=A$+LEFT$(EMPTY$,80-LEN(A$))
1060 NEXT
1070 LINE INPUT #2,A$
1090 RAD$(ANTALSIGNALER%+2)=MID$(A$,1,48)
1095 RAD$(ANTALSIGNALER%+2)=RAD$(ANTALSIGNALER%+2)+LEFT$(EMPTY$,80-LEN(RAD$(ANTA
LSIGNALER%+2)))
1100 SAMPELTIME=VAL(LEFT$(A$,14))
1110 RETURN
```

Omformad_insignal(x)
====================
Reell variabel
Innehåller samtliga konverterade mätvärden, dvs. de som får plats i RAMarean.

Insignal(x)
===========
Reell variabel
Innehåller de senast samplade mätvärdena.

Converted_signal(x)
===================
Reell variabel
Innehåller de senast samplade konverterade mätvärdena.

Specifikation över vad vissa variabler betyder i huvudprogrammet.
==========

Par_real(x,y)
==========
Reell variabel som innehåller konverterings-konstanterna för formlerna
a=k*b+m resp. a=k*sqr(b) . (b=Den mätbara signalen, a=Konverterad signal)
y-anger vilken insignal (mellan 0 och 23)
x=0- k-värdet
x=1- m-värdet

Par_str(x,y)
==========
Strang variabel som är max 8 tecken lång
y-anger vilken insignal (mellan 0 och 23)
x=0- mätenhet på konverterad signal
x=1- signalbeskrivning , upplysningar om signalen

Par_integ(x,y)
==========
Heltals variabel
y-anger vilken insignal (mellan 0 och 23)
x=0- kod ,anger vilket mätområde signalen ligger inom. De tillgängliga mätområdena är angivna i signalmenyn där val av kod skall ske.
x=1- anger hur konverteringen skall ske (0=linjärt, 1=sqr).
x=2- anger om signalen är deklarerad (1=deklarerad,0=ej deklarerad).
x=3- anger om signalen loggas (1=loggas, 0=loggas ej).

Gen_real_par(x)
==========
Reell variabel
x=0- samplingsintervallets längd
x=1- integrationstid, utnyttjas i AIN-kommandot. Är nu vald till 0.02s
x=2- tidpunkten i sek. då loggningen första gången startade. Dvs. ej tidpunkten efter någon mjölkning.
x=3- anger överskriden tid i RAMarean, anges i sek. Kan ske vid mjölkning om man har överskridit den tid då RAMarean blir full.
x=4- anger den tidpunkt då plottning kan starta
x=5- den senaste start-tidpunkten (tex. efter en mjölkning)

Gen_integ_par(x)
==========
Heltals variabel
x=0- anger hur många sampel som hittills gjorts.
x=1- anger numret på den fil som skall lagras nästa gång på disk. Numret ligger mellan 0-99.
x=2- mode. Anger om insamlade data lagras direkt till en fil på disk (Direkt överföring) eller om insamlade data skall lagras vid en senare tidpunkt tex. strax före RAMarean blir full (Mjölkning) Mjölkning=1, Direkt överföring=0
x=3- Används ej
x=4- Anger hur många platser som utnyttjas för lagring av insamlade mätvärden i RAMet.
x=5- antal loggbara signaler
x=6- anger hur många block som får plats i RAMarean .Längden på ett block=antalet loggbara signaler.

String Function Cursor

```
String Function Cursor
   INTEGER ARG: Y
   INTEGER ARG: X
   STRING: YposÄ2Å,XposÄ2Å

       5 Rem Function Som Placerar ut Cursorn på önskad Plats. Platsen anges
             som (Rad,Kol).
      10 If Y<=9 Then Ypos=Mid$(Str$(Y),2,1) Else Ypos=Mid$(Str$(Y),2,2)
      20 If X<=9 Then Xpos=Mid$(Str$(X),2,1) Else Xpos=Mid$(Str$(X),2,2)
      30 Result=Chr$(27)+Chr$(91)+Ypos+";"+Xpos+Chr$(72)
   EXIT 1
```

Procedure Clear_Screen

```
Procedure Clear_screen

      10 Rem Procedure Som Raderar Skärmen
      20 Print Chr$(27)+Chr$(91)+"2J"
   EXIT 1
```

Procedure Screen_display

```
Procedure Screen_display
   INTEGER ARG: Y
   INTEGER ARG: X
   STRING ARG: Text/VAR
   INTEGER ARG: Maxlangd
   INTEGER: Langd%,Pos%,Antal%
   EXTERNAL: Cursor
   STRING: AÄ1Å
   INTEGER: Klar,Move%

       5 Rem Procedure som skriver ut på skärmen inmatade tecken från
         tangentbordet. De inmatade tecknen lagras i en teckensträngen
         "text".
      10 Langd%=Len(Text)
      20 Pos%=0
      30 Antal%=0
      40 Print Cursor(Y,X+Pos%);
      50 Do
      55   Move%=0
      60   Gosub 420: Rem LäSer In FråN Bufferten
      70   Do If A=Chr$(27)
      80     Gosub 420
      90     Do If A=Chr$(91)
     100       Gosub 420
     110       Rem Flyttar MarköRen åT VäNster Eller åT HöGer
     120       If A=Chr$(67) Then If Pos%=Langd% Then Move%=1 Else
               If Pos%<>Langd% Then Pos%=Pos%+1 : Move%=1 :
               Print Cursor(Y,Pos%+X);
     130       If A=Chr$(68) Then If Pos%<>0 Then Pos%=Pos%-1 : Move%=1 :
               Print Cursor(Y,Pos%+X); Else Move%=1
     140     End Do
     150   End Do
     160   Do If A=Chr$(127)       :Rem DEL
     170     Do If Pos%<>0
     180       Pos%=Pos%-1
     190       If Langd%-Pos%-1>0 Then Print Cursor(Y,Pos%+X);
               Right$(Text,Langd%-Pos%-1);" ";Cursor(Y,Pos%+X);
               Else Print Cursor(Y,Pos%+X);" ";Cursor(Y,Pos%+X);
     200       Klar=0
     210       If Pos%=0 And Langd%-Pos%-1=0 Then Text="" : Klar=1
     220       If Pos%=0 And Klar=0 Then Text=Right$(Text,Langd%-Pos% 1) :
               Klar=1
     230       If Langd%-Pos%-1=0 And Klar=0 Then Text=Left$(Text,Pos%) :
               Klar=1
     240       If Pos%<>0 And Langd%-Pos%-1<>0 Then Text=Left$(Text,Pos%)+
               Right$(Text,Langd%-Pos%-1)
     250       Langd%=Len(Text)
     260     End Do
     270   End Do
```

```
280    Do If A<>Chr$(13) And A<>Chr$(127) And Move%=0
290      Do If Langd%<Maxlangd
300        If Langd%-Pos%=0 Then Print Cursor(Y,X+Pos%);A; Else Print
           Cursor(Y,X+Pos%);A;Right$(Text,Langd%-Pos%);Cursor(Y,X+Pos%+1);
310        Klar=0
320        If Pos%=0 And Langd%-Pos%=0 Then Text=A :Klar=1
330        If Pos%=0 And Klar=0 Then Text=A+Right$(Text,Langd%-Pos%):
           Klar=1
340        If Langd%-Pos%=0 And Klar=0 Then Text=Left$(Text,Pos%)+A
350        If Pos%<>0 And Langd%-Pos%<>0 Then Text=Left$(Text,Pos%)+A+
           Right$(Text,Langd%-Pos%)
360        Langd%=Len(Text)
370        Pos%=Pos%+1
380      End Do
390    End Do
400  Repeat If A<>Chr$(13)
410  Exit
420  While Antal%=0 Do
430    Antal%=Loc(0)
440  Repeat
450  Readchr A,1
460  Antal%=Antal%-1
470  Return
EXIT 1
```

Procedure Clear_rad

```
Procedure Clear_rad
   INTEGER ARG: Y
   INTEGER ARG: X
   INTEGER ARG: Nr
   EXTERNAL: Cursor

       5 Rem Procedure Som Raderar "Nr" Tecken På En Rad Och Placerar Cursorn
             I Position (Rad,Kol)
      20 Print Cursor(Y,X);Chr$(32,Nr);Cursor(Y,X);
   EXIT 1
```

Procedure Move_up_down

```
Procedure Move_up_down
   INTEGER ARG: Meny
   INTEGER ARG: Rad/VAR
   INTEGER ARG: Up_down
   INTEGER ARG: Y/VAR
   INTEGER ARG: X/VAR
   INTEGER: I%,Start

      10 Rem Procedure Som ange vilken position cursorn skall hamna på (för en
             speciell sida) om man trycker på någon av tangenterna "pil upp"
             resp. "pil ner". Up_down anger om "pil upp" eller "pil ner" har
             tryckts.
      20 Do If Meny=1                    :Rem Datemeny
      30   Do If Up_Down=1
      40     If Rad<>1 Then Rad=Rad-1
      50   End Do
      60   Do If Up_Down=0
      70     If Rad<>3 Then Rad=Rad+1
      80   End Do
      90   Data 5,32,10,33,20,30
     100   Restore
     110   For I%=1 To Rad
     120     Read Y,X
     130   Next
     140 End Do
     150 Do If Meny=2                    :Rem Signalmeny
     155   Start=3
     160   Do If Up_Down=1
     170     If Rad<>1 Then Rad=Rad-1
     180   End Do
     190   Do If Up_Down=0
     200     If Rad<>8 Then Rad=Rad+1
     210   End Do
     220   Data 1,12,2,20,13,39,16,46,17,18,18,18,20,32,22,33
     230   Restore
     240   For I%=1 To Rad+Start
     250     Read Y,X
     260   Next
     270 End Do
     280 Do If Meny=3                    :Rem Filemeny
     285   Start=11
     290   Do If Up_Down=1
     300     If Rad<>1 Then Rad=Rad-1
     310   End Do
     320   Do If Up_Down=0
     330     If Rad<>5 Then Rad=Rad+1
     340   End Do
     350   Data 1,10,4,47,6,32,21,4,23,30
     360   Restore
     370   For I%=1 To Rad+Start
     380     Read Y,X
     390   Next
     400 End Do
   EXIT 1
```

Integer Function Up_down_cursor

```
Integer Function Up_down_cursor
   STRING: Ch1Ä1Å,Ch2Ä2Å
   INTEGER: Ok%,Antal%,Ans%

      5 Rem Function Som Anger Vilken Av Tangenterna "CR", "Pil Upp",
            "Pil Ner" eller "F1" som Har Tryckts Ner.
     10 Ok%=0
     20 Do
     30   Do
     40     Antal%=Loc(0)
     50   Repeat If Antal%=0
     60   Readchr Ch1,1
     70   If Ch1=Chr$(13) Then Ok%=1 : Ans%=0        :Rem CR
     80   Do If Ch1=Chr$(27)
     90     Do
    100       Antal%=Loc(0)
    110     Repeat If Antal%<2
    120     Readchr Ch2,2
    130     If Ch2="ÄA" Then Ok%=1 : Ans%=1          :Rem Pil Upp
    140     If Ch2="ÄB" Then Ok%=1 : Ans%=2          :Rem Pil Ner
    145     If Ch2="OT" Then Ok%=1 : Ans%=4          :Rem "F1"
    147     If Ch2="OP" Then Ok%=1 : Ans%=4          :Rem "F1"
    150   End Do
    160 Repeat If Ok%=0
    170 Result=Ans%
   EXIT 1
```

Integer Function Convert_integ

```
Integer Function Convert_integ
   STRING ARG: Text
   INTEGER ARG: Length
   INTEGER: K%,Ans%

      10 Rem Function Som UndersöKer Om "text" äR Ett Heltal ,Ja-1, Nej-0
      20 Ans%=1
      25 If Len(Text)<=0 Then Ans%=0 :Goto 60
      30 For K%=1 To Length
      40   If Left$(Text(K%),1)<"0" Or Left$(Text(K%),1)>"9" Then Ans%=0 : Exit
      50 Next
      60 Result=Ans%
   EXIT 1
```

Integer Function Convert_real

```
Integer Function Convert_real
   STRING ARG: Text
   REAL: Y

       5 Rem UndersöKer Om "text" äR Ett Reelt Tal Ja-1, Nej-0
      10 If Len(Text)=0 Then Result=0 : Exit
      20 On Error Goto 50
      30 Y=Val(Text)
      40 If Len(Text)=Err Then Result=1 : Exit
      50 Result =0
   EXIT 1
```

Procedure Time_converter

```
Procedure Time_converter
   STRING ARG: Tid/VAR
   REAL ARG: Tid_i_sek
   INTEGER ARG: Include_dygn
   INTEGER: Dygn,Tim,Min,Sek
   REAL: Time_konst

       5 Rem Procedure som omvandlar en tid punkt i sekunder till en
             tidsangivelse på formen (dygn:tim:min:sek). Man kan välja
             om dygn skall vara med eller ej.
      10 Dygn=Int(Tid_I_Sek/86400)
      20 Time_Konst=Tid_I_Sek-86400*Dygn
      30 Tim=Int(Time_Konst/3600)
      40 Time_Konst=Time_Konst-3600*Tim
      50 Min=Int(Time_Konst/60)
      60 Time_Konst=Time_Konst-60*Min
      70 Sek=Int(Time_Konst)
      80 If Include_Dygn=1 Then Tid=Right$(Str$(Dygn),Len(Str$(Dygn))-1)+":"
         Else Tid=""
      90 Tid=Tid+Right$(Str$(Tim),Len(Str$(Tim))-1)+":"+Right$(Str$(Min),
         Len(Str$(Min))-1)+":"+Right$(Str$(Sek),Len(Str$(Sek))-1)
   EXIT 1
```

String Function Margin

```
String Function Margin
   INTEGER ARG: Top
   INTEGER ARG: Bottom
   STRING: UppÄ2Å,NerÄ2Å

       5 Rem Function som kan ändra övre och undre marginalgränserna på
             terminalen.
      10 If Top<=9 Then Upp=Mid$(Str$(Top),2,1) Else Upp=Mid$(Str$(Top),2,2)
      20 If Bottom<=9 Then Ner=Mid$(Str$(Bottom),2,1) Else
         Ner=Mid$(Str$(Bottom),2,2)
      30 Result=Chr$(27)+Chr$(91)+Upp+";"+Ner+Chr$(114)
   EXIT 1
```

String Function Save_cursor

```
String Function Save_cursor

       5 Rem Function som sparar Cursorns position.
      10 Result=Chr$(27)+Chr$(55)
   EXIT 1
```

String Function Restore_cursor

```
String Function Restore_cursor

       5 Rem Function som  hämtar den lagrade Cursorns värde.
      10 Result=Chr$(27)+Chr$(56)
   EXIT 1
```

Integer Function Correct_input_d

```
Integer Function Correct_input_d
   INTEGER ARG: Meny
   INTEGER ARG: Rad
   STRING ARG: Text
   EXTERNAL: Convert_integ,Convert_real
   INTEGER: Ans%
   STRING: TeckenÄ1Å
   INTEGER: Nr%
   STRING ARRAY(3)Ä2Å: Tal
   INTEGER: I%,K%
   STRING: NrÄ3Å

      10 Rem Function som undersöker Om Strängen "Text" Innehåller Rätt
            Information (Rätt-1,Fel-0). Sidorna Mainmeny,Datemeny och
            Signalmeny utnyttjar detta underprogram. Varje meny har ett
            speciellt nummer. "Rad" anger var på menyn den inmatade strängen
            kommer ifrån.
      20 Do If Meny=0                  :Rem Mainmeny
      30   Do If Rad=1                 :Rem "siffra mellan 1 och 4"
      40     Ans%=0
      50     Do If Convert_Integ(Text,Len(Text))=1
      60       If Val(Text)>=1 Or Val(Text)<=4 Then Ans%=1
      70     End Do
      80   End Do
      90 End Do
     100 Do If Meny=1                  :Rem  Datemenyn
     110   Do If Rad=1 Or Rad=2        :Rem "datum eller tidpunkt"
     120     Ans%=1
     130     If Rad=1 Then Tecken="/" Else Tecken=":"
     140     Nr%=0
     150     Tal(0)=""
     160     K%=Len(Text)
     170     If K%=0 Then Ans%=0 :Exit
     180     For I%=1 To  K%    :Rem Separerar Talen FråN TeckensträNgen
     190       If Left$(Text(I%),1)=Tecken Then Nr%=Nr%+1 : Tal(Nr%)="" Else
               If Left$(Text(I%),1)>="0" And Left$(Text(I%),1)<="9" Then
               Tal(Nr%)=Tal(Nr%)+Left$(Text(I%),1) Else Ans%=0 :Exit
     200       If Len(Tal(Nr%))>2 Then Ans%=0 :Exit
     210       If Nr%>2 Then Ans%=0 :Exit
     220     Next
     230     On Error Goto 280
     240     If Ans%=0 Or Nr%<>2 Then Ans%=0 :Exit
     250     If Rad=1 Then Sdate(Val(Tal(0)),Val(Tal(1)),Val(Tal(2)))
     260     If Rad=2 Then Stime(Val(Tal(0)),Val(Tal(1)),Val(Tal(2)))
     270     Exit
     280     Ans%=0
     290   End Do
     300   Do If Rad=3                 :Rem "Y/N"
     310     Gosub 650:Rem "Y/N"
     320   End Do
     330 End Do
```

```
 340 Do If Meny=2                          :Rem Signalmeny
 350   Do If Rad=1                         :Rem "signalnr"
 360     If Len(Text)>2 Or Len(Text)=0 Then Ans%=0 :Exit
 370     If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 :Exit
 380     If Val(Text)<0 Or Val(Text)>23 Then Ans%=0 :Exit
 390     Ans%=1
 400   End Do
 410   Do If Rad=2                         :Rem "signalbeskrivning"
 420     Ans%=1
 430   End Do
 440   Do If Rad=3                         :Rem "kode"
 450     If Len(Text)<>2 Then Ans%=0 :Exit
 460     If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 : Exit
 470     If (Val(Text)>=01 And Val(Text)<=06) Or (Val(Text)>=24 And
         Val(Text)<=27)        Then Ans%=1 :Exit
 480     Ans%=0
 490   End Do
 500   Do If Rad=4                         :Rem "linjär,sqr"
 510     Gosub 690
 520   End Do
 530   Do If Rad=5 Or Rad=6                :Rem "k,m-värdet"
 540     If Convert_Real(Text)=0 Then Ans%=0 : Exit
 550     Ans%=1
 560   End Do
 570   Do If Rad=7                         :Rem "enhet"
 580     Ans%=1
 590   End Do
 600   Do If Rad=8                         :Rem "(Y/N)-fler signaler"
 610     Gosub 650
 620   End Do
 630 End Do
 640 Goto 740
 650 If Len(Text)<>1 Then Ans%=0 : Return                    :Rem "Y/N"
 660 If Text<>"Y" And Text<>"N" Then Ans%=0 : Return
 670 Ans%=1
 680 Return
 690 If Len(Text)<>1 Then Ans%=0 : Return   :Rem VäRdena 0 Eller 1 Skall
     Vara Inmatade
 700 If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 : Return
 710 If Val(Text)<>0 And Val(Text)<>1 Then Ans%=0 :Return
 720 Ans%=1
 730 Return
 740 Result=Ans%
EXIT 1
```

Integer Function Correct_input_f

```
Integer Function Correct_input_f
   INTEGER ARG: Meny
   INTEGER ARG: Rad
   STRING ARG: Text
   EXTERNAL: Convert_integ,Convert_real
   INTEGER: Ans%,I%,K%
   STRING: NrÄ3Å
   INTEGER ARRAY ARG(4,24): Deklarerad

     10 Rem Function som undersöker om strängen "Text" innehåller rätt
           information (Rätt-1,Fel-0). Sidorna Filemeny och Show Logg.-
           Signals utnyttjar detta underprogram. Varje Meny har ett
           speciellt nummer. "Rad" anger var på sidan den inmatade strängen
           kommer ifrån.
     20 Do If Meny=3                    :Rem Filemeny
     30   Do If Rad=1                   :Rem "Filnamn"
     40     If Len(Text)>6 Then Ans%=0 Else Ans%=1
     50   End Do
     60   If Rad=2 Then Gosub 240       :Rem "Mjölkning eller direkt överföring"
     70   Do If Rad=3                   :Rem "Samplingsintervall"
     80     If Convert_Real(Text)=0 Then Ans%=0 :Exit
     85     If Val(Text)<=0 Then Ans%=0 : Exit
     90     Ans%=1
    100   End Do
    110   Do If Rad=4             :Rem "Anger vilka signaler som skall Loggas"
    120     Gosub 290:Rem UndersöKer Om De Signaler Som Skall Loggas
                         är Deklarerade
    130   End Do
    140   If Rad=5 Then Gosub 200       :Rem "Y/N"
    150 End Do
    160 Do If Meny=4                    :Rem Visning Av Loggade Signaler
    170   Gosub 290:Rem UndesöKer Om De Signaler Som Skall Visas äR Loggbara
    180 End Do
    190 Goto 580
    200 If Len(Text)<>1 Then Ans%=0 : Return                :Rem "Y/N"
    210 If Text<>"Y" And Text<>"N" Then Ans%=0 : Return
    220 Ans%=1
    230 Return
    240 If Len(Text)<>1 Then Ans%=0 : Return    :Rem VäRdena 0 Eller 1 Skall
                                                  Vara Inmatade
    250 If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 : Return
    260 If Val(Text)<>0 And Val(Text)<>1 Then Ans%=0 :Return
    270 Ans%=1
    280 Return
    290 Ans%=1       :Rem undersöker om de signaler som skall visas är loggbara
    300 Nr=""
    305 If Len(Text)=0 Then Ans%=0 :Return
```

```
    310 For I%=1 To Len(Text)
    320   Do If Left$(Text(I%),1)=","
    330     Gosub 420     :Rem undersöker Om Sinalen äR Deklarerad
    340   End Do
    350   Do If Left$(Text(I%),1)<>","
    360     If ( Left$(Text(I%),1)>="0") And (Left$(Text(I%),1)<="9") Then
            Nr=Nr+Left$(Text(I%),1) Else Ans%=0
    370   End Do
    380   If Ans%=0 Then Exit
    390 Next
    400 If Ans%=1 Then Gosub 420
    410 Return
    420 K%=1              :Rem undersöker om signalen är deklarerad
    430 If Len(Nr)>2 Then K%=0
    440 If K%=1 Then K%=Convert_Integ(Nr,Len(Nr))
    450 Do If K%=1
    460   K%=0
    470   Do If Val(Nr)>=0 And Val(Nr)<=23
    480     Do If Meny=3      :Rem Filemeny
    490       If Deklarerad(2,Val(Nr))=0 Then Ans%=0 Else Nr="":K%=1
    500     End Do
    510     Do If Meny=4      :Rem Visning av loggbara signaler
    520       If Deklarerad(3,Val(Nr))=0 Then Ans%=0 Else Nr="" : K%=1
    530     End Do
    540   End Do
    550 End Do
    560 If K%=0 Then Ans%=0
    570 Return
    580 Result=Ans%
  EXIT 1
```

Procedure Read_sentence

```
Procedure Read_sentence
   INTEGER ARG: Y
   INTEGER ARG: X
   INTEGER ARG: I
   INTEGER ARG: Nr
   INTEGER ARG: Meny
   STRING ARG: Text/VAR
   EXTERNAL: Clear_rad,Screen_display,Correct_input_d

      10 Rem Procedure Som läser In En Tecken -Sträng Och väntar tills den
             blir Ok. Kan endast utnyttjas för menyerna Mainmeny,Datemeny och
             Signalmeny.
      20 Do
      30   Clear_Rad(Y,X,Nr)
      40   Text=""
      50   Screen_Display(Y,X,Text,Nr)
      60 Repeat If Correct_Input_D(Meny,I,Text)=0
   EXIT 1
```

Procedure Date_meny

```
Procedure Date_meny
   INTEGER ARG: Showflag
   EXTERNAL: Cursor,Clear_screen

       5 Rem Procedure Som Skriver Ut DateMenyn.
      10 Clear_Screen
      20 Print Cursor(5,1);"Aktuellt datum (månad/dag/år): "
      30 Print Cursor(10,1);"Aktuell tidpunkt (tim:min:sek): "
      40 Do If Showflag=1
      50   Print Cursor(15,1);"Datum då loggningen startade: "
      60   Print Cursor(17,1);"Tidpunkt då loggningen startade: "
      70 End Do
      80 Print Cursor(20,1);"Alla ändringar klara (Y/N) : "
   EXIT 1
```

Procedure Signal_meny

```
Procedure Signal_meny
   EXTERNAL: Clear_screen
   EXTERNAL: Cursor

       5 Rem Procedure Som Skriver Ut Signalmenyn
      10 Clear_Screen
      20 Print Cursor(1,0);"Signal Nr:"
      30 Print "Signalbeskrivning:"
      40 Print
      50 Print "Möjliga mätområden för signalen"
      60 Print
      70 Print "Kod";Cursor(6,10);"Mätområde";Cursor(6,22);"Installation";
         Cursor(6,40);"Kod";Cursor(6,47);"Mätområde"
      75 Print Cursor(6,65);"Installation"
      80 Print Cursor(7,2);"01";Cursor(7,12);"25 mV";Cursor(7,41);"06";
         Cursor(7,47);"10 V";Cursor(7,65);"100:1 Resistor"
      90 Print Cursor(8,2);"02";Cursor(8,12);"50 mV";Cursor(8,41);"24";
         Cursor(8,47);"4-20mA (0-100%)";Cursor(8,65);"Resistor"
     100 Print Cursor(9,2);"03";Cursor(9,12);"100 mV";Cursor(9,41);"25";
         Cursor(9,47);"4-20mA (1-5V)";Cursor(9,65);"Resistor"
     110 Print Cursor(10,2);"04";Cursor(10,12);"1V";Cursor(10,22);
        "50:1 Resistor";Cursor(10,41);"26";Cursor(10,47);"+-20mA (+-100%)";
         Cursor(10,65);"Resistor"
     120 Print Cursor(11,2);"05";Cursor(11,12);"5V";Cursor(11,22);
         "100:1 Resistor";Cursor(11,41);"27";Cursor(11,47);"+-1mA (+-100%)";
         Cursor(11,65);"Resistor"
```

```
    130 Print
    140 Print "Typ av mätområde för insignalen, kod:"
    150 Print
    160 Print "Hur skall den uppmätta signalen konverteras"
    170 Print "( 0=linjärt, y=k*x+m  : 1=sqr.,  y=k*sqr(x):"
    180 Print Cursor(17,8);"k-värdet:"
    190 Print Cursor(18,8);"m-värdet:"
    200 Print
    210 Print "Enhet på konverterad insignal:"
    220 Print
    230 Print "Önskas fler signalmenyer (Y/N):"
  EXIT 1
```

Procedure File_meny

```
Procedure File_meny
   EXTERNAL: Clear_screen
   EXTERNAL: Cursor

     10 Rem Procedure Som Skriver Ut Filmenyn
     20 Clear_Screen
     30 Print Cursor(1,1);"Filnamn: "
     40 Print
     50 Print "Konfigurationen med omvärlden"
     60 Print "( 0=Ansluten direkt till disk, 1=Mjölkning ): "
     70 Print
     80 Print "Samplingsintervall i sekunder: "
     90 Print
    100 Print "De deklarerade signalerna"
    110 Print
    120 Print "Signal";Cursor(10,10);"Beskrivning";Cursor(10,28);"Signal";
        Cursor(10,37);"Beskrivning"
    130 Print Cursor(10,56);"Signal";Cursor(10,64);"Beskrivning"
    140 Print Cursor(20,1);"Signaler som skall lagras (signal 1,
        signal 2, ... )"
    150 Print Cursor(21,2);": "
    160 Print Cursor(22,1);"Loggningen kan pågå, vid mjölkning
        ( dygn:tim:min:sek ): "
    170 Print "Alla ändringar klara (Y/N) : "
  EXIT 1
```

Procedure Main_meny

```
Procedure Main_meny
   INTEGER ARG: Showflag
   EXTERNAL: Cursor,Clear_screen

       5 Rem Procedure som skriver ut Mainmeny
      10 Clear_Screen
      20 Print Cursor(2,1);Chr$(42,71)
      30 Print Chr$(42,25);" MÄTINSAMLINGSSYSTEM ";Chr$(42,25)
      40 Print Chr$(42,71)
      50 Print Cursor(7,1);"1.Ange nytt datum och klockslag "
      60 Print
      70 Print "2.Skapa signaler eller ändra skapade signaler "
      80 Print
      90 Print "3.Skapa eller ändra : Filnamn, Konfiguration med omvärlden,
         Samplingsintervall"
     100 Print Cursor(12,23);"Loggbara signaler"
     110 Print Cursor(13,4);"Då konfigurationen=mjölkning anges
         dumpningstidpunkt "
     120 Print
     130 If Showflag=1 Then Print Cursor(15,1);"4.Uthopp från menyn"
     140 Do If Showflag=0
     150   Print Cursor(15,1);"4.Startar loggningen"
     160   Print Cursor(18,1);"Vid uppstart skall valda siffror anges i
           ordningen (1-4) "
     170   Print "Vid ny loggning kan start ske från punkt 3"
     180 End Do
     190 Print Cursor(22,1);"Välj en siffra: "
  EXIT 1
```

Procedure Logg_time

```
Procedure Logg_time
   STRING ARG: RAM_loggtime/VAR
   REAL ARG: Samplingsintervall
   INTEGER ARG: Nr_RAM_places
   INTEGER ARG: Tot_Antal_RAM_P/VAR
   INTEGER ARG: Loggbara_signaler
   INTEGER: Antal%
   REAL: Full_in_RAM
   EXTERNAL: Time_converter

      10 Rem Procedure Som RäKnar Ut NäR RAMarean Blir Full
      20 Antal%=Int(Nr_RAM_Places/Loggbara_Signaler)     :
         Rem (Antal RAM Platser/Antal Loggbara Signaler )
      30 Tot_Antal_RAM_P=Antal%*Loggbara_Signaler        :
         Rem Totala Antalet MöJliga Platser I RAMMet ,Som Utnyttjas
      40 Full_In_RAM=Antal%*Samplingsintervall           :
         Rem Antalet Sekunder Som Har Gått Då RAMarean Blir Full
      50 Time_Converter(RAM_Loggtime,Full_In_RAM,1)      :
         Rem Lägger Tidpunkten Då RAMarean Blir Full I RAM_Loggtime
  EXIT 1
```

Procedure Date_declare

```
Procedure Date_declare
   INTEGER ARG: Showflag
   INTEGER ARG: Date_Mode/VAR
   STRING ARG: Start_date_logg,Start_time_logg
   STRING: TextÄ8Å
   EXTERNAL: Cursor,Clear_rad
   INTEGER: I%
   EXTERNAL: Up_down_cursor,Move_Up_Down,Date_meny
   INTEGER: Lage%,Y%,X%
   EXTERNAL: Read_sentence

     5 Rem Procedure Som Visar Aktuellt Datum och Tidpunkt,Kan även
           Visa Starttidpunkten FöR Loggningen. Aktuellt datum och tid-
           punkt måste anges vid uppstart.
    10 Date_Meny(Showflag)              :Rem Skriver ut Datemenyn
    20 Do If Showflag=0                 :Rem Loggning pågår ej
    30   Do If Date_Mode=1              :Rem Tidigare Inmatad , Changemode
    32     Print Cursor(1,40);"F1- MöJLIGGöR INMATNING Pä AKTUELL RAD"
    34     Print Cursor(2,40);Chr$(45,38)
    40     I%=3
    50     Gosub 470                    :Rem Utskrift Av Aktuellt Datum,Tid
    60     Print Cursor(20,30);
    70     Do          :Rem Flyttar cursorn till rätt rad
                        och väntar på inmatning då "F1" tryckts ner.
    80       Do
    90         Lage%=Up_Down_Cursor
   100         If Lage%=1 Then Move_Up_Down(1,I%,1,Y%,X%)
   110         If Lage%=2 Then Move_Up_Down(1,I%,0,Y%,X%)
   120         Print Cursor(Y%,X%);
   130       Repeat If Lage%=2 Or Lage%=1 Or Lage%=0    :Rem Leta Efter "F1"
   150       Gosub 410                   :Rem LäSer In En Tecken-SträNg Och
                                              UndersöKer Om Den äR Ok
   160       Move_Up_Down(1,I%,0,Y%,X%) :Print Cursor(Y%,X%);
   170     Repeat If Text<>"Y" Or I%<>3
   180   End Do
   190   Do If Date_Mode=0     :Rem Ej Tidigare Inmatad, Inputmode
   200     I%=1
   210     Y%=5 : X%=32
   220     While I%<>3 Do
   230       Gosub 410         :Rem LäSer In En Tecken-SträNg Och UndersöKer
                                    Om Den äR Ok
   240       Move_Up_Down(1,I%,0,Y%,X%)  :Rem Flyttar Cursorn Ett Steg Ner
   250       Print Cursor(Y%,X%);
   260     Repeat
   270     Do
   280       Gosub 410               :Rem VäNtar På "Y"
   290     Repeat If Text<>"Y"
   300     Date_Mode=1
   310   End Do
   320 End Do
   330 Do If Showflag=1              :Rem  Loggning PåGåR, Showmode
   340   Gosub 470                   :Rem Skriver Ut Aktuellt Datum,Tidpunkt
   342   Print Cursor(15,31);Start_date_logg
   344   Print Cursor(17,34);Start_time_logg
   350   Print Cursor(20,30);
   355   Y%=20 : X%=30  :I%=3
   360   Do
   370     Gosub 410:Rem VäNtar På "Y"
   380   Repeat If Text<>"Y"
   390 End Do
   400 Exit
```

```
  410 Clear_rad(Y%,X%,12)
  420 Read_Sentence(Y%,X%,I%,8,1,Text)
  460 Return
  470 Print Cursor(5,31);Date$
  480 Print Cursor(10,33);Time$
  490 Return
EXIT 1
```

Procedure File_declare

```
Procedure File_declare
   INTEGER ARG: Showflag/VAR
   INTEGER ARG: File_mode/VAR
   INTEGER ARRAY ARG(7): Gen_Integ_Par/VAR
   STRING ARG: Logg_Signals/VAR
   INTEGER ARRAY ARG(4,24): Par_Integ/VAR
   STRING ARG: Filename_Dat/VAR
   STRING ARG: Filename_Dok/VAR
   STRING ARG: RAM_Loggtime/VAR
   REAL ARG: Samplingsintervall/VAR
   STRING ARRAY ARG(2,24)Ä8Å: Par_Str/VAR
   INTEGER ARG: Nr_RAM_places
   STRING: Text1Ä61Å,TextÄ61Å
   EXTERNAL: Clear_screen,Cursor,Up_down_cursor,Clear_rad,Screen_display,
             Move_up_down,Correct_Input_f
   INTEGER: I%,Y%,X%
   EXTERNAL: Logg_time,File_meny
   INTEGER: Ok%,Lage%,Rad,Signal,Antal,Kol,First
   STRING: TalÄ2Å,TeckenÄ1Å

  10 Rem Procedure Som Läser In Eller Visar Filnamn,Konfiguration,
         Samplingsintervall,Loggbara Signaler.Visar tidpunkten Då
         RAMarean blir full ,dvs mjölkningen skall vara avklarad
         innan denna tidpunkt nås.
  20 File_Meny                  :Rem Skriver ut Filmenyn
  30 Do If Showflag=1           :Rem Loggning PåGåR  ,Showmode
  40   Signal=0 : Gosub 750                    :Rem Visar VäRdena
  50   Print Cursor(23,30);
  60   Do
  70     I%=5 : Y%=23 :X%=30 : Gosub 630     :Rem VäNtar På "Y"
  80   Repeat If Text<>"Y"
  90 End Do
 100 Do If Showflag=0           :Rem Inputmode Eller Changemode
 110   Do If File_Mode=1        :Rem Changemode
 115     Print Cursor(1,40);"F1- MöJLIGGöR INMATNING Pä AKTUELL RAD"
 116     Print Cursor(2,40);Chr$(45,38)
 120     Gosub 900              :Rem Laser Antalet Loggbara Signaler
 130     Rem IngåNgsrutin Som RäKnar Ut Maximal Loggtid
 140     Do                     :Rem MjöLkning
 150       If Gen_Integ_Par(5)>0 Then Logg_Time(RAM_Loggtime,
           Samplingsintervall,Nr_RAM_Places,Gen_Integ_Par(4),
           Gen_Integ_Par(5))  :Rem Om Antalet Loggbara Signaler är
           Större än 0 Räknas tiden Ut Da RAMMET Blir Fullt
 160     End Do
```

```
170     Y%=23 : X%=30
180     Signal=0 : Gosub 750 :Rem Visar VäRdena
185     I%=5
190     Print Cursor(23,30);
200     Do
210       Do                    :Rem Flyttar cursorn till rätt rad och
                                  väntar på inmatning då "F1" trycks ner.
220         Lage%=Up_Down_Cursor
230         If Lage%=1 Then Move_Up_Down(3,I%,1,Y%,X%)
240         If Lage%=2 Then Move_Up_Down(3,I%,0,Y%,X%)
250         Print Cursor(Y%,X%);
260       Repeat If Lage%=2 Or Lage%=1 Or Lage%=0  :Rem VäNtar På "F1"
270       Gosub 630:Rem LäSer In VäRden Då I%<>4
280       Gosub 690:Rem LäSer In VäRden Då I%=4  ,De Loggbara Signalerna
290       Do If I%=4    :Rem Skriver Ut Ny Tidpunkt Då RAMarean Blir Full
300         Rad=I% : Gosub 1000: I%=Rad          :Rem Lagrar Uppgifterna
310         Logg_Time(RAM_Loggtime,Samplingsintervall,Nr_RAM_Places,
            Gen_Integ_Par(4),Gen_Integ_Par(5)) : Clear_Rad(22,59,12) :
            Print RAM_Loggtime
320       End Do
330       Rem Innan Man Går Ut Måste Antalet Loggbara Signaler Deklareras,
              De Loggbara Signalerna Skall Läggas I Rätt Ordning I
              Strängen Logg_Signals. Detta görs Med Proceduren Logg_Values
340       If I%=5 And Text="Y" Then  Gosub 900:Exit  :Rem (Logg_Values)
345       If I%<>5 And I%<>4 Then Rad=I% : Gosub 1000 :I%=Rad
350       Logg_Time(RAM_Loggtime,Samplingsintervall,Nr_RAM_Places,
          Gen_Integ_Par(4),Gen_Integ_Par(5)): Clear_Rad(22,59,12):
          Print RAM_Loggtime          :Rem Skriver Ut Hur Länge Loggningen
                                       Kan Pågå
360       Do If I%=2
380         Logg_Time(RAM_Loggtime,Samplingsintervall,Nr_RAM_Places,
            Gen_Integ_Par(4),Gen_Integ_Par(5)) :Print Cursor(22,59);
            RAM_Loggtime
390       End Do
400       Move_Up_Down(3,I%,0,Y%,X%)
410       Print Cursor(Y%,X%);
420     Repeat
430   End Do
440   Do If File_Mode=0         :Rem Input Mode
450     Signal=1 : Gosub 750:Rem Skriver Ut De Signaler Som Ar Deklarerade
460     I%=1
470     Y%=1 : X%=10
480     Do
490       Gosub 630
500       Gosub 690
510       Rad=I% :  Gosub 1000 :I%=Rad       :Rem lagrar data
520       If I%=4 Then Logg_Time(RAM_Loggtime,Samplingsintervall,
          Nr_RAM_Places,Gen_Integ_Par(4),Gen_Integ_Par(5)) :
          Print Cursor(22,59);RAM_Loggtime  :Rem Skriver Ut Hur Länge
                                                 Loggningen Kan Pågå
530       Move_Up_Down(3,I%,0,Y%,X%)        :Rem Flyttar Cursorn Neråt
540     Repeat If I%<>5
545     Print Cursor(23,30);
550     Do
560       Gosub 630:Rem VäNtar På "Y"
570     Repeat If Text<>"Y"
580     File_Mode=1       :Rem Hädanefter Kommer Man Att Gå
                               In I Changemode
590     Gosub 900         :Rem (Loggvalues)
600   End Do
610 End Do
620 Exit
```

```
630 Do If I%<>4           :Rem LäSer In En TeckensträNg
640   Clear_Rad(Y%,X%,12)
650   Text=""
660   Screen_Display(Y%,X%,Text,12)
670 Repeat If Correct_Input_F(3,I%,Text,Par_Integ)=0
680 Return
690 Do If I%=4            :Rem LäSer In De Signaler Som Skall Loggas
700   Clear_Rad(Y%,X%,61)
710   If File_Mode=1 Then Text=Logg_Signals :Print Cursor(21,4);Text :
      Screen_Display(Y%,X%,Text,61) Else Text="" :
      Screen_Display(Y%,X%,Text,61)
720   Ok%=Correct_Input_F(3,I%,Text,Par_Integ)
730 Repeat If Ok%=0
740 Return
750 Do If Signal=0
760   Print Cursor(1,10);Filename_Dat         :Rem "Filnamn"
770   Print Cursor(4,46);Gen_Integ_Par(2)     :Rem "Konfigurationen"
780   Print Cursor(6,31);Samplingsintervall   :Rem "Samplingsintervall"
790   Print Cursor(21,4);Logg_Signals         :Rem "Loggade signaler"
800   Print Cursor(22,59);RAM_Loggtime        :Rem "Tidpunkten då RAM-
                                                    blir full"
810 End Do
820 Rad=10
830 Kol=0
840 For I%=0 To 23         :Rem Skriver Ut De Deklarerade Signalerna
850   Do If Par_Integ(2,I%)=1
860     If Rad=18 Then Kol=Kol+1 : Rad=11 Else Rad=Rad+1
870     Print Cursor(Rad,2+Kol*28);I%;Cursor(Rad,10+Kol*27);Par_Str(1,I%)
880   End Do
890 Next
895 Return
900 Tecken=","       :Rem Lägger De Loggbara Signalerna I Rätt
                          Ordning I En Tecken Sträng
910 Text1=""
920 First=1
930 Antal=0
940 For I%=0 To 23
950   If Par_Integ(3,I%)=1 Then If First=1 Then Text1=Text1+
      Right$(Str$(I%),Len(Str$(I%))-1) : First=0 : Antal=Antal+1 Else
      Text1=Text1+Tecken+Right$(Str$(I%),Len(Str$(I%))-1) :Antal=Antal+1
960 Next
970 Logg_Signals=Text1          :Rem De Loggbara Signalerna
980 Gen_Integ_Par(5)=Antal      :Rem Antal Loggbara Signaler
990 Return
1000 Do If Rad=1                :Rem Filnamn
1010   Filename_Dat=Upper$(Text)+Right$(Str$(Gen_Integ_Par(1)),
       Len(Gen_Integ_Par(1))-1)+".DAT"
1020   Filename_Dok=Upper$(Text)+Right$(Str$(Gen_Integ_Par(1)),
       Len(Gen_Integ_Par(1))-1)+".DOK"
1030 End Do
1040 Do If Rad=2
1050   Gen_Integ_Par(2)=Val(Text)     :Rem Mode
1060 End Do
1070 Do If Rad=3
1080   Samplingsintervall=Val(Text)  :Rem Sampelintervallets Längd
1090 End Do
```

```
 1100 Do If Rad=4
 1110   Logg_Signals=Text              :Rem De Loggbara Signalerna
 1120   Tal=""
 1130   Antal=0         :Rem Anger Vilka Signaler Som är Loggbara
 1140   For I%=0 To 23
 1150     Par_Integ(3,I%)=0
 1160   Next
 1170   For I%=1 To Len(Text)
 1180     If Left$(Text(I%),1)="," Then If Par_Integ(3,Val(Tal))=0 Then
          Par_Integ(3,Val(Tal))=1 :Tal="" :Antal=Antal+1 Else Tal=""
 1190     If Left$(Text(I%),1)<>"," Then Tal=Tal+Left$(Text(I%),1)
 1200   Next
 1210   If Par_Integ(3,Val(Tal))=0 Then Par_Integ(3,Val(Tal))=1 :
        Antal=Antal+1
 1230   Gen_Integ_Par(5)=Antal     :Rem Antal Loggbara Signaler
 1240 End Do
 1250 Return
EXIT 1
```

Huvudprogrammets deklarationsdel

```
REAL ARRAY(2,24): Par_real
STRING ARRAY(2,24)Ä8Å: Par_str
INTEGER ARRAY(4,24): Par_integ
INTEGER: Showflag
REAL ARRAY(6): Gen_real_par
INTEGER ARRAY(7): Gen_integ_par
STRING: Filename_datÄ12Å,Filename_dokÄ12Å,RAM_loggtimeÄ12Å
INTEGER: Date_mode,File_mode,First_start
STRING: Logg_signalsÄ61Å
INTEGER: Nr_RAM_places
INTEGER ARRAY(5): Sampel_show
INTEGER: Next_input,Start_plottning,Framforhallning
REAL ARRAY(24): Insignal
REAL ARRAY(24): Converted_signal
STRING: Logg_sampelÄ14Å
INTEGER: Logg_disp_flag,New_sampel_flag
REAL ARRAY(1800): Omformad_insignal
INTEGER: First,Last,Start_disk_uthamt,First_in_RAM,I%,Logg_flag,Init_Flag
STRING: Start_date_loggÄ12Å,Start_time_loggÄ12Å
INTEGER: Autoplot71,New_plot_flag,First_time_open,Start_tomning,Ny_start,
         Plot_flag,Mjolk_flag,Stang_av_flag
INTEGER: Just_start_up,Help_flag,Nr%
STRING: Text$Ä16Å
INTEGER: Mjolk_direkt_flag,Direkt_mjolk_flag,Direkt_flag,Start_disk,Num,Lage
STRING: SiffraÄ2Å
INTEGER: Antal_CTRLB
```

Huvudprogrammet

```
 10 Clear                          :Rem Nollställer RAMet
 20 File_Mode=0                    :Rem Inget har matats in i Filmenyn
 30 Showflag=0                     :Rem Loggning PåGåR Ej
 40 First_start=1  :Rem forsta gången som startas upp,dvs. ny loggning
 60 Nr_RAM_Places=1800
 70 Main_Declare                   :Rem Initieringsdelen
 80 First_In_RAM=1                 :Rem Ny loggning
 90 Just_Start_Up=1                :Rem Ny loggning
100 Gen_Integ_Par(6)=Gen_Integ_Par(4)/Gen_Integ_Par(5)  :Rem Antal Block
110 Gen_Real_Par(1)=0.02           :Rem integrationstiden
120 Last=Gen_Integ_Par(0)-1        :Rem Pekar På Det Sista Blocket I RAMet
122 Framforhallning=0              :Rem används vid plottning,anger hur många
    sampelvarden som man måste ha samlat in innan plottning kan ske
124 If Gen_real_par(0)>=1 And Gen_real_par(0)<5 Then Framforhallning=2
126 If Gen_real_par(0)<1 Then Framforhallning=5
130 Showflag=1
135 If Gen_integ_par(2)=1 Then Antal_CTRLB=1 Else Antal_CTRLB=0  :Rem Då
    konfigurationen=Mjolkning sätts Antal_CTRLB=1 så att loggningen kan
    upphöra när man begär det
140 Do If Gen_Integ_Par(2)=0  :Rem Direkt ansluten till disk
150   Input "Om uppkopplingen med WOS-programmet klart: skriv KLAR: "Text$
160 Repeat If Text$<>"KLAR"
170 For I%=0 To 23                :Rem Konfigurerar De Analoga Insignalerna
180   Do If Par_Integ(3,I%)=1          :Rem Signalen Loggbar
190     Intype(I%,Par_Integ(0,I%),Gen_Real_Par(1))
200   End Do
210 Next
220 Num=2
230 Do If Gen_Integ_Par(2)=0  :Rem Öppnar Filen Om Direkt Ansluten Till Disk
240   Open("o",Filename_Dat,Num)
250   Close(Num)
260   Open("a",Filename_Dat,Num)
270 End Do
280 First_Time_Open=1
290 Setint(3,Gen_Real_Par(0))     :Rem Tidsavbrott
300 Setint(7,2)                   :Rem Komunikationsavbrott
310 On Interrupt Inlasning
320 Do
330   If Stang_Av_Flag=1 Then Stang_Av_Flag=0 : Stang_Av_Loggningen :Exit
340   If Mjolk_Flag=1 Then Mjolk_Flag=0 : Mjolka
350   If Direkt_Mjolk_Flag=1 Then Direkt_Mjolk_Flag=0 :Direkt_Till_Mjolka
360   If Mjolk_Direkt_Flag=1 Then Mjolk_Direkt_Flag=0 :Mjolka_Till_Direkt
370   If Help_Flag=1 Then Help_Flag=0 : Help_Meny_v
380   If Logg_Flag=1 Then Logg_Flag=0 : Logg_Display
390   If Init_Flag=1 Then Init_Flag=0 : Main_Declare
400   If Plot_Flag=1 Then Plot_Flag=0 : Plottning
410 Repeat
420 Do If Ny_Start=1
422   Siffra=Right$(Str$(Gen_Integ_Par(1)),Len(Str$(Gen_Integ_Par(1)))-1)
424   Lage=Instr(1,Filename_Dat,Siffra)
426   Filename_Dat=Left$(Filename_Dat,Lage-1)+"0"+".DAT"
428   Filename_Dok=Left$(Filename_Dok,Lage-1)+"0"+".DOK"
430   Clear(Omformad_Insignal,Converted_Signal,Insignal,First,Last,
      Start_Disk,Next_Input,Start_Plottning,New_Sampel_Flag,Gen_Integ_Par(0),
      Gen_Integ_Par(1),Gen_Real_Par(2),Gen_Real_Par(3),Gen_Real_Par(4),
      Gen_Real_Par(5),Start_Date_Logg)
440   Clear(Start_Time_Logg,Showflag)
450 End Do
460 If Ny_Start=1 Then Ny_start=0 : Goto 60
```

Procedure Help_meny_v

```
Procedure Help_meny_v

     5 Rem Procedure som skriver ut Hjälp-menyn
    10 Clear_Screen
    20 Print Cursor(1,30);"HJÄLP_MENY"
    30 Print Cursor(2,1);"CTRLB          :Vid Mjölkning avslutas loggningen.
       Vid Direkt överföring"
    40 Print "                avslutas loggningen om samplings-intervallet
       är kort annars"
    50 Print "                övergång till Mjölkning. Använd WOS-programmet."
    60 Print
    70 Print "M-CTRLB         :Överför insamlad data i RAMarean till en fil på
       disk."
    80 Print Cursor(7,16);"Används vid Mjölkning. Använd WOS prog."
    90 Print
   100 Print "S-CTRLB         :Visar de senast samplade värdena. Visar verklig
       signal"
   110 Print Cursor(10,16);"och konverterad signal."
   120 Print
   130 Print "I-CTRLB         :Visar Initieringsdelen."
   140 Print
   150 Print "P-CTRLB         :Möjliggör Plottning av max 2 signaler."
   160 Print
   170 Print "H-CTRLB         :Hjälp-menyn"
   180 Print
   190 Print "*md-CTRLB       :Byte från Mjölkning till Direkt överföring ."
   200 Print "                WOS-programmet måste användas."
   210 Print
   220 Print "CTRLB--CTRLB    :Övergång från Direkt överföring till Mjölkning,
       därefter avbryts"
   230 Print "                loggningen. Det andra CTRLB får först komma då
       den första"
   240 Print "                övergången är klar. Andvänd WOS-programmet."
EXIT 1
```

Procedure Store_signal_descr

```
Procedure Store_signal_descr
   INTEGER ARG: Meny
   INTEGER ARG: Rad
   STRING ARG: Text
   INTEGER ARG: Signal
   EXTERNAL: Par_Real,Par_Str,Par_Integ
   STRING: TalÄ2Å
   INTEGER: I%
   EXTERNAL: Sampel_Show
   INTEGER: Nr%

      10 Rem Procedure Som överför vissa av de inknappade värdena
             till de Globala Värdena.
      20 Do If Meny=2                        :Rem Signalmenyn
      30   Do If Rad=2                       :Rem "signalbeskrivning"
      40     Par_Str(1,Signal)=Text
      50   End Do
      60   Do If Rad=3                       :Rem "kode"
      70     Par_Integ(0,Signal)=Val(Text)
      80   End Do
      90   Do If Rad=4                       :Rem "linjär,sqr"
     100     Par_Integ(1,Signal)=Val(Text)
     110   End Do
     120   Do If Rad =5                      :Rem "k-värdet"
     130     Par_Real(0,Signal)=Val(Text)
     140   End Do
     150   Do If Rad=6                       :Rem "m-värdet"
     160     Par_Real(1,Signal)=Val(Text)
     170   End Do
     180   Do If Rad=7                       :Rem "enhet"
     190     Par_Str(0,Signal)=Text
     200   End Do
     210 End Do
     480 Do If Meny=4                 :Rem Visning Av De Loggbara Signalerna
     490   For I%=0 To 4
     500     Sampel_Show(I%)=-1       :Rem Nollställer variablen,variablen
                                           anger nu att ej någon signal skall
                                           visas  Kontinuerligt under Loggningen
     510   Next
     520   Nr%=0  :Tal=""
     530   For I%=1 To Len(Text)
     540     If Left$(Text(I%),1)="," Then Sampel_Show(Nr%)=Val(Tal) : Tal="" :
             Nr%=Nr%+1
     550     If Left$(Text(I%),1)<>"," Then Tal=Tal+Left$(Text(I%),1)
     560   Next
     570   Sampel_Show(Nr%)=Val(Tal)
     580 End Do
   EXIT 1
```

Procedure Show_values

```
Procedure Show_values
   INTEGER ARG: Meny
   INTEGER ARG: Signal
   EXTERNAL: Par_str,Par_real,Par_integ

      10 Rem Procedure Som för Signalmenyn skriver ut tidigare inmatade värden.
      60 Do If Meny=2                                    :Rem Signalmenyn
      70   Print Cursor(2,20);Par_Str(1,Signal)          :Rem Signalbeskrivning
      80   Print Cursor(13,38);Par_Integ(0,Signal)       :Rem Kode
      90   Print Cursor(16,45);Par_Integ(1,Signal)       :Rem LinjäR Eller Sqr.
     100   If Par_Real(0,Signal)<0 Then Print Cursor(17,18);
           Par_Real(0,Signal)  Else Print Cursor(17,17);Par_Real(0,Signal):
           Rem K-VäRdet
     110   If Par_Integ(1,Signal)=0 Then If Par_Real(1,Signal)<0 Then
           Print Cursor(18,18);Par_Real(1,Signal) Else Print Cursor(18,17);
           Par_Real(1,Signal)                            :Rem M-VäRdet
     120   Print Cursor(20,32);Par_Str(0,Signal)         :Rem Enhet
     130 End Do
   EXIT 1
```

Procedure Signal_declare

```
Procedure Signal_declare
   INTEGER: Mode%
   STRING: TextÄG1Å,TeckenÄ1Å
   INTEGER: Signal%
   EXTERNAL: Showflag,Par_integ
   INTEGER: I%,Y%,X%
   EXTERNAL: Store_signal_descr,Show_values
   INTEGER: K%,Lage%,Signal2%
   EXTERNAL: Par_str,Par_real
   INTEGER: Ok%,Ut%,Antal,First
   EXTERNAL: RAM_Loggtime,Gen_Real_par,Gen_Integ_Par,Nr_RAM_places,Logg_Signals
   EXTERNAL: File_mode

       5 Rem Procedure som läser in eller visar signalnr.,signalbeskrivning,
         kod,konverterings-sätt,k-och m-värdet,Mätenhet.
         inmatningsmode=inget har varit inknappat tidigare
         changemode=man vill ändra något som tidigare har knappats in.
         showmode=visar de inknappade värdena, det går ej att ändra dem.
```

```
 10 Do
 20   Mode%=2                                        :Rem nollställer moden
 30   Clear_Screen
 40   Signal_Meny
 50   Read_Sentence(1,12,1,2,2,Text)                 :Rem LäSer In Ett Signalnr
100   Signal%=Val(Text)
110   Do If Showflag=0
120     Mode%=0                                      :Rem "inmatningsmode"
130     If Par_Integ(2,Signal%)=1 Then Mode%=1       :Rem "om signalen inmatad
                                                          -changemode"
140   End Do
150   Do If Mode%=0                                  :Rem "inmatningsmode"
160     Print Cursor(1,47);"Inmatningsmode"
170     For K%=1 To 7
180       I%=K%
190       Move_Up_Down(2,I%,0,Y%,X%)          :Rem "flyttar ett steg neråt"
200       If I%=6 And Par_Integ(1,Signal%)=1 Then Goto 280
210       Rem Då I=5 Måste man testa om Signalen är Linjär Eller Sqr.,
          för Att Se Om  M-Värdet också skall knappas in.
220       Read_Sentence(Y%,X%,I%,8,2,Text)     :Rem LäSer In Text
270       If I%<>8 Then Store_Signal_Descr(2,I%,Text,Signal%)
280       Rem
290     Next
300     Par_Integ(2,Signal%)=1
310   End Do
320   Do If Showflag=1                               :Rem showmode
330     Print Cursor(1,47);"showmode"
340     If Par_Integ(3,Signal%)=1 Then Show_Values(2,Signal%) Else
        Print Cursor(2,42);"signalen är ej loggbar"
350     Read_Sentence(22,33,0,8,2,Text) :Rem VäNtar På "Y"
400   End Do
410   Do If Mode%=1                                  :Rem changemode
420     Print Cursor(1,30);"Ändring av Översta raden:"
421     Print Cursor(2,35);"F1-CR              - Tar bort signalen"
422     Print Cursor(3,35);"F1-Nytt Signalnr.- Den nya signalen blir en "
423     Print Cursor(4,35);"                    kopia av föregående signal"
425     Print Cursor(23,42);"F1- Möjliggör inmatning på aktuell rad"
426     Print Cursor(24,42);Chr$(45,38);
430     Show_Values(2,Signal%)                       :Rem Skriver Ut VäRdena
440     Print Cursor(22,33);
450     Y%=22 : X%=33
460     I%=8
470     Do
480       Do              :Rem flyttar cursorn till rätt rad och väntar
                             på inmatning då "F1" trycks ner.
490         Lage%=Up_Down_Cursor
500         Do If Lage%=1
510           Move_Up_Down(2,I%,1,Y%,X%)
520           If I%=6 And Par_Integ(1,Signal%)=1 Then
              Move_Up_Down(2,I%,1,Y%,X%)             :Rem flyttar ett steg upp
530           Print Cursor(Y%,X%);
540         End Do
550         Do If Lage%=2
560           Move_Up_Down(2,I%,0,Y%,X%)
570           If I%=6 And Par_Integ(1,Signal%)=1 Then
              Move_Up_Down(2,I%,0,Y%,X%)             :Rem flyttar ett steg ner
580           Print Cursor(Y%,X%);
590         End Do
600       Repeat If Lage%=2 Or Lage%=1 Or Lage%=0  :Rem VäNtar På "F1"
610       Do                                  :Rem Inmatning Av Aktuell Rad
620         Text=""
```

```
630         Clear_Rad(Y%,X%,8)
640         Screen_Display(Y%,X%,Text,8)
650         If I%=1 And Text="" Then Ut%=1 :Exit Else Ut%=0  :Rem Signalen
                                                              Tas Bort
660       Repeat If Correct_Input_D(2,I%,Text)=0
670       If Ut%=1 Then Exit
680       Do If I%=4          :Rem UndesöK Om LinjäR Eller Sqr Om Sqr.
                                 Skall M-Värdet raderas På Skärmen
690         If Val(Text)=1 Then Clear_Rad(18,18,8) Else
            Par_Real(1,Signal%)=0 : Print Cursor(18,17);
            Par_Real(1,Signal%)    :Rem Om Sqr. ändras Till
                                        Linjärt så Blir M-VäRdet=0
700       End Do
710       Rem HäR ändras Aktuell Rad
720       Do If I%=1                :Rem kopiering av en signals parametrar
                                        till en annan signal
730         Signal2%=Signal%
740         Signal%=Val(Text)
750         Par_Str(1,Signal%)=Par_Str(1,Signal2%)
760         Par_Integ(0,Signal%)=Par_Integ(0,Signal2%)
770         Par_Integ(1,Signal%)=Par_Integ(1,Signal2%)
780         Par_Real(0,Signal%)=Par_Real(0,Signal2%)
790         Par_Real(1,Signal%)=Par_Real(1,Signal2%)
800         Par_Str(0,Signal%)=Par_Str(0,Signal2%)
810         Par_Integ(2,Signal%)=1   :Rem anger att Signalen är deklarerad
820       End Do
830       If I%<>1 And I%<>8 Then Store_Signal_Descr(2,I%,Text,Signal%)
840       If I%=8 Then I%=9                       :Rem sidan är klar
850       Do If I%<>9
860         If I%=5 And Par_Integ(1,Signal%)=1 Then
            Move_Up_Down(2,I%,0,Y%,X%)            :Rem linjär eller sqr.
870         Move_Up_Down(2,I%,0,Y%,X%) : Print Cursor(Y%,X%);
880       End Do
890     Repeat If I%<>9
900     Do If Ut%=1
910       Par_Integ(2,Signal%)=0    :Rem Signalen Anvands Ej
920       Par_Integ(3,Signal%)=0    :Rem Signalen Ar Ej Loggbar
922       Tecken=","
923       Text=""
924       Antal=0 : First=1
925       For I%=0 To 23            :Rem lägger de loggbara signalerna i
                                        rätt ordning.
927         If Par_Integ(3,I%)=1 Then If First=1 Then
            Text=Text+Right$(Str$(I%),Len(Str$(I%))-1) :First=0 :
            Antal=Antal+1 Else Text=Text+Tecken+Right$(Str$(I%),
            Len(Str$(I%))-1) : Antal=Antal+1
928       Next
929       Logg_Signals=Text         :Rem De Loggbara Signalerna
930       Gen_Integ_Par(5)=Antal    :Rem Antal Loggbara Signaler
935       If Antal=0 Then File_mode=0 : Goto 945
940       Logg_Time(RAM_Loggtime,Gen_Real_Par(0),Nr_RAM_Places,
          Gen_Integ_Par(4),Gen_Integ_Par(5)) :Rem räknar ut när RAMarean
                                                blir full.
945       Clear_Screen :Signal_Meny
950       Read_Sentence(2,33,8,8,2,Text)    :Rem Väntar På "Y"
960     End Do
970     I%=8
980   End Do
990 Repeat If Text="Y"
EXIT 1
```

Procedure Main_declare

```
Procedure Main_declare
   STRING: TextÄ2Å
   EXTERNAL: Showflag,First_start,Date_mode
   INTEGER: Ok,Nr
   EXTERNAL: Signal_declare,Gen_integ_par
   INTEGER: Anvands%,I%
   EXTERNAL: Par_integ,File_mode,Logg_Signals,Filename_dat,Filename_dok,
             RAM_Loggtime,Gen_real_par
   EXTERNAL: Par_Str
   EXTERNAL: Nr_RAM_places
   EXTERNAL: Start_date_logg
   EXTERNAL: Start_time_logg
   EXTERNAL: Help_meny_v

     5 Rem Procedure som läser in en siffra mellan 1 och 4, hoppar sedan
           till någon av menyerna : Datemeny,Signalmeny,Filemeny.
    10 Do If Showflag=1                  :Rem showmode
    20   Main_Meny(Showflag)             :Rem skriver ut Mainmenyn
    30   Gosub 330                       :Rem Läser In Vald Siffra
    40   Gosub 390                       :Rem Hoppar Till Någon Meny
    50 Repeat If Val(Text)<>4
    60 Ok=0  :Nr=1
    70 Do If Showflag=0                  :Rem Inputmode Eller Changemode
    80   Main_Meny(Showflag)
    90   Gosub 330                       :Rem Läser In Vald Siffra
   100   If Val(Text)=4 And Ok=1 Then Exit
   110   Do If First_Start=0
   120     While Val(Text)=4 Do
   130       Gosub 330                   :Rem Läser In Vald Siffra
   140     Repeat
   150     Rem Här Skall Test Ske Om Det Finns Några Signaler Deklarerade
   155     Anvands%=1
   157     Do If Val(Text)=3
   160       Anvands%=0
   170       For I%=0 To 23
   180         If Par_Integ(2,I%)=1 Then Anvands%=1 : Exit
   190       Next
   195     End Do
   200     If Anvands%=0 Then Text="4" :Goto 120
   210     Gosub 390                     :Rem Hoppar Till Någon Meny
   220     If Val(Text$)=3 Then Ok=1 Else Ok=0
   230   End Do
   240   Do If First_Start=1
   250     While Nr<>Val(Text) Do
   260       Gosub 330                   :Rem Läser In Vald Siffra
   270     Repeat
   280     Gosub 390                     :Rem Hoppar Till Någon Meny
   290     If Nr=2 Then Ok=1 :First_Start=0 Else Ok=0 :Nr=Nr+1
   300   End Do
   310 Repeat
   315 If Gen_Integ_Par(2)=1 Then Help_Meny_V Else Clear_Screen
   320 Exit
```

```
   330 Read_Sentence(22,17,1,2,0,Text) :Rem Läser In En Siffra
   380 Return
   390 Rem Hoppar Till Någon Meny
   400 If Val(Text)=1 Then Date_Declare(Showflag,Date_Mode,Start_Date_Logg,
       Start_Time_Logg)
   410 If Val(Text)=2 Then Signal_Declare
   420 If Val(Text)=3 Then File_Declare(Showflag,File_Mode,Gen_Integ_Par,
       Logg_Signals,Par_Integ,Filename_Dat,Filename_Dok,RAM_Loggtime,
       Gen_Real_Par(0),Par_Str,Nr_RAM_Places)
   430 Return
EXIT 1
```

Procedure Show_logg_meny

```
Procedure Show_logg_meny

     5 Rem Procedure som visar vilka signaler som loggas. Menyn utnyttjas
       då man vill kontinuerligt visa vissa signaler på skärmen.
    10 Clear_Screen
    20 Print Cursor(1,16);"De signaler som utnyttjas är"
    30 Print
    40 Print "Signal  Beskrivning  Mätområde  Enhet  Signal  Beskrivning
       Mätområde  Enhet"
    50 Print Cursor(19,1);"Ange de signaler som skall visas kontinuerligt.
       Ange dem på formen"
    60 Print "( signal 1,signal 2,... ) max 5st. : "
EXIT 1
```

Procedure Sampel_meny

```
Procedure Sampel_meny

     5 Rem Procedure som innehåller menyn där de senast samplade
       värdena visas.
    10 Clear_Screen
    20 Print Cursor(1,27);"De senaste samplade värdena"
    30 Print Cursor(3,1);"Signal nr"
    40 Print "Beskrivning"
    50 Print "Mätområde"
    60 Print "Enhet"
    70 Print
    80 Print "Samplat värde"
    90 Print"  Nr:"
   110 Print "Konv. värde"
   120 Print "Sampel nr"
   130 Print Cursor(24,1);"E-CTRLB -Avbryter visningen och återhopp sker
       till Hjälp-menyn";
EXIT 1
```

Procedure Logg_display

```
Procedure Logg_display
   EXTERNAL: Show_logg_meny
   EXTERNAL: Sampel_meny
   EXTERNAL: Store_signal_descr
   EXTERNAL: Par_integ
   EXTERNAL: Par_str
   EXTERNAL: Sampel_show
   EXTERNAL: Gen_integ_par
   STRING: TextÄ78Å,Text1Ä14Å
   EXTERNAL: Logg_sampel,Logg_disp_flag
   INTEGER: Rad%,Kol%,I%
   EXTERNAL: New_sampel_flag
   INTEGER: Lage%,K%
   EXTERNAL: Insignal,Converted_signal,Help_meny_v

     10 Rem Procedure Som Visar De Senast Samplade Värdena.
     20 Show_Logg_Meny
     30 Rad%=4  : Kol%=0
     40 For I%=0 To 23                :Rem Skriver Ut De Loggbara Signalera
     50   Do If Par_Integ(3,I%)=1     :Rem Signalen Loggas
     60     If Rad%=16 Then Kol%=1 : Rad%=5 Else Rad%=Rad%+1
     70     Print Cursor(Rad%,3+Kol%*40);I%;Cursor(Rad%,11+Kol%*40);
            Par_Str(1,I%);Cursor(Rad%,25+Kol%*40);Par_Integ(0,I%);
            Cursor(Rad%,33+Kol%*40);Par_Str(0,I%)
     80   End Do
     90 Next
    100 Do                                :Rem Inläsning Av De Signaler Som Skall
                                               visas kontinuerligt.
    110   Clear_Rad(20,38,15)
    120   Text1=Logg_Sampel
    130   Print Cursor(20,38);Text1
    140   Screen_Display(20,38,Text1,14)
    150 Repeat If Correct_Input_F(4,1,Text1,Par_Integ)=0
    160 Logg_Sampel=Text1 : Store_Signal_Descr(4,1,Text1,0) :Logg_Disp_Flag=1
    170 Rem Genom Att Logg_Disp_Flag=1 är Det Nu Möjligt Att Visa De Senast
        Samplade Värdena
    180 Sampel_Meny
    190 Kol%=0
    200 For I%=0 To 4
    210   Do If Sampel_Show(I%)<>-1    :Rem Signalen Skall Visas
    220     Print Cursor(3,21+Kol%*12);Sampel_Show(I%)
    230     Print Cursor(4,19+Kol%*12);Par_Str(1,Sampel_Show(I%))
    240     Print Cursor(5,21+Kol%*12);Par_Integ(0,Sampel_Show(I%))
    250     Print Cursor(6,19+Kol%*12);Par_Str(0,Sampel_Show(I%))
    260     Kol%=Kol%+1
    270   End Do
    280 Next
    290 Print Margin(12,23);Cursor(12,1);Save_Cursor
```

```
300 While Logg_Disp_Flag=1          :Rem Blir Noll Via Avbrott och då skall
                                        den kontinuerliga visningen av de
                                        senast samplade värdena upphöra.
310   Do If New_Sampel_Flag=1       :Rem Är Ett Då Ett Nytt Sampelvärde Har
                                        Kommit.
320     New_Sampel_Flag=0
330     Kol%=0
340     Print Cursor(9,6);Gen_Integ_Par(0)    :Rem skriver ut sampelnumret.
350     For I%=0 To 4
360       Do If Sampel_Show(I%)<>-1           :Rem Signalen Skall Visas
370         Print Cursor(8,16+Kol%*12);:Print Using "  #.###^^^^"
            Insignal(Sampel_Show(I%));        :Rem skriver ut senast
                                                   samplat mätvärde.
380         Kol%=Kol%+1
390       End Do
400     Next
410     Print
420     Print Restore_Cursor;
430     Print "  ";Gen_Integ_Par(0);Chr$(32,6-Len(Str$(Gen_Integ_Par(0)))+
        7);
440     For K%=0 To 4
450       Do If Sampel_Show(K%)<>-1           :Rem  Signalen Skall Visas,
                                                  skriver ut senast samplat
                                                  och konverterat mätvärde.
460         Print Using "  #.###^^^^"Converted_Signal(Sampel_Show(K%));
470       End Do
480     Next
490     Print: Print Save_Cursor;
500   End Do
510 Repeat
520 Print Margin(1,24)
530 Help_Meny_V                         :Rem skrier ut hjälpmenyn
EXIT 1
```

Procedure Dokumentfil

```
Procedure Dokumentfil
   STRING ARG: Start_date
   STRING ARG: Start_time
   REAL ARG: Time_overrun
   INTEGER ARG: Antal_tidpunkter
   EXTERNAL: Filename_dat,Filename_dok
   STRING: TextÄ80Å
   INTEGER: Langd
   EXTERNAL: Gen_integ_par,Par_integ,Par_str,Par_real,Gen_real_par
   INTEGER: Nr%,I%,Length%

    5 Rem Procedure Som överför All Dokumentation Till En Fil på disk.
   10 Nr%=3
   20 Open("o",Filename_Dok,Nr%)
   30 Close(Nr%)
   40 Open("a",Filename_Dok,Nr%)
   50 Text=Chr$(32,78)
   60 For I%=15 To 45 Step 5
   70   Mid$(Text,I%,1)="0"
   80 Next
   85 Rem öVersta Raden I Filen
   90 Langd=Len(Str$(Antal_Tidpunkter))-1    :Rem Antal Sampeltidpunkter
  100 Mid$(Text,6-Langd,Langd)=Right$(Str$(Antal_Tidpunkter),Langd)
  110 Langd=Len(Str$(Gen_Integ_Par(5)))-1    :Rem Antal Loggbara Signaler
  120 Mid$(Text,11-Langd,Langd)=Right$(Str$(Gen_Integ_Par(5)),Langd)
  130 Langd=Len(Str$(Gen_Integ_Par(5)+2))-1 :Rem Antal Inledande Textrader
  140 Mid$(Text,51-Langd,Langd)=Right$(Str$(Gen_Integ_Par(5)+2),Langd)
  150 Print #Nr% Text
  160 Print #Nr% "*PEREXJ.*        1"    :Rem Rad 2
  170 For I%=0 To 23                     :Rem Skriver Ut Signalerna till Filen
  180   Do If Par_Integ(3,I%)=1          :Rem Signalen Loggbar
  190     Text=Chr$(32,78)
  200     Left$(Text,8)=Par_Str(1,I%)              :Rem Signalbeskrivning
  210     Mid$(Text,11,8)=Par_Str(0,I%)            :Rem Enhet
  220     Mid$(Text,26,2)=Str$(Par_Integ(1,I%))    :Rem Linj/Sqr
  230     Mid$(Text,33,14)=Str$(Par_Real(0,I%))    :Rem K-Värdet
  240     If Par_Integ(1,I%)=0 Then Mid$(Text,52,14)=Str$(Par_Real(1,I%)) :
                                                     Rem M-Värdet
  250     Mid$(Text,70,Len(Str$(Par_Integ(0,I%))))=Str$(Par_Integ(0,I%))  :
                                                     Rem Kod
  260     Print #Nr% Text
  270   End Do
  280 Next
  290 Text=Chr$(32,78)                           :Rem 78 blanka tecken
  300 Mid$(Text,1,14)=Str$(Gen_Real_Par(0))      :Rem Samplingsintervall
  310 Mid$(Text,20,12)=Start_Date
  320 Mid$(Text,37,12)=Start_Time
  330 If Gen_Integ_Par(2)=1 Then Mid$(Text,53,14)=Str$(Time_Overrun) Else
      Mid$(Text,54,14)="0"                       :Rem överskrden Tid I RAMarean
  340 Print #Nr% Text
  350 Close(Nr%)
  355 Gen_Integ_Par(1)=Gen_Integ_Par(1)+1  :Rem Filnr.
  356 Length%=Len(Str$(Gen_Integ_Par(1)))-1
  357 I%=Instr(1,Filename_Dat,".")         :Rem Filnumret skall ökas med ett
  360 Filename_Dat=Left$(Filename_Dat,I%-Length%-1)+Right$(Str$(
      Gen_Integ_Par(1)),Length%)+".DAT"
  370 Filename_Dok=Left$(Filename_Dok,I%-Length%-1)+Right$(Str$(
      Gen_Integ_Par(1)),Length%)+".DOK"
EXIT 1
```

Procedure Mjolka

```
Procedure Mjolka
   EXTERNAL: First_time_open
   EXTERNAL: Filename_dat
   EXTERNAL: Gen_integ_par
   EXTERNAL: Gen_real_par
   EXTERNAL: Omformad_insignal
   EXTERNAL: Start_disk_uthamt
   EXTERNAL: Next_input
   EXTERNAL: First_in_ram
   EXTERNAL: Dokumentfil
   EXTERNAL: Start_tomning
   INTEGER: Nr%,I%
   EXTERNAL: Start_date_logg,Start_time_logg,First,Last,Start_disk
   INTEGER: A,B,St,Sl
   STRING: Old_dateÄ12Å,Old_timeÄ12Å
   INTEGER: Ok%
   REAL: Tov
   INTEGER: C,N

      10 Rem Procedure Som överför Data Från RAMarean Till En Fil på disk
            Samtidigt Som Loggning Pågår.
      20 N=2
      30 Do If First_Time_Open=1    :Rem Första Gången Filen öppnas
      40   First_Time_Open=0
      50   Open("o",Filename_Dat,N)
      60   Close(N)
      70   Open("a",Filename_Dat,N)
      80 End Do
      85 Ok%=0
      90 Old_Date=Start_Date_Logg : Old_Time=Start_Time_Logg
     100 A=Gen_Integ_Par(5) : B=Gen_Integ_Par(6) : C=A-1
     110 Start_Tomning=Start_Disk   :Rem Start_Tomning anger Numret På Första
         Sampelvärdet som skall överföras till en fil ,variabeln utnyttjas
         också för Att man Sedan skall Kunna Räkna Ut Antalet övrerförda
         Sampelvärden Till Filen.
     120 Do
     130   For I%=0 To C:Print #N Using "#.#####^^^^"Omformad_Insignal(
           (Start_Disk Mod B)*A+I%):Next:Start_Disk=Start_Disk+1:First=First+1:
           Last=Last+1:If Start_Disk=Next_Input Then Ok%=1:First_Time_Open=1:
           First_In_RAM=1:Sl=Start_Disk:St=Start_Tomning
     150 Repeat If Ok%=0
     152 Ok%=0
     155 Close(N):Tov=Gen_Real_Par(3):Gen_Real_Par(3)=0:Dokumentfil(
         Old_Date,Old_Time,Tov,Sl-St)
     157 Start_Tomning=Sl
     160 Print
     170 Print "Överföringen är klar"
     180 First_Time_Open=1
     190 Print:Print "H-CTRLB- Medför återhopp till Hjälpmenyn."
   EXIT 1
```

Procedure Stang_av_loggningen

```
Procedure Stang_av_loggningen
   EXTERNAL: Gen_integ_par
   EXTERNAL: Dokumentfil
   EXTERNAL: Mjolka
   EXTERNAL: Start_date_logg
   EXTERNAL: Start_time_logg
   STRING: TeckenÄ1Å
   EXTERNAL: Ny_start
   STRING: TextÄ4Å
   EXTERNAL: Start_tomning,Start_disk_uthamt,Gen_real_par,Num,Start_disk

    5 Rem Procedure Som Avslutar Loggningen Samt Gör Det Möjligt För Omstart
   10 Clearint(3) :Clearint(7)              :Rem stänger av avbrotten
   20 If Gen_Integ_Par(2)=1 Then Mjolka
   30 If Gen_Integ_Par(2)=0 Then Close(Num) :Dokumentfil(Start_Date_Logg,
      Start_Time_Logg,Gen_Real_Par(3),Start_Disk-Start_Tomning)
   40 Print
   50 Print "Överföringen av data klar"
   55 Do
   60   Input "Skall återstart ske (Y/N): "Tecken
   80 Repeat If Tecken<>"Y" And Tecken<>"N"
   90 If Tecken="Y" Then Ny_Start=1 Else Ny_Start=0
  100 Do If Ny_start=1
  105   Input "VTERM programmet skall vara laddat nu, skriv KLAR : "Text
  110 Repeat If Text<>"KLAR"
 EXIT 1
```

Procedure Mjolka_till_direkt

```
Procedure Mjolka_till_direkt
   EXTERNAL: Start_tomning
   EXTERNAL: Start_disk_uthamt
   EXTERNAL: Next_input
   EXTERNAL: Gen_integ_par
   EXTERNAL: Omformad_insignal
   EXTERNAL: First_time_open
   EXTERNAL: Filename_dat
   INTEGER: I%
   EXTERNAL: Start_disk,First,Last,Num

    1 Rem Procedure som ändrar konfigurationen från Mjölkning till
        Direkt ansluten.
    2 Clearint(7) :Rem stänger av kommunikations-avbrottet
    5 Num=2
   10 Do If First_Time_Open=1
   20   First_Time_Open=0
   30   Open("o",Filename_Dat,Num)
   40   Close(Num)
   50   Open("a",Filename_Dat,Num)
   60 End Do
```

```
   70 Start_Tomning=Start_Disk    :Rem Start_Tomning anger numret på
      första sampelvärdet som överförts till en fil på disk.Variabeln
      utnyttjas för att man senare skall kunna ange (i dokument-filen) hur
      många sampelvärden som överförts till filen.
   80 Do
   90   For I%=0 To Gen_Integ_Par(5)-1 : Print #Num Using "#.#####^^^^"
        Omformad_Insignal((Start_Disk Mod Gen_Integ_Par(6))*
        Gen_Integ_Par(5)+I%) : Next : Start_Disk=Start_Disk+1:First=First+1:
        Last=Last+1
  100 Repeat If Start_Disk<>Next_Input :Gen_Integ_Par(2)=0 :Rem övergången
      till Direkt överföring är nu klart.
  110 Setint(7,2)              :Rem sätter på kommunikations-avbrottet igen
  120 Print
  130 Print "Byte från Mjölkning till Direkt ansluten till disk är nu klart.
      "

EXIT 1
```

Procedure Direkt_till_mjolka

```
Procedure Direkt_till_mjolka
   EXTERNAL: Gen_integ_par
   EXTERNAL: First_in_RAM
   EXTERNAL: Start_date_logg
   EXTERNAL: Start_time_logg
   EXTERNAL: Gen_real_par
   EXTERNAL: Start_disk_uthamt
   EXTERNAL: Start_tomning
   EXTERNAL: First_time_open
   EXTERNAL: Dokumentfil
   EXTERNAL: Start_disk
   EXTERNAL: Num
   STRING: Old_dateÄ12Å,Old_timeÄ12Å
   INTEGER: Antal

     5 Rem Procedure som ändrar konfigurationen från Diirekt överföring till
       Mjölkning.
     7 Rem kommunikationsavbrottet stängs av
    10 Clearint(7):Close(Num):Gen_Integ_Par(2)=1:First_In_RAM=1:
       Old_Date=Start_Date_Logg:Old_Time=Start_Time_Logg:Antal=Start_Disk-
       Start_Tomning
    15 Dokumentfil(Old_Date,Old_Time,Gen_Real_Par(3),Antal)
    20 First_Time_Open=1
    22 Setint(7,2)  :Rem sätter på kommunikations-avbrottet igen
    25 Print
    30 Print "Byte från Direkt ansluten till Mjölkning klar"
EXIT 1
```

Interrupt Procedure Inlasning

```
Interrupt Procedure Inlasning
EXTERNAL: First
EXTERNAL: Last
EXTERNAL: Start_plottning
EXTERNAL: Start_disk_uthamt
EXTERNAL: Next_input
EXTERNAL: Gen_integ_par
EXTERNAL: Gen_real_par
EXTERNAL: New_sampel_flag
EXTERNAL: First_in_RAM
EXTERNAL: Insignal
EXTERNAL: Converted_signal
EXTERNAL: Omformad_insignal
EXTERNAL: Par_real
EXTERNAL: Par_integ
INTEGER: I%,Place_in_block,Hour,Min
STRING: Kommando$A16Å
EXTERNAL: Logg_disp_flag,Logg_Flag,Init_flag,Autoplot71,Plot_flag,
          New_plot_flag,Mjolk_flag,Stang_av_flag
EXTERNAL: Just_start_up
EXTERNAL: Start_date_logg
EXTERNAL: Start_time_logg
REAL: Sek
EXTERNAL: Help_flag,Mjolk_direkt_flag,Direkt_mjolk_flag
INTEGER: Nr%
EXTERNAL: Start_disk,Direkt_flag,Num,Start_tomning,Antal_CTRL3
INTEGER: CTRL8

   10 Rem Procedure Som Tar Hand Om Avbrotten
   20 On Intr Goto 880,880,30,880,880,880,580,880
   30 Rem Tidsavbrott
   40 Do If First_In_RAM=1      :Rem ny loggning eller då byte från Direkt
                                     överföring till Mjölkning har skett.
   50   GTime(Hour,Min,Sek)
   60   Start_Time_Logg=Time$
   70   Start_Date_Logg=Date$  :Rem Här Lagras Start-Tidpunkten för Loggningen
   80   Gen_Real_Par(5)=Hour*3600+Min*60+Sek
   90   If Just_Start_Up=1 Then Gen_Real_Par(4)=Gen_Real_Par(5) :
        Gen_Real_Par(2)=Gen_Real_Par(5) :Just_Start_Up=0   :Rem ny loggning
  100 End Do
  110 For I%=0 To 23            :Rem Läser In Signalerna
  120   If Par_Integ(3,I%)=1 Then Insignal(I%)=AIN(I%)     :Rem signalen loggas
  130 Next
  140 Place_In_Block=0
  150 For I%=0 To 23      :Rem Lägger De Inlästa Signalerna I RAMMET
  160   Do If Par_Integ(3,I%)=1
  170     Rem Först Konverteras Signalen
  180     If Par_Integ(1,I%)=0 Then Converted_Signal(I%)=Insignal(I%)*
          Par_Real(0,I%)+Par_Real(1,I%)
  190     If Par_Integ(1,I%)=1 Then Converted_Signal(I%)=Par_Real(0,I%)*
          SQR(Insignal(I%))
  200     Rem Nu Lagras Signalen I RAMMET
  210     Omformad_Insignal((Next_Input Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+
          Place_In_Block)=Converted_Signal(I%)
  220     Place_In_Block=Place_In_Block+1
  230   End Do
  240 Next
```

```
250 Do If First_In_RAM<>1
260   Do If Gen_Integ_Par(2)=1                    :Rem konfigurationen=Mjölkning
270     Do If Next_Input Mod Gen_Integ_Par(6)=First Mod Gen_Integ_Par(6)
280       First=First+1
290       Last=Last+1
300       Gen_Real_Par(3)=Gen_Real_Par(3)+Gen_Real_Par(0)    :Rem överskriden
                                                               Tid i RAMarean
310     End Do
320     Do If Next_Input Mod Gen_Integ_Par(6)=Start_Disk Mod Gen_Integ_Par(6)
330       Start_Disk=Start_Disk+1                  :Rem överskriden tid i RAMarean
340       Start_Tomning=Start_Tomning+1
350     End Do
360   End Do
370 End Do
380 Do If First_In_RAM<>1
390   Do If Next_Input Mod Gen_Integ_Par(6)=Start_Plottning Mod
      Gen_Integ_Par(6)
400     Start_Plottning=Start_Plottning+1
410     Gen_Real_Par(4)=Gen_Real_Par(4)+Gen_Real_Par(0)
420   End Do
430 End Do
440 Next_Input=Next_Input+1
450 Gen_Integ_Par(0)=Gen_Integ_Par(0)+1    :Rem Totalt Antal Samplade Värden
460 First_In_RAM=0
470 New_Sampel_Flag=1                  :Rem Anger Att Ett Nytt Sampel Har Kommit
480 New_Plot_Flag=1                    :Rem Anger att ett Nytt Sampel har kommit
                                           används vid Autoplottning.
490 Do If Gen_Integ_Par(2)=0           :Rem Direkt överföring till disk
500   For I%=0 To Gen_Integ_Par(5)-1
510     Nr%=(Start_Disk Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+I%
520     Print #Num Using "#.#####^^^^"Omformad_Insignal(Nr%)
530   Next
540   Start_Disk=Start_Disk+1
550   Last=Last+1 :First=First+1
560 End Do
570 Exit
580 On Error Goto 880                  :Rem Kommunikations-avbrott
590 Kommando$=""
600 Readchr Kommando$,Loc(0)
610 CTRLB=0
620 Do If Len(Kommando$)=1             :Rem CTRLB
630   If Asc(Left$(Kommando$,1))=2 Then CTRLB=1
640 End Do
650 Do If Len(Kommando$)=0 Or CTRLB=1
660   If Antal_CTRLB=1 Then Stang_Av_Flag=1  :Exit
670   Do If Antal_CTRLB=0
680     If Gen_Real_Par(0)<1.5 Then Stang_Av_Flag=1 Else Antal_CTRLB=1 :
        Direkt_Mjolk_Flag=1: Print "Övergång från Direkt överföring till
        Mjölkning sker nu."
690   End Do
700 End Do
```

```
  710 If Stang_Av_Flag=1 Then Print "Loggningen avslutas." :Exit
  720 Do If Len(Kommando$)>=1
  730   Do If Gen_Integ_Par(2)=1
  740     If Left$(Kommando$,1)="M" Then Mjolk_Flag=1 :Print "Mjölkning pågår.
          " :Exit                      :Rem MjöLkning
  750     If Left$(Kommando$,1)="S" Then Logg_Flag=1 :Exit   :Rem Visning Av
          de Loggbara Signaler
  760     If Left$(Kommando$,1)="E" Then Logg_Disp_Flag=0 :Exit   :Rem Avbryter
          Visningen Av De Loggbara Signalerna
  770     If Left$(Kommando$,1)="A" Then Autoplot71=0 :Exit  :Rem Avbryter
          Autoplottning
  780     If Left$(Kommando$,1)="H" Then Help_Flag=1 :Exit   :Rem Help-Menyn
  790     If Left$(Kommando$,1)="I" Then Init_Flag=1: Exit  :Rem  Visar
          Initieringsdelen
  800     If Left$(Kommando$,1)="P" Then Plot_Flag=1 : Exit   :Rem MöJligöR
          Plottning
  810     Do If Len(Kommando$)>=3
  820       Do If Left$(Kommando$,3)="*md"  :Rem Byte Fran Mjolka Till Direkt
  830         Antal_CTRLB=0 :Mjolk_Direkt_Flag=1:Print "Övergång från
              Mjolkning till Direkt overföring sker nu."
  840       End Do
  850     End Do
  860   End Do
  870 End Do
  880 Exit
EXIT !
```

Procedure Maximum

```
Procedure Maximum
   REAL ARG: Max/VAR
   INTEGER ARG: Last_place/VAR
   INTEGER ARG: Start
   INTEGER ARG: Sampel_avst
   INTEGER ARG: Last_input_nr
   INTEGER ARG: Blockplace
   INTEGER: K,I
   EXTERNAL: Omformad_insignal,Gen_integ_par

      10 Rem Procedure Som Beräknar Maximum På De Insamlade Värdena,
             Samt Anger platsen i RAMet för det sista värdet som skall plottas.
      20 Max=-1.0E+6
      30 I=0
      40 For K=Start To Last_Input_Nr Step Sampel_Avst
      50   If K>Last_Input_Nr Then Exit
      60   If I>70 Then Exit
      70   Last_Place=(K Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+Blockplace
      80   If Omformad_Insignal(Last_Place)>Max Then Max=Omformad_Insignal(
           Last_Place)
      90   Last_Place=K
     100   I=I+1
     110 Next
   EXIT 1
```

Real Function Minimum

```
Real Function Minimum
   INTEGER ARG: Start
   INTEGER ARG: Sampel_avst
   INTEGER ARG: Last_input_nr
   INTEGER ARG: Blockplace
   INTEGER: K,I,Nr
   REAL: Min
   EXTERNAL: Gen_integ_par,Omformad_insignal

      10 Rem Function Som Beräknar Minimum På De Värden Som Skall Plottas
      20 Min=1.0E+6
      30 I=0
      40 For K=Start To Last_Input_Nr Step Sampel_Avst
      50   If K>Last_Input_Nr Then Exit
      60   If I>70 Then Exit
      70   Nr=(K Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+Blockplace
      80   If Omformad_Insignal(Nr)<Min Then Min=Omformad_Insignal(Nr)
      90   I=I+1
     100 Next
     110 Result=Min
   EXIT 1
```

Integer Function Correct_input_p

```
Integer Function Correct_input_p
   INTEGER ARG: Meny
   INTEGER ARG: Rad
   STRING ARG: Text
   INTEGER ARG: Next_input
   INTEGER ARG: Start_plottning
   INTEGER ARG: Framforhallning
   INTEGER: Ans%,Nr%
   STRING ARRAY(2)Ä20Å: Tal
   INTEGER: I%,K%
   EXTERNAL: Par_Integ
   EXTERNAL: Gen_integ_par

     10 Rem Function som UndersöKer Om Strängen Text Innehåller Rätt
           Information (Rätt-1, Fel-0). Proceduren Terminal_input
           utnyttjar detta underprogram. "Rad" anger var den inmatade
           strängen kommer ifrån.
     20 Do If Meny=5              :Rem  "för plottning"
     30   Do If Rad=1             :Rem Antal Skalmarkeringar På X-Axeln
     40     Ans%=0
     50     If Len(Text)<1 Or Len(Text)>2 Then Exit
     60     If Convert_Integ(Text,Len(Text))=0 Then Exit
     70     If Val(Text)<2 Or Val(Text)>11 Then Exit
     80     If 70 Mod (Val(Text)-1) <>0 Then Exit
     90     Ans%=1
    100   End Do
    110   Do If Rad=2             :Rem Antal Plottade Signaler
    120     Ans%=0
    130     If Convert_Integ(Text,Len(Text))=0 Then  Exit
    140     If IVal(Text)=1 Or IVal(Text)=2 Then Ans%=1
    145     If IVal(text)>Gen_integ_par(5) Then Ans%=0
    150   End Do
    160   Do If Rad=3     :Rem "Signalnumrena" På De Signaler Som Skall Plottas
    170     Gosub 610
    180     If Ans%=0 Then Exit
    190     If Convert_Integ(Tal(0),Len(Tal(0)))=0 Then Ans%=0 : Exit
    200     If Convert_Integ(Tal(1),Len(Tal(1)))=0 Then Ans%=0 : Exit
    210     If Val(Tal(0))=Val(Tal(1)) Then Ans%=0 :Exit
    220     Gosub 680     :Rem Undersöker Om Signalerna Loggas
    230     If Nr%=2 Then Ans%=1 Else Ans%=0
    240   End Do
    250   Do If Rad=4     :Rem "Signalen" på den signal Som Skall Plottas
    260     If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 :Exit
    270     Gosub 680     :Rem Undersöker Om Signalen Loggas
    280     If Nr%=1 Then Ans%=1 Else Ans%=0
    290   End Do
    300   Do If Rad=5     :Rem Valalternativ 1,2 VäLj AUTO Eller VALDA Max,Min
    310     Ans%=0
    320     If Convert_Integ(Text,Len(Text))=0 Then Exit
    330     If Val(Text)<>1 And Val(Text)<>2 Then Exit
    340     Ans%=1
    350   End Do
    360   Do If Rad=6     :Rem Max-Värdet Resp. Min-Värdet
    370     Gosub 610
    380     If Ans%=0 Then Exit
    390     If Convert_Real(Tal(0))=0 Then Ans%=0: Exit
    400     If Convert_Real(Tal(1))=0 Then Ans%=0: Exit
    410     If Val(Tal(0))<=Val(Tal(1)) Then Ans%=0 :Exit
    420     Ans%=1
    430   End Do
```

```
  440   Do If Rad=7     :Rem Valalternativ 1,2,3,4
  450     If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 :Exit
  460     If Val(Text)<1 Or Val(Text)>4 Then Ans%=0 :Exit
  470     Ans%=1
  480   End Do
  490   Do If Rad=8     :Rem Nya Sampelavståndet
  500     If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 :Exit
  510     If Val(Text)>70 Or Val(Text)<=0 Or Val(Text)>Next_Input Then
          Ans%=0 :Exit
  520     Ans%=1
  530   End Do
  540   Do If Rad=9     :Rem Ny Sampel Start Punkt
  550     If Convert_Integ(Text,Len(Text))=0 Then Ans%=0 :Exit
  560     If Val(Text)>Next_Input-1-Framforhallning Or Val(Text)<
          Start_Plottning+Framforhallning Then Ans%=0 :Exit
  570     Ans%=1
  580   End Do
  590 End Do
  600 Goto 790
  610 Ans%=1            :Rem Läser In Signalnumrena
  620 If Len(Text)<3 Then Ans%=0 :Return
  630 K%=Instr(1,Text,",")
  640 If K%=0 Or Len(Text)=K% Or K%=1 Then Ans%=0 :Return
  650 Tal(0)=Left$(Text,K%-1)
  660 Tal(1)=Right$(Text,Len(Text)-K%)
  670 Return
  680 Nr%=0
  690 For I%=0 To 23
  700   Do If Rad=3
  710     If Val(Tal(0))=I% And Par_Integ(3,I%)=1 Then Nr%=Nr%+1
  720     If Val(Tal(1))=I% And Par_Integ(3,I%)=1 Then Nr%=Nr%+1
  730   End Do
  740   Do If Rad=4
  750     If Val(Text)=I% And Par_Integ(3,I%)=1 Then Nr%=Nr%+1
  760   End Do
  770 Next
  780 Return
  790 Result=Ans%
EXIT 1
```

Procedure Terminal_input

```
Procedure Terminal_input
   INTEGER ARG: Signal1/VAR
   INTEGER ARG: Signal2/VAR
   INTEGER ARG: Plott_alt/VAR
   INTEGER ARG: Sampel_avstand/VAR
   INTEGER ARG: Antal_signaler/VAR
   INTEGER ARG: Nr_xscale_points/VAR
   INTEGER ARG: Time_choice/VAR
   INTEGER ARG: Start_sampel/VAR
   INTEGER ARG: Last_input
   REAL ARG: Maxvalue/VAR
   REAL ARG: Minvalue/VAR
   EXTERNAL: Gen_integ_par
   INTEGER: Y%,X%,I%
   STRING: TextÄ20Å
   INTEGER: Pos%,Nr%
   EXTERNAL: Autoplot71,Correct_input_p,Start_plottning,Framforhallning

     10 Rem Procedure Som Läser In Värden Som Anger Hur Plottningen Skall Gå
          Till.
     20 Clear_Screen
     30 Print Cursor(1,1);Chr$(42,71)          :Rem ******
     40 Print Chr$(42,26);" PLOTTNING AV DATA ";Chr$(42,26)
     50 Print Chr$(42,71)
     60 Print Cursor(5,1);"Ange antal skalmarkeringar på x-axeln: "
     70 Y%=5 :X%=40 :I%=1
     80 Gosub 810  :Rem läser in en teckensträng och undersöker om den är rätt
     90 Nr_Xscale_Points=Val(Text)
    100 Print Cursor(7,1);"Ange antalet signaler som önskas plottas: "
    110 Y%=7 :X%=43 :I%=2
    120 Gosub 810
    130 Antal_Signaler=Val(Text)
    140 Do If Antal_Signaler=2
    150   Print Cursor(8,1);"Ange signal-numrena (signal 1,signal 2): "
    160   Y%=8 :X%=42 :I%=3
    170   Gosub 810
    180   Pos%=Instr(1,Text,",")
    190   Signal1=Val(Left$(Text,Pos%-1))
    200   Signal2=Val(Right$(Text,Len(Text)-Pos%))
    210 End Do
    220 Do If Antal_Signaler=1
    230   Print Cursor(8,1);"Ange signal: "
    240   Y%=8 :X%=14 :I%=4
    250   Gosub 810
    260   Signal1=Val(Text)
    270 End Do
    280 Print
    290 Print "1. Fysikaliska storheter _____ AUTO  -Max/Min"
    300 Print
    310 Print "2. Fysikaliska storheter _____ VALDA -Max/Min"
    320 Print Cursor(13,1);"Välj 1 eller 2 : "
    330 Y%=13 :X%=18 :I%=5
    340 Gosub 810
    350 Plott_Alt=Val(Text)
```

```
360 Do If Plott_Alt=2
370   Print Cursor(14,1);"Ange Max-värdet resp. Min-värdet (Max,Min): "
380   Y%=14 :X%=45 :I%=6
390   Gosub 810
400   Pos%=Instr(1,Text,",")
410   Maxvalue=Val(Left$(Text,Pos%-1))
420   Minvalue=Val(Right$(Text,Len(Text)-Pos%))
430 End Do
440 Sampel_Avstand=1
450 Time_Choice=0
460 Print Cursor(16,1);"1. Ändring av sampelavstånd"
470 Print "2. Möjlighet att ange start-tidpunkt för plottningen"
480 Print "3. Automatisk plottning av de 71 senast samplade värdena"
490 Print "4. Startar plottningen"
500 Print "Ange siffra: "
510 Do
530   Y%=20 :X%=14 :I%=7
540   Gosub 810
550   Nr%=Val(Text)
560   Clear_Rad(23,1,45) :Print
570   Do If Nr%=1
580     Print Cursor(21,1);"Nya sampelavstandet: "
590     Y%=21 :X%=22 :I%=8
600     Autoplot71=0
610     Gosub 810
620     Sampel_Avstand=Val(Text)
630   End Do
640   Do If Nr%=2
650     Time_Choice=1
660     Autoplot71=0
670     Print Cursor(22,1);"Plottningen skall starta från sampelnr. : "
680     Y%=22 :X%=43 :I%=9
690     Gosub 810
700     Start_Sampel=Val(Text)
710   End Do
720   Do If Nr%=3
730     Sampel_Avstand=1
740     Time_Choice=0
750     Clear_Rad(21,1,30) :Print: Clear_Rad(22,1,55):Print
760     Print Cursor(23,1);"Autoplott på de 71 senast samplade värdena"
770     Autoplot71=1          :Rem kontinuerlig plottning av de 71
                                   senast samplade värdena
780   End Do
790 Repeat If Nr%<>4
800 Exit
810 Do        :Rem läser in en teckensträng och undersöker om den är rätt
820   Text=""
830   Clear_Rad(Y%,X%,20)
840   Screen_Display(Y%,X%,Text,20)
850 Repeat If Correct_Input_P(5,I%,Text,Last_Input,Start_Plottning,
    Framforhallning)=0
860 Return
EXIT 1
```

Procedure Plottning

```
Procedure Plottning
   EXTERNAL: Gen_integ_par
   EXTERNAL: Omformad_insignal
   EXTERNAL: Gen_real_par
   EXTERNAL: Par_str
   EXTERNAL: Par_integ
   CONSTANT: Length_xkoord%=70,Length_ykoord%=16,Ykoord_start%=10,
             Scale_heights%=4
   STRING: Empty$Ä80Å
   INTEGER: Start%,Plottcounter%,Signal1%,Signal2%,Plott_alt%,Sampel_avst%,
            Antal_signaler%,Nr_xscale_points%
   INTEGER: Time_choice%
   REAL: Max_value,Min_value,Min_time
   INTEGER: Begin_start%
   STRING ARRAY(23)Ä80Å: Rad$
   INTEGER: K%
   REAL: Max_value1
   INTEGER: Nr%
   REAL: Max_time,Min_value2,Max_value2
   INTEGER: Typ_av_prec%,Scale_widths%,I%,Xpos%
   REAL: Scale_time
   STRING: Scale_tid$Ä16Å
   INTEGER: Ypos%,Start_sampel%
   REAL: Scale_value
   STRING: TeckenÄ1Å
   INTEGER: Continue_plott%
   REAL: Spann_vidd,Rad_avst,Skal_avst,Potens,New_scale,Data1,Data2,
         New_scale_dist
   REAL: First_scale_mark
   INTEGER: Sort%
   REAL: Scale_value1,Scale_value2
   INTEGER: Last_input_nr,Place,Blockplace1,Blockplace2,Last_place
   EXTERNAL: Next_input,Framforhallning,New_plot_flag,Autoplot71
   EXTERNAL: Start_plottning
   EXTERNAL: Terminal_input
   EXTERNAL: Maximum
   EXTERNAL: Minimum
   EXTERNAL: Filename_dat
   EXTERNAL: Help_meny_v

    5 Rem Procedure som  kan plotta max 2 signaler
   10 Last_Input_Nr=Next_Input-1    :Rem sista samplet
   15 If Last_input_nr<1 Then Exit :Rem måste ha läst in minst 2 värden
   20 Empty$=Chr$(32,80)            :Rem 80 blanktecken
   30 Plottcounter%=1
   40 Continue_Plott%=1             :Rem anger om en plottning skall ske
   50 Begin_Start%=0                :Rem anger var plottningen skall börja
   55 Rem Hämtar all information som behövs för att en plottning skall
         kunna ske
   60 Terminal_Input(Signal1%,Signal2%,Plott_Alt%,Sampel_Avst%,
```

```
    Antal_Signaler%,Nr_Xscale_Points%,Time_Choice%,Start_Sampel%,
    Last_Input_Nr,Max_Value,Min_Value)
 70 Place=0 : Blockplace1=0 : Blockplace2=0
 80 For I%=0 To 23      :Rem Letar Rätt på  Var Signalerna Ligger I RAMet
                           (de som skall plottas)
 90   Do If Par_Integ(3,I%)=1                    :Rem Signalen Loggbar
100     If Signal1%=I% Then Blockplace1=Place
110     Do If Antal_Signaler%=2
120       If Signal2%=I% Then Blockplace2=Place
130     End Do
140     Place=Place+1
150   End Do
160 Next
170 Do If Last_Input_Nr>Framforhallning       :Rem plottning får först ske
    när ett visst antal signaler har samlats in
180   Do If Time_Choice%=1 And Autoplot71=0   :Rem ny sampelstart-tidpunkt
190     Begin_Start%=Start_Sampel%
200     Start%=Begin_Start%
210   End Do
220   Do If Time_Choice%=0 And Autoplot71=0
230     Start%=Start_Plottning+Framforhallning
240     Begin_Start%=Start%
250   End Do
260   Do If Autoplot71=1                        :Rem automatisk plottning av
      de 71 senast samplade värdena
270     If Next_Input>=71 Then Start%=Next_Input-71 :Last_Input_Nr=
        Next_Input-1 Else Start%=0 : Last_Input_Nr=Next_Input-1
280     Begin_Start%=Start%
290     New_Plot_Flag=0    :Rem Blir Ett Då Ett Nytt Sampelvärde har kommit
300   End Do
310   For K%=17 To 21      :Rem huvudet på plottningsbilden, som innehåller
      information om signalerna
320     Rad$(K%)=Empty$
330   Next
340   Rad$(19)="Datafilnamn : "+Filename_Dat
350   Rad$(18)="Signal"+Str$(Signal1%)+" = *            Enhet : "+Par_Str(
      0,Signal1%)+"            Mätområde : "+Str$(Par_Integ(0,Signal1%))
360   Do If Antal_Signaler%=2
370     Rad$(17)="Signal"+Str$(Signal2%)+" = +           Enhet : "+Par_Str
       (0,Signal2%)+"            Mätområde : "+Str$(Par_Integ(0,Signal2%))
380   End Do
390   ""
400   "*****************************************************************"
410   "******************** Sjalva prog for plottningen*****************"
420   "*****************************************************************"
430   While Continue_Plott%<>0 Do  :Rem plotta tills Continue_Plott blir 0
440     Continue_Plott%=0
450     "*********************************************************"
460                     Rem Y-Axeln skapas
470     "*********************************************************"
480     For K%=0 To 16
490       Rad$(K%)=Empty$
500       Mid$(Rad$(K%),Ykoord_Start%,1)="!"
510     Next
520     "*********************************************************"
530                     Rem X-Axeln skapas
540     "*********************************************************"
550     For K%=Ykoord_Start% To 80
560       Mid$(Rad$(0),K%,1)="-"
570     Next
580     Rem Beräknar maximum och minimum på de signaler som skall plottas
```

```
590     Maximum(Max_Value1,Last_Place,Start%,Sampel_Avst%,Last_Input_Nr,
        Blockplace1)
600     Max_Time=Last_Place*Gen_Real_Par(0)+Gen_Real_Par(2)
610     Min_Time=Gen_Real_Par(2)+Gen_Real_Par(0)*Start%
620     ""
630     Do If Plott_Alt%=1          :Rem Auto Max,Min
640       Min_Value=Minimum(Start%,Sampel_Avst%,Last_Input_Nr,Blockplace1)
650       Do If Antal_Signaler%=2
660         Min_Value2=Minimum(Start%,Sampel_Avst%,Last_Input_Nr,
            Blockplace2)
670         If Min_Value2<Min_Value Then Min_Value=Min_Value2
680         Maximum(Max_Value2,Last_Place,Start%,Sampel_Avst%,
            Last_Input_Nr,Blockplace2)
690         If Max_Value2>Max_Value1 Then Max_Value=Max_Value2
            Else Max_Value=Max_Value1
700       End Do
710       Do If Antal_Signaler%=1
720         Max_Value=Max_Value1
730       End Do
740     End Do
750     If Max_Value-Min_Value<=0 Then Exit  :Rem FåR Ej Dividera Med Noll
760     Rem Väljer Precision på y-axeln (mätvärdena),dvs. om de skall
            skrivas ut på formen (a.aa+E-xx)  eller ej
770     Do If Abs(Max_Value)=0
780       If Abs(Min_Value)<1.0E+06  And Abs(Min_Value)>1.0E-06 Then
          Typ_Av_Prec%=0       Else  Typ_Av_Prec%=1
790     End Do
800     Do If Abs(Min_Value)=0
810       If Abs(Max_Value)<1.0E+06 And Abs(Max_Value)>1.0E-06 Then
          Typ_Av_Prec%=0        Else Typ_Av_Prec%=1
820     End Do
830     Do If Abs(Max_Value)<>0 And Abs(Min_Value)<>0
840       If (Abs(Max_Value)<1.0E+06) And (Abs(Min_Value)>1.0E-06) Then
          Typ_Av_Prec%=0   Else  Typ_Av_Prec%=1
850     End Do
860     Scale_Widths%=Length_Xkoord%/(Nr_Xscale_Points%-1)
870     "*********************************************************"
880             Rem Ritar Skal-Markeringar På X_Axeln
890     "*********************************************************"
900     For I%= 1 To (Nr_Xscale_Points%-1)
910       Mid$(Rad$(0),I%*Scale_Widths%+Ykoord_Start%,1)="!"
920     Next
930     "*********************************************************"
940             Rem Tidsmarkering Av X-Axeln
950     "*********************************************************"
960     For I%=0 To (Nr_Xscale_Points%-1)
970       Xpos%=I%*Scale_Widths%
980       Scale_Time=Xpos%*(Max_Time-Min_Time)/Length_Xkoord%+Min_Time
990       Time_Converter(Scale_Tid$,Scale_Time,0)
1000      Do If I%<>(Nr_Xscale_Points%-1)
1010        Mid$(Rad$(21),I%*Scale_Widths%+4,8)=Scale_Tid$
1020      End Do
1030      Do If I%=(Nr_Xscale_Points%-1)
1040        Mid$(Rad$(21),I%*Scale_Widths%+2,8)=Scale_Tid$
1050      End Do
1060    Next
1070    "*********************************************************"
1080            Rem Utplacering Av Värdena I Grafen
1090    "*********************************************************"
1100    I%=0
1110    For K%=Start% To Last_Input_Nr Step Sampel_Avst%
```

```
1120      If K%>Last_Input_Nr Then Exit
1130      If I%>70 Then Continue_Plott%=1:Exit
1140      Nr%=(K% Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+Blockplace1
1150      Xpos%=(Length_Xkoord%/(Max_Time-Min_Time))*(Gen_Real_Par(2)+
          Gen_Real_Par(0)*K%-Min_Time)
1160      Ypos%=(Length_Ykoord%/(Max_Value-Min_Value))*(Omformad_Insignal(
          Nr%)-Min_Value)
1170      Do If (Ypos%>=0) And (Ypos%<=16)
1180        Mid$(Rad$(Ypos%),Xpos%+Ykoord_Start%,1)="*"
1190      End Do
1200      Do If Antal_Signaler%=2
1210        Nr%=(K% Mod Gen_Integ_Par(6))*Gen_Integ_Par(5)+Blockplace2
1220        Ypos%=(Length_Ykoord%/(Max_Value-Min_Value))*(
            Omformad_Insignal(Nr%)-Min_Value)
1230        Do If (Ypos%>=0) And (Ypos%<=16)
1240          Mid$(Rad$(Ypos%),Xpos%+Ykoord_Start%,1)="+"
1250        End Do
1260      End Do
1270      I%=I%+1
1280    Next
1290    "************************************************************"
1300            Rem Utskrift Av HEADING, 4-Rader
1310    "************************************************************"
1320    For I%=20 To 17 Step -1
1330      Print Rad$(I%)
1340    Next
1350    "************************************************************"
1360    Rem Utskrift Av Värdena + Numrering Och Skalmarkering Av Y-Axeln
          Först beräknas en lämplig skala för y-axeln
1370    "************************************************************"
1380    Spann_Vidd=Max_Value-Min_Value
1390    Rad_Avst=Spann_Vidd/16
1400    Skal_Avst=Spann_Vidd/4
1410    Potens=INT(LOG(Skal_Avst)/LOG(10))
1420    New_Scale=Skal_Avst/(10^Potens)
1430    Data 2,2.5,3,5,10
1440    Restore
1450    Read Data1
1460    While Data1<>10 Do
1470      Read Data2
1480      If Abs(New_Scale-Data1)<=Abs(New_Scale-Data2) Then Exit
          Else    Data1=Data2
1490    Repeat
1500    New_Scale_Dist=Data1*10^Potens
1510    New_Scale=(Int(Max_Value/New_Scale_Dist))*New_Scale_Dist
1520    Scale_Value1=Length_Ykoord%*(Max_Value-Min_Value)/Length_Ykoord%+
        Min_Value
1530    For I%=Length_Ykoord% To 0 Step -1  :Rem Skalmarkerin Av Y-Axeln
1540      Sort%=0
1550      Do If I%<>0
1560        Scale_Value2=(I%-1)*(Max_Value-Min_Value)/Length_Ykoord%+
            Min_Value
1570        Do If Abs(New_Scale-Scale_Value1)<=Abs(New_Scale-Scale_Value2)
            And Abs(New_Scale-Scale_Value1)<=Rad_Avst/2
1580          If Mid$(Rad$(I%),Ykoord_Start%,1)<>"*" Then Mid$(Rad$(I%),
              Ykoord_Start%,1)="-"
1590          If Typ_Av_Prec%=1 Then Print Using "#.##^^^^"New_Scale;:
              Print Mid$(Rad$(I%),Ykoord_Start%,71); Else Mid$(Rad$(I%),1,
              Ykoord_Start%-1)=Str$(New_Scale):Print Rad$(I%);
1600          New_Scale=New_Scale-New_Scale_Dist
1610          Scale_Value1=Scale_Value2
```

```
1620            Sort%=1
1630          End Do
1640          Do If Sort%=0
1650            Print Rad$(I%);
1660            Scale_Value1=Scale_Value2
1670          End Do
1680        End Do
1690        Do If I%=0
1700          Do If Abs(New_Scale-Scale_Value1)<=Rad_Avst/2
1710            If Mid$(Rad$(I%),Ykoord_Start%,1)<>"*" Then Mid$(Rad$(I%),
                Ykoord_Start%,1)="-"
1720            If Typ_Av_Prec%=1 Then Print Using "#.##^^^^"New_Scale;:
                Print Mid$(Rad$(I%),Ykoord_Start%,71); Else Mid$(Rad$(I%),1,
                Ykoord_Start%-1)=Str$(New_Scale):Print Rad$(I%);
1730          End Do
1740          Do If Abs(New_Scale-Scale_Value1)>Rad_Avst/2
1750            Print Rad$(I%);
1760          End Do
1770        End Do
1780      Next
1790      Print Rad$(21)          :Rem tidsangivelserna på x-axeln skrivs ut
1800      Start%=71*Plottcounter%+Begin_Start%
1810      Plottcounter%=Plottcounter%+1
1820      Do If  Autoplot71=0
1830        Print Cursor(24,1);"Skall fler värden plottas (Y/N): ";
1840        Tecken="" : Clear_Rad(24,34,4): Screen_Display(24,34,Tecken,1)
1850        If Tecken<>"Y" And Tecken<>"N" Then Goto 1840
1860        If Tecken="N" Then Continue_Plott%=0
1870      End Do
1880      Do If Gen_Real_Par(0)<6 And Autoplot71=1
1890        Print Cursor(24,1);"Skall de 71 senast samplade vÄrdena
            plottas igen (Y/N): ";
1895        Print Cursor(1,1);:Print
1900        Do
1910          Tecken=""
1920          Clear_Rad(24,57,2)
1930          Screen_Display(24,57,Tecken,2)
1940        Repeat If Tecken<>"Y" And Tecken<>"N"
1950        If Tecken="N" Then Autoplot71=0
1960      End Do
1970      If Autoplot71=1 And Gen_real_par(0)>=6 Then Print Cursor(24,1);
          "A-CTRLB  Avslutar plottningen och återhopp sker till komando
          menyn";
1980      Do If Autoplot71=1    :Rem autmatisk plottning av de 71 senast
                                    samplade värdena
1990        Rem väntar på att ett nytt sampel skall komma
2000      Repeat If New_Plot_Flag=0
2010      Do If Autoplot71=1
2020        Continue_Plott%=1 : New_Plot_Flag=0
2030        If Next_Input>=71 Then Start%=Next_Input-71 : Last_Input_Nr=
            Next_Input-1 Else Start%=0 : Last_Input_Nr=Next_Input-1
2040        Begin_Start%=Start%
2050      End Do
2060    Repeat
2070  End Do
2080  Help_meny_v               :Rem skriver ut hjälpmenyn
EXIT 1
```