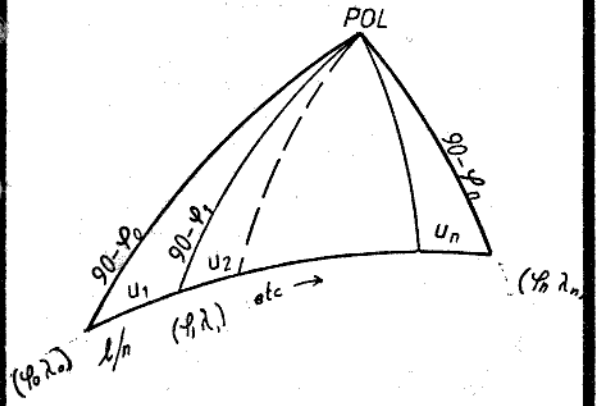
PROGRAM nittinian BITEN

84-4



Innehåll

ISSN 0281-1146

Redaktören. . .

Ja, nu har jag arbetat med tidningen ett år och likaså varit redaktör denna period. Det har varit slitigt, tungt och svårt att hinna med både skola, hemliv och samtidigt göra en bra tidning. Men oooooj-- vad roligt jag haft och så många underbara människor jag har lärt känna och nu har ett givande samarbete med. Jag tackar er alla av hela mitt hjärta för förmånen att ha upplevt detta och i synnerhet tackar jag Dig som har skickat in bidrag till tidningen. Det finns några jag vill ge en eloge av tacksamhet till och det är Claes Schibler, Lennar Lindberg, Anders Persson, Michael Öhman, Börje Häll, Bengt Fahlgren, Tony Rogvall, Björn Mårtensson, förre redaktören Björn Gustavsson, Örjan Gustavsson, Bo Nordlin, Lars Ekeroth, Michael Dahlquist, Raul Hofman, Jan Alexandersson, Arne Wennberg och Lars-Erik Svahn (säkert har jag glömt några). Jag tackar er alla för ert flit och er uthållighet i att hålla tidningen i gång och att ideligen sända in bidrag till ProgramBiten samt ert stora intresse för denna värld med datorer och räknare.

Utmaningen på sid 4 fortsätter i detta nummer med subrutiner och strukturerad programmering som påbörjades i förra numret. Och därmed avslutar Anders dessa aspekter och tankar genom att beskriva ett Telefonregister.

Dissasemblera GROM 0 (Basic tolken) eller CRU (kommunikations-register-enheten). Detta och mer därtill kan du göra med Tony Rogvall's program på sid 8. Programmet är skapat i Assembler och är drygt fyra sidor långt.

Har du en Speech Synthesizer (pratorn) kan du få den att 'prata' både det ena och det andra. Arne Wennberg har skapat ett program i Extended-Basic vilket utvidgar pratorns ordförråd. Du finner programmet på sid 12. Lars-Erik Svahn har forskat i det här med koderna till pratorn. Vad de heter och vad de gör. En fin introduktion till pratorns värld och lärorik för dig som tänker programmera i Assembler för pratorn. Artikeln börjar på sid 14.

Problem med tecken i listningar av program, vad är ett U med två prickar över och vad betyder det där konstiga h'et i listningarna ? Claes, vår allt-i-allo förklarar detta på sid 16. I anslutning till detta finns ett program från programbanken vilket skriver ut ASCII-koden för alla tangenter. Vi har publicerat det i detta nummer på sid 17.

På sidan 18 börjar TI59 programmen. Vi har först ett program av Hofman. Storcirkelnavigering vilket hjälper seglaren att hålla kursen. Harnevie har knappat in ett program för Reglerteknik eller kretsanalys. Nyttigt även för oss data-knuttar. Dessa program fyller några sidor och borde ge alla som har kunskaper i matematik intressanta uppslag.

Sprites med Mini-Memory och TI-Basic beskrivs på sid 24 av Jan Alexandersson. Vi har även publicerat två andra program han har skrivit för Mini-Memory. Det är Textmode på sid 30 och Diskkatalog i Textmode på sid 25. Tyvärr har Jan inte hunnit med att beskriva dem. Men jag tror att ni kan lista ut hur de fungerar. Det finns förklaringar i programmen.

Tips och Tricks är denna gången skriven av Jan Alexandersson. Det var så många tips och tricks att de var värda att publiceras i sin helhet. Slå upp sid 26 och du kan läsa om loppor (bugs) och icke omtalade möjligheter.

Vet du vad du har på dina kassetter, inte ? Knappa då in Sven-Erik Wind's program på sid 29. Katalogprogram för kassetthantering.

Rita och plotta i Forth tycker jag för min del är mycket roligt. Michael Dahlquist har på sid 30 tre skärmar med grafikrutiner och på sid 32 har Örjan Gustavsson 15 skärmar för plottning med Forthen. Jag skulle med glädje vilja se mer av Forth program i vår tidning.

Kommer ni ihåg P:TEXTIN och P:TEXTUT i nr. 3/84 ? Nu kommer Börje Häll med redaktionens redigeringsprogram för era insända texter med hjälp av P:TEXTIN. Det kallas P:TILLCOMP vilket är avsett att användas för konvertering till COMPANION-filer. Ett ordbehandlingsprogpram som vi använder. Har du COMPANION använd P:TILLCOMP. Du finner det på sid 35.

Bortglömda brev hittar man alltid i sina ägor. Vi har publicerat ett på sid 36 och ber om överseende med vår glömska. Vi hinner inte allt, tidningen först, det andra senare (men när ?).

Hur kan det komma sig att listningarna av Basicprogram är så snyggt utskrivna i tidningen ? Jo, Bo Nordlin's listprogram hjälper oss avsevärt med det problemet. Ett program som jag ber er att använda om ni vill skicka färdiga listningar till oss på tidningsredaktionen. För oss har det blivit ett måste att använda detta listprogram . Du finner det på sid 38.

Häng kvar nästa år och ös på med mer program till Skandinaviens bästa tidning för TI99/4A och TI59.

Göran Nygren

PS Vi hann i sista stund med att publicera ett brev om medlemsmötet i Lund. Det finns att läsa på sid 17. Och Programbanken har fått sitt nya utseende på sista sidorna. DS

I REDAKTIONEN:

Redaktör	Göran Nygren
Biträdande redaktör	Michael Öhman
Utmaningsredaktör	Anders Persson
Programförmedlare	Björn Mårtensson
Allt-i-allo	Claes S

Föreningens och redaktionens adress:

Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 6 1 tr ned
121 46 Johanneshov

Postgiro 19 83 00 - 6

+++++++++++++++++++++++++
+ DATAINSPEKTIONENs +
+ LICENSNUMMER +
+ +
+ 82100488 +
+++++++++++++++++++++++++

Medlemsavgift för 1985 är ..... 120 SEK
Nittinian, årgång 1983 (nr 1, 2, 3, 4/5) kostar 80 SEK
Programbiten, årgång 1984 (nr 1, 2, 3, 4)..... 100 SEK

ANVÄNDARTIPS MED MINI MEMORY

Denna bok skrevs av Björn Gustavsson på uppdrag av Texas Instruments. Precis när boken var klar lade TI ner hemdatorn. Föreningen fick då rätten till boken och har tryckt upp den. Boken innehåller 60 sidor och kostar för medlemmar 60 SEK som sätts in på föreningens postgiro 19 83 00 - 6.

VIKTIGT MEDDELANDE

Föreningen kan från och med nr 84-1 inte lämna någon ersättning för artiklar och program enligt notis i Nittinian 2.

Annonser, insatta av enskild medlem (ej företag), som gäller försäljning av moduler eller andra tillbehör i enstaka exemplar är gratis.

Övriga annonser kostar 2000 SEK per helsida, förutom baksidan som kostar 3000 SEK.

ORDFÖRANDEN har ORDET

Det här är fjärde gången jag skriver den här spalten. Det är sista ordinarie gången för i år. "Sista ordinarie" skriver jag därför att vi funderar på att komma ut med ett extranummer före årsmötet. Vi har en del material såväl från medlemmar som ur utländsk press. Det kan ju vara synd att låta det ligga i lådorna. Orkar vi med så kommer det!

Den förra tidningen tycker jag blev litet väl kraftigt peppad med diverse uppmaningar till Dig och andra medlemmar - att skriva, att ta kontakter mm. Kanske blev Du irriterad. Det kan jag väl förstå i så fall. Ändå förstår Du nog att vi menar väl. Jag är övertygad om att Du och andra medlemmar har en hel del saker som är värda att berätta om. Vi vill bara hjälpa Dig över tröskeln, så att säga. Det är faktiskt dessutom så att "delad glädje är dubbel glädje, delad sorg är halv sorg". Så vad har Du att berätta?

I den första spalten i våras talade jag om saker vi tänkte göra. I stort går det som vi tänkte, och nu har vi ett par saker på gång:

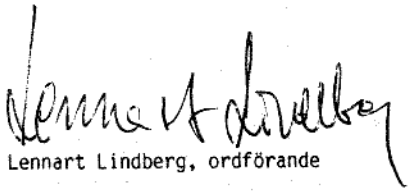
- en telefonjour har beslutats på styrelsemöte. Det blir ett stockholmsnummer som kommer att passas vissa tider av personer som kan och vill svara på frågor och resonera dator. Nummer och tider kommer att tillkännages snart.

- en begagnad dator med ett stort direktaccessminne har dykt upp. Den verkar lämpad för den föreningens databas vi talat om tidigare. Nu handlar det om pris, driftkostnader, driftorganisation, underhåll mm - samt förstås medlemsintresse. Vi återkommer!

Om Du vill vara med om de här arrangemangen och annat vi gör - fortsätt att vara medlem nästa år också! Värva dessutom gärna någon som har en 99:a och som Du tror skulle vara intresserad. Ju fler vi blir, desto mera sakkunskap och erfarenhet finns det att dela på.

Som jag rapporterat tidigare har Volvo skaffat några hundra 99:or som till fördelaktiga priser placerats hos anställda. Tyvärr har vi inte fått kontakt med dem i den omfattning jag hoppats. Har Du någon ide om hur vi kan bära oss åt med detta? Välkommen med den i så fall!

Låt mig sluta med att önska trevlig helg! Om det blir innesittarväder är datorn en utmärkt sysselsättning. Och helgen är lång. Såväl jag som styrelsen i övrigt och - tror jag - läsarna av vår tidning tar gärna del av resultatet av Din 'hacking'. Glöm heller inte årsmötet lördagen den 23 februari. Då bör Du ha gjort Din självdeklaration och kan behöva tänka på annat!



Lennart Lindberg, ordförande

PS

Ju tidigare Du för fram de förslag Du vill ha behandlade på årsmötet, desto säkrare är Du att de verligen blir förberedda och förhoppningsvis väl behandlade.

DS

Reservera LÖRDAGen den 23 FEBRUARI 1985 för

Årsmöte

PLATS meddelas senare

FÖRENINGENs TILLBEHÖRs-FÖRSÄLJNING

Följande finns att köpa för medlemmar genom att motsvarande belopp sätts in på postgiro 19 83 00 - 6.

Användartips med Mini Memory	60:-
FORTH Olika versioner, se annons på annan plats i tidningen (sid 37)	
Nittinian T-tröja	40:-
99'er Magazine nr 12/82 nr 1-5,7-9/83 (per styck)	20:-
Nittinian, årgång 1983	80:-
Astronomical Formulae for Calculators	Slut

Programbiten, årgång 1983	Kalkylatorer	80:-
Programbiten, årgång 1982	-"-	80:-
Programbiten, årgång 1981	-"-	60:-
Programbiten, årgång 1980	-"-	60:-
Programbiten, årgång 1978/79	-"-	60:-
Programbiten, fem årgångar 1978-1983		280:-
Katalog med belgiska och engelska program för räknare TI-57, TI-58, TI-59		20:-
Föreningens programmeringsblanketter (TI-59), olika typer, block om 50 blanketter (se pb 83-1 sidan 30)		11:-
Patenthandlingar TI-59		25:-
40 st tomma magnetkort med plånbok		150:-
Tom magnetkortsplånbok		10:-

DATAVISION MODUL
Modulen gör att du kan kommunicera med DATAVISION eller liknande baser.
Utskrift för printer finns tillgänglig, sparning av skärmar på kassett eller diskett.
Nödvändig utrustning:
RS232 interface
Televerkets modem eller liknade
Modulen kommer med instruktioner om hur du kopplar upp dig med Televerkets modem.

Pris: 565:-

16 KRAM CMOS MODUL
utan hölje. Modulen kan simulera ett ROM, inladdning sker från T.ex FORTH vilken medför att minimum utrustning är X-basic, MINIMENORY, eller ED/ASM och 32 K minnesutökning. Ram'et är batteribackupat så programmet ligger kvar minst 2 veckor. Denna modul används i princip för att testa ut hur egna program i modulform fungerar innan man bränner prommar vilket blir klart jobbigare.

Pris: 875:-

UTMANINGEN!

Redaktörens adress:
Anders Persson
Kämnärsvägen 4:1078
222 45 LUND

* TELEFONREGISTER

Som avslutning på all teori om underprogram i allmänhet och SUB program i synnerhet, tar jag upp ett exempel från dataregistrens värld.

Den programmeringsmetodik som används här gör väl inte direkt anspråk på att vara den bästa som finns. Avsikten är snarare att visa hur man kan koda ett program någorlunda vettigt i Extended BASIC. Dessutom finns det i alla fall ett visst mått av ordning, vilket är mer än vad många så kallade "hackers" håller sig med. Lägg särskilt märke till att man bestämmer datastrukturen först, innan det blir fråga om hur den ska manipuleras. Dock bör man veta vilka operationer som kan komma ifråga, eftersom det annars är lätt att välja en datastruktur som inte passar för vissa operationer.

Den här varianten av registerprogram är i enklaste laget för att göra någon verklig nytta, men eftersom det formligen kryllar av SUB i programmet som blir resultatet, tjänar det i alla fall som en god demonstration av hur man kan dela upp ett program i självständiga delar. Dessutom är det inte speciellt svårt att bygga ut programmet så att det blir mer praktiskt användbart. Redan vid den grundläggande konstruktionen ska vi ta hänsyn till eventuella framtida utbyggnader.

Låt oss ställa upp en liten kravspecifikation för vårt program. Den kan se ut så här:

1. Registret ska innehålla namn och telefonnummer. Vi bryr oss inte om adresser eller annat nu, men lämnar så att säga dörren öppen för eventuella framtida tillägg.

2. Man ska kunna söka på ett visst namn och få reda på alla namn som stämmer överens. Givetvis ska även tillhörande telefonnummer visas.

3. Det ska gå att få hela registret, i bokstavsordning med avseende på namnen, utskrivet på skrivare, eller på skärmen om man inte har någon skrivare.

4. Det ska gå att lagra registret på kassett eller diskett. Att man ska kunna läsa in det igen hoppas jag är uppenbart.

5. Man ska kunna lägga till nya namn och telefonnummer när som helst. I ett riktigt register måste man även kunna ändra namn och telefonnummer som har matats in tidigare, samt ta bort namn. Dessa möjligheter bryr vi oss dock inte om nu.

Antag att vi är nöjda nu. Då är det dags att titta lite på vilka funktioner programmet behöver ha för att det ska klara av att uppfylla kraven.

Vi kan börja med att ställa upp de olika funktionerna som behövs i en tabell. Denna tabell kan sedan kanske bli programmets huvudmeny, där man väljer olika funktioner när programmet har startats. Säg att vi vill ha det så här:

1. Inmatning
2. Sortering
3. Sökning
4. Utskrift
5. Lagring (på kassett eller disk)
6. Hämtning (från kassett eller disk)
7. Radering (av registret)
8. Avsluta (programkörningen)

Nu var det tänkt att det här skulle vara så enkelt som möjligt. Vi kan rationalisera lite om vi tänker oss att vi alltid har registret sorterat. Då behöver vi inte ha någon särskild funktion för det. Vi kan ju sortera automatiskt varje gång nya namn matas in. Det ställer naturligtvis krav på sorteringsrutinen. Den får inte vara för långsam. Annars somnar kanske den som ska använda programmet. Dessutom kan vi säga att vi raderar registret genom att stanna programmet och starta det igen. Då ser det ut så här:

1. Inmatning
2. Sökning
3. Utskrift
4. Lagring
5. Hämtning
6. Avsluta

Nästa steg blir att bestämma hur informationen ska representeras i datorn. Eftersom vi ska lagra namn är det givet att någon form av strängvariabel måste utnyttjas. Enklast blir då att låta även telefonnumret lagras som text. I regel vill man ju ha ett tecken mellan riktnummer och abonnentnummer. Då är det enklast om även numret representeras som text.

Det blir ju åtskilliga namn som vi ska hålla reda på, varför vi måste ha ett fält att lagra dem i. Eftersom vi har mer än en uppgift för varje post, blir det då lämpligt med ett tvådimensionellt fält. Den lösningen tillåter oss att ange olika poster med ett index och olika delar i posten med ett annat index. Att varje post består av två delar, namn och nummer, har vi redan bestämt. Låt oss nu bestämma att vi låter registret omfatta maximalt 100 personer. Då är det inga problem att få plats med både program och data, även för de som inte har någon minnesexpansion.

Det ända som behövs nu är att bestämma vilket variabelnamn registret ska ha, t.ex. REG$.

I programmet kommer det alltså att finnas en sats som ser ut så här:

```
DIM REG$(100,2)
```

Genom att utnyttja 99:ans instruktion

```
OPTION BASE 1
```

blir vi av med index noll i fälten. Det indexet behöver vi ändå inte i det här programmet, så det är lika bra det.

Med den här datastrukturen kan vi alltså adressera det 14:e namnet med

```
REG$(14,1)
```

och motsvarande telefonnummer med

```
REG$(14,2)
```

Nu ska vi tänka till ordentligt en stund. Vi sa tidigare att vi skulle sortera registret varje gång något nytt matades in. Eftersom vi då sorterar ganska ofta får sorteringen inte ta för lång tid. Om nu varje post i registret har två delar, betyder det att varje gång vi vill låta två poster byta plats, måste vi göra sex tilldelningar. Med strängar. Det tar tid. Visserligen kan det väl gå an när varje post består av två delar, men vi vill ju kunna bygga ut senare. Hur blir det om varje post har 10 delar? 30 tilldelningar vid varje byte? Hu jeda mej, som lilla Ida sa i Emil i Lönneberga. Att något intelligentare måste till är uppenbart.

Antag att vi har ett endimensionellt numeriskt fält, med lika många element som det kan finnas poster i registret. 100 i det här fallet alltså. Då kan vi låta det första elementet i det numeriska fältet ange vilket element i REG$ som innehåller det första namnet i bokstavsordning. Om det numeriska fältet heter PEK, får man alltså tag i det första namnet med

```
REG$(PEK(1),1)
```

Om vi har fyra namn i registret, och dessa namn är Peter, Katarina, Bengt och Lisbeth, kan det se ut som i tabellen nedan.

Index	PEK	Innehåll REG$(Index,1)
1	3	Peter
2	2	Katarina
3	4	Bengt
4	1	Lisbeth

Om man bara tar namnen i REG$ i nummerordning, får man inte dem i bokstavsordning. Om man däremot tar det första namnet på den plats som PEK(1) anger, och sedan fortsätter med de andra namnen i samma stil, får man dem i bokstavsordning!

Genom att utnyttja den här tekniken med pekare blir sorteringen mycket effektivare. Varje gång två poster ska byta plats behöver vi bara låta pekarna byta med varandra. För att det här ska fungera måste vi börja med att låta PEK(1) vara 1, PEK(2) 2 osv. I det färdiga programmet, vilket finns efter den här artikeln, göres detta på raderna 150-170.

Nackdelar då? Det finns en. Det tar mera minne. Särskilt i 99:ans BASIC, eftersom den inte har några heltal. Här tar varje pekare upp åtta bytes, vilket är nästan lika mycket som ett telefonnummer. Om det hade funnits heltal, vilket det gör i t.ex. Forth och Pascal, hade vi bara behövt slösa bort två bytes per pekare. Då hade det varit ännu mer fördelaktigt än vad det är nu. Å andra sidan har vi inte direkt något val. Om sorteringen ska bli klar inom vettiga tidsramar måste vi tillgripa något i den här stilen.

Vår DIM sats har nu vuxit till

```
DIM REG$(100,2),PEK(100)
```

Som det ser ut nu ska del ett i posten vara ett namn och del två ett telefonnummer. För att göra utskrifterna någorlunda lätta att begripa behöver vi någon rubrik på varje del. Eftersom vi vill ha det hela så flexibelt som möjligt, är det lämpligt att lägga rubrikerna i en variabel. Då kan vi lätt anpassa rubrikerna till nya innehåll i registret. Nu har vi ju två delar i varje post, varför vår DIM sats slutgiltiga utseende blir detta:

```
DIM REG$(100,2),PEK(100),RUBRIK$(2)
```

När vi nu är igång och bestämmer, kan vi lika gärna bestämma oss för namnen på två andra variabler som det är lätt att se att vi behöver. En skall tala om hur många namn som är inmatade. Den kallar vi ANTAL. En annan skall tala om hur många namn man maximalt kan mata in. Den får logiskt nog heta MAXANTAL.

Nu har vi kommit så långt att vi kan börja fundera över vilka underprogram som vi kommer att behöva.

Programmets grundfunktion skall vara att visa en huvudmeny med de alternativ som vi kom fram till tidigare, låta användaren välja ett alternativ, utföra detta och sedan visa menyn igen. Det kan vara lämpligt med en SUB som visar menyn och frågar efter vilket alternativ man väljer. Detta val måste det anropande programmet få reda på. Menyrutinen behöver alltså en numerisk parameter. Proceduren bör också kontrollera att det är ett giltigt alternativ som valdes. Då kan man med fördel använda ACCEPT med dess VALIDATE funktion. SUB programmet MENY finns på raderna 1540-1660 i programmet.

När ett alternativ har valts ska det utföras. Då kan vi dela upp programmet så att varje alternativ får en egen SUB. På det sättet eliminerar vi risken för att de ska störa varandra. Det enda alternativet som inte får någon egen SUB är Avslutning. Den funktionen är nämligen speciell så tillvida att den aldrig kommer att återvända till menyrutinen. När vi har sållat bort slutvarianten kan vi välja ut en av de andra funktionerna med en ON GOSUB konstruktion. I programmet göres detta på raderna 220-300.

Hur skriver man då en procedur för ett alternativ? Vi kan titta på Inmatning.

Inmatningen, som kan få heta DATAENTRY, ska fungera så att programmet skriver ut rubriken för en del av posten i taget, och läser in den information som ska finnas där. När alla delar av en post är inmatade, ska posten sättas in i registret. Detta kan göras på olika sätt. Låt oss deklarera en särskild SUB, kallad INSERT, för att sätta in en post i registret. Sedan ska rutinen fråga om fler poster ska matas in. När alla posterna har matats in, uppstår frågan om registret behöver sorteras eller ej. Om INSERT sätter in posterna i ordning i registret är en sortering nu överflödig. Om däremot INSERT bara sätter in posterna på första lediga plats, kan registret hamna i oordning. Då måste vi sortera registret innan DATAENTRY återvänder till huvudprogrammet. I det färdiga programmet sätter INSERT in en ny post längst bak i registret, oberoende av den nya postens och registrets tidigare innehåll. Därför måste vi här sortera registret innan vi återvänder till huvudprogrammet. Dessutom måste DATAENTRY kontrollera att man inte matar in fler poster än vad registret är dimensionerat för.

Sammantaget ger detta att DATAENTRY måste ha följande parametrar:

REG$(,) eftersom man naturligtvis måste kunna komma åt registret när man ska sätta in poster i det.

PEK() behövs för sorteringens skull.

RUBRIK$() måste till så att DATAENTRY kan fråga efter rätt saker vid inmatningen.

ANTAL används bland annat för att tala om för sorteringen hur många poster som har matats in.

MAXANTAL behövs för att kontrollera att ANTAL inte kommer utanför det tillåtna antalet poster i registret.

Ett liknande resonemang använder man till varje underprogram, för att avgöra vilka parametrar som behövs.

DATAENTRY utnyttjar även ett SUB program som heter WAIT. WAIT finns på raderna 1500-1530, och gör inget annat än väntar tills man trycker på en tangent. Själva inmatningen sker i en FOR - NEXT loop. Det gör det lättare att lägga till fler delar i varje post om man vill det. Efter varje inmatad post anropas INSERT för att sätta in posten i registret. QSORT behöver vi dock inte kalla in förrän samtliga nya poster har satts in i registret. Innan dess kan vi inte skriva ut registret, och därför störs vi inte av den oordning som denna enkla version av INSERT skapar i registret. INSERT (1140-1190) sätter helt enkelt in en ny post efter de andra i registret.

Sorteringen sköter ett underprogram vid namn QSORT om. QSORT, vilket finns på raderna 1200-1460, är samma program som fanns i Nittinian 83-4/5, modifierat för att kunna sortera med vår variant på pekare. Alla jämförelser sker med namnet i REG$(,) (formell parameter A$(,)), men alla byte av värden göres i PEK(). För att minska på skrivandet i QSORT finns det en särskild SUB, kallad SWAP, som skiftar två numeriska värden.

QSORT sorterar alltså posterna med avseende på namnet. Men för att göra det lättare att ändra bestäms vilken del som sortering ska ske på i rad 1210. Där tilldelas variabeln DEL värdet 1 (ett). I hela QSORT refereras till DEL för att få reda på vilken del i en post som är den betydelsefulla för sorteringsordningen. Om man vill kan man låta DEL vara en parameter till QSORT. Då kan man vid anropet av QSORT bestämma om sortering skall ske med avseende på namn, telefonnummer, ålder eller vad man nu har i sitt register.

Just den variant av QSORT som finns i miniregistret är hämtad från A-REGISTER 2.7. Det är ett registerprogram med betydligt fler möjligheter än detta. Det går bra att höra av sig om någon är intresserad.

Sökningen i programmet administreras av MAINSEARCH. Själva sökningen sker i undeprogrammet SEARCH. Det letar helt enkelt igenom hela registret från början till slut. I och med att funktionen POS används, behöver man inte komma ihåg exakt hur ett visst namn stavas. Det räcker om man vet hälften. Å andra sidan betyder det också att om man frågar efter ANDERS får man med alla ANDERSSON osv också.

SEARCH letar igenom registret med hjälp av PEK(). Om man nu ändå ska leta igenom hela registret, är det egentligen inte nödvändigt att utnyttja PEK(). Men genom att använda PEK() får man utskriften av påträffade namn i bokstavsordning, och det ser snyggare ut. Det kan vara lämpligt att lägga till en möjlighet att stanna upp utskriften mellan olika poster. Om man får många överensstämmelser kommer de kanske utanför skärmen innan man hinner skriva upp dem. Om det inte finns någon med det efterfrågade namnet skrivs bara "Hela registret genomsökt" ut.

PRINTREG, på raderna 1030-1130, skriver ut hela registret på skrivare. Utskriften sker i bokstavsordning, eftersom adressering av REG$(,) sker via PEK(). Även här sker utskriften av varje post i en FOR - NEXT loop, för att underlätta tillägg av fler delar. Filnamnet finns på rad 1040. Kontrollera där om du har din skrivare likadant ansluten.

Om du inte har någon skrivare kan du ta bort filnumret vid alla PRINT satser och skriva på skärmen i stället. Men, som vi ska se senare, genom att mata in namnen på ett vettigt sätt kan man då uppnå samma effekt med sökningen.

STOREREG (650-750) och READREG (760-870) har hand om lagring respektive läsning av ett register på exempelvis diskett. Lagringssättet är valt så att det ska fungera även med den enklare kassettbandspelaren. Först i filen lagras antalet poster som finns inmatade. Sedan kan man lagra rubrikerna för varje del, om man vill ha olika rubriker i olika register. Det gör jag inte här, utan rubrikerna finns i en DATA sats på rad 310. Efter uppgiften om antalet poster lagras posterna, en i varje record på filen. Här behövs den största ändringen om man ska ha fler delar än två i varje post. Hur man då vill göra beror på hur lång tid lagringen får ta. Särskilt om man lagrar på kassett är det viktigt att man inte skriver fler records än absolut nödvändigt. Observera att posterna lagras i bokstavsordning. Genom att göra det slipper man att lagra PEK() på filen. Den hade ändå bara ockuperat utrymme som kan vara till något bättre.

Detta innebär att inläsningen måste låta PEK(1) bli 1, PEK(2) 2 osv igen. Dessutom kan det ju tänkas att man har ett register med 75 poster i minnet när man börjar läsa in ett annat med bara 40 poster. För att man då inte ska blanda ihop registren med varandra raderas det register som finns i minnet innan inläsning av ett nytt sker. Underprogrammet CLEARREG (880-960) sköter om detta. Endast utnyttjade delar av REG$(,) raderas, för att inte spilla tid i onödan. Vid inläsningen bör man också kontrollera att filen inte innehåller fler poster än vad som ryms i minnet. Rad 830 ser till att så många poster som möjligt läses in.

Dessutom bör man ta hand om eventuella fel vid inläsningen. Det är ju inte meningen att programmet ska krascha bara för att man råkade glömma att vrida om låsspaken på diskenheten eller knuffade ner volymkontrollen på bandspelaren. Med ON ERROR kan man ta hand om fel av det slaget.

Därmed har vi gått igenom funktionen hos hela programmet. Tilläggas bör kanske att det är lämpligt att mata in namnen som PERSSON,ANDERS. Genom att göra så får man dem sorterade i bokstavsordning med avseende på i första hand efternamnet, i andra hand förnamnet. Dessutom kan man då, om man vill ha tag i någon med förnamnet ANDERS, söka efter ",ANDERS". Då slipper man alla ANDERSSON etc.
Som en extra finess kan man utnyttja sökningen till att skriva ut hela registret på skärmen. Eftersom det med inmatningssättet ovan alltid finns ett komma i varje namn, behöver man bara leta efter ett komma.

Än en gång: Det här programmet är knappast praktiskt användbart i sin nuvarande form. En adressbok fungerar bättre. Men genom att lägga till en del funktioner, kan man få ett ganska vettigt registerprogram.

Det som står på önskelistan är väl närmast det här (jag läser i bruksanvisningen till A-REGISTER 2):

- Möjlighet att ändra och radera inmatade poster.

- Möjlighet att specificera flera olika villkor vid sökningen.

- Möjlighet att bestämma utskriftsformatet så att man t.ex. kan skriva ut adressetiketter, om man vill lagra adresser också i registret.

- Möjlighet att koppla ihop sökningen och utskriften så att man kan få utskrifter av de delar av registret som uppfyller vissa villkor.

- Snyggare utseende på skärmutskrifter. Att menyn kommer skrollande underifrån är inte speciellt stiligt. Dessutom är det lätt att åtgärda med DISPLAY AT. Men det får ni göra själva.

- Felhantering. Vad händer om man anger ett felaktigt filnamn, om minnet tar slut, om det blir fel på kassettbandet, om... Mycket kan gå galet. Dessutom bör man inte låta den som använder programmet göra något som kan radera registret utan att först fråga en gång till om det verkligen är meningen att informationen ska försvinna.

Här är programmet i alla fall. Jag har härjat i utskriften med ordbehandlaren för att det ska vara lättare att se var underprogrammen börjar och slutar.

```
100 !Mycket enkelt telefonregister
110 PAT$="00280038447C44440028007C4444447C00100028447
    C4444"
120 CALL CHAR(91,PAT$,123,PAT$)!Svenska tecken
130 MAXANTAL=100 :: OPTION BASE 1
140 DIM REG$(100,2),PEK(100),RUBRIK$(2)
150 FOR DEL=1 TO MAXANTAL
160 PEK(DEL)=DEL
170 NEXT DEL
180 RESTORE 310
190 FOR DEL=1 TO 2
200 READ RUBRIK$(DEL)
210 NEXT DEL

220 CALL MENY(NR)
230 IF NR=6 THEN CALL CLEAR :: STOP
240 ON NR GOSUB 260,270,280,290,300 :: GOTO 220
250 !De olika alternativen
```

```
260 CALL DATAENTRY(REG$(,),PEK(),RUBRIK$(),ANTAL,MAXA
    NTAL):: RETURN
270 CALL MAINSEARCH(REG$(,),PEK(),RUBRIK$(),ANTAL)::
    RETURN
280 CALL PRINTREG(REG$(,),PEK(),RUBRIK$(),ANTAL):: RE
    TURN
290 CALL STOREREG(REG$(,),PEK(),RUBRIK$(),ANTAL):: RE
    TURN
300 CALL READREG(REG$(,),PEK(),RUBRIK$(),ANTAL,MAXANT
    AL):: RETURN

310 DATA "Namn ","Tele "

320 SUB MAINSEARCH(REG$(,),PEK(),RUBRIK$(),ANTAL)
330 CALL CLEAR :: START=1
340 PRINT "Vilket namn sökes?"
350 LINPUT NAMN$
360 PRINT
370 CALL SEARCH(REG$(,),PEK(),NAMN$,START,ANTAL,PLATS
    )
380 IF PLATS=0 THEN 450
390 FOR DEL=1 TO 2
400 PRINT RUBRIK$(DEL);REG$(PEK(PLATS),DEL)
410 NEXT DEL
420 PRINT
430 START=PLATS+1
440 GOTO 370
450 PRINT "Hela registret genomsökt"
460 CALL WAIT
470 SUBEND

480 SUB DATAENTRY(REG$(,),PEK(),RUBRIK$(),ANTAL,MAXAN
    TAL)
490 DIM TEMP$(2)
500 CALL CLEAR
510 IF ANTAL>=MAXANTAL THEN PRINT "Det får inte plats
     fler!" :: CALL WAIT :: SUBEXIT
520 !Inmatning

530 FOR DEL=1 TO 2
540 PRINT RUBRIK$(DEL)
550 LINPUT TEMP$(DEL)
560 PRINT
570 NEXT DEL
580 CALL INSERT(REG$(,),PEK(),TEMP$(),ANTAL)
590 PRINT "Fler (J/N)?"
600 INPUT SVAR$
610 IF SVAR$="J" OR SVAR$="j" THEN PRINT :: GOTO 510
620 PRINT :"Sorterar..."
630 CALL QSORT(REG$(,),PEK(),ANTAL)
640 SUBEND

650 SUB STOREREG(REG$(,),PEK(),RUBRIK$(),ANTAL)
660 CALL CLEAR
670 LINPUT "Filnamn? ":FILE$
680 OPEN #2:FILE$,OUTPUT,FIXED 64,INTERNAL
690 PRINT #2:ANTAL
700 !Här kan man lagra rubrikerna, om man vill att de
     ska kunna vara olika för olika register
710 FOR POST=1 TO ANTAL
720 PRINT #2:REG$(PEK(POST),1),REG$(PEK(POST),2)
730 NEXT POST
740 CLOSE #2
750 SUBEND

760 SUB READREG(REG$(,),PEK(),RUBRIK$(),ANTAL,MAXANTA
    L)
770 CALL CLEAR
780 LINPUT "Filnamn? ":FILE$
790 CALL CLEARREG(REG$(,),PEK(),ANTAL,MAXANTAL)
800 OPEN #2:FILE$,INPUT ,FIXED 64,INTERNAL
810 INPUT #2:ANTAL
820 !Här kan man läsa rubrikerna, om man har lagrat d
    em också
830 FOR POST=1 TO MIN(ANTAL,MAXANTAL)!
840 INPUT #2:REG$(POST,1),REG$(POST,2)
850 NEXT POST
860 CLOSE #2
870 SUBEND

880 SUB CLEARREG(REG$(,),PEK(),ANTAL,MAXANTAL)
890 FOR POST=1 TO ANTAL
900 FOR DEL=1 TO 2
910 REG$(POST,DEL)=""
920 NEXT DEL
930 PEK(POST)=POST
940 NEXT POST
950 ANTAL=0
960 SUBEND

970 SUB SEARCH(REG$(,),PEK(),NAMN$,START,ANTAL,PLATS)
980 PLATS=0
990 FOR POST=START TO ANTAL
1000 IF POS(REG$(PEK(POST),1),NAMN$,1)>0 THEN PLATS=PO
     ST :: SUBEXIT
1010 NEXT POST
1020 SUBEND


1030 SUB PRINTREG(REG$(,),PEK(),RUBRIK$(),ANTAL)
1040 OPEN #1:"RS232/2.BA=9600.DA=8"
1050 PRINT #1:TAB(5);"TELEFONREGISTER": :
1060 FOR POST=1 TO ANTAL
1070 FOR DEL=1 TO 2
1080 PRINT #1:TAB(5);RUBRIK$(DEL);REG$(PEK(POST),DEL)
1090 NEXT DEL
1100 PRINT #1
1110 NEXT POST
1120 CLOSE #1
1130 SUBEND

1140 SUB INSERT(REG$(,),PEK(),TEMP$(),ANTAL)
1150 ANTAL=ANTAL+1
1160 FOR DEL=1 TO 2
1170 REG$(PEK(ANTAL),DEL)=TEMP$(DEL)
1180 NEXT DEL
1190 SUBEND

1200 SUB QSORT(A$(,),PEK(),ANTAL)
1210 LVL=0 :: DEL=1
1220 L=1
1230 R=ANTAL
1240 IF R-L<9 THEN 1400
1250 CALL SWAP(PEK((L+R)/2),PEK(L+1))
1260 IF A$(PEK(L+1),DEL)>A$(PEK(R),DEL)THEN CALL SWAP(
     PEK(L+1),PEK(R))
1270 IF A$(PEK(L),DEL)>A$(PEK(R),DEL)THEN CALL SWAP(PE
     K(L),PEK(R))
1280 IF A$(PEK(L+1),DEL)>A$(PEK(L),DEL)THEN CALL SWAP(
     PEK(L+1),PEK(L))
1290 I=L+1 :: J=R :: KEY$=A$(PEK(L),DEL)
1300 I=I+1 :: IF A$(PEK(I),DEL)<KEY$ THEN 1300
1310 J=J-1 :: IF A$(PEK(J),DEL)>KEY$ THEN 1310
1320 IF J<I THEN 1350
1330 CALL SWAP(PEK(I),PEK(J))
1340 GOTO 1300
1350 CALL SWAP(PEK(L),PEK(J))
1360 IF MAX(J-L,R-I+1)<=9 THEN IF LVL=0 THEN 1400 ELSE
      L=LST(LVL):: R=RST(LVL):: LVL=LVL-1 :: GOTO 1250
1370 IF J-L>=R-I+1 THEN LSFL=L :: LSFR=J-1 :: SSFL=I :
     : SSFR=R ELSE LSFL=I :: LSFR=R :: SSFL=L :: SSFR=
     J-1
1380 IF MIN(J-L,R-I+1)<=9 THEN L=LSFL :: R=LSFR ELSE L
     VL=LVL+1 :: LST(LVL)=LSFL :: RST(LVL)=LSFR :: L=S
     SFL :: R=SSFR
1390 GOTO 1250
1400 FOR I=ANTAL-1 TO 1 STEP -1
1410 IF A$(PEK(I),DEL)<A$(PEK(I+1),DEL)THEN 1450
1420 KEY=PEK(I):: J=I+1
1430 PEK(J-1)=PEK(J):: J=J+1 :: IF J<=ANTAL THEN IF A$
     (PEK(J),DEL)<A$(KEY,DEL)THEN 1430
1440 PEK(J-1)=KEY
1450 NEXT I
1460 SUBEND

1470 SUB SWAP(A,B)
1480 SW=A :: A=B :: B=SW
1490 SUBEND

1500 SUB WAIT
1510 CALL KEY(5,K,S)
1520 IF S=0 THEN 1510
1530 SUBEND

1540 SUB MENY(NR)
1550 CALL CLEAR
1560 PRINT "MINIREGISTER"
1570 PRINT :"1 Inmatning"
1580 PRINT :"2 Sökning"
1590 PRINT :"3 Utskrift"
1600 PRINT :"4 Lagra"
1610 PRINT :"5 Hämta"
1620 PRINT :"6 Sluta"
1630 PRINT
1640 DISPLAY AT(24,1):"Ditt val?"
1650 ACCEPT AT(24,11)VALIDATE("123456")SIZE(1)BEEP:NR
1660 SUBEND
```

■

DIS-ASSEMBLER-SKOLAN DEL III

Av Tony Rogvall.

Det är meningen att det här programmmet skall ingå som en del av assembler-skolan. Kanske kan programmet vid första anblicken vara lite svårt att förstå.
Men låt Dig inte skrämmas. Allt är förklarat. Möjligen kan Du behöva en del tid. Syftet är att ge Dig litet hjälp med rutiner och med knep, knåp och teknik.

Disassembleraren i sig själv är ett utmärkt hjälpmedel när det gäller att kopiera rutiner och alla slags program. Programmet är dessutom fristående. Med det menar jag att det inte tar hjälp utav de inbyggda rutinerna för t ex VDP eller av DSR-rutinerna. Dessa funktioner finns som delar utav mitt program.

Disassembleraren kan:

=> Disassemblera DSRROM, VRAM, GROM - förutom vanligt CPU-minne förstås.
=> Skriva ut resultatet av disassembleringar på printer eller diskdrive (det senare för att kunna tä in det disassemblerade i editorn och redigera det där).
=> Addera en offset till adressen om programmet ligger felplacerat i minnet. Användbart t ex om man har kod i VRAM.

Utskriften sker i följande format:

=> adress, innehåll, ascii, mnemonic

OBS!--> Om Du är för lat för att knappa in hela det här programmet kan det vara skönt att veta att jag kommer att göra det tillgängligt i föreningens programbank.

```
*****************************************
*                                       *
*            DISASSEMBLER               *
*                                       *
*****************************************
      DEF  RUN

*---------------------------------------*
*              EQUATES                  *
*---------------------------------------*
KEYBRD EQU  >8375        * ASCII ADRESS I KSCAN.
KEYNUM EQU  >8374        * DEL AV TANGENTBORD. HELA=0.
STATUS EQU  >837C        * GPL STATUS BYTE.
SCAN   EQU  >000E        * TANGENTBORDS SCANNING RUTIN ADRESS.
GRMRD  EQU  >9800        * GROM READ DATA.
GRMRA  EQU  >9802        * GROM READ ADDRESS.
GRMWA  EQU  >9C02        * GROM WRITE ADDRESS.
VDPWA  EQU  >8C02        * VDP WRITE ADDRESS.
VDPWD  EQU  >8C00        * VDP WRITE DATA.
VDPRD  EQU  >8800        * VDP READ DATA.
PAB    EQU  >F80         * PAB ADRESS I VRAM.
PABBUF EQU  >1000        * PAB-BUFFER ADRESS.
DEVLEN EQU  >8354        * ENHETSNAMN LÆNGD BYTE FØR DSRLNK. (MSBY)
DEVLB  EQU  >8355        * --   II  --     --   II  --    . (LSBY)
PNTR   EQU  >8356        * PEKARE TILL NAMNET I PAB.
LINK   EQU  >83D0        * LINK ADRESS I DSR LAGRAS HÆR.
NLINK  EQU  >83D2        * NÆSTA LINK   --    II   --  .
GPLWS  EQU  >83E0        * GPL WORKSPACE PEKARE.
*---------------------------------------*
*             PAB DATA                  *
*            OCH BUFFRAR                *
*---------------------------------------*
PDATA  DATA >0012,PABBUF * DIS/VAR OCH PAB-BUFFER PEKARE
       BYTE >50          * FIL LÆNGDEN ER 80
RECL   BYTE >00          * DEN EGENTLIGA BUFFER LÆNGDEN.
       DATA >0000        * ANVÆNDS EJ
       BYTE >00          * -- II --
NAMEL  BYTE >00          * NAMN LÆNGD I DSR-NAMN
*
BUFFER BSS  40           * PRINTER BUFFER OCH SCROLL
KODST  BSS  6            * KOD BUFFER FØR DISASSEMBELERING.
MYREG  BSS  >20          * PROGRAMMETS ARBETSREGISTER.
WS1    BSS  >20          * WORKSPACE1
WS2    BSS  >20          * WORKSPACE2
*---------------------------------------*
*              TEXTER                   *
*---------------------------------------*
TEXT0  TEXT 'DISASSEMBLER UTILITY'
TEXT1  TEXT 'PRINTER DESCRIPTION:'
TEXT2  TEXT 'CRUBASE:'
TEXT3  TEXT 'EQUATE:'
TEXT4  TEXT '(G)ROM V)DP C)PU:C'
TEXT5  TEXT 'START ADDRESS:'
TEXT6  TEXT 'STOP  ADDRESS:'
TEXT7  TEXT 'PRESS CLEAR TO RESTART'
TEXT8  TEXT '** I/O ERROR   **'
TEXT9  TEXT 'PRESS ENTER TO CONTINUE'
UND    TEXT 'UNDEFINED'
*---------------------------------------*
*             VARIABLER                 *
*---------------------------------------*
PRON   DATA >0000        * PRINTER ON FLAGGA
DSRON  DATA >0000        * DISASSEMBLERING AV DSRROM FLAGGA
EQUON  DATA >0000        * EQUATE OFFSET
GVCSAV DATA >0000        * INNEHÅLLER ASCI G,V ELLER C I MSBY
ADDR   DATA >0000        * ADRESS PEKARE.
SAVGRM DATA >0000        * SPARA GROM ADRESS FØR ATERHOPP.
PARA   DATA >0000        * PARAMETER VERDE LAGRAS HÆR.
START  DATA >0000        * START ADRESS
STOP   DATA >0000        * STOP ADRESS
LEV1   DATA >0000        * RETURN ADRESS NIVA 1
LEV2   DATA >0000        *    - II -  - II -  2
LEV3   DATA >0000        *    - II -  - II -  3
*---------------------------------------*
*             KONSTANTER                *
*---------------------------------------*
JUMP   DATA F1,F2,F3,F4,F5,F6,F7,F8,F3 * HOPP TABELL FØR DE OLIKA FORMATEN.
BITS   DATA 12,8,10,10,8,6.5,5,10      * BIT JUSTERING FØR ATT NÅ VIKTIGA BITAR.
*---------------------------------------*
SET    DATA >2000        * TEST OM NY TANGENT I INKEY.
CURSOR BYTE >1E          * MARKØR DATA FØR INPUT.
SD     BYTE '*'          * STJÆRNA FØR INDIREKT ADRESSERING.
RD     BYTE 'R'          * REGISTER.
PD     BYTE '+'          * AUTO INCREMENT.
AD     BYTE '@'          * MEMORY ADDRESSING.
VD     BYTE '('          * PARENTES TILL INDEX.
HD     BYTE ')'          * --   II  --  II --
D1     BYTE '1'          * EN ETTA VID REG NUMMER >9.
KD     BYTE ','          * KOMMA MELLAN TS OCH TD FÆLT.
PUNKT  BYTE '.'          * SØK DATA I DSRNAMN.
SPACE  BYTE ' '          * ANVÆND PÅ FLERA STÆLLEN.
PRINTD BYTE >03          * DATA FØR ATT ØPPNA FIL
CLOSED BYTE >01          * DATA FØR ATT STÆNGA FIL
CR     BYTE >0D          * CARRIAGE RETURN, FØR PRINTER UTSKRIFT.
ALL    BYTE >AA          * VALID DSR ROM.
*---------------------------------------*
*           BLWP VEKTORER               *
*---------------------------------------*
*               VEKTOR FØR:
INPUT  DATA WS1
       DATA INPUT1       * INPUT FRAN SKÆRM.
SCROL  DATA WS1
       DATA SCR1         * SKÆRM SCROLLNING.
VSBR   DATA WS2
       DATA VSBR1        * VDP SINGLE BYTE READ.
VMBR   DATA WS2
       DATA VMBR1        * VDP MULTI BYTE READ.
VSBW   DATA WS2
       DATA VSBW1        * VDP SINGLE BYTE WRITE.
VMBW   DATA WS2
       DATA VMBW1        * VDP MULTI BYTE WRITE.
KSCAN  DATA WS2
       DATA KSCAN1       * TANGENT BORDS AVKÆNNING.
DSRLNK DATA WS1
       DATA DSRLBL       * DSR LINK FØR FIL HANTERING.
*---------------------------------------*
*  TEXT-DATA VID UTSKRIFT PA SKÆRM      *
*---------------------------------------*
TXT0   DATA 10,80,200,280,360,520,560,929  * DATA FØR REG R0
       DATA 892,929
TXT1   DATA TEXT0,TEXT1,TEXT2,TEXT3,TEXT4  * --   II  --  R1
       DATA TEXT5,TEXT6,TEXT7,TEXT8,TEXT9
TXT2   DATA 20,20,8,7,17,14.14,22,17,23    * --   II  --  R2
*---------------------------------------*
*    VALIDATE STRENGAR FØR INPUT        *
*---------------------------------------*
HV     BYTE 16                 * 16 DATA ATT TESTA.
       TEXT '0123456789ABCDEF' * STRENG ATT TESTA I INPUT.
GVC    BYTE 3                  * 3 DATA ATT TESTA.
       TEXT 'GVC'              * STRENG.
*---------------------------------------*
*            OP-CODE-DATA               *
*---------------------------------------*
OP     DATA >4000,'SZ','C '
       DATA >5000,'SZ','CB'
       DATA >6000,'S ','  '
       DATA >7000,'SB','  '
       DATA >8000,'C ','  '
       DATA >9000,'CB','  '
       DATA >A000,'A ','  '
       DATA >B000,'AB','  '
       DATA >C000,'MO','V '
       DATA >D000,'MO','VB'
       DATA >E000,'SO','C '
       DATA >F000,'SO','CB'
FTEST  DATA >FFFF              * >FFFF MARKERAR NYTT FORMAT.
       DATA >1000,'JM','P '
       DATA >1100,'JL','T '
       DATA >1200,'JL','E '
       DATA >1300,'JE','Q '
       DATA >1400,'JH','E '
       DATA >1500,'JG','T '
       DATA >1600,'JN','E '
       DATA >1700,'JN','C '
       DATA >1800,'JO','C '
       DATA >1900,'JN','O '
       DATA >1A00,'JL','  '
       DATA >1B00,'JH','  '
       DATA >1C00,'JO','P '
       DATA >1D00,'SB','O '
       DATA >1E00,'SB','Z '
       DATA >1F00,'TB','  '
       DATA >FFFF
       DATA >2000,'CO','C '
       DATA >2400,'CI','C '
       DATA >2800,'XO','R '
       DATA >FFFF
       DATA >3000,'LD','CR'
       DATA >3400,'ST','CR'
       DATA >FFFF
       DATA >0800,'SR','A '
       DATA >0900,'SR','L '
       DATA >0A00,'SL','A '
       DATA >0B00,'SR','C '
       DATA >FFFF
       DATA >0400,'BL','WP'
       DATA >0440,'B ','  '
       DATA >0480,'X ','  '
       DATA >04C0,'CL','R '
       DATA >0500,'NE','G '
       DATA >0540,'IN','V '
       DATA >0580,'IN','C '
       DATA >05C0,'IN','CT'
       DATA >0600,'DE','C '
       DATA >0640,'DE','CT'
       DATA >0680,'BL','  '
       DATA >06C0,'SW','PB'
       DATA >0700,'SE','TO'
       DATA >0740,'AB','S '
       DATA >FFFF
       DATA >0340,'ID','LE'
       DATA >0360,'RS','ET'
       DATA >0380,'RT','WP'
       DATA >03A0,'CK','ON'
       DATA >03C0,'CK','OF'
       DATA >03E0,'LR','EX'
       DATA >FFFF
       DATA >0200,'LI','  '
       DATA >0220,'AI','  '
       DATA >0240,'AN','DI'
       DATA >0260,'OR','I '
       DATA >0280,'CI','  '
       DATA >02A0,'ST','WP'
       DATA >02C0,'ST','ST'
       DATA >02E0,'LW','PI'
       DATA >0300,'LI','MI'
       DATA >FFFF
       DATA >2C00,'XO','R '
       DATA >3800,'MP','Y '
       DATA >3C00,'DI','V '
       DATA >FFFF
```

```
*-------------------------*
*       DSR LINK          *
*-------------------------*
DSRLBL MOV  *R14+,R5     * SPARA DATA
       SZCB @SET,R15     * NOLLSTÄLL EQUAL
       MOV  @PNTR,R0     * KOPIERA PEKARE TILL NAMN LÄNGD
       MOV  R0,R9        * KOPIERA TILL R9
       AI   R9,-8        * R9 PEKAR PÅ FLGSTA
       BLWP @VSBR        * LÄS IN ANTAL TECKEN I NAMNET
       MOVB R1,R3        * KOPIERA TILL R3
       SRL  R3,8         * BYTE JUSTERA
       SETO R4           * SÄTT R4 TILL -1
       LI   R2,>208C     * LADDA R2 MED BUFFER ADRESS
*-------------------------*
HOPP2  INC  R0           * ÖKA TILL NÄSTA TECKEN
       INC  R4           * ÖKA NAMN TECKEN RÄKNARE
       C    R4,R3        * ÄR NAMNET KOPIERAT?
       JEQ  HOPP1        * JA FORTSÄTT!
       BLWP @VSBR        * LÄS IN TECKEN FRÅN PAB
       MOVB R1,*R2+      * LÄGG TECKEN I BUFFER
       CB   R1,@PUNKT    * TESTA OM TECKNET ÄR EN PUNKT
       JNE  HOPP2        * OM INTE EN PUNKT, KOPIERA NÄSTA!
*-------------------------*
HOPP1  MOV  R4,R4        * ÄR ANTALET TECKEN I NAMNET 0 ?
       JEQ  HOPP3        * JA HOPPA TILL ERROR/UT
       CI   R4,7         * ÄR DSR NAMNET STÖRRE ÄN SJU
       JGT  HOPP3        * JA HOPPA TILL ERROR/UT
       CLR  @LINK        * RENSA ROM SEARCH POINTER
       MOV  R4,@DEVLEN   * KOPIERA DSRNAMN LÄNGDEN
       MOV  R4,@>2036    * --     II   --
       INC  R4           * ÖKA TILL EFTER PUNKT
       A    R4,@PNTR     * ADDERA R4 SÅ ATT PEKAR PÅ PARAMETER
       MOV  @PNTR,@>2038 * KOPIERA PARAMETER PEKARE
*-------------------------*
* LETA EFTER DSR
*-------------------------*
       LWPI GPLWS        * LADDA GPLWS
       CLR  R1           * NOLLSTÄLL ERROR RÄKNARE.
       LI   R12,>0F00    * LADDA BAS ADRESS CRU
*-------------------------*
* LOOP FÖR ATT SÖKA DSR  *
*-------------------------*
HOPP6  MOV  R12,R12
       JEQ  HOPP4
       SBZ  0
HOPP4  AI   R12,>0100    * ADDERA TILL NÄSTA BAS
       CLR  @LINK        * RENSA LÄNK ADRESS
       CI   R12,>2000    * ÄR HELA REGISTRET GENOMSÖKT?
       JEQ  HOPP5        * JA HOPPA ERROR/UT
       MOV  R12,@LINK    * NEJ, KOPIERA BAS TILL LÄNK
       SBO  0            * SKIFTA IN DSR
       LI   R2,>4000     * LADDA R2 MED BASADRESS TILL DSRROM
       CB   *R2,@ALL     * FINNS DET ETT DSR?
       JNE  HOPP6        * NEJ, LETA VIDARE!
*-------------------------*
       A    @WS1+10,R2   * ADDERA WS1'S R5 TILL DSRLÄNK
       JMP  HOPP7        *
*-------------------------*
* SÖK IGENOM DSR         *
*-------------------------*
HOPP9  MOV  @NLINK,R2
       SBO  0            * SKIFTA IN DSR
HOPP7  MOV  *R2,R2       * HÄMTA LÄNK TILL NÄSTA NAMN
       JEQ  HOPP6        * OM ADRESS ÄR 0, SÖK VIDARE!
       MOV  R2,@NLINK    * SPARA ADRESS I NLINK
       INCT R2           * ÖKA TILL DSRNAME LENGHT
       MOV  *R2+,R9      * SPAR RUTIN ADRESS
       MOVB @DEVLB,R5    * DSR NAMN LÄNGD -> R5
       JEQ  HOPP8        * OM LIKA MED 0 HOPPA VIDARE
       CB   R5,*R2+      * ÄR NAMNEN AV SAMMA LÄNGD?
       JNE  HOPP9        * NEJ, TESTA NÄSTA LÄNK
*-------------------------*
* TESTA DSR NAMN         *
*-------------------------*
       SRL  R5,8         * BYTE JUSTERA NAMN LÄNGD
       LI   R6,>208C     * LADDA R6 MED DSRNAMN BUFFER
HOPP10 CB   *R6+,*R2+    * JÄMFÖR TECKEN MELLAN DSRNAMN OCH BUFFERNAMN
       JNE  HOPP9        * OM EJ LIKA HÄMTA NÄSTA LÄNK!
       DEC  R5           * MINSKA NAMN LÄNGD RÄKNARE
       JNE  HOPP10       * OM NAMN EJ ÄR SLUT TA NÄSTA TECKEN
*-------------------------*
* NAMNET HITTAT KALLA DSR*
*-------------------------*
HOPP8  INC  R1           * ÖKA ERROR RÄKNARE.
       MOV  R1,@>203A    * SPARA DEN
       MOV  R9,@>2034    * SPARA RUTIN ADRESS!
       MOV  R12,@>2032   * SPARA CRUBASE ADRESS!
       BL   *R9          * KALLA PÅ DSR RUTIN.
       JMP  HOPP9        * LETA EFTER DSR MED SAMMA NAMN
*-------------------------*
       SBZ  0            * SKIFTA UR DSR
       LWPI WS1          * LADDA DSRLNK WORKSPACE
       MOV  R9,R0        * LADDA R0 MED FLGSTA ADRESS
       BLWP @VSBR        * LÄS INNEHÅLL.
       SRL  R1,13        * SKIFTA TILL BETYDELSEFULLA BITAR
       JNE  HOPP11       * SKILT FRÅN NOLL?
       RTWP              * JA, GE TILLBAKA KONTROLL
*-------------------------*
HOPP5  LWPI WS1          * LADDA DSRLNK'S ARBETSREGISTER.
HOPP3  CLR  R1           * ERROR INGET SADANT DSR!
HOPP11 SWPB R1           * BYTE JUSTERA R1
       MOVB R1,*R13      * LÄGG KOPIA I R0 AV KALLANDE WP
       SOCB @SET,R15     * SÄTT EQUAL FLAGGA
       RTWP              * GE TILLBAKA KONTROLL
*-------------------------*
* TANGENTBORDS AVKÄNNING *
*-------------------------*
KSCAN1 LWPI GPLWS        * LADDA GPL'S ARBETSREGISTER.
       MOV  R11,@WS2+22  * SPARA GPL-RETURN I WS2'S R11.
       BL   @SCAN        * KALLA PÅ TANGENTBORDS RUTIN.
       LWPI WS2          * LADDA WS2'S REGISTER.
       MOV  R11,@GPLWS+22 * FLYTTA GPL-RETURN TILL GPLWS'S R11.
       RTWP              * RETURNERING.
*-------------------------*
*   VDP SINGLE BYTE READ  *
*-------------------------*
VSBR1  BL   @SETRDA      * SÄTT LÄS ADRESS.
       MOVB @VDPRD,@2(R13) * FLYTTA FRÅN VDP TILL KALLANDE R1.
       RTWP              * RETURNERING.
*-------------------------*
*   VDP MULTI BYTE READ   *
*-------------------------*
VMBR1  BL   @SETRDA      * SÄTT LÄS ADRESS.
VRDLOP MOVB @VDPRD,*R1+  * FLYTTA DATA FRÅN VDP TILL BUFFER.
       DEC  R2           * MINSKA LOOP RÄKNARE.
       JNE  VRDLOP       * FORTSÄTT LÄS OM EJ FÄRDIG.
       RTWP              * RETURNERING.
*-------------------------*
*   VDP SINGLE BYTE WRITE *
*-------------------------*
VSBW1  BL   @SETWDA      * SÄTT SKRIV ADRESS.
       MOV  @2(R13),@VDPWD * FLYTTA FRÅN KALLANDE R1 (MSBY) TILL VDP.
       RTWP              * RETURNERING.
*-------------------------*
*   VDP MULTI BYTE WRITE  *
*-------------------------*
VMBW1  BL   @SETWDA      * SÄTT SKRIV ADRESS.
VWTLOP MOVB *R1+,@VDPWD  * SKRIV FRÅN BUFFER TILL VDP.
       DEC  R2           * MINSKA LOOP RÄKNARE.
       JNE  VWTLOP       * FORTSÄTT SKRIV OM EJ FÄRDIG.
       RTWP              * RETURNERING.
*-------------------------*
SETWDA LI   R1,>4000     * BIT 1 SÄTTS FÖR ATT KUNNA SKRIVA I VDP.
       JMP  WVADD        * SKRIV ADRESS.
*-------------------------*
SETRDA CLR  R1           * NOLLSTÄLLS, ENDAST LÄSNING.
*-------------------------*
WVADD  MOV  *R13,R2       * HÄMTA VDP ADRESS (MYREG R1-> WS2 R2).
       MOVB @WS2+5,@VDPWA * SKRIV LSBY AV VDP-ADRESS.
       SOC  R1,R2        * SÄTT MODE BITAR.
       MOVB R2,@VDPWA    * SKRIV MSBY AV VDP-ADRESS.
       MOV  @2(R13),R1   * HÄMTA MYREG'S R1.
       MOV  @4(R13),R2   * --  II --     R2.
       RT                * RETURN.
*-------------------------*
* SKÄRM SCROLLNINGS RUTIN *
*-------------------------*
SCR1   LI   R0,40        * VDP START ADRESS.
       LI   R1,BUFFER    * BUFFER ADRESS
       LI   R2,40        * ANTAL TECKEN PER RAD.
SCR2   BLWP @VMBR        * LÄS IN EN RAD.
       AI   R0,-40       * GÅ TILLBAKA EN RAD.
       BLWP @VMBW        * SKRIV RADEN.
       AI   R0,80        * GÅ NED TVÅ RADER.
       CI   R0,920       * FÄRDIG MED SCROLL?
       JNE  SCR2         * NEJ.
       MOV  R1,R3        * KOPIERA BUFFER ADRESS.
       LI   R0,879       * NÄST NEDERSTA RADEN!
       LI   R1,>2000     * BLANK TECKEN.
       LI   R2,39        * ANTAL TECKEN ATT BLANKSTÄLLA - 1
       BLWP @VSBW        * SKRIV EN BLANK.
       MOVB R1,*R3+      * SKRIV EN EXTRA BLANK I BUFFER.
SCR3   MOVB R1,*R3+      * BLANKSTÄLL BUFFER.
       MOVB R1,@VDPWD    * SKRIV BLANK TILL VDP.
       DEC  R2           * MINSKA LOOP RÄKNARE.
       JNE  SCR3         * FÄRDIG?
       RTWP              * RETURNERING.
*-------------------------*
*     INPUT RUTIN         *
*-------------------------*
INPUT1 MOV  *R14+,R0      * START ADRESS.
       MOV  *R14+,R5      * LÄNGD
       DEC  R5            * JUSTERA LÄNGDEN.
       CLR  @KEYNUM       * VÄLJER HELA TANGENTBORDET.
       CLR  R4            * TECKEN RÄKNARE.
READ   CLR  R3            * FLASH RÄKNARE.
       CLR  R1            *
       BLWP @VSBR         * LÄS TECKEN UNDER MARKÖR.
       MOVB R1,R2         * SPARA TECKEN.
       MOVB @CURSOR,R1    * LADDA MARKÖR DATA.
       BLWP @VSBW         * SKRIV MARKÖR.
*
GET    BL   @KEY          * ETT NYTT TECKEN?
       JEQ  WRITE         * JA, SKRIV TECKEN.
*
       INC  R3            * ANNARS ÖKA FLASH RÄKNARE.
       CI   R3,400        * SKALL FLASH GÖRAS?
       JNE  GET           * NEJ, VÄNTA PÅ NYTT TECKEN.
       CLR  R3            * NOLLSTÄLL FLASH.
       BLWP @VSBR         * LÄS TECKEN.
       CB   R1,@CURSOR    * ÄR DET MARKÖREN?
       JEQ  FLASH1        * JA!
       MOVB @CURSOR,R1    * ANNARS, LADDA MARKÖR.
       JMP  FLASH2        * SKRIV DEN.
FLASH1 MOVB R2,R1         * HÄMTA TECKEN UNDER MARKÖREN.
FLASH2 BLWP @VSBW         * SKRIV TECKEN.
       JMP  GET           * HÄMTA NYA TECKEN.
*
WRITE  MOVB @KEYBRD,R1    * HÄMTA ASCII FÖR TECKNET.
       CI   R1,>0D00      * ENTER?
       JEQ  ENTER
       CI   R1,>0800      * VÄNSTER?
       JEQ  LEFT
       CI   R1,>0900      * HÖGER?
       JEQ  RIGHT
       CI   R1,>0F00      * BACK?
       JEQ  BACK
*-------------------------*--------
* VALIDATE LOOP FÖR SPECIELL INMATNING.
*-------------------------*--------
       CLR  R7            * TECKEN RÄKNARE.
       LI   R6,BUFFER     * VALIDATE STRÄNG BUFFER.
       MOVB *R6+,R7       * HÄMTA ANTAL TECKEN I STRÄNG.
       CB   R7,@ALL       * ALLA TECKEN?
       JEQ  WRITE1        * JA SKRIV TECKEN!
       SRL  R7,8          * HÖGER JUSTERA ANTALET.
KTEST  CB   *R6+,R1       * TESTA TECKEN, HITTAT?
       JEQ  WRITE1        * JA, SKRIV DET.
       DEC  R7            * ANNARS MINSKA RÄKNARE.
       JNE  KTEST         * STRÄNG GENOMSÖKT? NEJ,FORTSÄTT.
       JMP  FLASH1        * ANNARS SKRIV TECKEN UNDER MARKÖR.
*
WRITE1 BLWP @VSBW         * SKRIV TECKEN
       C    R4,R5         * SLUTET NÄTT?
       JEQ  READ          * JA, STANNA KVAR PÅ POSITION.
       INC  R0            * ÖKA SKÄRM ADRESS.
       INC  R4            * ÖKA TECKEN RÄKNARE.
       JMP  READ          * LÄS NÄSTA TECKEN.
*
LEFT   MOVB R2,R1         * HÄMTA TECKEN UNDER MARKÖR.
       BLWP @VSBW         * SKRIV DET.
       CI   R4,0          * VÄNSTER KANT?
       JEQ  READ          * JA, FLYTTA INTE LÄNGRE.
       DEC  R0            * ANNARS, MINSKA SKÄRM ADRESS
       DEC  R4            * OCH TECKEN RÄKNARE.
       JMP  READ          * LÄS NÄSTA TECKEN.
*
RIGHT  MOVB R2,R1         * HÄMTA TECKEN UNDER MARKÖR
       JMP  WRITE1        * OCH UTNYTTJA WRITE1.
*
```

```
ENTER   MOVB R2,R1              * HÄMTA TECKEN UNDER MARKÖR.
        BLWP @VSBW              * SKRIV DET.
        RTWP                    * RETURNERING TILL KALLANDE PROGRAM.
*
BACK    MOVB @SAVGRM,@GRMWA     * SKRIV SPARAD GROM ADRESS (MSBY)
        NOP
        MOVB @SAVGRM+1,@GRMWA   * SKRIV ADRESS (LSBY)
        MOV  @LEV1,R11          * HÄMTA SPARAD RETURN ADRESS.
        CLR  @STATUS            * ERROR FRI RETURN!
        RT                      * HOPPA TILLBAKA TILL KALLANDE PROGRAM.
*-------------------------------*

*---------------------------------------*
INFOUT MOV  R11,R10             * SPAR ÅTERHOPP.
       LI   R4,BUFFER           * R4 = PRINTER BUFFER PEKARE.
       MOV  @ADDR,R6            * R6 = PARAMETER.
       A    @EQUON,R6           * ADDERA EQUON TILL R6
       BL   @HEXUT              * SKRIV ADDRES I PBUFF.
       INC  R4                  * ÖKA PBUFF PEKAREN TILL HEXFÄLT
       MOV  *R9,R6              * FLYTTA FRAN ADDRES TILL PARAMETER.
       BL   @HEXUT              * SKRIV INNEHALL.
       INC  R4                  * ÖKA PBUFF PEKAREN TILL ASCIIFÄLT
       CLR  R6                  * RENSA PARAMETER.
       MOVB *R9,R6              * FYTTA MSBY TILL PARAMETER.
       BL   @ASCII              * SKRIV ASCII I PBUFF.
       MOV  *R9,R6              * FLYTT HELA ORDET TILL PARAMETER.
       SLA  R6,8                * SKIFTA TILL LSBY.
       BL   @ASCII              * SKRIV ASCII.
       INC  R4                  * ÖKA PBUFF PEKAREN TILL MNEMONIC FÄLT
       B    *R10                * HOPP TILLBAKA.
*---------------------------------------*
INKEY  CLR  @STATUS             * RENSA STATUS FLAGGAN.
       BLWP @KSCAN              * KALLA PA TANGENTBORDSRUTIN.
       MOVB @STATUS,R1          * HÄMTA STATUS.
       COC  @SET,R1             * NY TANGENT ?
       RT                       * RETURN.
*---------------------------------------*
*  LÄSER 6 BYTES FRAN CPU TILL KODST    *
*---------------------------------------*
CREAD  MOV  @ADDR,R1            * HÄMTA ADRESS
       MOV  *R1+,@KODST         * FLYTTA IN INNEHÅLL I KOD BUFFER.
       MOV  *R1+,@KODST+2
       MOV  *R1,@KODST+4
       RT
*---------------------------------------*
*  LÄSER 6 BYTES FRÅN GROM TILL KODST   *
*---------------------------------------*
GREAD  MOVB @ADDR,@GRMWA        * SKRIV LÄS ADRESS MSBY.
       NOP
       MOVB @ADDR+1,@GRMWA      * --  II --  II --  LSBY.
       LI   R1,KODST            * LADDA BUFFER ADRESS.
       LI   R2,6                * ANTAL BYTES ATT LÄSA.
GREAD1 MOVB @GRMRD,*R1+         * FLYTTA FRÅN GROM TILL BUFFER.
       DEC  R2                  * MINSKA LOOP RÄKNARE.
       JNE  GREAD1              * FÄRDIG?
       RT                       * JA, RETURN.
*---------------------------------------*
*  LÄSER 6 BYTES FRAN VDP TILL KODST    *
*---------------------------------------*
VREAD  MOV  @ADDR,R0            * HÄMTA VDP ADRESS.
       LI   R1,KODST            * BUFFER ADRESS.
       LI   R2,6                * ANTAL BYTES ATT LÄSA.
       BLWP @VMBR               * LÄS TILL BUFFER FRÅN VDP.
       RT                       * RETURN.
*---------------------------------------*
* SKRIVER TEXT MEDELLANDEN PÅ SKÄRMEN   *
*---------------------------------------*
OUTTXT MOV  *R11+,R4            * HÄMTA TEXTNUMMER.
       SLA  R4,1                * TABELL JUSTERA.
       MOV  @TXT0(R4),R0        * HÄMTA SKÄRM ADRESS.
       MOV  @TXT1(R4),R1        * HÄMTA TEXT ADRESS.
       MOV  @TXT2(R4),R2        * HÄMTA ANTAL BYTES ATT SKRIVA.
       BLWP @VMBW               * SKRIV TEXT.
       RT                       * RETURN.
*---------------------------------------*
* SKRIVER ASCII TECKEN I BUFFER         *
*---------------------------------------*
ASCII  CI   R6,>2000            * ÄR ASCII MINDRE ÄN 32?
       JL   ASCII1              * JA.
       CI   R6,>7F00            * ÄR ASCII STÖRRE ÄN 127?
       JH   ASCII1              * JA.
       MOVB R6,*R4+             * ANNARS SKRIV ASCII I BUFFER.
       JMP  ASCII2              * HOPPA UR RUTIN.
ASCII1 MOVB @SPACE,*R4+         * SKRIV ETT MELLANSLAG.
ASCII2 RT                       * RETURN.

*---------------------------------------*
* STOPPAR SCROLLNING TILLS NY TANGENT   *
*---------------------------------------*
HOLD   MOV  R11,@LEV3           * SPARA ÅTERHOPP
       LI   R10,HOLD3           * LADDA TEST ADRESS.
       BL   @INKEY              * ÄR TANGENT NEDTRYCKT?
       JEQ  HOLD1               * JA, VÄNTA PÅ NÄSTA.
       JMP  HOLD2               * NEJ, HOPPA UR RUTIN.
*
HOLD3  BL   @INKEY              * TANGENT?
       JNE  HOLD3               * NEJ, VÄNTA.
       LI   R10,HOLD2           * LADDA NY ADRESS.
*
HOLD1  MOVB @KEYBRD,R14         * HÄMTA ASCII.
       SRL  R14,8               * HÖGER JUSTERA.
       CI   R14,2               * CLEAR?
       JNE  HOLD4               * NEJ VÄNTA PÅ NY TANGENT!
       BL   @CLOSE              * STÄNG EVENTUELL FIL.
       MOV  @LEV2,R10           * HOPPA TILL RUN4
HOLD4  B    *R10                * RETURN
*
HOLD2  MOV  @LEV3,R10           * LADDA ÅTERHOPPS ADRESS.
       JMP  HOLD4               * FORTSÄTT LISTA.
*---------------------------------------*
* INITIERAR SKÄRM OCH VDP REGISTER      *
*---------------------------------------*
INIT   LI   R0,>87F1
       BLWP @VSBW               * SKRIV >F1 TILL VDP REGISTER 7
       LI   R0,>81D0
       BLWP @VSBW               * SKRIV >D0 TILL VDP REGISTER 1
       SWPB R0
       MOVB R0,@>83D4           * LÄGG EN KOPIA FÖR ATT BEHÄLLA TEXT-MODE.
       RT                       * RETURN.

*---------------------------------------*
*          RENSAR SKÄRMEN               *
*---------------------------------------*
CLS    CLR  R0                  * SKÄRM ADRESS.
       LI   R1,>2000            * BLANKTECKEN.
       LI   R2,959              * 960 TECKEN ATT BLANKSTÄLLA.
       BLWP @VSBW               * SKRIV ETT BLANKTECKEN.
CLS1   MOVB R1,@VDPWD           * SKRIV DIREKT TILL VDP.
       DEC  R2                  * MINSKA LOOP RÄKNARE.
       JNE  CLS1                * SKÄRMEN TOM?
       RT                       * JA, RETURN.
*---------------------------------------*
* LADDNING AV VALIDATE STRÄNGAR         *
*---------------------------------------*
HEXVAL LI   R1,HV               * LADDA HEX-STRÄNG ADRESS.
       JMP  VALOAD
GVCVAL LI   R1,GVC              * LADDA "GVC"-STRÄNS ADRESS.
VALOAD LI   R2,BUFFER           * I BUFFER TESTAS TECKEN I INPUT.
       MOVB *R1,R3              * HÄMTA ANTAL TECKEN.
       SRL  R3,8                * HÖGER JUSTERA.
       INC  R3                  * ÖKA FÖR ATT FÅ MED ANTAL.
VAL1   MOVB *R1+,*R2+           * SKRIV STRÄNG TILL BUFFER.
       DEC  R3                  * FÄRDIG?
       JNE  VAL1                * NEJ, FORTSÄTT.
       RT                       * RETURN.
*---------------------------------------*
* ASCII HEX PÅ SKÄRM TILL BINÄR         *
*---------------------------------------*
HEXIN  CLR  R2                  * R2 = VÄRDET UT
HEXI4  CLR  R1                  *
       BLWP @VSBR               * HÄMTA HEX SIFFRA.
       CI   R1,>2000            * ÄR DET ETT MELLANSLAG?
       JEQ  HEXI1               * JA, KLAR.
       SLA  R2,4                * ANNARS SKIFTA R2 TILL NY HEX SIFFRA.
       SRL  R1,8                * HÖGER JUSTERA ASCII (HEX) SIFFRAN.
       AI   R1,-48              * MINSKA FÖR ATT NÅ NOLL.
       CI   R1,9                * ÄR SIFFRAN STÖRRE ÄN NIO?
       JH   HEXI2               * JA.
       JMP  HEXI3               * NEJ.
HEXI2  AI   R1,-7               * DRA IFRÅN 7 DIFF MELLAN ASCII(0) OCH ASCII(9).
HEXI3  A    R1,R2               * ADDERA SIFFRA TILL JUSTERAD R2.
       INC  R0                  * ÖKA INLÄSNINGS ADRESS.
       JMP  HEXI4               * LÄS VIDARE.
HEXI1  RT                       * RETURN.
*---------------------------------------*
* BINÄR TILL ASCII HEX I BUFFER         *
*---------------------------------------*
HEXUT  LI   R2,4                * 4 TECKEN ATT SKRIVA.
HEXU3  MOV  R6,R1               * KOPIERA PARAMETER.
       ANDI R1,>F000            * SILA FRAM DEN FÖRSTA NYBBELN.
       SRL  R1,12               * HÖGER JUSTERA.
       AI   R1,48               * ADDERA ASCII.
       CI   R1,57               * STÖRRE ÄN ASCII(9)?
       JH   HEXU1               * JA.
       JMP  HEXU2               * NEJ.
HEXU1  AI   R1,7                * ADDERA TILL BOKSTAV.
HEXU2  SWPB R1                  * SKIFTA BYTE FÖR UTSKRIFT.
       MOVB R1,*R4+             * SKRIV TILL BUFFER.
       SLA  R6,4                * FIXA NÄSTA HEX SIFFRA.
       DEC  R2                  * MINSKA LOOP RÄKNARE.
       JNE  HEXU3               * FÄRDIG?
       RT                       * JA, RETURN.
*---------------------------------------*
*           ÖPPNAR EN FIL               *
*---------------------------------------*
OPEN   LI   R0,PAB              * HÄMTA PAB ADRESS I VDP.
       LI   R1,PDATA            * HÄMTA PAB ADRESS I CPU.
       MOV  @NAMEL,R2           * HÄMTA FILNAMNS LÄNGD.
       AI   R2,10               * ADDERA PAB LÄNGD.
       BLWP @VMBW               * SKRIV PAB + NAMN.
       JMP  DSR                 * KALLA DSR.
*---------------------------------------*
* SKRIVER ETT RECORD TILL FILEN         *
*---------------------------------------*
PRINT  MOVB @CR,*R4+            * LÄGG EN VAGNRETUR I SLUTET AV BUFFERN.
       LI   R0,PAB              * HÄMTA PAB ADRESS I VDP.
       MOVB @PRINTD,R1          * HÄMTA PRINT KOMMANDO.
       BLWP @VSBW               * SKRIV KOMMANDO.
       LI   R1,BUFFER           * HÄMTA BUFFER ADRESS.
       S    R1,R4               * BUFFER PEKARE MED BUFFER START.
       LI   R0,PABBUF           * LADDA PABBUFFER ADRESS.
       MOV  R4,R2               * KOPIERA POST LÄNGD.
       BLWP @VMBW               * SKRIV BUFFER TILL PABBUFFER.
       SWPB R2                  * JUSTERA LÄNGD.
       MOVB R2,R1               *
       LI   R0,PAB+5            * ADRESS TILL POSTLÄNGD.
       BLWP @VSBW               * SKRIV DEN EGENTLIGA LÄNGDEN.
       JMP  DSR                 * KALLA DSR.
*---------------------------------------*
*       STÄNGER ÖPPEN FIL               *
*---------------------------------------*
CLOSE  MOV  @PRON,@PRON         * ÄR FIL ÖPPEN?
       JEQ  DSRUT               * NEJ HOPPA UR RUTIN.
       LI   R0,PAB              * LADDA PAB ADRESS.
       MOVB @CLOSED,R1          * HÄMTA STÄNG KOMMANDO.
       BLWP @VSBW               * SKRIV DET.
*---------------------------------------*
*  JUSTERAR PEKARE , KALLAR DSRRUTIN    *
*---------------------------------------*
DSR    LI   R0,PAB+9            * FIXA PAB PEKARE.
       MOV  R0,@PNTR            * LÄGG DEN PÅ SIN PLATS.
       BLWP @DSRLNK             * KALLA DSR.
       DATA 8                   *
DSRUT  RT                       * RETURN

*    RUTIN FÖR ERROR UTSKRIFTER         *
*---------------------------------------*
ERRRUT MOV  R11,R10             * SPAR ÅTERHOPP.
       AI   R0,>3000            * R0 INNEHÅLLER ERROR KOD.
       MOVB R0,@TEXTB+13        * SKRIV KOD DIREKT I TEXTEN.
       BL   @OUTTXT             * SKRIV TEXT->
       DATA 8                   * "** I/O ERROR X **"
       JMP  PRESS1              * HOPPA OCH VÄNTA PÅ TANGENT.
PRESS  MOV  R11,R10             * SPARA ÅTERHOPP.
PRESS1 BL   @OUTTXT             * SKRIV TEXT->
       DATA 9                   * "PRESS ENTER TO ..."
PRESS2 BL   @INKEY              * TANGENT?
       JNE  PRESS2              * NEJ.
       MOVB @KEYBRD,R0          * HÄMTA ASCII.
       SRL  R0,8                * HÖGER JUSTERA.
       CI   R0,13               * ENTER?
       JNE  PRESS2              * NEJ, VÄNTA.
       B    *R10                * RETURN.
```

```
*---------------------------------------*
* CHECKAR LÄNGD PÅ FILNAMN OCH JUSTERAR *
*---------------------------------------*
CHECK  CLR  R0              * TEST REGISTER.
       LI   R2,BUFFER       * JUSTERING EFTER MELLANSLAG.
       LI   R1,BUFFER       * SÖKNING FÖRE MELLANSLAG.
CHECK1 MOVB *R1+,R0         * FLYTTA FRÅN BUFFER TILL TEST.
       CI   R1,BUFFER+39    * ÄR BUFFERN IGENOM SÖKT?
       JH   CHOUT           * JA.
       CI   R0,>2000        * ÄR TECKNET ETT MELLAN SLAG?
       JEQ  CHECK1          * JA, LÄS VIDARE.
       MOVB R0,*R2+         * ANNARS LÄGG FRÅN BÖRJAN.
       JMP  CHECK1          * FORTSÄTT LÄS.
CHOUT  LI   R1,BUFFER       * LADDA BUFFER ADRESS IGEN.
       NEG  R1              * GÖR ADRESS NEGATIV
       A    R1,R2           * DRA I FRÅN JUSTERAD LÄNGD.
       SLA  R2,8            * VÄNSTER JUSTERA LÄNGD.
       MOVB R2,@NAMEL       * SKRIV LÄNGD.
       RT                   * RETURN.
*---------------------------------------*
*           INITIERING                  *
*---------------------------------------*
RUN    MOV  R11,@LEV1       * SPAR ÅTERHOPP.
       LWPI MYREG           * LADDA ARBETS REGISTER.
       BL   @INIT           * INITIERA SKÄRM.
       LI   R0,RUN4         * LADDA R0 MED RUN4 ADRESSEN.
       MOV  R0,@LEV2        * LAGRA RUN4 ADRESS FÖR CLEAR.
*---------------------------------------*
       MOVB @GRMRA,@SAVGRM  * SPAR MSBY GROM ADRESS.
       NOP                  * VÄNTA!
       MOVB @GRMRA,@SAVGRM+1 * SPAR LSBY GROM ADRESS.
       DEC  @SAVGRM         * JUSTERA SPARAD ADRESS.
*---------------------------------------*
RUN4   BL   @CLS            * RENSA SKÄRMEN.
       CLR  @PRON           * PRINT FLAGGA AV.
       CLR  @DSRON          * DSR   --  II --.
*-------------------------*
       BL   @OUTTXT         * SKRIV TEXTEN->
       DATA 0               * DISASSEMBLER UTILITY
       BL   @OUTTXT         * SKRIV TEXTEN->
       DATA 1               * PRINTER DESCRIPTION:
       MOVB @ALL,@BUFFER    * ALLA TECKEN I INPUT.
       BLWP @INPUT          * TA FIL NAMN.
       DATA 100             * POSITION 100 PÅ SKÄRMEN.
       DATA 40              * LÄNGDEN 40.
       LI   R0,100          * SKÄRM ADRESS.
       LI   R1,BUFFER       * BUFFER ADRESS FÖR FIL NAMN.
       LI   R2,40           * MAX ANTAL TECKEN ATT LÄSA.
       BLWP @VMBR           * LÄS NAMNET.
       BL   @CHECK          * ANNARS TESTA LÄNGDEN.
       CI   R2,0            * NÅGOT SKRIVET I DSRNAMN?
       JEQ  RUN7            * NEJ, TESTA DSRROM.
       SETO @PRON           * SÄTT PRINTER FLAGGA.
       BL   @OPEN           * ÖPPNA FILEN.
       JNE  RUN7            * ERROR?
       BL   @ERRRUT         * NEJ, SKRIV ERROR NUMMER.
       JMP  RUN4            * STARTA OM.
*-------------------------*
* DISASSEMBLERING AV DSRROM ? *
*                         *
RUN7   BL   @OUTTXT         * SKRIV TEXT->
       DATA 2               * CRUBASE:
       BL   @HEXVAL         * ANVÄND HEX SIFFOR I INPUT.
       BLWP @INPUT          * KALLA INPUT RUTIN.
       DATA 208,4           * START ADRESS, LÄNGD.
       LI   R0,208          * LADDA VDP ADRESS I PARAMETER.
       BL   @HEXIN          * TA IN HEXSIFFROR FRÅN SKÄRM.
       MOV  R2,R12          * KOPIERA RESULTAT TILL CRUBAS.
       CI   R12,>1000       * ÄR BAS MINDRE ÄN 1000?
       JL   RUN8            * JA, HOPPA TILL EQUATE.
       SETO @DSRON          * ANNARS SÄTT DSR FLAGGA.
*-------------------------*
* SKALL EQUATE SÄTTAS?    *
*
RUN8   BL   @OUTTXT         * SKRIV TEXT->
       DATA 3               * EQUATE:
       BLWP @INPUT          * KALLA INPUT.
       DATA 287,4           * VDP ADRESS, LÄNGD.
       LI   R0,287          * VDP ADRESS->PARAMETER
       BL   @HEXIN          * HÄMTA HEX.
       MOV  R2,@EQUON       * FLYTTA RESULTAT TILL EQUON.
*-------------------------*
* VÄLJ GROM,VRAM ELLER CPU *
*
       BL   @OUTTXT         * SKRIV TEXT->
       DATA 4               * G)ROM V)DP C)PU:C.
       BL   @GVCVAL         * G, V ELLER C I INPUT.
       BLWP @INPUT          * KALLA INPUT.
       DATA 376,1           * VDP ADRESS, LÄNGD.
       LI   R0,376          * VDP ADDRESS
       BLWP @VSBR           * LÄS TECKEN FRÅN SKÄRM.
       MOVB R1,@GVCSAV      * SPARA TECKEN FÖR TEST.
*-------------------------*
* START OCH STOP ADRESSER
*
       BL   @OUTTXT         * SKRIV TEXT->
       DATA 5               * START ADRESS:
       BL   @OUTTXT         * SKRIV TEXT->
       DATA 6               * STOP  ADRESS:
       BL   @HEXVAL         * HEX INPUT.
       BLWP @INPUT          * KALLA INPUT.
       DATA 534,4           * VDP ADRESS, LÄNGD.
       LI   R0,534          * VDP ADRESS->PARAMETER
       BL   @HEXIN          * LÄS HEX.
       ANDI R2,>FFFE        * JÄMN START
       MOV  R2,@ADDR        * SPARA START I ADDR.
RUN1   BLWP @INPUT          * HÄMTA IN STOP.
       DATA 574,4           * VDP ADRESS, LÄNGD.
       LI   R0,574          * VDP ADRESS->PARAMETER.
       BL   @HEXIN          * HÄMTA HEX.
       C    @ADDR,R2        * ÄR START STÖRRE ÄN STOP?
       JH   RUN1            * JA, HÄMTA NYTT STOP.
       MOV  R2,@STOP        * SPARA STOP ADRESS.
       BL   @OUTTXT         * SKRIV TEXT->
       DATA 7               * PRESS CLEAR...
       BLWP @SCROL          * SCROLLA SKÄRMEN OCH RENSA BUFFER.
```

```
*---------------------------------------*
*           MAIN LOOP                   *
*---------------------------------------*
DSRTST CLR  R5              * NOLLSTÄLL SCROLL RÄKNARE.
       MOV  @DSRON,@DSRON   * DISASSEMBLERING AV DSR?
       JEQ  CPUTST          * NEJ, HOPPA TILL NÄSTA TEST!
       SBO  0               * ANNARS, SKIFTA IN DSR.
*---------------------------------------*
CPUTST CB   @GVCSAV,@GVC+3  * DISASSEMBLERA CPU?
       JNE  VDPTST          * NEJ, TESTA VDP.
       BL   @CREAD          * JA, FLYTTA 6 BYTES TILL MNEMST.
       JMP  LOADR9          * HOPPA VIDARE.
VDPTST CB   @GVCSAV,@GVC+2  * DISASSEMBLERA VDP?
       JNE  GRMTST          * NEJ, TESTA GROM.
       BL   @VREAD          * ANNARS FLYTTA 6 BYTES FRÅN VDP TILL MNEMST.
       JMP  LOADR9          * HOPPA VIDARE.
GRMTST BL   @GREAD          * OM OVAN FLYTTA FRÅN GROM TILL MNEMST.
*---------------------------------------*
LOADR9 LI   R9,KODST        * R9 = PEKARE TILL KOD.
*---------------------------------------*
       BL   @INFOUT         * SKRIV ADRESS, HEX, ASCII.
*---------------------------------------*
       CLR  R3              * R3 = FORMAT RÄKNARE.
       CLR  R0              * R0 = SKIFT RESULTAT FÖR DEST.
       LI   R6,OP           * R6 = OP-CODE TABELL PEKARE.
OP1    C    *R6,@FTEST      * NYTT FORMAT?
       JNE  OP2             * NEJ, TESTA KOD.
       INCT R6              * ÖKA TILL NYTT FORMAT.
       INCT R3              * ÖKA FORMAT RÄKNARE.
       CI   R3,18           * SLUT PÅ FORMAT?
       JNE  OP1             * NEJ, TESTA KOD.
       JMP  UNDEF           * JA, KODEN HAR INGEN MNEMONIC.
*
OP2    MOV  *R9,R2          * KOPIERA KODEN.
       MOV  @BITS(R3),R0    * HÄMTA ANTAL SKIFTBITAR FÖR FORMATET.
       SRA  R2,0            * SKIFTA AT HÖGER MED AVSEENDE PÅ R0.
       SLA  R2,0            * -- II -   VÄNSTER    -- II -     .
       C    *R6,R2          * JÄMFÖR JUSTERAD KOD MED TABELL.
       JEQ  FOUND           * OM LIKA HOPPA TILL FOUND.
       AI   R6,6            * ÖKA TABELL PEKAREN.
       JMP  OP1             * FORSÄTT SÖK I TABELLEN.
*---------------------------------------*
UNDEF  LI   R6,UND          * LADDA R6 MED TEXT ADRESS.
       LI   R2,10           * R2 = ANTAL TECKEN.
       JMP  COPY            * HOPPA TILL ÖVERFÖRING AV TEXT.
*
FOUND  LI   R2,4            * LADDA R2 MED 4 (ANTAL TECKEN I MNEM KOD!.
       INCT R6              * ÖKA TABELL RÄKNAREN TILL MNEM TEXT.
*
COPY   MOVB *R6+,*R4+       * FLYTTA TEXT TILL BUFFER.
       DEC  R2              * MINSKA TECKEN RÄKNARE.
       JNE  COPY            * OM EJ FÄRDIG FORTSÄTT KOPIERING.
*---------------------------------------*
       CI   R3,18           * ÄR DET EN UNDEF TEXT?
       JEQ  F7              * JA SKRIV DEN.
       INC  R4              * ÖKA PBUFF PEKAREN TILL ADRESS FORMAT FÄLT.
       MOV  @JUMP(R3),R3    * FLYTTA FRÅN HOPPTABELL TILL R3.
       B    *R3             * HOPPA TILL FORMAT.
*---------------------------------------*
PRINT4 BL   @INFOUT         * SKRIV ADRESS, HEX OCH ASCII.
*---------------------------------------*
PRINT1 MOV  @DSRON,@DSRON   * DSR OPTION?
       JEQ  PRINT2          * NEJ, TESTA PRINTER.
       SBZ  0               * ANNARS SKIFTA UT DSR.
PRINT2 MOV  @PRON,@PRON     * PRINTER OPTION?
       JEQ  PRINT3          * NEJ, SKRIV PÅ SKÄRM.
       BL   @PRINT          * JA, SKRIV PÅ PRINTER.
*---------------------------------------*
PRINT3 LI   R0,880          * UTSKRIFTS ADRESS PÅ SKÄRM.
       LI   R1,BUFFER       * TEXT FINNS I PBUFF.
       LI   R2,40           * ANTAL TECKEN ÄR 40.
       BLWP @VMBW           * SKRIV.
       BL   @HOLD           * STÄNNA FÖR ATT KIKA?
       BLWP @SCROL          * SCROLLA SKÄRMEN EN RAD.
       INCT @ADDR           * ÖKA ADRESS RÄKNARE.
       INCT R9              * ÖKA FYSISK ADRESS I MNEMST.
       DEC  R5              * MINSKA SCROLL RÄKNARE.
*---------------------------------------*
       CI   R5,-1           * FÄRDIG MED INSTRUKTION?
       JNE  PRINT4          * NEJ, SKRIV FLER ADRESSER.
*
       C    @ADDR,@STOP     * SLUT PÅ DISASSEMBLERING?
       JLE  DSRTST          * NEJ FORTSÄTT.
       BL   @CLOSE          * STÄNG EVENTUELL FIL.
       BL   @PRESS          * GE "PRESS ENTER.." MEDDELANDE.
       B    @RUN4           * BÖRJA OM.
*---------------------------------------*
*                             * FORMAT 7
F7     B    @PRINT1         * ENKELT FORMAT ENDAST MNEMONIC.
*-------------------------------------*
*                         * FORMAT 2
F2     MOV  *R9,R7         * KOPIERA KOD.
       SLA  R7,8           * SKIFTA BORT ONÖDIG INFORMATION.
       SRL  R7,8           * JUSTERA.
       AI   R6,-6          * LÄT TABELLPEKAREN PEKA PÅ KODEN.
       MOV  *R6,R6         * HÄMTA KOD FRÅN TABELL.
       CI   R6,>1C00       * ÄR KODEN INTE EN JUMP?
       JH   F22            * NEJ, DET ÄR SBO, SBZ ELLER TB.
       CI   R7,128         * HOPP TILLBAKA?
       JL   F21            * NEJ.
       ORI  R7,>FF00       * JUSTERA RELATIVT HOPP BAKÅT.
F21    SLA  R7,1           * MULTIPLICERA MED 2.
       MOV  @ADDR,R6       * HÄMTA ADRESS.
       A    @EQUON,R6      * ADDERA EQUATE.
       A    R7,R6          * LÄGG TILL RELATIV ADRESS.
       AI   R6,2           *
       JMP  F23            * SKRIV UT
F22    MOV  R7,R6          * FLYTTA BIT NUMMER TILL PARAMETER.
F23    BL   @HEXUT         * SKRIV HEX.
       JMP  F7             * SKRIV UT.
*-------------------------------------*
*                         * FORMAT 1
F1     JMP  TS             * SKRIV KÄLLA.
       JMP  TD             * SKRIV DESTINATION.
       JMP  F7             * SKRIV UT.
*-------------------------------------*
*                         * FORMAT 3 OCH 9
F3     JMP  TS             * SKRIV KÄLLA.
       MOVB @KD,*R4+       * SKRIV ','.
       AI   R3,20          * FIXA ÅTER HOPPSADRESS VID TS2 KALL.
       CLR  R6             * REGISTER ADRESSERING.
       MOV  *R9,R7         * KOPIERA KOD.
       ANDI R7,>03C0       * SÅLLA UT DESTINATION.
       SRL  R7,6           * HÖGER JUSTERA.
       JMP  TS2            * SKRIV REGISTER OCH NUMMER.
       JMP  F7             * SKRIV UT.
```

```
*--------------------------------------*
*                                      * FORMAT 4
F4      JMP   TS                       * SKRIV KÆLLA.
        MOVB  @KD,*R4+                 * SKRIV ','.
        MOV   *R9,R6                   * KOPIERA KOD.
        ANDI  R6,>03C0                 * SALLA UT ANTAL CRU-BITAR.
        SRL   R6,6                     * HØGER JUSTERA.
F54     CI    R6,10                    * MINDRE EN 10?
        JL    F41                      * JA.
        MOVB  @D1,*R4+                 * SKRIV '1'.
        AI    R6,-10                   * MINSKA ANTAL MED 10.
F41     AI    R6,48                    * ADDERA ASCII OFFSET.
        SWPB  R6                       * SKIFTA FØR UTSKRIFT.
        MOVB  R6,*R4+                  * SKRIV I BUFFER.
        JMP   F7                       * SKRIV UT.
*--------------------------------------*
*                                      * FORMAT 5
F5      CLR   R6                       * REGISTER ADRESSERING.
        AI    R3,14                    * FIXA ATERHOPP FRAN TS2.
        MOV   *R9,R7                   * HÆMTA KOD.
        ANDI  R7,>000F                 * HÆMTA REGISTER NUMMER.
        JMP   TS2                      * SKRIV REGISTER.
        MOVB  @KD,*R4+                 * SKRIV ','.
        MOV   *R9,R6                   * KOPIERA KOD.
        ANDI  R6,>00F0                 * ANTAL SKIFT BITAR.
        SRL   R6,4                     * HØGER JUSTERA.
        JMP   F54                      * ANVÆND FORMATET OVAN.
*--------------------------------------*
*                                      * FORMAT 6
F6      JMP   TS                       * SKRIV KÆLLA.
        JMP   F7                       * SKRIV UT.
*--------------------------------------*
*                                            * FORMAT 8
F8      AI    R6,-6                    * MINSKA TABELL PEKAREN TILL KODEN.
        MOV   *R6,R6                   * HÆMTA TABELL KOD.
        CI    R6,>02C0                 * ÆR DET EN IMMEDIATE UTAN REGISTER?
        JH    F81                      * JA
        AI    R3,24                    * FIXA ATERHOPPS ADRESS FØR TS2.
        MOV   *R9,R7                   * KOPIERA KOD.
        ANDI  R7,>000F                 * SALLA UT REGISTER NUMMER.
        JMP   TS2                      * SKRIV REGISTER.
        CI    R6,>0280                 * BARA REGISTER? (STWP OCH STST).
        JH    F7                       * JA, SKRIV UT.
        MOVB  @KD,*R4+                 * ANNARS SKRIV ','.
F81     MOV   R9,R13                   * HÆMTA ADRESS TILL VÆRDE
        INCT  R13                      * SOM LIGGER UNDER KOD.
        INC   R5                       * ØKA SCROLL RÆKNARE.
        MOV   *R13,R6                  * FLYTTA VÆRDE TILL PARAMETER.
        BL    @HEXUT                   * SKRIV VÆRDET.
        JMP   F7                       * SKRIV UT.
*--------------------------------------*
* RUTIN FØR KÆLLA
*
TS      INCT  R3                       * ØKA ATERHOPPS-ADRESS.
        MOV   *R9,R6                   * KOPIERA KOD.
        MOV   *R9,R7                   * KOPIERA KOD.
        ANDI  R7,>000F                 * R7 = KÆLLA
        ANDI  R6,>0030                 * R6 = T-FÆLT.
        SRL   R6,4                     * SKIFTA TILL TESTVÆRDE.
        JMP   TDS                      * GØR TESTER.
*--------------------------------------*
* RUTIN FØR DESTINATION
*
TD      INCT  R3                       * ØKA ATERHOPPS-ADRESS.
        MOVB  @KD,*R4+                 * SKRIV KOMMATECKEN.
        MOV   *R9,R6                   * KOPIERA KOD
        MOV   *R9,R7                   * --  II  --
        ANDI  R6,>0C00                 * R6 = T-FÆLT
        ANDI  R7,>03C0                 * R7 = DESTINATION.
        SRL   R6,10                    * SKIFTA TILL TEST  VÆRDE.
        SRL   R7,6                     * SKIFTA TILL TESTVÆRDE.
*--------------------------------------*
TDS     CI    R6,2                     * MINNES ADRESSERING?
        JNE   TS1                      * OM INTE, TESTA ANDRA T-VÆRDEN.
*--------------------------------------*
        INC   R5                       * ØKA SCROLL RÆKNARE
        MOV   R5,R13                   * KOPIERA FØR ATKOMST AV ADRESS.
        SLA   R13,1                    * MULTIPLICERA MED TVÅ.
        A     R9,R13                   * R13 PEKAR PA ADRESSEN.
        MOVB  @AD,*R4+                 * SKRIV ALPHA SLANG.
        MOV   R6,R14                   * SPARA R6
        MOV   *R13,R6                  * ADRESS->PARAMETER.
        BL    @HEXUT                   * SKRIV ADRESS.
        MOV   R14,R6                   * HÆMTA TILLBAKA SPARAD R6
        CI    R7,0                     * INDEXERAD MINNES ADRESSERING?
        JEQ   TSOUT                    * JA, DA ÆR VI FÆRDIGA.
        MOVB  @VD,*R4+                 * ANNARS SKRIV EN VÆNSTER PARENTES.
        JMP   TS2                      * FYLL I PARENTES
*
TS1     CI    R6,0                     * REGISTER ADRESSERING?
        JEQ   TS2                      * JA SKRIV ENKEL REGISTER.
        MOVB  @SD,*R4+                 * ANNARS INDEXERAD, SKRIV '+'.
TS2     MOVB  @RD,*R4+                 * SKRIV 'R'
        CI    R7,10                    * REGISTER NUMMER  >=10?
        JL    TS21                     * NEJ SKRIV ENSIFFRIGT NUMMER.
        MOVB  @D1,*R4+                 * SKRIV '1'
        AI    R7,-10                   * MINSKA MED 10
TS21    AI    R7,48                    * LÆGG TILL ASCII OFFSET.
        SWPB  R7                       * SKIFTA BYTES FØR UTSKRIFT
        MOVB  R7,*R4+                  * SKRIV SIFFRA.
        CI    R6,3                     * AUTO INCREMENT?
        JNE   TS4                      * NEJ GÅR SLUT TEST.
        MOVB  @PD,*R4+                 * SKRIV '+'
        JMP   TSOUT                    * AVSLUTA.
TS4     CI    R6,2                     * MINNES ADRESSERING?
        JNE   TSOUT                    * NEJ, AVSLUTA.
        MOVB  @HD,*R4+                 * SKRIV HØGER PARENTES.
TSOUT   B     *R3                      * HOPPA TILLBAKA.
*--------------------------------------*
```

```
*****************************************
*          DIS COPYFILE                 *
*****************************************
        COPY "DSK1.DISHEAD"
        COPY "DSK1.DISBLWP"
        COPY "DSK1.DISROUT"
        COPY "DSK1.DISMAIN"
        END
```

PRATORN

av ARNE WENNBERG

Utvidga ditt ordförråd.

Med hjälp av Speech Synthesizer och lämplig programmodul kan man få TI99/4A att tala. Använder man Terminal Emulator II som programmodul behärskar den alla ord i engelskan och med lite list och knep kan också svenska ord framställas med Texas-accent. Dock endast i TI-BASIC.
I Extended-Basic finns bara tillgång till c:a 400 ord, vilka är listade i den tillhörande manualen under Appendix L. Detta är en rätt besvärande begränsning. Med mitt program kan man väsentligt utvidga ordförrådet i Extended-Basic. I manualen kan man lära sig att tillfoga vissa ändelser till de ord som ingår i ordförrådet. Innan en ändelse läggs till måste ordet för det mesta trunkeras, stympas dvs. ett antal bytes i slutet på ordet tages bort. Ett program för detta finns under Appendix M. Att trunkera ett ord i slutet är lätt. Man kan ta bort en del av ett ljud, en stavelse eller nästan hela ordet och bara behålla första ljudet. Att trunkera ord i början är mera besvärligt. Tar man slumpmässigt bort ett visst valt antal bytes blir resultatet för der mesta en serie mystiska och oförståeliga ljud från Pratorn. Men man kan finna platser där det går att dela orden och använda slutet av detta ord. Programmet som kallas "TRUNKERING" är avsett att användas för att sönderdela ord och sätta ihop dessa orddelar till nya ord. Man prövar sig fram till rätt antal bytes med trunkering av det valda ordet. Sedan kan de nya orden användas i alla Extended-Basic program. Mitt program är hela tiden självförklarande och blir därmed lätt att använda. Detta förklaras bäst genom ett exempel. Antag att man vill skapa ordet STATION (engelskt uttal ungefär "stejtion"). Det gäller då att först hitta lämpliga ord i det fasta ordförrådet att arbeta med. Börja t.ex. med att slå in START. Trunkera det i slutet med 58 bytes. Kvar blir då ljudet "ST". När rätt trunkering hittats lagras det stympade ordet i en sträng. Välj sträng 1 (ett). Därefter går du vidare med att trunkera ett nytt ord. Nu gäller det att finna ett ord som kan ge oss ljudet "EJ", bokstaven "A" kan användas med trunkering=0. Det går också att använda EIGHT. Mata in detta ord. Trunkera med siffran 25 i slutet av ordet. Lagra i sträng nr 2 (två). Nu återstår ändelsen TION och för att finna den måste man trunkera i början av ett ord. Slå in INSTRUCTION. Här är det lite besvärligt att hitta rätt trunkering vilket i detta fall är 60 bytes. Försök med tal som ligger nära detta. Andra tal ger helt omöjliga resultat. Lagra detta i sträng 3. Tryck sedan på sammanställning. Programmet frågar om man skall sammanställa två eller tre strängar. I detta fall är det tre. Man får vidare förslag på olika sätt att sätta samman dessa strängar. I vårt fall är det modellen 1-2-3.
Pratorn säger nu det nya ordet STATION. Man har möjlighet att lyssna på ordet hur många gånger man vill och även ändra trunkering eller ord om man inte är nöjd med resultatet. Exemplet är inte helt tillfredställande det kan säkert göras bättre.
Programmet kan också hantera nya ord vilka sammanställts av två eller tre delar. Ett nytt och längre ord kan givetvis delas upp i två bitar vilka behandlas var för sig.
Det nya ordet kan skrivas in i ett Extended-Basic program enligt listning 2. Rad 100-340 skrives lämpligen in i början av ditt program. Rad 400 kan sedan anropas varje gång man vill att pratorn skall uttala ordet ex. STATION.

Lycka till "

```
100 REM ÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑ
110 REM Ñ   TRUNKERING   Ñ
120 REM Ñ ETT PROGRAM AV Ñ
130 REM Ñ  ARNE WENNBERG Ñ
140 REM ÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑÑ
150 CALL CLEAR
160 CALL CHAR(91,"00280038447C4444")
170 CALL CHAR(92,"0028007C4444447C")
180 CALL CHAR(93,"00382838447C4444")
190 CALL CHAR(94,"C3C7666637361E1E")
200 CALL CHAR(95,"C3E36666EC6C7878")
```

```
210 CALL SCREEN(9)
220 PRINT "   ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ"
230 PRINT "   Ä T R U N K E R I N G Ä"
240 PRINT "   ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ"
250 PRINT : : : : :
260 PRINT "            19^_84"
270 PRINT : : : :
280 GOSUB 1500
290 CALL SAY("HELLO. . . PARTNER"):: CALL SAY(,NI$&TO
    $&SE$):: CALL SAY(,YO$)
300 FOR I=1 TO 1000
310 NEXT I
320 CALL CLEAR
330 CALL SCREEN(12)
340 DISPLAY AT(1,1):"SLÅ IN ETT ORD SOM FINNS I";"ORD
    FÖRRÅDET FÖR EX-BASIC"
350 CALL SAY("TYPE WORD")
360 INPUT "VILKET ORD? ":O$
370 CALL SPGET(O$,Q$)
380 CALL SAY("WHICH . . END")
390 INPUT "VILKEN ÄNDA (B/S) ":F$
400 CALL SAY("TYPE NUMBER")
410 INPUT "TRUNKERING? ":T
420 IF T>LEN(Q$)THEN 430 ELSE 440
430 DISPLAY AT(1,1):"FÖR STORT TAL." :: GOTO 400
440 IF F$="B" THEN GOSUB 600 ELSE 460
450 GOTO 470
460 IF F$="S" THEN GOSUB 550 ELSE 380
470 C$=A$
480 CALL SAY(,C$)
490 DISPLAY AT(1,1):"TRYCK 1=EN GÅNG TILL":"TRYCK 2=Ä
    NDRA TRUNKERING":"TRYCK 3=NYTT ORD":"TRYCK 4=LAGR
    ING"
500 CALL KEY(0,K,S):: S=0
510 IF K=49 THEN 480
520 IF K=50 THEN 400
530 IF K=51 THEN 320
540 IF K=52 THEN 650 ELSE 500
550 REM TRUNKERING SLUTET
560 CALL SPGET(O$,B$)
570 L=LEN(B$)-T-3
580 A$=SEG$(B$,1,2)&CHR$(L)&SEG$(B$,4,L)
590 RETURN
600 REM TRUNKERING BÖRJAN
610 CALL SPGET(O$,B$)
620 L=LEN(B$)-T-3
630 A$=SEG$(B$,1,2)&CHR$(L)&SEG$(B$,4+T,L)
640 RETURN
650 REM LAGRING
660 CALL CLEAR
670 PRINT "SPARA DEN BEHANDLADE ORD-":"DELEN I EN STR
    ÄNG (1-3).": : :
680 INPUT "STRÄNG NR: ":X
690 TR(X)=T
700 W$(X)=C$
710 FOR I=1 TO 300 :: NEXT I :: CALL CLEAR
720 DISPLAY AT(1,1):"TRYCK 1=NYTT ORD":"TRYCK 2=SAMMA
    NSTÄLLNING"
730 CALL KEY(0,K,S):: S=0
740 IF K=49 THEN 320
750 IF K=50 THEN 760 ELSE 730
760 REM SAMMANSTÄLLNING
770 CALL CLEAR
780 PRINT "HUR MÅNGA STRÄNGAR SKALL":"SÄTTAS SAMMAN?"
    : : :
790 PRINT "TRYCK 2 ELLER 3": : :
800 FOR I=1 TO 500 :: NEXT I
810 CALL KEY(0,K,S):: S=0
820 IF K=50 THEN 840
830 IF K=51 THEN 1130 ELSE 810
840 REM TVÅ STRÄNGAR
850 PRINT "VÄLJ MODELL:"
860 PRINT "TRYCK 1= 1 - 1":"TRYCK 2= 1 - 2":"TRYCK 3=
     2 - 1":"TRYCK 4= 2 - 2"
870 FOR I=1 TO 500 :: NEXT I
880 CALL KEY(0,K,S):: S=0
890 IF K=49 THEN 930
900 IF K=50 THEN 940
910 IF K=51 THEN 950
920 IF K=52 THEN 960 ELSE 880
930 P$=W$(1)&W$(1):: GOTO 970
940 P$=W$(1)&W$(2):: GOTO 970
950 P$=W$(2)&W$(1):: GOTO 970
960 P$=W$(2)&W$(2):: GOTO 970
970 GOSUB 990
980 GOTO 1080
990 CALL SAY(,P$)
1000 CALL CLEAR
1010 DISPLAY AT(8,1):"TRUNKERING (BYTES):"
1020 DISPLAY AT(10,1):"STRÄNG NR 1:   ";TR(1)
1030 DISPLAY AT(11,1):"STRÄNG NR 2:   ";TR(2)
1040 DISPLAY AT(12,1):"STRÄNG NR 3:   ";TR(3)
1050 PRINT "LÄTER ORDET BRA?":"TRYCK 1=EN GÅNG TILL":"
     TRYCK 2=ÄNDRA NÅGON DEL"
1060 PRINT "TRYCK 3=ÄNDRA ORDNINGSFÖLJD":"TRYCK S=SLUT
     "
1070 RETURN
1080 CALL KEY(0,K,S):: S=0
1090 IF K=49 THEN 970
1100 IF K=50 THEN 320
1110 IF K=51 THEN 760
1120 IF K=83 THEN 1440 ELSE 1080
1130 REM TRE STRÄNGAR
1140 CALL CLEAR
1150 PRINT "VÄLJ MODELL: ="
1160 PRINT "TRYCK 1= 1 - 2 - 3"
1170 PRINT "TRYCK 2= 1 - 3 - 2"
1180 PRINT "TRYCK 3= 2 - 1 - 3"
1190 PRINT "TRYCK 4= 2 - 3 - 1"
1200 PRINT "TRYCK 5= 3 - 1 - 2"
1210 PRINT "TRYCK 6= 3 - 2 - 1"
1220 FOR I=1 TO 500 :: NEXT I
1230 CALL KEY(0,K,S):: S=0
1240 IF K=49 THEN 1300
1250 IF K=50 THEN 1310
1260 IF K=51 THEN 1320
1270 IF K=52 THEN 1330
1280 IF K=53 THEN 1340
1290 IF K=54 THEN 1350 ELSE 1230
1300 P$=W$(1)&W$(2)&W$(3):: GOTO 1360
1310 P$=W$(1)&W$(3)&W$(2):: GOTO 1360
1320 P$=W$(2)&W$(1)&W$(3):: GOTO 1360
1330 P$=W$(2)&W$(3)&W$(1):: GOTO 1360
1340 P$=W$(3)&W$(1)&W$(2):: GOTO 1360
1350 P$=W$(3)&W$(2)&W$(1):: GOTO 1360
1360 GOSUB 990
1370 GOTO 1390
1380 CALL CLEAR
1390 CALL KEY(0,K,S):: S=0
1400 IF K=49 THEN 1360
1410 IF K=50 THEN 350
1420 IF K=51 THEN 760
1430 IF K=83 THEN 1440 ELSE 1390
1440 CALL CLEAR :: CALL SCREEN(9)
1450 DISPLAY AT(10,9):"S  L  U  T"
1460 DISPLAY AT(14,11):"19^ 84"
1470 CALL SAY("GOODBYE PARTNER")
1480 FOR I=1 TO 2000 :: NEXT I
1490 END
1500 REM TAL
1510 CALL SPGET("ÑNICE TRYÑ",NI$)
1520 L=LEN(NI$)-88
1530 NI$=SEG$(NI$,1,2)&CHR$(L)&SEG$(NI$,4,L)
1540 CALL SPGET("TO",TO$)
1550 L=LEN(TO$)-33
1560 TO$=SEG$(TO$,1,2)&CHR$(L)&SEG$(TO$,4,L)
1570 CALL SPGET("SEE",SE$)
1580 CALL SPGET("YOU",YO$)
1590 RETURN
```

LISTNING 2

```
100 REM ST AV START
110 TR=58
120 CALL SPGET("START",ST$)
130 L=LEN(ST$)-TR-3
140 ST$=SEG$(ST$,1,2)&CHR$(L)&SEG$(ST$,4,L)

200 REM EI AV EIGHT
210 TR=25
220 CALL SPGET("EIGHT",EI$)
230 L=LEN(EI$)-TR-3
240 EI$=SEG$(EI$,1,2)&CHR$(L)&SEG$(EI$,4,L)

300 REM TION AV INSTRUCTION
310 TR=60
320 CALL SPGET("INSTRUCTION",TI$)
330 L=LEN(TI$)-TR-3
340 TI$=SEG$(TI$,1,2)&CHR$(L)&SEG$(TI$,4+TR,L)

XXX REM HÄR LIGGER DITT PROGRAM MED GOSUB SATSER ELLER
CALL XYZ TILL CALL SAY(XX$)


400 REM UTTALAR ORDET
410 CALL SAY(,ST$&EI$&TI$)
420 RETURN
```

PRATORN och dess koder

av Lars-Erik Svahn

Koderna som styr pratorn

Det finns ett mikrospråk som styr pratorn, och med vilket man kan kontrollera denna, utan att behöva använda Extended-Basic-Instruktionerna CALL SAY och CALL SPGET. För att kunna programmera pratorn i detta mikrospråk fordras att du har någon av modulerna: Mini-Memory, Extended-Basic eller Editor/Assembler Om programmeringen ska ske i Basic (eller X-Basic), och du inte använder Mini-Memory, krävs att Expansionsminnet är inkopplat.

De viktigaste komponenterna i pratorn är 1 st TMS 5200 Voice Synthesis Processor (VSP). Via två adresser i 99'ans CPU-minne kan man skriva och läsa data i VSP/VSM:

SPRD = -28672 (HEX 9000) SPEACH-READ

SPWR = -27648 (HEX 9400) SPEACH-WRITE

Det är via SPWR som det går att programmera VSP med följande kommandon:

```
NOP1  (HEX 00)  CALL LOAD(SPWR,0)
(Ingen operation)

READ  (HEX 10)  CALL LOAD(SPWR,16)
(Initierar läsning från VSM)

NOP2  (HEX 20)  CALL LOAD(SPWR,32)
(Ingen operation)

R&B   (HEX 30)  CALL LOAD(SPWR,48)
(Läs och Sök-rutin för tabell i VSM)

LOAD X  (HEX 4X)  CALL LOAD(SPWR,X+64)
(Lagrar X i VSM's adressregister)

SPEAK  (HEX 50)  CALL LOAD(SPWR,80)
(Säger VSM-ord)

SPEXT  (HEX 60)  CALL LOAD(SPWR,96)
(Säger egna "ord")

RESET  (HEX 70)  CALL LOAD(SPWR,112)
(Avbryter SPEAK-kommandot)
```

En förklaring av dessa kommandon följer, men först några ord om VSM.

TMS 6100 är ett 128 kBit seriellt ROM-chip och VSP kan arbeta med 16 st sådana samtidigt utan ytterligare extern hårdvara. Kommunikationen till och från VSM sker via en 4 bitars adress-buss och en 8 bitars data-buss. Internt är VSM byte-orienterat och varje TMS 6100 ökar minnet med 16 kByte (128 kBit). Pratorns VSM på 32 kByte är uppdelat i två delar:

00000 - 013C2 Tabell-delen: Innehåller en lista över alla ord i VSM, adressen till dess data i ord-data-delen m.m.

013C3 - 07FFF Ord-data-delen: Innehåller de data som bestämmer hur ordet skall uttalas.

Tabell-delen är uppdelad i poster, en post för varje ord. Varje post är uppdelad i sex fält av olika längd. Posten för ordet 'MORE' ser ut så här:

```
ASCII-Längd  1 Byte
(Innehållande  HEX 4)

ASCII-Kod  4 Byte
(Innehållande  HEX 4D,4F,52,45 (M,O,R,E))

UPP-pekare  2 Byte
(Innehållande  HEX 04B6)

NED-pekare  2 Byte
(Innehållande  HEX 0EB5)

ORD-DATA-pekare  3 Byte
(Innehållande  HEX 004642)

ORD-DATA-längd  1 Byte
(Innehållande  HEX 51)
```

ASCII-Kod-Fältet är det enda med variabel längd, givet av innehållet i föregående fält. Ned-pekaren pekar nedåt med alfabetisk nyckel i ett binärt träd och upp-pekaren pekar uppåt i samma träd av poster. I posten ser man att Data-Blocket för 'MORE' är HEX 51 byte långt och börjar på adressen HEX 04642 i VSM.

Ord-Data-Delen är uppdelad i datablock, ett block för varje ord. Sista bytet i varje block innehåller (någonstans) den binära koden 1111, vilket är den STOP-kod som markerar slutet på data-blocket.

Åter till kommandona !

Kommandona LOADX, SPEAK och RESET:

Operationen LOADX lagrar nybblen X i VSM's adressegister (1 nybble = 4 bitar = 1/2 byte). Med denna operation kan man lagra ett ords data-blocks-adress, nybble för nybble (5 st) i VSM's adressregister. Men man skall börja "från höger" med den minst signifikanta nybblen. För att lagra adressen till 'MORE' (HEX 04642) skriver man således: LOAD 2, LOAD 4, LOAD 6. LOAD 4, LOAD 0. Detta åstadkommes genom att lagra byten HEX 42, HEX 44, HEX 46, HEX 44, HEX 40 i denna ordning på CPU-adressen SPWR. När adressen ovan är lagrad och man beordrar operationen SPEAK, d.v.s. lagrar bytet HEX 50 i SPWR, så säger pratorn 'MORE'. CPU väntar inte tills pratorn är färdig utan fortsätter ganska snart med sitt eget program, till skillnad från då CALL SAY har anropats från X-Basic. Förutom att detta snabbar upp programmen, har det den fördelen att man (som i Parsec) kan avbryta pratorn mitt i ett ord. Detta avbrott sker med kommandot RESET: HEX 70 lagras i SPWR. Pratorn tystnar då tvärt och utan missljud. Följande Basic (X-Basic) program demonstrerar dessa kommandon genom att börja säga 'TEXAS INSTRUMENT' för att plötsligt avbryta tvärt.

```
100 CALL INIT
110 HEX$="123456789ABCDEF"
120 SPWR= -27648
130 LOAD=64
140 SPEAK=80
150 RESET=112
160 REM HEX KOD I STR-VAR
170 KOD$="06696"
180 REM LADDAR VSM-ADRESSEN
190 FOR I=5 TO 1 STEP -1
200 CALL LOAD(SPWR,LOAD+POS
    (HEX$,SEG$(KOD$,I,1),1))
210 NEXT I
220 CALL LOAD(SPWR,SPEAK)
230 REM PAUS
240 FOR I=1 TO 255
250 NEXT I
260 CALL LOAD(SPWR,RESET)
270 END
```

VARNING ! CALL INIT ändrar i expansionsminnet och i MINI-MEMORY. Behövs ej med MINI-MEMORY Modulen !

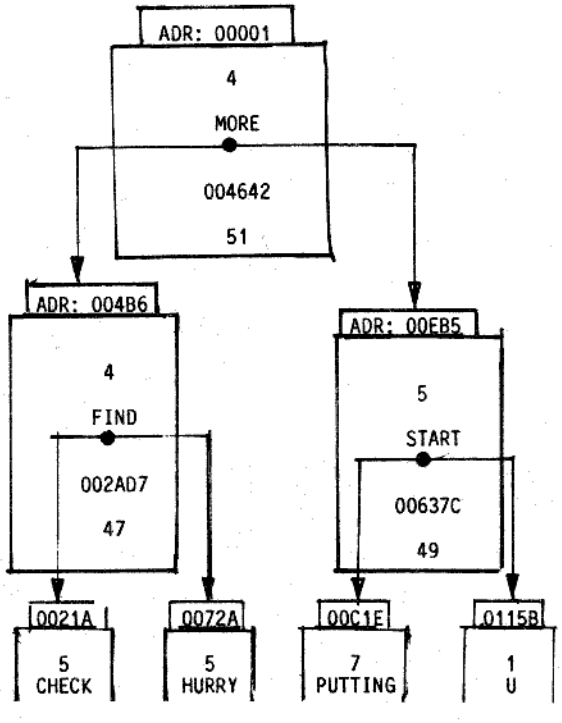
Kommandona R&B, NOP1 och NOP2:

R&B är en Read-and-Branch instruktion. Den läser två byte som ett 16 bitars-ord i VSM, med början från adressen i adressregistret. Den skippar de två mest signifikanta bitarna i ordet och ersätter de 14 minst signifikanta bitarna i adressregistret med de återstående bitarna. Denna instruktion används för att söka i binära träd, liknande det som finns i tabell-delen av VSM. Men detta kommando kan ej användas med pratorn eller andra multipla VSM-system. Enligt TMS 5200 manualen uppträder "bus-contention" i sådana fall.

Jag har provat det men misslyckats i såväl Basic som i Forth och Assembler. NOP1 och NOP2 är "No-OPerations" och är nog oanvändbara de också.

Kommandot READ och det binära trädet i tabell-delen.

När READ beordrats (HEX 10 i SPWR) kan det byte i VSM, vars adress står i adressregistret, läsas från SPRD. När ett byte har lästs, ökas innehållet i adressregistret med 1. Och hela förfarandet kan upprepas med READ och läsning. Innan läs proceduren påbörjas måste alltså rätt adress lagras i adressregistret med LOAD-kommandot enligt tidigare. Med READ kan man få fram ett ords tal-data, byte för byte, om man känner till adressen där dessa börjar. Gör man inte det kan man söka den i tabell-delen, där det finns ett binärt träd av poster, med ordet 'MORE':s post som rot:



Det finns två möjliga vägar att gå om man vill använda de i den här artikeln beskrivna kommandona för att få pratorn att prata i Basic:

1. Man gör ett program (t.ex. i Basic) som söker i det binära trädet och skriver ut en ordlista över alla orden och dess adresser i taldata-delen av VSM som man sedan använder med kommandot SPEAK. (Man kan förstås också använda den ordlista som finns i Assembler manualen.)

2. Man gör ett Assembler program som söker i trädet och som från Basic kan anropas med t.ex. CALL LINK("SPEAK,"MORE"). Nedan finns ett Basic program som gör detta, men tyvärr är det för långsamt.

Kommandot SPEXT

De HEX 51 bytes som finns med början från adress 04642 i VSM, kan läsas till CPU- eller VDP-RAM enligt ovan. Där kan man sedan trunkera bort bytes bakifrån eller från vissa ställen framifrån, mata in den trunkerade listan genom att omväxlande skriva HEX 60 (SPEXT) och aktuellt byte i listan, på adress SPWR. Om man trunkerat riktigt säger då pratorn en del av ordet 'MORE'. Jag har inte provat detta eftersom jag tror att Basic är för långsamt för att hinna med pratorn. (Pratorn börjar prata när ett visst antal byte på detta sätt lästs till VSP's FIFO-buffert.
OBS! RESET-kommandot kan ej avbryta SPEAK-EXTERNAL-kommandot.

Följande program demonstrerar hur READ-kommandot kan användas för att söka upp adressen till ett givet ord, samt kanske hur man kan handskas med HEXADECIMAL-NYBBLE-BYTE-ADRESS-PROBLEMATIKEN. Programmet är för långsamt för att kunna ersätta CALL SAY.

```
100 CALL INIT  * Behövs ej med Mini-Memory
110 SPWR= -27648  * Adressen för att skriva i VSP
120 SPRD= -28672  * Adressen för att läsa från VSM
130 READV=16
140 LOAD=64
150 SPEAK=80
160 INPUT "ORD: ":WANT$
170 BH=0  * VSM-adressens mest signifikanta byte
(bortsett från 0)
180 BL=1  * VSM-adressens minst signifikanta byte
190 GOSUB 460  * Laddar VSM-adressen
200 GOSUB 550  * Läser VSM-byte (funna ordets längd)
210 IF B=170 THEN 160  * Ordet fanns ej om B=170 (VSM-
adr. 0000, HEX AA)

220 LET FIND$=""
230 FOR I=1 TO B  * Sätter FIND$= det funna ordet
240 GOSUB 550
250 LET FIND$=FIND$&CHR$(B)
260 NEXT I
270 IF WANT$=FIND$ THEN 360  * Testar om det är klart
280 IF WANT$<FIND$ THEN 310  * Testar om önskade ordet
kommer före funna ordet
290 GOSUB 550  * Stegar fram 2 bytes
300 GOSUB 550  *  -    "   "     -
310 GOSUB 550  * Läser nästa ords adress (mest
signifikanta bytet)
320 LET BH=B
330 GOSUB 550  * (Minst signifikanta bytet)
340 LET BL=B
350 GOTO 190  * Börja om och tag fram nästa post
360 FOR I=1 TO 6 * Rätt post hittad ! Stegar fram 6
byte för att
370 GOSUB 550    * hitta ord-data-adressen
380 NEXT I
390 LET BH=B   * (Mest signifikanta bytet)
400 GOSUB 550  * (Minst signifikanta bytet)
410 LET BL=B
420 GOSUB 460  * Laddar ord-data-adressen
430 CALL LOAD(SPWR,SPEAK)  * Uttalar ordet
440 GOTO 160  * Färdigt, klart för nästa ord
450 REM LOAD ADRESS  * Load Subrutinen *
460 LET N=INT(BL/16)  * BL's mest signifikanta nybble
470 CALL LOAD(SPWR,LOAD+BL-16*N)  * Laddar BL's minst
sign. nybble
480 CALL LOAD(SPWR,LOAD+N)  * Laddar BL's mest sign.
nybble
490 LET N=INT(BH/16)  * BH's mest sign. nybble
500 CALL LOAD(SPWR,LOAD+BH+16*N)  * Laddar BH's minst
sign. nybble
510 CALL LOAD(SPWR,LOAD+N)  * Laddar BH's mest sign.
nybble
520 CALL LOAD(SPWR,LOAD)  * Laddar första nybblen (0)
530 RETURN
540 REM READ BYTE  * READ Subrutin *
550 CALL LOAD(SPWR,READV)  * Initierar läsning
560 CALL PEEK(SPRD,B)  * Läser byte till B
570 RETURN
580 END
```

TYPSNITT

Som Ni sett i olika nr av PB har vi använt olika skrivmaskiner för utskrifter av texter och programlistningar. Detta har satt myror i huvudet på många vilka inte haft en aning om att ASCII-koderna skiftar mellan länder och även mellan maskiner med samma tillverkare.
Nedan har vi samlat ett antal ASCII-listningar och från vilka vi senare ska ta bara de specialtecken som ändras för att ha med dem vid listningar.

ASCII-KODER OCH MOTSVARANDE TECKEN: SVERIGE,Sense PO1
Skönskrivare med typhjul PICA,12 tkn/tum

35	36	64	91	92	93	94	96	123	124	125	126
#	$	§	Ä	Ö	Å	^	`	ä	ö	å	ü

```
100 ! ASCII-KODER FÖR OLIKA TECKENUPPSÄTTNINGAR
110 OPEN #9:"RS232.BA=2400.PA=N",OUTPUT
120 IMAGE "####"
130 DATA 35,36,64,91,92,93,94,96,123,124,125,126
140 PRINT #9:CHR$(27);"F";!12 TKN/TUM
150 ! PRINT #9:CHR$(27);"R";CHR$(0);! VAL AV TECKENUPPSÄTTNING EPSON
160 PRINT #9:CHR$(27);"W";!DUBBEL SKRIFT EPSON
170 PRINT #9:CHR$(27);"9";CHR$(10)! VÄNSTERMARGINAL
180 PRINT #9:"ASCII-KODER OCH MOTSVARANDE TECKEN: SVERIGE,Sense PO1"
190 PRINT #9:"Skönskrivare med typhjul PICA,12 tkn/tum"
200 FOR I=1 TO 12 :: READ A :: PRINT #9,USING 120:A;:: NEXT I
210 PRINT #9
220 RESTORE
230 FOR I=1 TO 12 :: READ A :: PRINT #9:TAB(I*4);CHR$(A);:: NEXT I
240 CLOSE #9
```

SKRIVREGLER

från Claes S.

Som Ni ser i detta nummer av Programbiten så saknar vi skrivregler.....
I vanliga fall försöker vi hålla en spaltbredd av 11,5 cm vilket ger 46 tecken vid 10 tecken per tum, eller 55 tecken vid 12 tecken per tum.
Nu har vi emellertid erhållit material skrivet med 55 teckens bredd meeeeen.. med hemskrivmaskinens bredd av 11 tecken per tum. I några fall är det inknappat och utkört med 10 tpt eller 12 tpt, dock Tips och Tricks blev så mycket specialtecken så den tog vi som den var. Sen är det FORTH-skärmarna som håller 64 teckens bredd där vi nu haft en textmassa att kunna anpassa till smalare spalt, det blir dock en del extra knappande. Ni kommer att se resultat av det i nästa nummer, 5 sid.

Det här med teckenbredder, spaltbredder mm får dock inte hindra ER från att skriva bidrag till tidningen, men hänsyn till dem underlättar redaktionens arbete. Använd gärna textinskrivningsprogrammet P:TEXTIN i förra numret, -84-3, så kan vi erhålla texten på kassett eller diskett.

Nedan ser Ni en utskrift från maskin vilken med hjälp av kontrollkoder kan ställas in på olika teckenbredder. Man har 120 punkter per tum och delar 120 med ett tal för att erhålla 'pitch width' 120/8=15, 120/9=13.3 120/10=12, 120/11=11 och 120/12=10, max är 120/60=2 och min 120/1=120 samt 120/0= proportionell text.

```
15 pitch
0000011111222223333344444555555
.....
135791357913579135791357913579

13.3 pitch
00000111111222222333333444444555555
.....
135791357913579135791357913579

12 pitch
000001111112222223333334444445555 55
.....
135791357913579135791357913579

11 pitch
0000011111222223333334444445 55555
.....
135791357913579135791357913579 1 3 5 7 9

10 pitch
0 0 0 0 0 011 1 1 1 122 2 2 2 233 3 3 3 344 4 4 4 455 5 5 5 56
.....
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

ASCII-KODER OCH MOTSVARANDE TECKEN: SVERIGE,EPSON RX

35	36	64	91	92	93	94	96	123	124	125	126
#	¤	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü

ASCII-KODER OCH MOTSVARANDE TECKEN: USA,EPSON RX

35	36	64	91	92	93	94	96	123	124	125	126
#	$	@	[	\	]	^	`	{	¦	}	~

ASCII-KODER OCH MOTSVARANDE TECKEN: DANMARK1,EPSON RX

35	36	64	91	92	93	94	96	123	124	125	126
#	$	@	Æ	Ø	Å	^	`	æ	ø	å	~

Cannon PW-80

ASCII-KODER OCH DERAS TECKEN

32 =	64 = @	96 = `
33 = !	65 = A	97 = a
34 = "	66 = B	98 = b
35 = #	67 = C	99 = c
36 = $	68 = D	100 = d
37 = %	69 = E	101 = e
38 = &	70 = F	102 = f
39 = '	71 = G	103 = g
40 = (	72 = H	104 = h
41 =)	73 = I	105 = i
42 = *	74 = J	106 = j
43 = +	75 = K	107 = k
44 = ,	76 = L	108 = l
45 = -	77 = M	109 = m
46 = .	78 = N	110 = n
47 = /	79 = O	111 = o
48 = 0	80 = P	112 = p
49 = 1	81 = Q	113 = q
50 = 2	82 = R	114 = r
51 = 3	83 = S	115 = s
52 = 4	84 = T	116 = t
53 = 5	85 = U	117 = u
54 = 6	86 = V	118 = v
55 = 7	87 = W	119 = w
56 = 8	88 = X	120 = x
57 = 9	89 = Y	121 = y
58 = :	90 = Z	122 = z
59 = ;	91 = Ä	123 = ä
60 = <	92 = Ö	124 = ö
61 = =	93 = Å	125 = å
62 = >	94 = ^	126 = ~
63 = ?	95 = _	127 = ■

Diablo Scandia Elite 12

ASCII-KODER OCH DERAS TECKEN

32 =	64 = '	96 = `
33 = ¨	65 = A	97 = a
34 = "	66 = B	98 = b
35 = ñ	67 = C	99 = c
36 = $	68 = D	100 = d
37 = %	69 = E	101 = e
38 = &	70 = F	102 = f
39 = ´	71 = G	103 = g
40 = (	72 = H	104 = h
41 =)	73 = I	105 = i
42 = *	74 = J	106 = j
43 = +	75 = K	107 = k
44 = ,	76 = L	108 = l
45 = -	77 = M	109 = m
46 = .	78 = N	110 = n
47 = /	79 = O	111 = o
48 = 0	80 = P	112 = p
49 = 1	81 = Q	113 = q
50 = 2	82 = R	114 = r
51 = 3	83 = S	115 = s
52 = 4	84 = T	116 = t
53 = 5	85 = U	117 = u
54 = 6	86 = V	118 = v
55 = 7	87 = W	119 = w
56 = 8	88 = X	120 = x
57 = 9	89 = Y	121 = y
58 = :	90 = Z	122 = z
59 = ;	91 = Ä	123 = ä
60 = <	92 = Ö	124 = ö
61 = =	93 = Å	125 = å
62 = >	94 = ^	126 = ñ
63 = ?	95 = _	127 =

Tangent-definitioner

TANGENTER OCH DERAS ASCII-KODER

I anslutning till artikeln på förra sidan publicerar vi här ett Basicprogram från programbanken. Det skriver ut vilken tangent du har tryckt ner och ASCII-koden för den tangenten. Du kan alltså undersöka vilka koder ex. CNTRL-tangenterna och FCTN-tangenterna har.

Hälsningar Redaktionen.

```
100 REM  CALL KEY, LFñ 4346, BY: BERNARD FALKIN
110 REM    T.I. YOU CAN'T
120 REM FOOL US ANYMORE"
130 CALL CLEAR
140 PRINT "THIS PROGRAM DEFINES ALMOST":"ALL KEYS ON
    THE KEYBOARD."
150 PRINT :"EXAMPLE:":"DID YOU KNOW KEYñ 13 WAS":"YOU
    R ""ENTER"" KEY?"
160 PRINT "THIS WILL CURE YOUR":"CURIOSITY OF HOW T.I
    . DOES IT IN THEIR MODULES."
170 PRINT "BUT, BETTER THAN THAT YOU":"CAN DO IT TOO"
    ""
180 PRINT :"HOW THIS WORKS:":"PRESS ANY KEY LIKE ""L"
    " IT":"WILL PRINT ""L"" AND THE KEY":"NUMBER ""76
    """."
190 PRINT "OR, TRY SHIFT ""G""(INS) AND":"SEE WHAT HA
    PPENS."
200 PRINT : : :
210 PRINT "PRESS PROC'D TO BEGIN."
220 CALL KEY(0,K,S)
230 IF K=12 THEN 240 ELSE 220
240 CALL CLEAR
250 PRINT TAB(7);"PROGRAM BY:";TAB(34);"BERNARD FALKI
    N"
260 PRINT : : : : : : : : : : : : : : : : :
270 CALL KEY(0,K,S)
280 IF S<1 THEN 270
290 IF K<>32 THEN 320
300 PRINT "SPACE BAR",K
310 GOTO 270
320 IF K<16 THEN 350
330 PRINT CHR$(K),K
340 GOTO 270
350 ON K GOSUB 370,390,410,430,450,470,490,510,530,55
    0,570,590,610,630,650
360 GOTO 270
370 PRINT "A(AID)",K
380 RETURN
390 PRINT "C(CLEAR)",K
400 RETURN
410 PRINT "F(DELETE)",K
420 RETURN
430 PRINT "G(INS)",K
440 RETURN
450 PRINT "NOTHING.....)",K
460 RETURN
470 PRINT "R(REDO)",K
480 RETURN
490 PRINT "T(ERASE)",K
500 RETURN
510 PRINT "S(LEFT)",K
520 RETURN
530 PRINT "D(RIGHT)",K
540 RETURN
550 PRINT "X(DOWN)",K
560 RETURN
570 PRINT "E(UP)",K
580 RETURN
590 PRINT "V(PROC'D)",K
600 RETURN
610 PRINT "ENTER",K
620 RETURN
630 PRINT "W(BEGIN)",K
640 RETURN
650 PRINT "Z(BACK)",K
660 RETURN
670 END
```

■

RAPPORT FRÅN

Medlemsträff

Av Anders Persson.
Artikeln bearbetad av Göran Nygren.

Vi fick ett brev från Anders Persson där han berättar om medlemsmötet i Lund. Det kom 38 personer.
Tre danskar hade åkt över sundet samt medlemmar från Småland hade också kommit till mötet.
Anders höll ett föredrag i ca 1.5 timma om det mesta från Extended-Basic till Multiplan och Pascalsystemet. Intresset var aktivt och frågorna många. Vad frågades det om och vad diskuterades ?
Jo, utrustning till 99:an. Det kom frågor om skrivare, RS232, parallellinterface, expansionsboxar osv.
Lennart Thelander visade sitt specialbygge av minnes-expansion och fick genast kuta iväg och kopiera fler blad med kopplingsanvisningar vilka han hade ritat. Mini-Memory är tydligen en eftertraktad Modul.
Telekommunikation frågades och svarades det om. Överhuvudtaget var det huvudsakliga intresset koncentrerat något över Extended-Basic men inte så långt som till Pascal. Intresset för Forth var svalt. Detta beror nog på konstnaden att införskaffa allt som behövs för dessa system. TI-Writer och särskilt Multiplan kom det frågor om. Några ville ha mer information om de avancerade finesserna i Multiplan, men skrivet på svenska.
Styrning och reglering av annan elektronisk utrustning fanns det intresse för.

Sammanfattningsvis:
På hårdvarusidan tycks medlemmar vara intresserade av allt som de kan tänka sig och mer därtill. På mjukvarusidan fanns intresset från Basic och nästan upp till Pascal och Forth. Två 99:or stod i lokalen med det mesta av tillbehören.

Anders skriver i brevet några tankar om Assembler och Assemblerprogrammering vilket i korthet går ut på följande:
Börja med att publicera ett färdigt Assemblerprogram. Det ska vara lagom stort och publiceras i sin helhet. Det skall kunna anropas från Extended-Basic med CALL LOAD satser och innehålla parameteröverföring till/från Extended-Basic. Det skall ge exempel på god programmeringskonst dvs. strukturerat och modulariserat (JSP).
Det skall göra något nyttigt som blir svårt och långsamt helst helt omöjligt i Basic. Sedan skall det i kommande nummer av tidningen finnas en detaljerad förklaring av vad programmet gör och varför det är skrivet på detta speciella sätt. Vad varje instruktion gör kan alla engelskkunniga läsa mer om i Editor/Assembler manualen. Det är vad man skall använda instruktionerna till som är det väsentliga att förklara. Förklaringen skall gå igenom allt. Från den första kommentar-asterisken till det sista END:et. Förklaringarna skall vara enkla och lätta att förstå för den oinvigde. Alltså inte en massa svengelska och "inne" snack. Förklara gärna en liten bit i taget av programmet. Använd gärna blockscheman och instruktiva teckningar över hur programmet fungerar.

Vi tackar Anders för dessa tankar och anammar detta. Redaktören håller fullkomligt med Anders i detta och tycker för övrigt att det mesta som skrivs i datortidningar om Assembler är luddig innejargong.
"Vi som vet hur allt fungerar och inte behöver förklara det för andra" osv. Det här blir en utmaning för dig som kan Assembler att bevisa att du verkligen kan förklara det här med koder och mnenonics.

Hälsningar Redaktionen.

Som har assemblerat det här numret.

■

TI 59

STORCIRKELNAVIGERING

Pargas, den 12 april 1984

Hej Claes.

Olika program som man med olika möda knåpat ihop värderas självfallet olika mycket. Det jag nu sänder dig hör i min egen skala till 10 i topp. I marinmodulen NG-26 beräknas latituden då utseglingspunkt och inseglingspunkt samt longitudsdifferensen inmatats. Vidare fås storcirkelavståndet. Nu är felet det att avståndet mellan två på varandra följande punkter är olika (meridianerna konvergerar eller divergerar). Detta är i praktiskt hänseende olämpligt.

Jag såg nyligen i TV hur den salta skepparen hade kontakt med en motordriven följebåt som hade en större dator ombord som kunde sända den optimerade kursen till segelbåten vilken självfallet var utrustad med satellitnavigeringssystem.

Navigatörens huvudsakliga problem syntes härvid bestå i att se till att cognacen räckte till över atlanten. Ej annan föda att förglömma.

Nu får du se vad 59-an gör.

E = + och W = -

E': Init ger ledtext.
A : Lat. (DD.MMss) Utseglingspunkt
A': Long " "
B : Lat. " Inseglingspunkt
B': Long " "

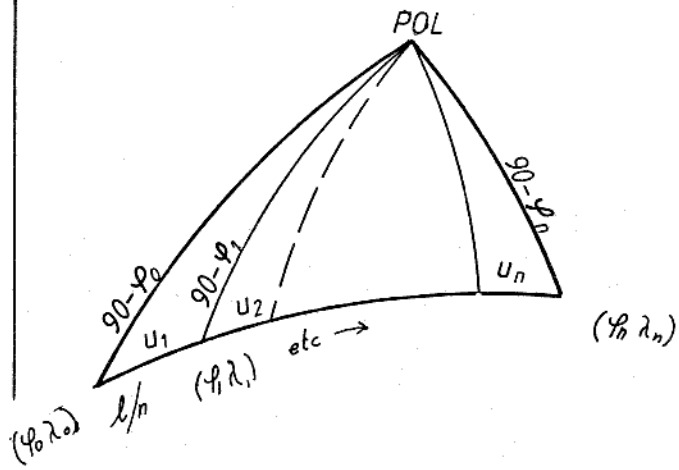
Efter B' utskrives storcirkelavståndet. Slå in antalet önskade ekvidistanta storcirkelbågar med C (1 - N) och du får 1:sta inseglingspunktens Lat., Long., kordalängd och kompasskurs. Sedan följer hela sviten av grupper om fyra för en navigatör så viktiga värden tills hela storcirkeln är genomgången. Text följer med varje värde.

Jag hoppas programmet tilltalar någon icke datoriserad seglare. Jag skall ge formlerna man behöver så att någon 99-fantast kan utnyttja dem och komplettera med grafik (settdottare obs).

De fyra första formlerna är den sfäriska geometrins cosinusteorem tillämpat på den vänstra triangeln (se fig). De undre är de som används vid medellatitudmetoden.

Programmet duger lika väl vid aviation. Rakt norrut skriver programmet rena smörjan såvida man inte ökar den ena longituden med ett epsilon tillskott (t.ex 0.1")

1,3	PROGRAMMERARE: HOFMAN			2
4 OP 17	STORCIRKELNAVIGERING			
LONG IN	LONG UT			INIT
LAT IN	LAT UT	ANTAL N		



$$\cos\ell = \sin\varphi_o \sin\varphi_n + \cos\varphi_o \cos\varphi_n \cos(\lambda_n - \lambda_o) \quad \text{(ger avst.)}$$

$$\cos u_1 = \frac{\sin\varphi_n - \sin\varphi_o \cos\ell}{\cos\varphi_o \sin\ell} \quad \text{(hjälpstorhet)}$$

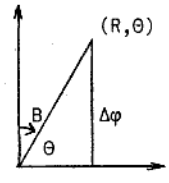
$$\sin\varphi_1 = \sin\varphi_o \cos\frac{\ell}{n} + \cos\varphi_o \sin\frac{\ell}{n} \cos u_1 \quad \text{(ger lat.)}$$

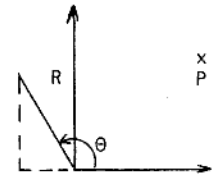
$$\cos(\lambda_1 - \lambda_o) = \frac{\cos\frac{\ell}{n} - \sin\varphi_o \sin\varphi_1}{\cos\varphi_o \cos\varphi_1} \quad \text{(ger long.)}$$

$\lambda_1 = \lambda_o + \Delta$ då $\theta \leq 90$ (polära koord.)

$\lambda_2 = \lambda_o - \Delta$ då $\theta \leq 90$ "

Kurs och Distans





$-90 \leq \theta \leq 90$ $\qquad$ $90 < \theta < 270$

$B = 90 - \theta$ $\qquad$ $B = 450 - \theta$ (kurs)

$$\varphi_m = \frac{\varphi_o + \varphi_n}{2} \quad \text{(medellatituden)}$$

$$dep = \Delta\lambda \cos\varphi_m * 60 \quad \text{(departur)}$$

$$\Delta\varphi = (\varphi_n - \varphi_o) * 60 \quad \text{(lat. diff.)}$$

$$\ell_k = R \quad \text{(kordalängd)}$$

TI 59

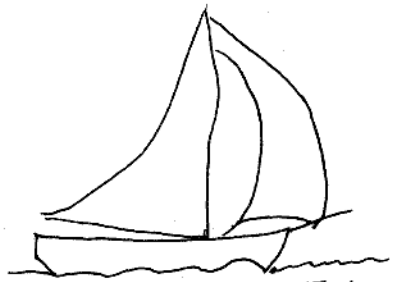
```
        ↑E'
STORCIRKELNAVIGERING

  LAT          ↑A          LAT          ↑A
                59.                      60.          LAT
 LONG        ↑A'          LONG        ↑A'
                17.          LONG        22.          LONG
  LAT          ↑B          LAT          ↑B
                60.          LAT         59.          LAT
 LONG        ↑B'          LONG        ↑B'
                22.          LONG        17.          LONG

              163.6         DIST        163.6         DIST
ANTAL          N    ↑C     ANTAL         N    ↑C
                2.           N            2.           N

          59.314     LAT             59.314     LAT
          19.278     LONG            19.278     LONG
            81.8     KORD              81.8     KORD
            67.4     KURS             249.6     KURS

          60.000     LAT             59.000     LAT
          22.000     LONG            17.000     LONG
            81.8     KORD              81.8     KORD
            69.6     KURS             247.4     KURS
```



Navigare Necesse Est /claes

```
  LAT          ↑A
                60.          LAT
 LONG        ↑A'
                22.          LONG
  LAT          ↑B
                58.          LAT
 LONG        ↑B'
                12.          LONG

              331.1         DIST
ANTAL          N    ↑C
                3.           N

          59.252     LAT
          18.323     LONG
           110.4     KORD
           251.6     KURS

          58.480     LAT
          15.122     LONG
           110.4     KORD
           248.7     KURS

          58.000     LAT
          12.000     LONG
           110.4     KORD
           245.9     KURS
```

```
 271337.  17  LAT
27323122.  18  LONG
16243637.  19  DIST
1331371327.20  NTAL
26323516.  21  KORD
26413536.  22  KURS
```

```
000 76 LBL
001 34 FX
002 53  (
003 53  (
004 43 RCL
005 15  15
006 65  ×
007 01  1
008 00  0
009 00  0
010 54  )
011 42 STO
012 16  16
013 59 INT
014 42 STO
015 15  15
016 53  (
017 53  (
018 43 RCL
019 16  16
020 22 INV
021 59 INT
022 65  ×
023 05  5
024 55  ÷
025 03  3
026 85  +
027 43 RCL
028 15  15
029 54  )
030 55  ÷
031 01  1
032 00  0
033 00  0
034 54  )
035 54  )
036 92 RTN
037 76 LBL
038 19 D'
039 32 X:T
040 69 OP
041 04  04
042 32 X:T
043 69 OP
044 06  06
045 69 OP
046 00  00
047 92 RTN
048 76 LBL
049 10 E'
050 69 OP
051 00  00
052 06  6
053 00  0
054 01  1
055 07  7
056 06  6
057 05  5
058 69 OP
059 02  02
060 69 OP
061 05  05
062 03  3
063 06  6
064 03  3
065 07  7
066 03  3
067 02  2
068 03  3
069 05  5
070 01  1
071 05  5
072 69 OP
073 01  01
074 02  2
075 04  4
076 03  3
077 05  5
078 02  2
079 06  6
080 01  1
081 07  7
082 02  2
083 07  7
084 69 OP
085 02  02
086 03  3
087 01  1
088 01  1
089 03  3
090 04  4
091 02  2
092 02  2
093 04  4
094 02  2
095 02  2
096 69 OP
097 03  03
098 01  1
099 07  7
100 03  3
101 05  5
102 02  2
103 04  4
104 03  3
105 01  1
106 02  2
107 02  2
108 69 OP
109 04  04
110 69 OP
111 05  05
112 69 OP
113 00  00
114 98 ADV
115 02  2
116 07  7
117 01  1
118 03  3
119 03  3
120 07  7
121 42 STO
122 17  17
123 02  2
124 07  7
125 03  3
126 02  2
127 03  3
128 01  1
129 02  2
130 02  2
131 42 STO
132 18  18
133 01  1
134 06  6
135 02  2
136 04  4
137 03  3
138 06  6
139 03  3
140 07  7
141 42 STO
142 19  19
143 01  1
144 03  3
145 03  3
146 01  1
147 03  3
148 07  7
149 01  1
150 03  3
151 02  2
152 07  7
153 42 STO
154 20  20
155 02  2
156 06  6
157 03  3
158 02  2
159 03  3
160 05  5
161 01  1
162 06  6
163 42 STO
164 21  21
165 02  2
166 06  6
167 04  4
168 01  1
169 03  3
170 05  5
171 03  3
172 06  6
173 42 STO
174 22  22
175 04  4
176 69 OP
177 17  17
178 76 LBL
179 66 PAU
180 06  6
181 00  0
182 01  1
183 03  3
184 69 OP
185 03  03
186 76 LBL
187 68 NOP
188 43 RCL
189 17  17
190 76 LBL
191 69 OP
192 69 OP
193 01  01
194 29 CP
195 32 X:T
196 69 OP
197 05  05
198 69 OP
199 00  00
200 91 R/S
201 76 LBL
202 11  A
203 19 D'
204 88 DMS
205 42 STO
206 00  00
207 06  6
208 00  0
209 01  1
210 03  3
211 06  6
212 05  5
213 69 OP
214 03  03
215 76 LBL
216 60 DEG
217 43 RCL
218 18  18
219 61 GTO
220 69 OP
221 76 LBL
222 16 A'
223 19 D'
224 88 DMS
225 42 STO
226 01  01
227 06  6
228 00  0
229 01  1
230 04  4
231 69 OP
232 03  03
233 61 GTO
234 68 NOP
235 76 LBL
236 12  B
237 19 D'
238 88 DMS
239 42 STO
240 02  02
241 06  6
242 00  0
243 01  1
244 04  4
245 06  6
246 05  5
247 69 OP
248 03  03
249 61 GTO
250 60 DEG
251 76 LBL
252 17 B'
253 19 D'
254 88 DMS
255 42 STO
256 03  03
257 98 ADV
258 43 RCL
259 19  19
260 69 OP
261 04  04
262 43 RCL
263 00  00
264 38 SIN
265 65  ×
266 43 RCL
267 02  02
268 38 SIN
269 85  +
270 43 RCL
271 00  00
272 39 COS
273 65  ×
274 43 RCL
275 02  02
276 39 COS
277 65  ×
278 53  (
279 43 RCL
280 03  03
281 75  -
282 43 RCL
283 01  01
284 54  )
285 42 STO
286 04  04
287 39 COS
288 95  =
289 22 INV
290 39 COS
291 42 STO
292 05  05
293 65  ×
294 06  6
295 00  0
296 95  =
297 58 FIX
298 01  01
299 69 OP
300 06  06
301 69 OP
302 00  00
303 22 INV
304 58 FIX
305 43 RCL
306 20  20
307 69 OP
308 01  01
309 06  6
310 00  0
311 01  1
312 05  5
313 69 OP
314 03  03
315 03  3
316 01  1
317 69 OP
318 02  02
319 29 CP
320 32 X:T
321 69 OP
322 05  05
323 69 OP
324 00  00
325 91 R/S
326 76 LBL
327 13  C
328 42 STO
329 06  06
330 19 D'
331 98 ADV
332 43 RCL
333 05  05
334 55  ÷
335 43 RCL
336 06  06
337 95  =
338 42 STO
339 06  06
340 76 LBL
341 44 SUM
342 43 RCL
343 02  02
344 38 SIN
345 75  -
346 43 RCL
347 00  00
348 38 SIN
349 65  ×
350 43 RCL
351 05  05
352 39 COS
353 95  =
354 55  ÷
355 43 RCL
356 00  00
357 39 COS
358 55  ÷
359 43 RCL
360 05  05
361 38 SIN
362 95  =
363 42 STO
364 07  07
365 43 RCL
366 00  00
367 38 SIN
368 65  ×
369 43 RCL
370 06  06
371 39 COS
372 85  +
373 43 RCL
374 00  00
375 39 COS
376 65  ×
377 43 RCL
378 06  06
379 38 SIN
380 65  ×
381 43 RCL
382 07  07
383 95  =
384 22 INV
385 38 SIN
386 42 STO
387 08  08
388 22 INV
389 88 DMS
390 42 STO
391 15  15
392 71 SBR
393 34 FX
394 32 X:T
395 22 INV
396 58 FIX
397 43 RCL
398 17  17
399 69 OP
400 04  04
401 32 X:T
402 58 FIX
403 03  03
404 69 OP
405 06  06
406 43 RCL
407 06  06
408 39 COS
409 75  -
410 43 RCL
411 08  08
412 38 SIN
413 65  ×
414 43 RCL
415 00  00
416 38 SIN
417 95  =
418 55  ÷
419 43 RCL
420 08  08
421 39 COS
422 55  ÷
423 43 RCL
424 00  00
425 39 COS
426 95  =
427 22 INV
428 39 COS
429 50 IX1
430 42 STO
431 09  09
432 43 RCL
433 04  04
434 32 X:T
435 43 RCL
436 02  02
437 75  -
438 43 RCL
439 00  00
440 95  =
441 22 INV
442 37 P/R
443 42 STO
444 10  10
445 09  9
446 00  0
447 32 X:T
448 43 RCL
449 10  10
450 22 INV
451 77 GE
452 85  +
453 43 RCL
454 01  01
455 75  -
456 61 GTO
457 75  -
458 76 LBL
459 85  +
460 43 RCL
461 01  01
462 85  +
463 76 LBL
464 75  -
465 43 RCL
466 09  09
467 95  =
468 42 STO
469 10  10
470 22 INV
471 88 DMS
472 42 STO
473 15  15
474 71 SBR
475 34 FX
476 32 X:T
477 22 INV
478 58 FIX
479 43 RCL
480 18  18
481 69 OP
482 04  04
483 32 X:T
484 58 FIX
485 03  03
486 69 OP
487 06  06
488 76 LBL
489 18 C'
490 43 RCL
491 10  10
492 75  -
493 43 RCL
494 01  01
495 95  =
496 42 STO
497 11  11
498 43 RCL
499 00  00
500 85  +
501 43 RCL
502 08  08
503 95  =
504 55  ÷
505 02  2
506 95  =
507 39 COS
508 65  ×
509 43 RCL
510 11  11
511 65  ×
512 06  6
513 00  0
514 95  =
515 42 STO
516 12  12
517 43 RCL
518 08  08
519 75  -
520 43 RCL
521 00  00
522 95  =
523 65  ×
524 06  6
525 00  0
526 95  =
527 42 STO
528 13  13
529 43 RCL
530 12  12
531 32 X:T
532 43 RCL
533 13  13
534 22 INV
535 37 P/R
536 42 STO
537 14  14
538 22 INV
539 58 FIX
540 43 RCL
541 21  21
542 69 OP
543 04  04
544 32 X:T
545 58 FIX
546 01  01
547 69 OP
548 06  06
549 09  9
550 00  0
551 32 X:T
552 43 RCL
553 14  14
554 77 GE
555 23 LNX
556 09  9
557 00  0
558 61 GTO
559 95  =
560 76 LBL
561 23 LNX
562 04  4
563 05  5
564 00  0
565 76 LBL
566 95  =
567 75  -
568 43 RCL
569 14  14
570 95  =
571 32 X:T
572 22 INV
573 58 FIX
574 43 RCL
575 22  22
576 69 OP
577 04  04
578 32 X:T
579 58 FIX
580 01  01
581 69 OP
582 06  06
583 98 ADV
584 43 RCL
585 17  17
586 69 OP
587 04  04
588 43 RCL
589 08  08
590 42 STO
591 00  00
592 43 RCL
593 10  10
594 42 STO
595 01  01
596 43 RCL
597 05  05
598 75  -
599 43 RCL
600 06  06
601 95  =
602 42 STO
603 05  05
604 32 X:T
605 93  .
606 00  0
607 00  0
608 01  1
609 77 GE
610 25 CLR
611 61 GTO
612 44 SUM
613 76 LBL
614 25 CLR
615 98 ADV
616 98 ADV
617 22 INV
618 58 FIX
619 69 OP
620 00  00
621 25 CLR
622 61 GTO
623 66 PAU
624 68 NOP
625 76 LBL
626 91 R/S
627 91 R/S
```

```
001 34 FX
038 19 D'
049 10 E'
179 66 PAU
187 68 NOP
191 69 OP
202 11  A
216 60 DEG
222 16 A'
236 12  B
252 17 B'
327 13  C
341 44 SUM
459 85  +
464 75  -
489 18 C'
561 23 LNX
566 95  =
614 25 CLR
626 91 R/S
```

■

TI 59
REGLERTEKNIK

Lysekil 19/11-84

Hej Programbiten!

Detta är i första hand ett program för den som sysslar med reglerteknik eller kretsanalys. Programmet listar frekvensfunktionen G(s):s amplitud och fas med utskrift varje tiondels dekad på frekvensaxeln. Den listar därutöver amplitud och fas för det enhetsåterförda slutna systemets frekvensfunktion M(s). En fördel med programmet är att det använder sådana begrepp som känns hemma i dessa sammanhang, såsom dödtid(deadtime) och dämpning (attenuation).

Programmet är till fullo interaktivt, och detta på ett sätt som blivit en slags "personlig standard" i mina längre TI-59-program. På ett av de oskrivna plastkort som följer med vid köp av modul har jag skrivit enligt fig. 1. Jag använder sedan detta plastkort till alla program skrivna med denna typ av promtning. Det gör att magnetkorten förblir oskrivna med undantag av sid- och blocknummer. Man slipper sudda och ändra på dem. Den största fördelen är dock att man slipper att för varje gång tänka på hur ett program skall köras, alla körs ju på samma sätt, vilket snabbar upp programanvändningen. (För den som fortfarande sörjer över den uteblivna TI-88 kan nämnas att det var ungefär så här användarmiljön vid programkörning var tänkt.)

Knappbeskrivning

"E'"	- START	Innebär start av program eller återstart vid fel vid magnetkortsinläsning eller annat programstopp.
"E"	- ENTER	Inmatning av data.
"C"	- CONTINUE	Avbrott och hopp till nästa programavsnitt eller behållning av vid tidigare körning inmatade data.
"A"	- YES	Svar vid (YES/NO)-fråga.
"B"	- NO	Svar vid (YES/NO)-fråga.

Jag tänker inte ge någon djupare beskrivning av programmets funktion, men nämnas bör att sida 1 innehåller subrutiner, sida 2 & 3 innehåller inmatningsrutiner, sida 4 & 6 innehåller utmatningsrutiner och sida 5 gör beräkningar.

Inmatningsinstruktioner

1. Stäng av och sätt på räknaren.
2. Mata in sida 1 från steg 000 till 239 och spela in på framsidan av magnetkort nr 1 med 1 2nd Write.
3. Mata in sida 2 från steg 240 till 479 och spela in på baksidan av kort nr 1 med 2 2nd Write.
4. Mata in sida 3 från steg 240 till 479 och spela in på framsidan av kort nr 2 med 2 2nd Write.
5. Mata in sida 4 från steg 240 till 479 och spela in på baksidan av kort nr 2 med 2 2nd Write.
6. Mata in sida 5 från steg 240 till 479 och spela in på framsidan av kort nr 3 med 2 2nd Write.
7. Tryck 3 Op 17, dvs ställ in uppdelning 719.29.
8. Mata in sida 6 från steg 480 till 719 och spela in på baksidan av kort nr 3 med 3 2nd Write.
9. Tryck 6 Op 17.
10. Märk kortsidorna med respektive sidnummer.
11. Provkör programmet!

Körinstruktioner

1. Stäng av och sätt på räknaren.
2. Läs in sida 1 och 2.
3. Tryck START (="E'") och följ instruktionerna.

Testkörningar

I testkörning nr 1 har kretsen

$$G(s) = 2000 \frac{(1 + 0.2s)^2}{s(1 + 18s)^2 \ (1 + 0.01)^2}$$

analyserats mellan vinkelfrekvenserna 1 rad/s till 10 rad/s. Märk att tidskonstantinmatning avslutas med CONTINUE. Vid frågan "2ND ORDER FACTOR IN DENOMINATOR?" svaras NO.

Testkörning nr 2 har startats genom att svara YES på frågan "AGAIN?". Här har kretsen

$$G(s) = 2.5 \frac{e^{-3s}}{1 + 18s + 144s^2}$$

analyserats mellan vinkelfrekvenserna 0.02 rad/s till 0.1 rad/s. Vid frågan "2ND ORDER FACTOR IN DENOMINATOR?" svaras YES. Körningen avslutas genom att svara NO på frågan "AGAIN?".

Glöm ej att vinkelfrekvensen $\omega = 2 * \pi * f$, där f är frekvensen. I en andragradsterm $s^2 + sA + B$ gäller att:

Tidskonstanten $T = 1/A$

Dämpningen $zäta = 1/(2 * T * B^{\frac{1}{2}})$

Oftast beskriver man dock en andragradsterm direkt i tidskonstant och dämpning.

Utskrift

Utskriften följer helt den standard som normalt används vid plottning av Bode-diagram. Först anges vinkelfrekvensen i rad/s, sedan absolutvärdet av G(s) i decibel och fasen i grader. Därefter ges M(s) på samma sätt. En komihåglista för dessa storheter skrivs ut innan sifferlistan.

Lycka till!

Martin Harnevie, Mölndal

TI 59

PROGRAMMERARE: HARNEVIE				
REGLERTEKNIK				
				START
YES	NO	CONTINUE		ENTER

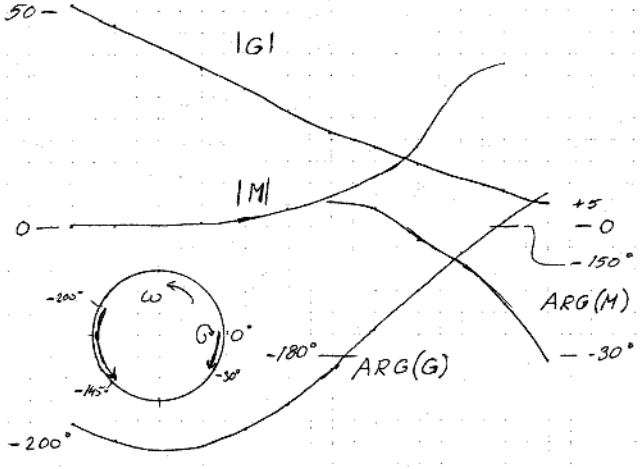
```
TESTKOERNING

NO. OF INTEGRATIONS?
          1.
CIRCUIT GAIN?                          1.          3.981071706
       2000.                    52.38099807         22.09451744
DEADTIME (SEC)?                -195.3959101        -183.1878674
          0.                    .0201538698         .7095389671
                               -359.9633458        -359.728338

TIMECONSTANTS(MAX 5)
NOMINATOR:                       1.258925412         5.011872336
  T1?                            47.23970601         17.91281322
          0.2                   -199.8552171        -174.2081861
  T2?                            .0355619211         1.173938175
          0.2                   -359.9150936        -.8416791713
  T3?
ENTER CARDSIDE NO. 3             1.584893192         6.309573445
DENOMINATOR:                     41.98470946         14.16465157
  T1?                           -201.6222477        -164.9487248
         0.01                    .0644535137         1.80322267
  T2?                           -359.8307491        -3.587394339
         0.01
  T3?
          2.                     1.995262315         7.943282347
                                 36.7304238          10.83071792
  T4?                           -200.6404136        -156.2574211
          2.                     .1191324581         2.545609174
  T5?                           -359.7016532        -8.923359913
2ND ORDER FACTOR IN
DENOMINATOR(YES/NO)?
ENTER CARDSIDE NO. 4             2.511886432                10.
WMIN(RAD/S)?                     31.59115225         7.850685072
          1.                     -197.01441         -148.8264782
WMAX(RAD/S)?                     .2212140691         3.270002087
         10.                    -359.547183         -17.78720937

W (RAD/S)=                       3.16227766          12.58925412
IGI(DB)=                         26.67880194         5.142542865
ARG(G)=                         -191.0296758        -143.1243513
IMI(DB)=                         .4040600843         3.757357171
ARG(M)=                         -359.4677125        -30.77199934

ENTER CARDSIDE NO. 5                                 DONE?
ENTER CARDSIDE NO. 6                                 ****************
                                                     AGAIN? (YES/NO)
```



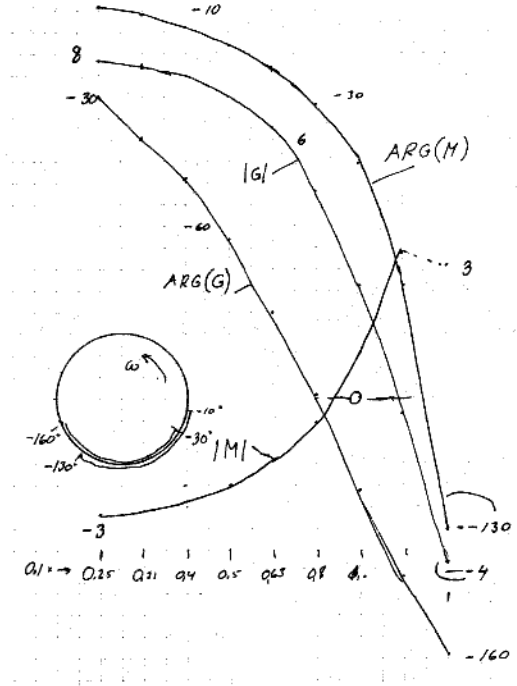

```
AGAIN? (YES/NO)
ENTER CARDSIDE NO. 2

NO. OF INTEGRATIONS?
          0.                                          .0630957344
CIRCUIT GAIN?                                         6.279634089
          2.5                                        -80.25263496
DEADTIME (SEC)?                                      -1.460783392
          3.                                         -23.84436196

TIMECONSTANTS(MAX 5)             .0251188643         .0794328235
NOMINATOR:                       7.826341516         4.835630672
  T1?                           -30.75994411        -99.99496088
ENTER CARDSIDE NO. 3            -2.70206606         -.5288421786
DENOMINATOR:                    -8.753660402        -32.07635518
  T1?
2ND ORDER FACTOR IN
DENOMINATOR(YES/NO)?             .0316227766                 0.1
                                 7.719135145         2.601303171
                                -39.05806774        -120.9250022
TIMECONSTANT?                   -2.571175268         1.037227144
         12.                    -11.11084967        -45.76445815
ENTER CARDSIDE NO. 4
ATTENUATION?
         0.75                    .0398107171         .1258925412
WMIN(RAD/S)?                     7.508926188         -.3529902213
         0.02                   -49.71964281        -141.1425493
WMAX(RAD/S)?                    -2.360785847         3.347394321
          0.1                   -14.17550922        -73.86792205

W (RAD/S)=                       .0501187234         .1584893192
IGI(DB)=                         7.090707371         -3.798561793
ARG(G)=                         -63.23424935        -159.7750395
IMI(DB)=                        -2.019885259         3.081379138
ARG(M)=                         -18.24396781        -130.2423005

ENTER CARDSIDE NO. 5                                 DONE?
ENTER CARDSIDE NO. 6                                 ****************
```



REGLERTEKNIK

TI 59

REGLERTEKNIK

Kortsida 1 SUBrutiner

000	91	R/S	060	85	+	120	03	3	180	01	1								
001	76	LBL	061	03	3	121	07	7	181	06	6								
002	17	B'	062	07	7	122	61	GTO	182	16	A'								
003	42	STO	063	00	0	123	01	01	183	03	3								
004	03	03	064	01	1	124	62	62	184	06	6								
005	25	CLR	065	07	7	125	76	LBL	185	02	2								
006	42	STO	066	01	1	126	87	IFF	186	04	4								
007	01	01	067	54	)	127	82	HIR	187	01	1								
008	76	LBL	068	69	OP	128	05	05	188	06	6								
009	16	A'	069	01	0.	129	82	HIR	189	01	1								
010	84	OP+	070	69	OP	130	06	06	190	07	7								
011	01	01	071	05	05	131	82	HIR	191	00	0								
012	69	OP	072	43	RCL	132	07	07	192	00	0								
013	21	21	073	05	05	133	92	RTN	193	16	A'								
014	92	RTN	074	69	OP	134	08	08	194	53	(								
015	76	LBL	075	00	00	135	69	OP	195	03	3								
016	91	R/S	076	92	RTN	136	05	05	196	01	1								
017	22	INV	077	61	GTO	137	00	0	197	03	3								
018	57	ENG	078	00	00	138	92	RTN	198	02	2								
019	22	INV	079	82	82	139	68	NOP	199	04	4								
020	58	FIX	080	83	GO+	140	68	NOP	200	00	0								
021	68	NOP	081	02	02	141	68	NOP	201	00	0								
022	00	0	082	72	ST*	142	68	NOP	202	00	0								
023	42	STO	083	04	04	143	68	NOP	203	00	0								
024	00	00	084	99	PRT	144	68	NOP	204	00	0								
025	42	STO	085	69	OP	145	68	NOP	205	85	+								
026	01	01	086	24	24	146	68	NOP	206	32	X:T								
027	69	NOP	087	69	OP	147	68	NOP	207	54	)								
028	70	RAD	088	25	25	148	68	NOP	208	16	A'								
029	42	STO	089	43	RCL	149	68	NOP	209	16	A'								
030	03	03	090	05	05	150	68	NOP	210	00	0								
031	92	RTN	091	22	INV	151	68	NOP	211	68	NOP								
032	76	LBL	092	77	GE	152	68	NOP	212	22	INV								
033	28	LOG	093	00	00	153	68	NOP	213	96	WRT								
034	53	(	094	53	53	154	68	NOP	214	83	GO*								
035	28	LOG	095	83	GO*	155	68	NOP	215	03	03								
036	65	×	096	02	02	156	76	LBL	216	76	LBL								
037	01	1	097	76	LBL	157	96	WRT	217	12	B								
038	00	0	098	69	OP	158	32	X:T	218	22	INV								
039	54	)	099	07	7	159	02	2	219	76	LBL								
040	92	RTN	100	32	X:T	160	03	3	220	11	A								
041	76	LBL	101	03	3	161	07	7	221	86	STF								
042	97	DSZ	102	69	OP	162	17	B'	222	01	01								
043	42	STO	103	17	17	163	01	1	223	83	GO*								
044	04	04	104	86	STF	164	07	7	224	00	00								
045	01	1	105	02	02	165	03	3	225	76	LBL								
046	42	STO	106	04	4	166	01	1	226	13	C								
047	05	05	107	07	7	167	03	3	227	24	CE								
048	06	6	108	07	7	168	07	7	228	03	3								
049	32	X:T	109	61	GTO	169	01	1	229	44	SUM								
050	07	7	110	01	01	170	07	7	230	03	03								
051	07	7	111	62	62	171	03	3	231	83	GO+								
052	17	B'	112	76	LBL	172	05	5	232	03	03								
053	53	(	113	81	RST	173	16	A'	233	76	LBL								
054	43	RCL	114	06	6	174	01	1	234	15	E								
055	05	05	115	69	OP	175	05	5	235	83	GO+								
056	65	×	116	17	17	176	01	1	236	03	00								
057	01	1	117	03	3	177	03	3	237	68	NOP								
058	00	0	118	32	X:T	178	03	3	238	76	LBL								
059	00	0	119	02	2	179	05	5	239	10	E'								

Kortsida 3 INmatningsrutiner

240	69	OP	300	05	5	360	01	1	420	07	7								
241	35	:35	301	01	1	361	03	3	421	07	7								
242	43	RCL	302	07	7	362	16	A'	422	01	1								
243	05	05	303	03	3	363	01	1	423	00	0								
244	42	STO	304	06	6	364	06	6	424	00	0								
245	14	14	305	16	A'	365	16	A'	425	00	0								
246	17	B'	306	06	6	366	06	6	426	00	0								
247	01	1	307	03	3	367	06	6	427	16	A'								
248	06	6	308	03	3	368	06	6	428	00	0								
249	01	1	309	01	1	369	01	1	429	16	A'								
250	07	7	310	03	3	370	06	6	430	16	A'								
251	03	3	311	00	0	371	06	6	431	16	A'								
252	01	1	312	42	STO	372	04	4	432	04	4								
253	03	03	313	03	03	373	16	A'	433	05	5								
254	16	A'	314	04	4	374	69	OP	434	05	5								
255	43	RCL	315	03	3	375	05	05	435	01	1								
256	06	06	316	03	3	376	17	B'	436	06	6								
257	16	A'	317	00	0	377	02	2	437	16	A'								
258	00	0	318	01	1	378	04	4	438	01	1								
259	16	A'	319	03	3	379	03	3	439	04	4								
260	16	A'	320	04	4	380	02	2	440	05	5								
261	92	RTN	321	04	4	381	02	2	441	06	6								
262	42	STO	322	05	5	382	04	4	442	06	6								
263	16	16	323	05	5	383	05	5	443	04	4								
264	99	PRT	324	69	OP	384	05	5	444	00	0								
265	03	3	325	01	01	385	01	1	445	00	0								
266	00	0	326	69	OP	386	06	6	446	00	0								
267	06	6	327	05	05	387	16	A'	447	00	0								
268	17	B'	328	00	0	388	01	1	448	16	A'								
269	04	4	329	92	RTN	389	04	4	449	69	OP								
270	03	3	330	42	STO	390	05	5	450	05	05								
271	03	3	331	18	18	391	06	6	451	17	B'								
272	00	0	332	99	PRT	392	06	6	452	01	1								
273	02	2	333	43	RCL	393	04	4	453	03	3								
274	04	4	334	17	17	394	00	0	454	03	3								
275	03	3	335	71	SBR	395	00	0	455	05	5								
276	01	1	336	28	LOG	396	00	0	456	02	2								
277	05	5	337	59	INT	397	00	0	457	02	2								
278	05	5	338	42	STO	398	16	A'	458	05	5								
279	16	A'	339	17	17	399	69	OP	459	05	5								
280	03	3	340	43	RCL	400	05	05	460	03	3								
281	05	5	341	18	18	401	17	B'	461	00	0								
282	01	1	342	71	SBR	402	01	1	462	16	A'								
283	03	3	343	28	LOG	403	03	3	463	05	5								
284	01	1	344	59	INT	404	03	3	464	06	6								
285	06	6	345	42	STO	405	05	5	465	06	6								
286	06	6	346	18	18	406	02	2	466	04	4								
287	03	3	347	01	1	407	02	2	467	00	0								
288	03	3	348	44	SUM	408	05	5	468	00	0								
289	06	6	349	18	18	409	05	5	469	00	0								
290	16	A'	350	98	ADV	410	02	2	470	00	0								
291	05	5	351	17	B'	411	02	2	471	00	0								
292	06	6	352	04	4	412	16	A'	472	00	0								
293	07	7	353	03	3	413	05	5	473	16	A'								
294	01	1	354	00	0	414	06	6	474	69	OP								
295	00	0	355	00	0	415	06	6	475	05	05								
296	00	0	356	05	5	416	04	4	476	98	ADV								
297	00	0	357	05	5	417	00	0	477	06	6								
298	00	0	358	03	3	418	00	0	478	61	GTO								
299	00	0	359	05	5	419	00	0	479	96	WRT								

Kortsida 2 INmatningsrutiner

240	71	SBR	300	01	1	360	07	7	420	03	3								
241	91	R/S	301	05	5	361	00	0	421	04	4								
242	02	2	302	02	2	362	00	0	422	04	4								
243	09	9	303	04	4	363	05	5	423	00	0								
244	03	3	304	03	3	364	05	5	424	00	0								
245	17	B'	305	05	5	365	16	A'	425	00	0								
246	03	3	306	01	1	366	03	3	426	06	6								
247	01	1	307	05	5	367	06	6	427	05	5								
248	03	3	308	04	4	368	01	1	428	06	6								
249	02	2	309	01	1	369	07	7	429	16	A'								
250	04	4	310	16	A'	370	01	1	430	16	A'								
251	00	0	311	02	2	371	05	5	431	17	B'								
252	00	0	312	04	4	372	05	5	432	03	3								
253	00	0	313	03	3	373	06	6	433	01	1								
254	03	3	314	07	7	374	07	7	434	03	3								
255	02	2	315	00	0	375	01	1	435	02	2								
256	16	A'	316	00	0	376	16	A'	436	03	3								
257	02	2	317	02	2	377	00	0	437	00	0								
258	01	1	318	02	2	378	16	A'	438	02	2								
259	00	0	319	01	1	379	16	A'	439	04	4								
260	00	0	320	03	3	380	92	RTN	440	03	3								
261	02	2	321	16	A'	381	42	STO	441	01	1								
262	04	4	322	02	2	382	12	12	442	16	A'								
263	03	3	323	04	4	383	99	PRT	443	01	1								
264	01	1	324	03	3	384	17	B'	444	03	3								
265	03	3	325	01	1	385	98	ADV	445	03	3								
266	07	7	326	07	7	386	03	3	446	07	7								
267	16	A'	327	01	1	387	07	7	447	03	3								
268	01	1	328	00	0	388	02	2	448	02	2								
269	07	7	329	00	0	389	04	4	449	03	3								
270	02	2	330	00	0	390	03	3	450	05	5								
271	02	2	331	00	0	391	00	0	451	06	6								
272	03	3	332	16	A'	392	01	1	452	02	2								
273	05	5	333	00	0	393	07	7	453	16	A'								
274	01	1	334	16	A'	394	01	1	454	00	0								
275	03	3	335	16	A'	395	05	5	455	16	A'								
276	03	3	336	92	RTN	396	16	A'	456	16	A'								
277	07	7	337	42	STO	397	03	3	457	16	A'								
278	16	A'	338	11	11	398	02	2	458	04	4								
279	02	2	339	99	PRT	399	03	3	459	06	6								
280	04	4	340	03	3	400	01	1	460	07	7								
281	03	3	341	08	8	401	03	3	461	42	STO								
282	02	2	342	01	1	402	06	6	462	02	02								
283	03	3	343	17	B'	403	03	3	463	02	2								
284	01	1	344	01	1	404	07	7	464	00	0								
285	03	3	345	06	6	405	01	1	465	61	GTO								
286	06	6	346	01	1	406	03	3	466	97	DSZ								
287	07	7	347	07	7	407	16	A'	467	04	4								
288	01	1	348	01	1	408	03	3	468	61	GTO								
289	16	A'	349	03	3	409	01	1	469	96	WRT								
290	16	A'	350	01	1	410	03	3	470	68	NOP								
291	00	0	351	06	6	411	07	7	471	68	NOP								
292	92	RTN	352	03	3	412	03	3	472	68	NOP								
293	42	STO	353	07	7	413	06	6	473	68	NOP								
294	10	10	354	16	A'	414	05	5	474	68	NOP								
295	99	PRT	355	02	2	415	05	5	475	68	NOP								
296	03	3	356	04	4	416	03	3	476	68	NOP								
297	03	3	357	03	3	417	00	0	477	68	NOP								
298	07	7	358	00	0	418	16	A'	478	68	NOP								
299	17	B'	359	01	1	419	01	1	479	68	NOP								

Kortsida 4 UTmatningsrutiner

240	22	INV	300	00	0	360	01	1	420	00	0								
241	87	IFF	301	16	A'	361	03	3	421	00	0								
242	01	01	302	00	0	362	16	A'	422	00	0								
243	02	02	303	16	A'	363	01	1	423	16	A'								
244	65	65	304	16	A'	364	06	6	424	69	OP								
245	02	2	305	92	RTN	365	06	6	425	05	05								
246	06	6	306	42	STO	366	03	3	426	17	B'								
247	02	2	307	17	17	367	03	3	427	02	2								
248	17	B'	308	99	PRT	368	06	6	428	04	4								
249	43	RCL	309	03	3	369	05	5	429	03	3								
250	04	04	310	03	3	370	06	6	430	00	0								
251	16	A'	311	00	0	371	06	6	431	02	2								
252	43	RCL	312	42	STO	372	04	4	432	04	4								
253	05	05	313	03	03	373	16	A'	433	05	5								
254	16	A'	314	04	4	374	69	OP	434	05	5								
255	43	RCL	315	03	3	375	05	05	435	01	1								
256	06	06	316	03	3	376	17	B'	436	06	6								
257	16	A'	317	00	0	377	02	2	437	16	A'								
258	00	0	318	01	1	378	04	4	438	01	1								
259	16	A'	319	03	3	379	03	3	439	04	4								
260	16	A'	320	04	4	380	02	2	440	05	5								
261	92	RTN	321	04	4	381	02	2	441	06	6								
262	42	STO	322	05	5	382	04	4	442	06	6								
263	16	16	323	05	5	383	05	5	443	04	4								
264	99	PRT	324	69	OP	384	05	5	444	00	0								
265	03	3	325	01	01	385	01	1	445	00	0								
266	00	0	326	69	OP	386	06	6	446	00	0								
267	06	6	327	05	05	387	16	A'	447	00	0								
268	17	B'	328	00	0	388	01	1	448	16	A'								
269	04	4	329	92	RTN	389	04	4	449	69	OP								
270	03	3	330	42	STO	390	05	5	450	05	05								
271	03	3	331	18	18	391	06	6	451	17	B'								
272	00	0	332	99	PRT	392	06	6	452	01	1								
273	02	2	333	43	RCL	393	04	4	453	03	3								
274	04	4	334	17	17	394	00	0	454	03	3								
275	03	3	335	71	SBR	395	00	0	455	05	5								
276	01	1	336	28	LOG	396	00	0	456	02	2								
277	05	5	337	59	INT	397	00	0	457	02	2								
278	05	5	338	42	STO	398	16	A'	458	05	5								
279	16	A'	339	17	17	399	69	OP	459	05	5								
280	03	3	340	43	RCL	400	05	05	460	03	3								
281	05	5	341	18	18	401	17	B'	461	00	0								
282	01	1	342	71	SBR	402	01	1	462	16	A'								
283	03	3	343	28	LOG	403	03	3	463	05	5								
284	01	1	344	59	INT	404	03	3	464	06	6								
285	06	6	345	42	STO	405	05	5	465	06	6								
286	06	6	346	18	18	406	02	2	466	04	4								
287	03	3	347	01	1	407	02	2	467	00	0								
288	03	3	348	44	SUM	408	05	5	468	00	0								
289	06	6	349	18	18	409	05	5	469	00	0								
290	16	A'	350	98	ADV	410	02	2	470	00	0								
291	05	5	351	17	B'	411	02	2	471	00	0								
292	06	6	352	04	4	412	16	A'	472	00	0								
293	07	7	353	03	3	413	05	5	473	16	A'								
294	01	1	354	00	0	414	06	6	474	69	OP								
295	00	0	355	00	0	415	06	6	475	05	05								
296	00	0	356	05	5	416	04	4	476	98	ADV								
297	00	0	357	05	5	417	00	0	477	06	6								
298	00	0	358	03	3	418	00	0	478	61	GTO								
299	00	0	359	05	5	419	00	0	479	96	WRT								

TI 59

Kortsida 5 BERäkningar REGLERTEKNIK

240	43	RCL	300	01	1	360	02	2	420	85	+	480	43	RCL	540	71	71	600	53	(	660	71	SBR
241	17	17	301	54	)	361	05	5	421	01	1	481	09	09	541	53	(	601	43	RCL	661	87	IFF
242	42	STO	302	23	LNX	362	42	STO	422	54	)	482	99	PRT	542	53	(	602	07	07	662	17	B'
243	04	04	303	55	÷	363	05	05	423	53	(	483	53	(	543	02	2	603	65	×	663	01	1
244	53	(	304	02	2	364	53	(	424	33	X²	484	43	RCL	544	85	+	604	01	1	664	03	3
245	43	RCL	305	54	)	365	73	RC*	425	85	+	485	08	08	545	82	HIR	605	08	8	665	02	2
246	04	04	306	44	SUM	366	05	05	426	43	RCL	486	22	INV	546	17	17	606	00	0	666	02	2
247	55	÷	307	08	08	367	65	×	427	19	19	487	23	LNX	547	69	OP	607	55	÷	667	01	1
248	01	1	308	43	RCL	368	43	RCL	428	33	X²	488	82	HIR	548	10	10	608	89	π	668	03	3
249	00	0	309	19	19	369	09	09	429	65	×	489	08	08	549	54	)	609	54	)	669	02	2
250	54	)	310	70	RAD	370	54	)	430	04	4	490	33	X²	550	65	×	610	99	PRT	670	04	4
251	22	INV	311	22	INV	371	42	STO	431	65	×	491	85	+	551	89	π	611	53	(	671	03	3
252	28	LOG	312	30	TAN	372	19	19	432	43	RCL	492	01	1	552	55	÷	612	43	RCL	672	01	1
253	42	STO	313	44	SUM	373	53	(	433	16	16	493	85	+	553	02	2	613	09	09	673	16	A'
254	09	09	314	07	07	374	53	(	434	33	X²	494	02	2	554	94	+/-	614	28	LOG	674	07	7
255	43	RCL	315	43	RCL	375	33	X²	435	54	)	495	65	×	555	75	-	615	65	×	675	01	1
256	11	11	316	05	05	376	85	+	436	53	(	496	82	HIR	556	53	(	616	01	1	676	00	0
257	23	LNX	317	67	EQ	377	01	1	437	23	LNX	497	18	18	557	82	HIR	617	00	0	677	00	0
258	42	STO	318	03	03	378	54	)	438	55	÷	498	65	×	558	16	16	618	54	)	678	05	5
259	08	08	319	25	25	379	23	LNX	439	02	2	499	43	RCL	559	55	÷	619	99	PRT	679	05	5
260	53	(	320	69	OP	380	55	÷	440	54	)	500	07	07	560	82	HIR	620	53	(	680	04	4
261	43	RCL	321	25	25	381	02	2	441	22	INV	501	39	COS	561	17	17	621	82	HIR	681	05	5
262	12	12	322	61	GTO	382	54	)	442	44	SUM	502	54	)	562	54	)	622	18	18	682	01	1
263	65	×	323	02	02	383	22	INV	443	08	08	503	53	(	563	22	INV	623	65	×	683	07	7
264	43	RCL	324	87	87	384	44	SUM	444	53	(	504	35	1/X	564	30	TAN	624	01	1	684	16	A'
265	09	09	325	53	(	385	08	08	445	53	(	505	65	×	565	54	)	625	08	8	685	03	3
266	94	+/-	326	43	RCL	386	43	RCL	446	01	1	506	82	HIR	566	82	HIR	626	00	0	686	06	6
267	54	)	327	09	09	387	19	19	447	75	-	507	18	18	567	08	08	627	55	÷	687	06	6
268	42	STO	328	23	LNX	388	22	INV	448	43	RCL	508	33	X²	568	61	GTO	628	89	π	688	03	3
269	07	07	329	65	×	389	30	TAN	449	19	19	509	54	)	569	05	05	629	54	)	689	03	3
270	43	RCL	330	43	RCL	390	22	INV	450	33	X²	510	42	STO	570	87	87	630	98	PRT	690	11	A
271	14	14	331	10	10	391	44	SUM	451	54	)	511	09	09	571	53	(	631	98	ADV	691	03	3
272	29	CP	332	54	)	392	07	07	452	55	÷	512	43	RCL	572	69	OP	632	69	OP	692	02	2
273	67	EQ	333	22	INV	393	43	RCL	453	02	2	513	07	07	573	55	÷	633	24	24	693	05	5
274	03	03	334	44	SUM	394	05	05	454	55	÷	514	38	SIN	574	02	2	634	43	RCL	694	06	6
275	25	25	335	08	08	395	67	EQ	455	43	RCL	515	82	HIR	575	65	×	635	04	04	695	16	A'
276	53	(	336	53	(	396	04	04	456	16	16	516	07	07	576	53	(	636	32	X:T	696	00	0
277	46	INS	337	43	RCL	397	03	03	457	55	÷	517	53	(	577	82	HIR	637	43	RCL	697	16	A'
278	85	+	338	10	10	398	69	OP	458	43	RCL	518	43	RCL	578	16	16	638	18	18	698	16	A'
279	01	1	339	65	×	399	25	25	459	19	19	519	07	07	579	69	OP	639	77	GE	699	07	7
280	09	9	340	89	π	400	61	GTO	460	54	)	520	39	COS	580	10	10	640	02	02	700	00	0
281	54	)	341	55	÷	401	03	03	461	53	(	521	85	+	581	75	-	641	44	44	701	06	6
282	32	X:T	342	02	2	402	64	64	462	22	INV	522	82	HIR	582	01	1	642	00	0	702	42	STO
283	02	2	343	54	)	403	22	INV	463	30	TAN	523	18	18	583	54	)	643	17	B'	703	00	00
284	00	0	344	22	INV	404	87	IFF	464	75	-	524	54	)	584	54	)	644	01	1	704	00	0
285	42	STO	345	44	SUM	405	01	01	465	89	π	525	82	HIR	585	82	HIR	645	06	6	705	81	RST
286	05	05	346	07	07	406	04	04	466	55	÷	526	06	06	586	08	08	646	03	3	706	87	IFF
287	53	(	347	43	RCL	407	72	ST*	467	02	2	527	01	1	587	53	(	647	02	2	707	01	01
288	73	RC*	348	15	15	408	53	(	468	54	)	528	53	EE	588	43	RCL	648	03	3	708	81	RST
289	05	05	349	69	OP	409	43	RCL	469	68	NOP	529	09	9	589	08	08	649	01	1	709	25	CLR
290	65	×	350	67	EQ	410	13	13	470	44	SUM	530	94	+/-	590	65	×	650	01	1	710	81	RST
291	43	RCL	351	04	04	411	65	×	471	07	07	531	22	INV	591	01	1	651	07	7	711	68	NOP
292	09	09	352	03	03	412	43	RCL	472	22	INV	532	52	EE	592	22	INV	652	07	7	712	68	NOP
293	54	)	353	53	(	413	09	09	473	87	IFF	533	32	X:T	593	23	LNX	653	03	3	713	68	NOP
294	42	STO	354	46	INS	414	54	)	474	02	02	534	82	HIR	594	28	LOG	654	16	A'	714	68	NOP
295	19	19	355	85	+	415	42	STO	475	69	OP	535	17	17	595	65	×	655	69	OP	715	68	NOP
296	53	(	356	02	2	416	19	19	476	68	NOP	536	50	I×I	596	02	2	656	05	05	716	68	NOP
297	53	(	357	04	4	417	53	(	477	68	NOP	537	22	INV	597	00	0	657	06	6	717	68	NOP
298	33	X²	358	54	)	418	33	X²	478	68	NOP	538	77	GE	598	54	)	658	06	6	718	68	NOP
299	85	+	359	32	X:T	419	94	+/-	479	68	NOP	539	05	05	599	99	PRT	659	35	1/X	719	68	NOP

Kortsida 6 UTmatningsrutiner

NU är glada PB året 1984

SLUT SLUT SLUT

Men till nästa år

kommer vi igen

ty det lovar VI

BARA vi får MERa TEXT

och PROGRAM från ER !

Claes S

GOD JUL 1984

OCH

ETT

GOTT NYTT ÅR 1985

Gun-Wigge

SPRITES med MINI MEMORY

Av Jan Alexandersson

32 SPRITES MED MINI MEMORY

Det är fullt möjligt att fixa 32 sprites med Mini Memory med enbart Basic-komandon som POKEV och LOAD. I NITTINIAN NR 83-01 fanns en artikel om sprites i Mini Memory. En sammanfattning av denna artikel ger följande programlistning:

```
 80 CALL CLEAR
 90 CALL SCREEN(16)
100 CALL POKEV(768,98,128,161,1)
110 CALL POKEV(772,35,200,162,10,208)
120 CALL POKEV(1920,50,50)
130 CALL POKEV(1924,-30,120)
140 CALL LOAD(-31872,2)
150 CALL KEY(3,KEY,STA)
160 IF STA<1 THEN 150
170 CALL LOAD(-31878,0)
```

Det frågades om varför ej fler än 4 sprites kunde ordnas på detta sätt. Det beror på att COLOR TABLE och SPRITE ATRIBUTE LIST överlappar varandra (se fig 1). Eftersom SPRITE ATRIBUTE LIST alltid måste avslutas med D0 = 208 ser man att 3 sprites (# 0-#2) alltid får plats. Eftersom D0 = 208 ger bakgrundsfärg 0 kan man även få in 4 sprites (# 0-#3). CHR 32 som alltid finns på skärmen får då rätt bakgrundsfärg.

Man kan dock fylla skärmen med andra tecken med högre ASCII-kod. Slutsatsen är att det är möjligt att ha 7 sprites (# 0-#6) om skärmen fylls med tecken från SET 14-16 = CHR 136 - 159. Det går dock ej att ta tecken som används av SPRITE MOTION TABLE (se fig 2). Följande rader kan användas:

```
 95 CALL CHAR(136,"")
 97 CALL HCHAR(1,1,136,768)
```

Den enda möjligheten att få alla 32 sprites är att flytta någon av tabellerna:

- SPRITE MOTION TABLE — går aldrig att flytta
- SPRITE ATRIBUTE TABLE — går att flytta men tillåter då ej automatisk rörelse
- COLOR TABLE — går alltid att flytta till t.ex. DEC 1856 = 29 * 64 dvs 29 skall skrivas till VDP-register 3.

Ett demonstrationsprogram för upp till 32 sprites med valfri MAGNIFY (1-4) visas nedan för Mini Memory eller Editor/Assembler:

```
100 REM AUTO SPRITE MM/EA
110 REM JAN ALEXANDERSSON
120 REM VERSION 1984-11-11
130 REM REF NITTINIAN 83-01
140 CALL CLEAR
150 RANDOMIZE
160 DEF RN=90*RND-45
170 INPUT "ANTALET SPRITE ":ANTAL
180 IF (ANTAL>32)+(ANTAL<0) THEN 170
190 PRINT
200 INPUT "MAGNIFY        ":MAGN
210 IF (MAGN>4)+(MAGN<1) THEN 200
220 REM BLANK SCREEN
230 REM VDPREG1 SET A0=160
    DVS HEX 81A0
240 CALL POKEV(-32352,0)
250 CALL SCREEN(16)
260 REM COLTAB DEC 1856
270 REM VDPREG3 SET 1D=29
    DVS HEX 831D
280 CALL POKEV(-31971,0)
290 REM INIT COLTAB
300 FOR I=0 TO 14
310 CALL POKEV(1856+I,0)
320 NEXT I
330 FOR I=15 TO 31
340 CALL POKEV(1856+I,16)
350 NEXT I
360 REM SPRITE DESCRIPTION
370 FOR I=0 TO ANTAL-1
380 CALL POKEV(768+I*4,90,120,160+I,RND*12+1)
390 NEXT I
400 CALL POKEV(768+ANTAL*4,208)
410 SPRITE MOTION TABLE
420 FOR I=0 TO ANTAL-1
430 CALL POKEV(1920+4*I,RN,RN)
440 NEXT I
450 CALL CLEAR
460 PRINT "SPRITE TEST"::"AVSLUTA MED <ENTER>"
470 REM SPRITE MAGNIFY OCH
    AKTIVERA SCREEN
480 REM VDPREG1 SET E0-E3=
    224-227 DVS HEX 81E0-81E3
490 CALL POKEV(-32288+MAGN-1,0)
500 CALL LOAD(-31878,ANTAL)
510 CALL KEY(3,KEY,STA)
520 IF STA<1 THEN 510
530 CALL LOAD(-31878,0)
540 CALL CLEAR
```

Eftersom ingen tangenttryckning används annat än vid avslutning av programet behöver man ej skriva till VDPREG1 kopia. I ett mer avancerat program kan man lägga till följande rader:

```
495 CALL LOAD(-31788,224+MAGN-1)
525 CALL LOAD(-31788,224)
```

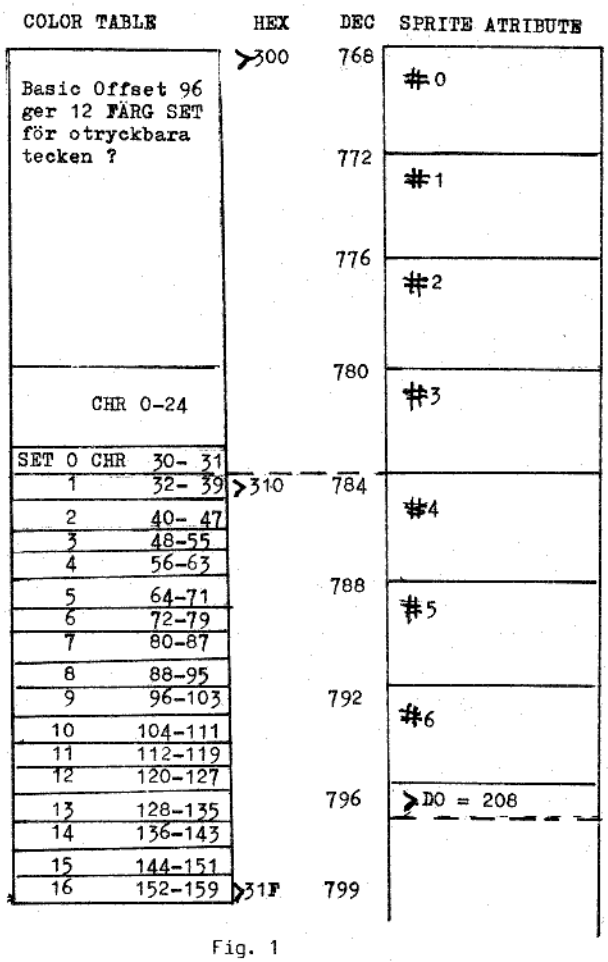


Fig. 1

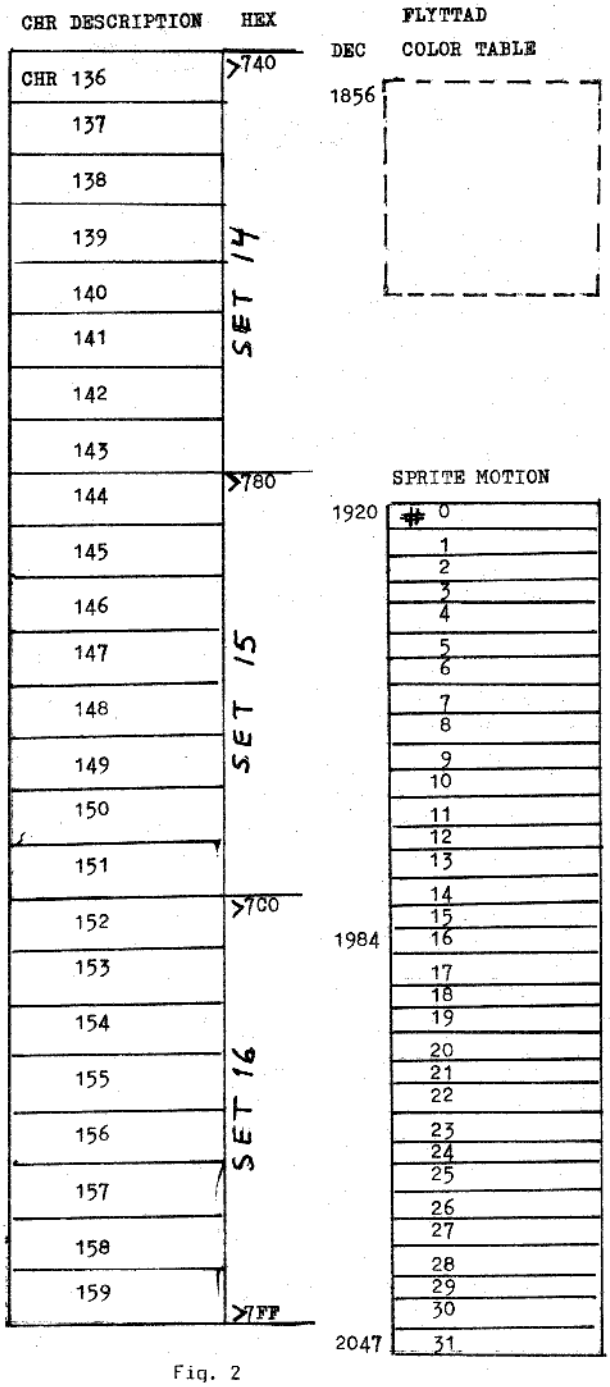


Fig. 2

DISK-KATALOG I TEXTMODE MED MINI MEMORY

```
100 REM CAT TEXTMODE MM/EA
110 REM JAN ALEXANDERSSON
120 REM VERSION 1984-11-30
130 REM REF DSK CNTR MANUAL
140 CALL CLEAR
150 CALL CHAR(91,"00280038447C4444")
160 SPACE$="                "
170 TYP$(1)="DF"
180 TYP$(2)="DV"
190 TYP$(3)="IF"
200 TYP$(4)="IV"
210 TYP$(5)="PROGR"
220 GOSUB 9000
230 OPEN #1:"DSK1.",INPUT ,RELATIVE,INTERNAL
240 INPUT #1:A$,J,J,K
250 TEXT$="DISK= "&A$&"  LEDIG= "&STR$(K)&" ANVÄNT= "
    &STR$(J-K)
260 RAD=1
270 KOL=1
280 GOSUB 9200
290 FOR KOL=1 TO 22 STEP 21
300 TEXT$="filnamn   sekt typ"
310 RAD=2
320 GOSUB 9200
330 FOR RAD=4 TO 24
340 INPUT #1:A$,A,J,K
350 IF LEN(A$)=0 THEN 400
360 TEXT$=A$&SEG$(SPACE$,1,13-LEN(A$)-LEN(STR$(J)))&S
    TR$(J)&CHR$(32-10*(A<0))&TYP$(ABS(A))&SEG$(STR$(K
    )&SPACE$,1,-3*(ABS(A)<>5))
370 GOSUB 9200
380 NEXT RAD
390 NEXT KOL
400 CLOSE #1
410 CALL KEY(3,KEY,STA)
420 IF STA<1 THEN 410
430 GOSUB 9100
440 END
```

```
9000 REM INITIERA TEXT MODE
9010 CALL POKEV(-32272,0)
9020 CALL LOAD(-31788,240)
9030 CALL CLEAR
9040 FOR I=759 TO 959 STEP 10
9050 CALL POKEV(I,128,128,128,128,128,128,128,128,128,
     128)
9060 NEXT I
9070 CALL POKEV(-30731,0)
9080 RETURN
9100 REM AVSLUTA TEXTMODE
9110 CALL CLEAR
9120 CALL POKEV(-32288,0)
9130 CALL LOAD(-31788,224)
9140 CALL SCREEN(8)
9150 FOR I=0 TO 16
9160 CALL POKEV(783+I,16)
9170 NEXT I
9180 RETURN
9200 REM PRINT TEXTMODE
9210 FOR I=1 TO LEN(TEXT$)
9220 CALL POKEV((RAD-1)*40+KOL-2+I,ASC(SEG$(TEXT$,I,1)
     )+96)
9230 NEXT I
9240 RETURN
```

▶31

Tips och tricks

Jag bifograr ett antal programeringstips för både BASIC oc EXTENDED BASIC som jag tror kan vara intressant att publicera i PROGRAMBITEN NITTINIAN. De flesta av tipsen har jag aldrig sett i någon amerikansk eller svensk tidning.

Trångsund 1984-11-14

Jan Alexandersson

JAN ALEXANDERSSON
Springarvägen 5³
142 00 TRÅNGSUND
Tel. 08-771 05 69

VARIABELNAMN

Det är tillåtet att använda Å, Ä och Ö i variabelnamn.

] = Å ASCII-kod 93
[= Ä ASCII-kod 91
\ = Ö ASCII-kod 92

Följande program fungerar:

```
100 ]=3
110 [=4
120 \=]*[
130 PRINT \
```

Det är tillåtet att använda underprogramens namn även som variabel. Prova följande:

```
100 SCREEN=14
110 CALL SCREEN(SCREEN)
120 CALL KEY(3,KEY,STA)
130 IF KEY<>74 THEN 120
```

Observera att även @ och _ kan ingå i variabelnamn.

RADLÄNGDER

Maximal radlängd är 112 i basic och 140 i extended basic. Dessa kan förlängas genom editering av raden. Gränsen sätts ytterst av "Chrunch Buffer" som är 158 tecken både för basic och extended basic.

Gör så här i basic:

100 PRINT "AAAA...A"	100 st A
(ENTER)	112 tecken/rad
100 ↓ (ENTER)	editera flera A
100 PRINT "AAAA...A"	128 st A
(ENTER)	140 tecken/rad
100 ↓ (ENTER)	editera flera A
100 PRINT "AAAA...A"	155 st A
(ENTER)	rad med 167 tecken

Om du försöker med 156 st A blir det följande felmeddelande LINE TO LONG.

I extended basic är det något enklare. Du kan redan från början få 140 tecken på raden. Fortsätt sedan att editera som i basic-exemplet.

Hur kan man få in 167 tecken i en buffert på 158 tecken? Jo, det görs en översättning (kompilering) till en mer kompakt kod. Alla basic-komandon lagras med en byte.

Pröva följande i extended basic:

```
100 GOTO :: GOTO :: GOTO :: GOTO :: GOTO :: osv.
```

Genom upprepade editeringar kan man klämma in 78 st GOTO och 77 st ::. En programrad täcker hela skärmen! Med basic måste editeringen upprepas för varje rad om 28 tecken. I extended basic räcker det med var annan rad dvs efter 56 tecken.

Vid DATA-satser kan ej fulla rader användas.

SIZE

Efter det att ett program har körts eller det att beräkningar har gjorts direkt från tangentbordet(utan radnummer) finns minnesutrymme okuperat av numeriska variabler och strängvariabler mm. Om du endast vill veta hur långt programet är för att beräkna hur mycket plats det tar på kassetband eller flexskiva måste variabelminnet frigöras. Detta görs alltid vid editering. Ett lämpligt komando du kan använda är:

```
1 (ENTER)
SIZE
```

Observera att även om du har expansionsminne måste du tömma det statiska variabelminnet före SIZE.

Följande gäller nämligen:

PROGRAM SPACE = program + numeriska variabler
STACK SPACE = strängvariabler + återhoppsadresser

För att alltid kunna använda 1 (ENTER) på detta sätt får ej radnummer 1 användas i programet.

Även SAVE-komandod rensar variabelminnena.

TRACE

Om man använder TRACE för att söka fel kommer alla CALL CLEAR man passerar att sudda ut radnummer som TRACE skriver. Om man har extended basic kan man tillfälligt skriva till ett nytt underprogram i slutet av listningen som ändrar datorns inbyggda underprogram CLEAR. Prova följande:

```
100 CALL CLEAR
9000 SUB CLEAR
9010 SUBEND
```

På samma sätt kan man definiera om alla inbyggda CALL så att de gör något annat än TEXAS har tänkt.

INMATNING AV TEXTSTRÄNGAR

Det gäller att välja rätt komando för inmatning av text. I Basic finns det bara INPUT att välja på men i Extended Basic finns stor valfrihet. Det är dock stor skilnad på maximalt antal tecken:

INPUT i BASIC	111-112 tecken
INPUT i Extended Basic (eller LINPUT)	138-139 tecken
ACCEPT	255 tecken
ACCEPT AT	28 tecken

INPUT

Du bör ej använda (CTRL) , A B C i INPUT-satsernas ledtexter i Extended Basic. (CTRL) , A B C motsvarar ASCII-koderna 128-131. Om någon av dessa används vid numeriska variabler och du trycker på en bokstav borde man få WARNING och en ny chans att skriva rätt. I detta fall får man WARNING följt av SYNTAX ERROR och programet stoppar. Övriga ASCII-koder fungerar bra. I vanlig Basic finns inga problem med ASCII-koderna 128-131.

PRINT

Det är även möjligt att använda flera kolon i rad i PRINT-satser i Extended Basic. Tänk bara på att använda mellanslag mellan närliggande kolon vid editering. Efter det du tryckt på (ENTER) gör datorn en översättning till intern kod. Då lagras : för radmatning med ett unikt värde ASCII-kod 181 (som ej används i textsträngar) och dubbelkolon :: med ett eget värde ASCII-kod 130.
Datorn lagrar alltid dessa kolon(181) på samma sätt i både Basic och Extended Basic. Det är endast vid editering som det ser olika ut. Du kan alltid flytta program mellan Basic och Extended Basic när det gäller PRINT-satser.

TAB

I normala PRINT-satser kan fler än 28 tecken användas så att flera rader kan skrivas med samma komando t.ex.
PRINT " ABCDEFGHIJKLMNOPQRSTUVXYZ123"
Om man i stället använder TAB kommer det ej att fungera om texten skall skrivas på flera rader t.ex.
PRINT TAB(4);"ABCDEFGHIJKLMNOPQRSTUVXYZ123"
I detta fall skrivs texten ut längst till vänster.

DATA och READ

Man bör alltid lägga till en extra DATA-post på den sissta raden med DATA. Denna extra post skall aldrig läsas med READ men kommer att snabba upp programet. När datorn läser den sista posten kommer den annars att stanna upp en kort stund (någon sekund). Det kan vara lämpligt att använda något ovanligt tecken t.ex. @. Se följande exemplel:

```
1900 DATA ADAM,23,10,BERTIL,24,10,@
```

AND, OR, NOT, XOR

Dessa komandon saknas i basic men kan lätt omskrivas:

```
IF A=8 AND B=6 THEN ...          extended basic
IF (A=8)*(B=6) THEN ...          basic
IF A=8 OR  B=6 THEN ...          extended basic
IF (A=8)+(B=6) THEN ...          basic
IF A=8 XOR B=6 THEN ...          extended basic
IF (A=8)+(B=6)=-1 THEN ...       basic
IF NOT A=8 THEN ...              extended basic
IF A<>8    THEN ...              basic
```

MATEMATISKA FUNKTIONER

DEF av ARCSIN och ARCCOS i beskrivningen för Extended BASIC fungerar ej om X=1 respektive X=0. Använd i stället följande:

```
DEF ARCSIN(X)=2*ATN(X/(1+SQR(1-X*X)))
DEF ARCCOS(X)=2*ATN(1)-ARCSIN(X)
```

Dessa definitioner fungerar i både Basic och Extended Basic. Om du endast använder Extended Basic kan man byta ut 2*ATN(1) mot PI/2. Om endast ARCCOS används kan definitionen för ARCSIN bakas in i ARCCOS.
På motsvarande sätt kan man definiera PI i Basic:

```
DEF PI=4*ATN(1)
```

Man kan sedan använda PI i matematiska beräkningar som om PI fanns från början.

CALL CHAR

Om du har definierat om tecken 128-159 kommer dessa ej att försvinna när programet är slut. NEW kommer ej att sudda ut dessa.
Om du använder kassett kan du utnyttja denna finess. Om flexskiva används kommer dock tecknen att ändras till något oönskat. De nollställs nämligen ej utan får ett slumpartat innehåll.

CALL HCHAR och PRINT

Det är en viktig skillnad mellan basic och extended basic om CALL HCHAR(24,...,...) åtföljs av PRINT.
Prova följande:

```
100 CALL HCHAR(24,3,65)
110 PRINT :"B"
```

I basic får man A och B under varandra på skärmen som önskat. I extended basic kommer först A att skrivas som omedelbart suddas bort av B.
Detta är saker man måste tänka på när program skall flyttas från basic till extended basic.

CALL CHARSET

Detta återställer endast ASCII-kod 32-95 till ursprungligt CHAR men ej små bokstäver 96-143. Samtidigt återställs COLOR till ursprungligt värde för alla ASCII-koder 30-143 vilket är mycket ologiskt (Bug:).

CALL KEY

Om du använder delat tangentbord dvs CALL KEY(1,KEY,STA) eller CALL KEY(2,KEY,STA) är det ej tillåtet att testa IF KEY=0 THEN ... i basic utan man måste skriva IF KEY+1=1 THEN ... Det går dock bra att testa alla andra tangentvärden än 0.
I extended basic är det även tillåtet att testa KEY=0. CALL KEY(0,KEY,STA) testar endast tidigare tangentbord 3, 4 eller 5 men ej 1 och 2. I normala program skall man alltid använda CALL KEY(3,KEY,STA) så slipper man extra programrader som titta på J och j respektive N och n. Anledningen till att 0 står i alla programlistningar är att man vill kunna köra i både TI 99/4 och TI 99/4A. Vi som bara har 99/4A bör skriva 3.

CALL JOYST

Om man saknar joystick eller tycker att programet fungerar bättre med tangenttryckningar kan man ändra kommandot i extended basic på följande sätt:

```
9000 SUB JOYST(NR,X,Y)
9010 CAL KEY(NR,KEY,STA)
9020 X=-4*(KEY=3)+4*(KEY=2)
9030 Y=-4*(KEY=5)+4*(KEY=0)
9040 SUBEND
```

Detta underprogram kan man knappa in i slutet av ett gamalt program. Någon ändring i själva huvudprogramet behöver ej göras. Detta gör att du inte behöver leta upp alla olika programrader där CALL JOYST finns. Det är ingen risk med att NR, X, Y, KEY och STA även används i huvudprogramet för andra saker. KEY och STA är lokala variabler. NR, X och Y är endast utfyllnad för att markera ett överfört talvärde.

Om man har tillgång till flexskiva går det att spara ovanstående program med SAVE DSK1.JOYST,MERGE
När man sedan har ett färdigt program i datorn som man vill ändra från joystick till tangenter måste man först ta reda på om radnummer 9000 osv redan finns i programet. Skriv LIST 9000. Om datorn då tar fram en rad som är lägre än 9000 är det bra, använd annars RES av huvudprogramet eller underprogramet så att de ej kolliderar. Skriv sedan MERGE DSK1.JOYST

Om man endast har Basic eller programet ej fungerar i Extended Basic måste alla rader med CALL JOYST letas upp och ersättas med två nya rader:

```
xxxx NR=1
xxxy GOSUB 9000
```

I slutet av programet skrivs följande:

```
9000 REM JOYST(NR,X,Y)
9010 CALL KEY(NR,KEY,STA)
9020 X=-4*(KEY=3)+4*(KEY=2)
9030 Y=-4*(KEY=5)+4*(KEY+1=1)
9040 RETURN
```

Jämfört med det tidigare programet behöver du endast ändra SUB till REM och SUBEND till RETURN. I basic kan dock de nya variablerna kollidera med andra i huvudprogramet. Det är viktigt att KEY ej lagrar något värde för t.ex. FIRE från joystick som skall användas senare. Vid problem använd andra variabelnamn. Om du vill använda joystick 2 skriv xxxx NR=2. Om programet använder en variabel skriv xxxy NR=VARIABEL

Inget av de ovan beskrivna programen tar hänsyn till diagonalerna för JOYST. Den som vill använda dessa måste modifiera programet. Det är dock ganska krångligt att komma ihåg var diagonalerna sitter i snabba spelprogram.

CALL SOUND

Datorn är konstruerad för USA där man har 60 Hz i elnätet. Eftersom vi endast har 50 Hz kommer datorn att räkna för långsamt. CALL SOUND(1000,FRE,VOL) kommer att hålla på (60/50)*1000 ms dvs 1,2 sek. På samma sätt blir den längsta tillåtna tiden 5,12 sek om man skriver CALL SOUND(4250,FRE,VOL).
Denna förlängning med 20 % gäller endast om datorn bara har CALL SOUND att tänka på. I praktiken vill man ha ljud och musik samtidigt som grafiken förändras eller datorn gör uträkningar. Datorn hinner då inte med att räkna ned ljudkomandot utan man får en avsevärd ytterligare fördröjning. I värsta fall kan det ta dubbelt så lång tid för CALL SOUND när det följs av andra beräkningar. I praktiken är det svårt att förutse tiden. Man måste i stället prova sig fram med stoppur och ändra siffervärdet i CALL SOUND tills det stämmer.

Till slut. Tips och Tricks återkommer nästa år med mer CALL LOAD kommandon och Basic buggar. Vi har en del på lager att sukta er med. Många av oss har en okontrollerbar lust att gräva sig djupt ner i datorns inre, jag är en av dem som vill veta allt, och då menar jag verkligen allt, om en dator och dess byggnad hur systemprogrammeraren har kodat maskinen och hans idee' om hur en dator skall fungera.

Redaktionen.

■

◀◀◀◀

```
290 IF S=1 THEN 195
295 PRINT " ": " TRYCK MELLANSLAGSTANGENTEN"
300 CALL KEY(3,KEY,STAT)
305 IF STAT=0 THEN 300 ELSE 200
310 PRINT # S:"    KATALOGEN AVSLUTAD."
315 IF S=0 THEN 330
320 CLOSE # S
325 REM
330 LET ORD=ORD+1
335 PRINT "Detta var sista programmet" :
" nytt program har nr";ORD
340 PRINT
345 RADNR=ORD x 5+15
350 PRINT "Skriv: NUM";RADNR;",1  -ENTER"
355 PRINT "SKRIV IN DATASATSERNA PÅ DE" :
"FÖLJANDE FEM RADNUMREN."
360 PRINT
365 PRINT " på radnr";RADNR;"skrivs in:" :
" " : "PROGR NR(";ORD;"), PROGNAMN,"
370 PRINT "PROGSPRÅK, PROGSTART, PROG- SLUT
och KRINGUTRUSTNING."
375 PRINT
380 PRINT "RADNR";RADNR+1;"och de tre följ
ande"
385 PRINT "beskriver programmet." : " " :
"avsluta med  SAVE CS1  -ENTER. "
390 FOR PAUS=0 TO 2000
395 NEXT PAUS
400 END
```

■

KASSETT INNEHÅLL

Gävle 1984.11.16

Tidningsredaktionen

Programbiten NITTINIAN

Först vill jag framhålla att Ni gör en bra tidning för oss 99-användare.

Jag har försökt hörsamma Lennarts upprop om mera aktivitet från oss som programmerar i TI BASIC / EXTENDED BASIC.

Kan Ni ha någon användning i tidningen för bifogade program? Ni får klippa och ändra som Ni tycker men ange gärna källan.

Med vänlig datorhälsning

Sven-Erik Wind

Takaregat 21

80238 Gävle

Allt eftersom programbiblioteket har vuxit i omfattning, har jag funnit det mer och mer angeläget, att på ett enkelt sätt lagra kortfattade beskrivningar om de olika program, som finns på respektive kassett.

Nedanstående program lagras på vanligt sätt i början av varje kassettsida, gärna redan vid bandstart för startremsan hinner passera innan programmet börjar spelas in. På så sätt blir det enkelt att senare hitta programmens plats på bandet.

Beskrivning av programmet.

Rad 7-9 svenska tecken med rätt ASCII-kod
FCTN] = Å
FCTN [= Ä
FCTN \ = Ö

Rad 20-94 information om programmen ligger här i datasatser

Rad 105 ger namn åt kassetten. Du väljer här Din egen beteckning på bandet

Rad 125-195 dessa kan strykas om Du ej har printer

Rad 155 ändra denna rad om Du har serieingång på printern

Rad 205-215 två första data läses för att se om fler program finns

Rad 235-280 data om programmet läses och skrivs ut

Rad 295-305 nästa program läses

Rad 315-320 filen till skrivare stänges. Stryk dessa rader om Du ej har printer

Rad 345 ger algoritmen för radnumret till första datasatsen i programbeskrivn

Rad 350 automatiska radnumreringen ökas 1

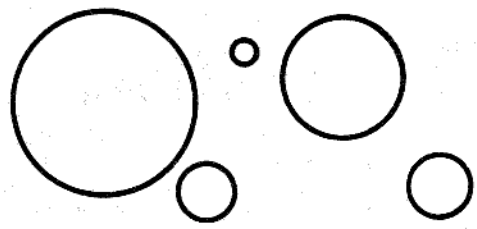
Rad 365-385 här visas hur /99,SLUT/ ersätts med uppgifter för att identifiera programmet och dess läge på bandet samt om det behövs styrspak t ex. Kommatecken sätts mellan varje data

De fyra följande raderna kan innehålla kortfattad beskrivning om programmet. Börja varje rad med DATA. Använd ej kommatecken för det ställer till problem (dataavskiljare).

Glöm ej att SAVEa från bandstart när nya uppgifter matats in.

KATALOGPROGRAM för KASSETTHANTERING I TI-BASIC/ EXTENDED BASIC

```
1 REM PROGRAMREG I TI-BASIC/ EXT BASIC
2 REM 1984.10.15
3 REM SVEN-ERIK WIND
4 REM TAKAREGAT 21
5 REM 802 38 GEFLE
6 REM TEL 026-191733
7 CALL CHAR(93,"0010003844C4444")
8 CALL CHAR(91,"0028003844C4444")
9 CALL CHAR(92,"0028007C4444447C")
10 REM
20 DATA 99,SLUT
25 DATA 99,SLUT
30 DATA 99,SLUT
35 DATA 99,SLUT
40 DATA 99,SLUT
45 DATA 99,SLUT
50 DATA 99,SLUT
55 DATA 99,SLUT
60 DATA 99,SLUT
65 DATA 99,SLUT
70 DATA 99,SLUT
75 DATA 99,SLUT
80 DATA 99,SLUT
85 DATA 99,SLUT
90 DATA 99,SLUT
95 DATA 99,SLUT
100 CALL CLEAR
105 PRINT TAB(5);"PROGRAMKATALOG FÖR:" : " " :
"TI-99 PROGRAMBAND 1 sid A"
110 PRINT " ":" UPPGIFTERNA VISAS FÖR ETT PRO
GRAM ÅT GÅNGEN." : " " : " " :
115 PRINT
120 REM
125 PRINT "VILL DU HA PROGRAMKATALOGEN PÅ
SKÄRMEN= 0 PÅ PRINTER= 1" : " " :
130 INPUT "SVARA MED SIFFRA ":S
135 PRINT
140 IF S=0 THEN 200
145 REM
150 REM
155 OPEN #S:"PIO/1"
160 INPUT " DAGENS DATUM":DAT$
165 LET BAND$ =" TI-99 PROGRAMBAND 1 sid A"
170 PRINT #S:"Band : ";BAND$
175 PRINT #S
180 PRINT #S:"KATALOGUTSKRIFT ";DAT$
185 PRINT #S
190 PRINT #S
195 PRINT #S
200 PRINT #S
205 READ N, PRO$
210 IF N=99 THEN 310
215 LET ORD=N
220 REM
225 REM
230 READ SP$, ST, SL, UTR$
235 PRINT #S:"PROGRAM :";N; PRO$
240 PRINT #S:"PLATS   :";ST;"-";SL
245 PRINT #S:"SPRÅK   :";SP$
250 PRINT #S:"UTRUSTN :";UTR$
255 REM
260 PRINT #S:"BESKRIV :";
265 FOR I=1 TO 4
270 READ TXT$
275 PRINT #S:TXT$
280 NEXT I
285 PRINT #S
```

RITA  i FORTH

Av Micael Dahlquist.

När jag hade inhandlat föreningens FORTH var mitt stora mål att i FORTH göra saker som inte kunde göras lika bra (eller inte alls) i BASIC (TI BASIC eller EXTENDED BASIC). Nyligen presenterade jag en rutin som sorterade tal. Det gick att göra i BASIC men med längre exekveringstid. Den här gången har jag gjort vissa rutiner för att kunna "plotta" cirklar i grafik med högupplösning (till alla som vill försöka i BASIC önskar jag lycka till!). I senare nummer hoppas jag kunna visa rutiner som exempelvis drar streck. Därefter hoppas jag kunna skriva om delar av FORTH-programmen i ASSEMBLER, vilket ger ännu högre hastighet (som bekant är det ju lätt att inom FORTH använda sig av ASSEMBLER).
Med de rutiner som presenteras här går det lätt att plotta och rita cirklar i alla de 16 (om man räknar genomskinlig som en färg...) olika färgerna. För att kunna skriva något på skärmen - för att ge FORTH-kommandon - var jag dock tvungen att offra en bit av grafikskärmen till text. På grund därav är endast 160*256 punkters upplösning (av de 192*256 möjliga) tillgängliga för grafik. Detta ger ett tecken-fönster som är 32 tecken långt och 4 tecken högt.

Redaktörens kommentar:

Du som läser det här - skriv nu gärna egna plottingrutiner flitigt. Använd de publicerade som underlag om Du vill. Skicka dem sedan till tidningen med vidhängande artikel. Om Dina program är snabba eller smarta eller roliga eller av någon annan anledning intressanta skall vi försöka publicera dem. Lägg märke till att grafiken - gärna i färg - är ett område med stora och intressanta möjligheter. Häng med!

Vi vill tacka både Micael och Örjan för smarta rutiner - men vi vill också framhålla att de, liksom många andra, säkerligen kan ytterligare utveckla sitt sätt att lägga upp FORTH-programmen. Det handlar om ordens lämpliga längd, utnyttjandet av utrymmet på FORTH-skärmarna, benämningarna på orden, kommentarerna - såväl mängd som redigering - , vissa konventioner rörande t ex BASE etc. Det finns intressanta synpunkter på sådant bl a i Brodies senaste bok Thinking FORTH. Både han och FORTH's skapare Charles Moore brukar framhålla hur viktigt det är med hanterbara, lättbegripliga, välförklarade FORTH-program. Vi skall försöka få med en del av detta i tidningen - kanske vi lyckats redan på annan plats i detta nummer.

```
SCR #81
  0 ( GRAFIK I FORTH !  1 AV 3 ) HEX
  1 : S SP@ S0 @ = IF CR ." EMPTY " ELSE . THEN ;
  2
  3   F4 VARIABLE BCO 80 VARIABLE X 70 VARIABLE Y
  4 : VWTR 8 SYSTEM ;
  5
  6 : BIT-MAP 0 6 VLOAD 1800 SCRN_START ! 1B80 SCRN_END !
  7   20 SCRN_WIDTH ! 1C00 PABS ! 3800 DISK_BUF !
  8   E200 83D4  ! 2 0 VWTR E2 1 VWTR  6 2 VWTR FF 3 VWTR
  9     D0 1B00 V! 3 4 VWTR 36 5 VWTR  7 6 VWTR F4 7 VWTR
 10   3 0 DO 100 0 DO I 1800 I J 100 * + + V! LOOP LOOP ;
 11
 12 : BPC 1B00 0 DO I 3FF > IF 0 I V! THEN F4 I 2000 + V! LOOP ;
 13
 14 : TO-B BIT-MAP BPC PAGE ;
 15 -->
SCR #82
  0 ( GRAFIK I FORTH !  2 AV 3 )
  1
  2 : P0 <BUILDS 0 DO C, LOOP DOES> + C@ ;
  3   01 02 04 08 10 20 40 80 8 P0 P1
  4
  5 : PLOT SWAP DUP 7 AND P1 >R F8 AND OVER 7 AND + SWAP F8 AND 20
  6   * + DUP V@ R> OR OVER V! 2000 + BCO @ SWAP V! ; DECIMAL
  7
  8 : DIV SWAP OVER /MOD ROT ROT 2/ SWAP < IF 1+ THEN ;
  9
 10 : 2ROT 1 BEGIN 2DUP DIV OVER + 2/ SWAP OVER = UNTIL SWAP DROP ;
 11
 12 : CIRCLE DUP DUP * SWAP 4 * 5 / 1+ DUP -1 * 1+ DO DUP I I * -
 13   2ROT DUP Y @ + I X @ + SWAP PLOT DUP -1 * Y @ + I X @ + SWAP
 14   PLOT DUP X @ + I Y @ + PLOT -1 * X @ + I Y @ + PLOT LOOP DROP
 15   ; -->
SCR #83
  0 ( GRAFIK I FORTH !  3 AV 3 )
  1
  2 : T1 15 -15 DO 148 108 DO J J * 2ROT 16 * 4 + BCO ! I 112 J 2 *
  3   + PLOT LOOP LOOP ;
  4
  5 : T2 112 Y ! 210 40 DO I X ! 7 1 DO I CIRCLE LOOP 20 +LOOP ;
  6
  7 : T3 128 X ! 80 50 112 DO I Y ! 5 - DUP CIRCLE -5 +LOOP DROP ;
  8
  9 : T4 11 -10 DO 128 I 2 * + 80 I I * + PLOT LOOP ;
 10 : T5 5 -4 DO 128 I 3 * + 112 I I I * * + PLOT LOOP ;
 11
 12 : T6 128 112 BEGIN ?KEY CASE 2 OF 1 ENDOF 69 OF 1 - 2DUP PLOT 0
 13   ENDOF 83 OF SWAP 1 - SWAP 2DUP PLOT 0 ENDOF 68 OF SWAP 1 +
 14   SWAP 2DUP PLOT 0 ENDOF 88 OF 1 + 2DUP PLOT 0 ENDOF 0 ENDCASE
 15   UNTIL DROP DROP ;
```

De olika variablerna i rutinerna är:

BCO (BitCOlor): innehåller förgrundsfärg samt bakgrundsfärg för all grafik (ej teckenfärg). Används i orden PLOT och CIRCLE.

Ex. HEX 14 BCO ! DECIMAL

Förgrundsfärg: 1 (svart)
Bakgrundsfärg: 4 (mörkblå)

X (X position): innehåller värdet på cirkelns mittpunkts X-koordinat. Används i ordet CIRCLE.

Ex. 128 X !

X värde: 128

Y (Y position): innehåller värdet på cirkelns mittpunkts Y-koordinat. Används i ordet CIRCLE.

Ex. 112 Y !

Y värde: 112

De olika orden är:

VWTR (Vdp Write To Register): kräver två värden på stacken - talet som skall skrivas till registret och registernumret. Används i ordet BIT-MAP.

Ex. HEX F4 7 VWTR DECIMAL

Skriver värdet F4 (hex) till register 7 (i det här fallet ställs förgrundsfärg = F (vit) och bakgrundsfärg = 4 (mörkblå)).

BIT-MAP (BIT-MAP mode): går in i bit-map mode samt ordnar screen image table osv. Behöver ej några tal på stacken. Används i ordet TO-B.

BPC (Bit-map mode Pattern and Color clear): rensar pattern descriptor table (nollställer) och color table (vitt på blått). Behöver ej några tal på stacken. Används i ordet TO-B

TO-B (TO Bit-map mode): initierar bit-map mode.

P0 och PLOT: P0 är ett definierande ord som används för att skapa ordet P1, vilket i sin tur används av ordet PLOT. Den här konstruktionen är en något modifierad version av motsvarande i WYCOVE FORTH. (WYCOVE FORTH är en FORTH-variant för TI 99/4A som tagits fram av ett kanadensiskt företag Wycove Systems Limited. Red anm.). P1 kräver X- och Y-koordinater på stacken samt önskad färgkombination i variabeln BCO. Använder alltså inte variablerna X och Y. (1<=X<=256, 32<=Y<=192)

Ex. 128 112 PLOT

Sätter en punkt på X-koordinaten 128 och Y-koordinaten

112 i den färgkombination som valts i BCO.

DIV (DIVide): kräver två tal på stacken och ger kvoten av dessa - avrundad! Används i ordet 2ROT.

Ex. 2 3 DIV

Lämnar talet 1 på stacken (2/3).

2ROT (kvadratroten): ger kvadratroten ur det tal som finns på stacken. Används i ordet CIRCLE.

Ex. 81 2ROT

Lämnar talet 9 på stacken (9*9=81).

CIRCLE: med given radie på stacken ritar ordet en cirkel med mittkoordinater hämtade från variablarna X och Y. Cirkeln ritas med färgen som angivits i variabeln BCO.

Ex. 15 CIRCLE

Ritar en cirkel med radien 15 med mittkoordinater bestämda i X och Y samt med färgen i BCO.

Utöver dessa ord finns också ett antal test-ord (T1-T6).

T1-T5 ritar förutbestämda figurer/mönster.

Med hjälp av T6 kan Du 'rita'. Styr med E,S,D OCH X (piltangenterna).

OBS ! Om plottning sker utanför angivna värden på X- och Y-axeln kan datorn 'bryta' eller på annat sätt bete sig underligt. Det har heller ej kommit till min kännedom hur man undviker att stänga av disk-enheten då man kommit in i bit-map mode (datorn är stendöd då disk anropas!).Därför har jag inte gjort något ord för att komma tillbaka till textmode (men detta och mer kan komma redan i nästa nummer...).
Vid vissa felmeddelanden läser datorn från disk och skulle då bli hängande. Därför har jag försökt ta bort åtmonstone en av dessa orsaker till irritation. Använd således ordet S istället för . (punkt) när Du ska skriva ut tal från stacken i samband med de här rutinerna. Ordet S skriver EMPTY (precis som S.) då stacken är slut (. läser felmeddelande från disk) ■

Spelprogram från ATARI

Av Tony Hall

Jag har fått tag i ett antal av de ATARI-spel som nu anpassats till TI 99/4A och spelat igenom dem. Antagligen har det ett visst intresse att tala om för andra hur de är. Därför har jag skrivit ned en del av vad jag tycker om dem. Jag börjar här med Donkey Kong. Fler kanske följer senare.

Donkey Kong.

Det här är ju ett välkänt spel för alla som sysslat med de små batteridrivna handspelen. Det handlar alltså om den stora apan/gorillan Donkey Kong som rövat bort Din flickvän och håller henne fången högst uppe på en byggnadsställning. Vad han skall ha henne till där är väl inte helt klart, men vi känner ju igen motivet från t ex filmerna om King Kong.
Din uppgift är att ta Dig upp till flickan för att rädda henne. Du klättrar på byggnadsställningen allt medan Kong försöker hindra Dig genom att från sitt överläge kasta tunnor på Dig. Det gäller för Dig att med skicklighet undvika tunnorna och avancera mot flickan.
Spelet innehåller totalt fyra olika omväxlande spelplaner. Grafiken är mycket bra - i samma klass som på originalspelet, och det är gott betyg. För den som tycker att originalspelet är för svårt är detta perfekt. Man behöver bara spela ett par gånger på varje bana för att klara av spelet.
Vad jag inte tycker om är att man inte får se när Kong tar med sig flickan till nästa bana eller när han ramlar ned från ställningen.
Vid en betygsättning från 1 till 10 vill jag ge grafiken en sjua. Den bristande omväxlingen drar ned helhetsbetyget - en tio-femton spelplaner kunde man väl ha kostat på sig - och jag slutar på sex som sammanfattning.

Jag har mera på väg: Moon Patrol, Defender, Pac Man etc. Spelen finns i handeln nu. ■

TEXTMODE I MINI MEMORY

◀ 25

```
100 REM TEXT MODE MM/EA
110 REM JAN ALEXANDERSSON
120 REM VERSION 1984-11-29
130 REM REF PROGRBIT 84-01
140 GOSUB 9000
150 TEXT$="TEST AV TEXT-MODE MED MINI-MEMORY ELLER ED
    ITOR/ASSEMBLER"
160 RAD=10
170 KOL=1
180 GOSUB 9200
190 TEXT$="TRYCK <ENTER>"
200 RAD=24
210 KOL=1
220 GOSUB 9200
230 CALL KEY(3,KEY,STA)
240 IF STA<1 THEN 230
250 GOSUB 9100
260 END
9000 REM INITIERA TEXT MODE
9010 CALL POKEV(-32272,0)
9020 CALL LOAD(-31788,240)
9030 CALL CLEAR
9040 FOR I=759 TO 959 STEP 10
9050 CALL POKEV(I,128,128,128,128,128,128,128,128,128,
     128)
9060 NEXT I
9070 CALL POKEV(-30731,0)
9080 RETURN
9100 REM AVSLUTA TEXTMODE
9110 CALL CLEAR
9120 CALL POKEV(-32288,0)
9130 CALL LOAD(-31788,224)
9140 CALL SCREEN(8)
9150 FOR I=0 TO 16
9160 CALL POKEV(783+I,16)
9170 NEXT I
9180 RETURN
9200 REM PRINT TEXTMODE
9210 FOR I=1 TO LEN(TEXT$)
9220 CALL POKEV((RAD-1)*40+KOL-2+I,ASC(SEG$(TEXT$,I,1)
     )+96)
9230 NEXT I
9240 RETURN
```

■

Plotter till FORTH

Av Örjan Gustavsson

(Denna artikel är skriven med en skönskrivare med begränsad teckenuppsättning. "Alfa-slang" representeras därför av paragraf-tecken och "tak" (exponentiering i matematiska formler) är ritat för hand.
I programlistningen däremot används en EPSON-skrivare med dansk uppsättning tecken, som ger de ovannämnda tecknen rätt men som ger danska å, ä och ö.)

Här kommer ett program som borde intressera de flesta FORTH- programmerare. Det är en plotter som ritar grafen till en funktion. Med den följer även en hel del användbar grafik och flyttalsrutiner. Plottern har jag själv tillverkat, men grafik- och flyttalsrutinerna kommer från TI FORTH version 3.0. Jag har gjort vissa kompletteringar.

Flyttalsrutinerna.

Flyttalsrutinerna anropar GPL-rutiner, alltså samma rutiner som används av TI Basic. Därför blir de nästan skrämmande långsamma, men jag jobbar på att få fram snabbare rutiner. De kanske kommer i ett senare nummer.

Ett flyttal enligt de rutiner som här används representeras i 99-an av fyra ord - åtta bytes. Flyttal kräver alltså fyra gånger så mycket minne som heltal. En sak att ha i minnet (häpp) när man programmerar, blir alltså att använda så få flyttalsvariabler som möjligt.

För att markera att man avser att mata in ett flyttal skriver man en ampersand (&) före talet, som eljest skrivs precis som i Basic med punkt i stället för komma. På stacken hamnar då fyra ord som representerar talet. Givetvis bör man undvika att använda de vanliga stackhanteringsorden, som är avsedda att användas för heltal. I stället bör man använda de speciella stackhanteringsord för flyttal som kommer med flyttalspaketet: FDUP, FSWAP, FROT och FOVER.

Flyttalspaketet innehåller i övrigt följande ord:

F. Skriver ut det flyttal som finns på stacken.

& Lägger ett flyttal på stacken.

F§ (a -- f1) Hämtar ett flyttal vid adressen a.

F! (f1 a -- f1) Lagrar flyttalet f1 vid adressen a.

F->S (f1 -- n) Konverterar flyttalet f1 till ett heltal n.

S->F (n -- f1) konverterar heltalet n till ett flyttal f1.

F+, F-, F*, F/ Fungerar som heltals varianterna fast med flyttal.

```
SCR #20
  0 ( PLOTTER  1 )
  1 23 LOAD   FVARIABLE X
  2 FVARIABLE XMIN FVARIABLE XMAX  FVARIABLE YMIN
  3 FVARIABLE YMAX FVARIABLE SCALX FVARIABLE SCALY
  4 FVARIABLE STAX FVARIABLE SLUX  FVARIABLE SCALI
  5 : TDOT  ( x y -- )  ( Sätter dot om 0<=X<=255 0<=Y<=192 )
  6    MINUS 192 + OVER OVER DUP 192 > IF 2DROP 2DROP ;S THEN
  7    0 < IF 2DROP DROP ;S THEN DUP 0=<  IF 2DROP DROP ;S THEN
  8    255 > IF 2DROP ;S THEN DOT ;
  9 : Xn  ( -- n )  SLUX F@ STAX F@ F- SCALX F@ F/ ;
 10 HEX
 11 : RDTEMP ( -- )   03C0 HERE 20 VMBR ;
 12 : WRTEMP ( -- )   HERE 03C0 20 VMBW ;
 13 : F(x)   ( f11 -- f12 ) ( anropar funktion )
 14     RDTEMP LATEST PFA CFA EXECUTE WRTEMP ;
 15     DECIMAL -->

SCR #21
  0 ( PLOTTER 2 )
  1 : SCALE ( -- )   XMAX F@ XMIN F@ F- & 256 F/ SCALX F!
  2                  YMAX F@ YMIN F@ F- & 190 F/ SCALY F! ;
  3 : SINT  ( -- )   SCALE SLUX F@ STAX F@ F- Xn F/ SCALI F! ;
  4 : AXIS  ( -- )   & 0 YMIN F@ F- SCALY F@ F/ F->S MINUS 192 +
  5                  DUP 0 SWAP ROT 256 SWAP LINE
  6                  & 0 XMIN F@ F- SCALX F@ F/ F->S
  7                  DUP 0 ROT 192 LINE ;
  8 HEX
  9 : COLOR ( -- )   0000 1800 DCOLOR @ VFILL DCOLOR @ 7 VWTR ;
 10 DECIMAL
 11 : INIT  ( -- )   STAX F@ X F! SINT Xn GRAPHICS2 COLOR AXIS ;
 12 : COLAD ( -- c ) XMIN F@ F- SCALX F@ F/ F->S ;
 13 : RADAD ( -- r ) YMIN F@ F- SCALY F@ F/ F->S ;
 14 : PLOT  ( -- )   INIT F->S 1+ 0 DO X F@ F(x) RADAD X F@ COLAD
 15                  SWAP TDOT SCALI F@ X F@ F+ X F! LOOP ; -->

SCR #22
  0 ( PLOTTER 3 )
  1 : X-MAX   ( -- )   ¢COMPILE& & XMAX F! ;
  2 : X-MIN   ( -- )   ¢COMPILE& & XMIN F! ;
  3 : Y-MAX   ( -- )   ¢COMPILE& & YMAX F! ;
  4 : Y-MIN   ( -- )   ¢COMPILE& & YMIN F! ;
  5 : START-X ( -- )   ¢COMPILE& & STAX F! ;
  6 : SLUT-X  ( -- )   ¢COMPILE& & SLUX F! ;
  7 : FX2     ( -- )   FDUP F* ;   ( X*X )
  8 : FX3     ( -- )   FDUP FX2 F* ; ( X*X*X )
  9 : DUMMY ;
 10
 11
 12
 13
 14
 15

SCR #23
  0 0 VARIABLE VDPMDE
  1 : VSBW 0 SYSTEM ;
  2 : VWTR 8 SYSTEM ;
  3 : VSBR 4 SYSTEM ;
  4 : VFILL 20 SYSTEM ;
  5 : VOR 24 SYSTEM ;
  6 : VAND 22 SYSTEM ;
  7 : VXOR 26 SYSTEM ;
  8 24 LOAD HEX
  9 : TEXT-MODE ( Sätt i text mode )
 10 0 3C0 20 VFILL 28 SCRN_WIDTH ! 0 SCRN_START !
 11 3C0 SCRN_END ! 460 PABS ! SETVDP1 2 VDPMDE !
 12 1 6 VWTR 0F4 7 VWTR 0F0 SETVDP2 ;
 13 DECIMAL
 14
 15

SCR #24
  0 ( CONVERT TO GRAPHICS2 MODE CONFIG 14SEP82 LAO)
  1 HEX : GRAPHICS2  0A0 1 VWTR
  2 -1 1B00 1800 DO 1+ DUP 0FF AND I VSBW LOOP DROP
  3 1 PABS @ VSBW 16 PABS @ 1+ VSBW 1 ( #FILE) 834C C! PABS @ 8356 !
  4  0A 0E SYSTEM ( SUBROUTINE TYPE DSRLNK TO SET 2 DISK BUFFERS )
  5 0 1800 0F0 VFILL ( INIT COLOR TABLE )
  6 2000 1800 0 VFILL ( INIT BIT MAP )
  7 20 SCRN_WIDTH ! 1800 SCRN_START ! 1B00 SCRN_END ! 1B00 PABS !
  8 1C00 DISK_BUF ! ( USER VARIABLES NOW SET UP )
  9  2 0 VWTR      6 2 VWTR  ( SET VDP REGISTERS )
 10 07F 3 VWTR      0FF 4 VWTR
 11 70 5 VWTR       7 6 VWTR
 12 0F1 7 VWTR      0E0 DUP 83D4 C! 1 VWTR    1BC0 836E ! ( VSPTR )
 13 0 0 GOTOXY 4 VDPMDE !  0 837A C!  ;
 14 -->
 15
```

Även F>, F<, F=, F0=, F0< fungerar som heltalsvarianterna.

Flyttalsrutinerna beskrivs bättre - med fullständiga screens - i manualen till TI-FORTH version 3.0, som kan köpas från föreningen.

Grafikrutinerna.

Grafikrutinerna består av bl a dessa ord:

DOT (x y --)

LINE (x1 y1 x2 y2 --)

DRAW (--)

DTOG (--)

UNDRAW (--)

GRAPHICS2 sätter TI 99/4A i bitmap-mode med upplösningen 256x192 pixels. DOT tänder en pixel på skärmen. LINE drar en linje från x1 y1 till x2 y2. DRAW sätter skärmen i DRAW-mode vilket gör att orden DOT och LINE kommer att tända punkterna som definieras. DTOG gör att DOT och LINE kommer att skifta alla punkter som defineras, d.v.s en tänd punkt släcks och en släckt punkt tänds. UNDRAW gör att orden DOT och LINE släcker alla punkter som defineras. TEXT-MODE sätter TI 99/4A i textmode (FORTH initieras i text-mode). DCOLOR är en variabel som innehåller pixelfärgen på de punkter som skall plottas - OBS! inte på dem som redan är plottade. Enklaste sättet att välja färg är att mata in värdet i DCOLOR som ett hexadecimalt tal. Den första siffran är då pixelfärgen och den andra bakgrundsfärgen. F0 t ex ger vita pixels på svart bakgrund. DCOLOR styr även färgen på skärmen och på grafen i plotter-rutinen.

Plottern

Plottern är ganska flexibel och klarar alla funktioner som kan uttryckas som ett FORTH-ord och som är en funktion från x till y. Det aktuella x-värdet ligger på stacken när funktionen anropas. Plottern förväntar sedan att det ligger ett flyttal på stacken när funktionen är uträknad. Exempelvis funktionen sin (x)/x programmeras så här:

```
: FUNC FDUP SIN FSWAP F/ ;
```

OBS! eftersom plottern alltid anropar det sista ordet i ordlistan måste man se till att funktionen är definierad sist av alla definitioner i ordlistan. När plottern laddas in ligger ordet DUMMY sist i ordlistan. DUMMY är definitionen för funktionen Y=X, vilken ger en rak linje.

Innan man plottar en kurva måste man ange min och maxvärden för x och y-axlarna. De matas här in med orden X-MIN, X-MAX, Y-MIN , Y-MAX, START-X och SLUT-X. Man skriver först ordet och sedan värdet som flyttal.

Inmatning av lämpliga värden för funktionen ovan kan se ut så här:

```
X-MIN     -12.56637061 ( -4PI )
X-MAX      12.56637061 (  4PI )
Y-MIN      -0.2
Y-MAX       1.0
START-X   -12.56637061
SLUT-X     12.56637061
```

START-X och SLUT-X anger definitionsmängden för x, alltså inom vilket område x skall variera. OBS! START-X och SLUT-X måste alltid anges även om de har samma värde som

```
SCR #25
  0 ( VDPMODES 14SEP82 LAO )
  1  HEX
  2  : SETVDP1 0B0 1 VWTR      ( BLANK THE SCREEN )
  3     800 B00 OFF VFILL      ( INIT 256 CHAR PATTERNS TO FF )
  4     800 6 VLOAD ;
  5 : SETVDP2 ( n --- )  460 PABS !
  6     1000 DISK_BUF !        ( RESTORE USER VARIABLES )
  7     ( SET VDP REGISTERS )
  8     0 0 VWTR  0 2 VWTR  0E 3 VWTR
  9     1 4 VWTR  6 5 VWTR
 10     3E0 836E !             ( VSPTR )
 11 1 PABS @ VSBW 16 PABS @ 1+ VSBW 3 ( #FILE) 834C C! PABS @ 8356 !
 12     0A 0E SYSTEM           ( SUB TYPE DSRLNK TO SET 3 DISK BUF )
 13     0 0 GOTOXY  0 837A C!
 14     DUP 83D4 C! 1 VWTR ;  -->
 15

SCR #26
  0 HEX
  1 0 VARIABLE DMODE F5 VARIABLE DCOLOR
  2 : DRAW 0 DMODE ! ; : UNDRAW 1 DMODE ! ; : DTOG 2 DMODE ! ;
  3 8040 VARIABLE DTAB 2010 , 804 , 201 , 7FBF , DFEF , F7FB ,
  4 FDFE , 8040 , 2010 , 804 , 201 ,
  5 -->
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

SCR #27
  0 ( GRAPHICS PRIMITIVES )  HEX
  1 CREATE DDOT   C079 ,
  2      C0D9 , C081 , C103 , 0241 ,
  3      0007 , 0243 , 0007 , 0242 ,
  4      00FB , 0244 , 00F8 , 0A52 ,
  5      A042 , A044 , 0221 , 2000 ,
  6      04C4 , D123 , DTAB , 06C4 ,
  7      C644 , 0649 , C641 , 045F , SMUDGE
  8 : DOT ( X Y --- )
  9   DDOT DUP 2000 - >R DMODE @
 10   CASE  0 OF VDR  ENDOF  ( DRAW )
 11         1 OF SWAP FF XOR SWAP VAND ENDOF  ( UNDRAW )
 12         2 OF VXOR ENDOF  ( TOGGLE )
 13   DROP DROP ENDCASE R>
 14   DCOLOR @ 0 < IF DROP ELSE DCOLOR @ SWAP VSBW ENDIF ;
 15 -->

SCR #29
  0 ( FLOATING POINT <4 WORD> STACK ROUTINES 12JUL82 LCT)
  1 HEX
  2 : FVARIABLE  0 VARIABLE 6 ALLOT  HERE 6 - 6 0 FILL ;
  3 : FDUP SP@ DUP 2- SWAP 6 + DO I @ -2 +LOOP ;
  4 : FDROP DROP DROP DROP DROP ;
  5 : FOVER SP@ DUP 6 + SWAP E + DO I @ -2 +LOOP ;
  6 : FSWAP FOVER >R >R >R >R >R >R >R >R
  7         FDROP R> R> R> R> R> R> R> R> ;
  8 : F! 4 0 DO DUP >R ! R> 2+ LOOP DROP ;
  9 : F@ 6 + 4 0 DO DUP >R @ R> 2- LOOP DROP ;
 10 834A CONSTANT FAC  835C CONSTANT ARG
 11 : >FAC FAC F! ;   : >ARG ARG F! ;  : FAC> FAC F@ ;
 12 : FROT  >FAC FSWAP FAC> FSWAP ;
 13 : SETFL >FAC >ARG ;
 14 : FADD 0600 C SYSTEM ;  : FSUB 0700 C SYSTEM ;
 15 : FMUL 0800 C SYSTEM ;  : FDIV 0900 C SYSTEM ; -->

SCR #28
  0 ( GRAPHICS PRIMITIVES 12JUL82 LCT)  HEX
  1 : SGN DUP IF DUP 0< IF -1 ELSE 1 ENDIF ELSE 0 ENDIF + ;
  2 : LINE >R R ROT >R R - SGN SWAP >R R ROT >R R - SGN OVER ABS
  3   OVER ABS < >R R 0= IF SWAP ENDIF 100 ROT ROT */ R>
  4   IF ( X AXIS ) R> R> OVER OVER >
  5      IF ( MAKE L TO R ) SWAP R> DROP R>
  6      ELSE R> R> DROP
  7      ENDIF 100 * ROT ROT 1+ SWAP
  8      DO I OVER 0 100 M/ SWAP DROP DOT OVER + LOOP
  9   ELSE ( Y AXIS ) R> R> R> R> ROT >R ROT >R OVER OVER >
 10      IF ( MAKE T TO B ) SWAP R> DROP R>
 11      ELSE R> R> DROP
 12      ENDIF 100 * ROT ROT 1+ SWAP
 13      DO DUP 0 100 M/ SWAP DROP I DOT OVER + LOOP
 14   ENDIF DROP DROP ;
 15 -->
```

X-MIN och X-MAX.

OBS! OBS! Detta är mycket viktigt:

Om man första gången man kör plottern inte har matat in max- och minvärden samt start- och slutvärden får man ganska säkert en maskinkrasch!

PLOT startar plottningen av grafen. Programmet räknar själv ut skalfaktorer för x och y axlarna, och ritar ut axlarna - dock ograderade, men man kan ju inte få allt.

När plottningen är klar stannar datorn kvar i bit map-mode. För att komma tillbaks till text-mode skriver man: TEXT-MODE. Eftersom det man skriver inte syns på skärmen är det lätt att skriva fel. Skriv en gång till om inget händer!

Med flyttalsrutinerna följer även ord för SIN, COS, TAN, ATN, e^x (EXP), x^y (^), SQR, LOG. Om man använder dessa ord tillstöter det en komplikation - de använder nämligen delar av VDP-minnet till att lagra mellanresultat och det är samma VDP-minne som används även till färgtabellen i bitmap-mode. Detta resulterar i att om man t ex anropar SIN i funktionen tänds det en plump uppe i högra hörnet på skärmen. För att komma tillrätta med det här problemet måste man spara undan den utsatta minnesarean för att - när funktionen är utförd - skriva tillbaka den i minnet. Detta i sin tur medför att eftersom GPL-rutinerna knappast kan kallas snabba, kommer 'plumpen' att blinka irriterande långsamt. Det här får man nog stå ut med emellertid. Plumpen försvinner ju i alla fall när plottningen är färdig.

Några funktioner som kan vara intressanta att prova kommer här:

	x	y
SIN (x)/x	-4pi, 4pi	-0.2, 1
1/x	-10, 10	-5, 5
SIN x	-2pi, 2pi	-1, 1
x^2	-5, 5	-1,25
x^3	-5, 5	-25,25

Det första talet är MIN, det andra MAX.

Ha så trevligt med plottningen! Den som upptäcker några bugar får gärna kontakta mig.

Örjan Gustavsson
Nordlandervägen 12a
777 00 SMEDJEBACKEN

Red anm: Se kommentar i anslutning till artikel om plottning av Micael Dahlquist!

```
SCR #30
  0 ( FLOATING POINT ARITHMETIC ROUTINES 12JUL82 LCT)
  1 : F+ SETFL FADD FAC> ;
  2 : F- SETFL FSUB FAC> ;
  3 : F* SETFL FMUL FAC> ;
  4 : F/ SETFL FDIV FAC> ;
  5 : S->FAC FAC ! 2300 C SYSTEM ;
  6 : FAC->S 1200 C SYSTEM FAC @ ;
  7 : FAC>ARG FAC ARG 8 CMOVE ;
  8 : F->S >FAC FAC->S ;
  9 : S->F S->FAC FAC> ;
 10 -->
 11
 12
 13
 14
 15

SCR #31
  0 ( FLOATING POINT CONVERSION ROUTINES CONTINUED 12JUL82 LCT)
  1 : DOSTR FAC B + C! 14 GPLLNK
  2   FAC B + C@ 8300 + FAC C + C@ DUP PAD C!
  3   PAD 1+ SWAP CMOVE ;
  4
  5 ( NUMBER IN FAC CONVERTED TO BASIC STRING AND PLACED AT PAD)
  6 : STR 0 DOSTR ;
  7
  8 ( NUMBER IN FAC CONVERTED TO FIXED STRING AND PLACED AT PAD)
  9 : STR. FAC D + C! FAC C + C! DOSTR ;
 10
 11 ( STRING AT PAD CONVERTED TO NUMBER IN FAC)
 12 : VAL PAD 1+ 1000 DUP FAC C + ! PAD C@ OVER OVER + 20 SWAP VSBW
 13   VMBW 1000 XMLLNK ;
 14 -->
 15

SCR #32
  0 ( FLOATING POINT - COMPILE NO TO STACK 12JUL82 LCT)
  1 : F$ PAD 1+ SWAP >R R CMOVE R> PAD C! VAL FAC> ;
  2 : (&)  R COUNT DUP 1+ =CELLS R> + >R F$ ;
  3 : &  20 STATE @
  4     IF   COMPILE (&) WORD HERE C@
  5          1+ =CELLS ALLOT
  6     ELSE WORD HERE COUNT F$
  7     ENDIF ; IMMEDIATE
  8
  9 ( FLOATING POINT OUTPUT ROUTINES )
 10 : JST PAD C@ - SPACES PAD COUNT TYPE ;
 11 : F.R >R >FAC STR R> JST ;
 12 : F. 0 F.R ;
 13 : FF.R >R >R >R >FAC R> 0 R> STR. R> JST ;
 14 : FF. 0 FF.R ;
 15 -->

SCR #33
  0 ( FLOATING POINT COMPARE ROUTINES 12JUL82 LCT)
  1 : FCLEAN >R DROP DROP DROP R> ;
  2
  3 : F0< 0< FCLEAN ;
  4
  5 : F0= 0= FCLEAN ;
  6
  7 : FCOM SETFL 0A00 C SYSTEM 837C C@ ;
  8 : F> FCOM 40 AND MINUS 0< ;
  9 : F= FCOM 20 AND MINUS 0< ;
 10 : F< FCOM 60 AND 0= ;
 11 : FLERR 8354 C@ ;
 12 : ?FLERR FLERR A ?ERROR ;
 13 -->
 14
 15

SCR #34
  0 ( FLOATING POINT TRANSCENDENTAL FUNCTIONS 12JUL82 LCT)
  1 0 VARIABLE LNKSAV
  2 : GLNK 83C4 @ LNKSAV ! GPLLNK LNKSAV @ 83C4 ! ;
  3 : INT >FAC 22 GLNK FAC> ;
  4 : ^   SETFL ARG 836E @ 8 VMBW 24 GLNK FAC> 8 836E +! ;
  5 : SQR >FAC 26 GLNK FAC> ;
  6 : EXP >FAC 28 GLNK FAC> ;
  7 : LOG >FAC 2A GLNK FAC> ;
  8 : COS >FAC 2C GLNK FAC> ;
  9 : SIN >FAC 2E GLNK FAC> ;
 10 : TAN >FAC 30 GLNK FAC> ;
 11 : ATN >FAC 32 GLNK FAC> ;
 12 : PI  &  3.141592653590 ;
 13
 14
 15
```

P:TILLCOMP

Av Börje Häll

Denna text är skriven med programmet P:TEXTIN vilket publicerades i PB 84/3 och sedan omformats till COMPANION-filer med programmet P:TILLCOMP. Texten har, efter redigering med COMPANION skrivits ut med detsamma.

Programmet P:TILLCOMP förutsätter att P:TEXTIN INTE har ändrats vad det gäller inputformatet av texten och filhanteringen d.v.s filernas uppläggning. P:TILLCOMP är skriven i Extended BASIC eftersom den modulen krävs för att köra COMPANION.

Det speciella med COMPANION-filer är att första ordet i första record alltid är 0000 eller 0001 hexadecimalt. 0000 om inga tabulatormarkeringar finns i filen, 0001 om fasta tabuleringslägen finns. Program P:TILLCOMP förutsätter att det inte finns några tabbar. Det andra ordet innehåller antal tecken i filen. Mer om detta står under (2.12) NOTE (5) på sida 223 i manualen till COMPANION. Vidare lagras alla blanktecken av COMPANION med ASCII-kod 128 och inte ASCII-kod 32. P:TILLCOMP tar hänsyn till dessa specialiteter, i COMPANION-filerna, vid omformningen av P:TEXTIN-filer till COMPANION-filer.

Om flexskiva har använts vid inmatningen av text med P:TEXTIN så letar P:TILLCOMP efter filer med namnen TEXT1, TEXT2 osv. COMPANION-filernas namn blir TEXTA, TEXTB o.s.v.

Ingen redigering sker vid omvandlingen varför jag föreslår att omformningen sker varefter den omformade filen läses in med COMPANION och skrivits ut, varefter redigering relativt lätt kan ske. Därefter kan texten lagras under samma filnamn eller under ett annat namn.

Red. anmärkning. I förra numret publicerade vi P:TEXTIN, vilket skulle användas till att skriva artiklar för att sedan publiceras i ProgramBiten. Här ser ni nu VÅRT redigeringsprogram. Detta program använder vi i redaktionellt syfte för exempelvis utskrift. Har ni ordbehandlingsprogrammet COMPANION kan ni använda detta program. Skriver du artiklar själv så skriv direkt i COMPANION och skicka oss filerna som de är.

```
100 REM P:TILLCOMP
110 REM
120 REM Extended BASIC
130 REM
140 REM Börje Häll
150 REM Vasavägen 101
160 REM 175 32  JÄRFÄLLA
170 REM
180 REM Programet läser filer skapade med program P:T
    EXTIN och omvandlar
190 REM dessa till Companion-filer
200 REM
210 REM Dimensionera textvariabeln
220 DIM TEXT$(63)
230 REM Bestäm skärmfärg och rensa skärmen
240 CALL SCREEN(8):: CALL CLEAR
250 REM Definiera svenska tecken
260 REM Stora bokstäver
270 CALL CHAR(91,"00280038447C44440028007C4444447C003
    8283844 7C4444")
280 REM Små bokstäver
290 CALL CHAR(123,"0000280038447C44000028007C44447C00
    00382838447C44")
300 REM Initialisera räknaren för filerna
310 INFILE=1 :: UTFILE=65
320 REM Fråga om kassett eller floppy
330 INPUT "Kassett eller Floppy? (K/F) ":M$
340 IF M$<>"K" AND M$<>"k" AND M$<>"F" AND M$<>"f" TH
    EN 330 ELSE PRINT : :
350 IF M$<>"K" AND M$<>"k" THEN 390
360 REM öppna file på kassett
370 OPEN #1:"CS1",SEQUENTIAL,DISPLAY ,INPUT ,FIXED 12
    8 :: GOTO 430
380 REM Kontrollera om file finns
390 GOSUB 840
400 REM öppna file på floppyn
410 OPEN #1:"DSK1.TEXT"&STR$(INFILE),INPUT ,SEQUENTIA
    L,DISPLAY ,FIXED 105
420 REM öppna Companion-file på floppy
430 OPEN #2:"DSK1.TEXT"&CHR$(UTFILE),OUTPUT,INTERNAL,
    VARIABLE 254
440 REM läs in texten
450 FOR I=0 TO 64 :: LINPUT #1:TEXT$(I):: CALL TRIMMA
    (TEXT$(I))
460 REM Om strängen är tom så är texten slut
470 IF TEXT$(I)="" THEN 520
480 REM Räkna antal tecken i texten
490 BYTES=BYTES+LEN(TEXT$(I))
500 NEXT I
510 REM I är 1 mer än antal inlästa strängar
520 I=I-1
530 REM Skriv texten
540 FOR J=0 TO I :: PRINT TEXT$(J): : :: X=1
550 REM Anpassa texten till Companion-format
560 X=POS(TEXT$(J)," ",X):: IF X=0 THEN 590
570 TEXT$(J)=SEG$(TEXT$(J),1,X-1)&CHR$(128)&SEG$(TEXT
    $(J),X+1,255)
580 GOTO 560
590 NEXT J
600 REM Första ordet i första record i Companionfilen
    är 0
610 REM Antal tecken lagras som andra ord i första re
    cord i Companionfilen
620 A=INT(BYTES/256):: B=BYTES-A*256
630 TEXT$(0)=CHR$(0)&CHR$(0)&CHR$(A)&CHR$(B)&TEXT$(0)
640 REM Lagra texten i Companion-format
650 FOR J=0 TO I :: PRINT #2:TEXT$(J):: NEXT J
660 REM öka file-räknarna och stäng filerna
670 INFILE=INFILE+1 :: UTFILE=UTFILEL+1 :: CLOSE #2 :
    : CLOSE #1
680 REM Inläsning från floppy
690 IF M$<>"K" AND M$<>"k" THEN 390
700 REM Inläsning från kassett
710 INPUT "Finns det fler TEXT-filer   på kasetten? (
    J/N) ":S$
720 REM Kontrollera att svaret är rätt
730 IF S$<>"J" AND S$<>"j" AND S$<>"N" AND S$<>"n" TH
    EN 710
740 REM Inga fler filer på kassetten?
750 IF S$="N" OR S$="n" THEN 770 ELSE 370
760 REM Avsluta programmet
770 CALL CLEAR
780 INPUT "Ska andra TEXT-filer om-    vandlas? (J/N)
    ":S$
790 IF S$<>"J" AND S$<>"j" AND S$<>"N" AND S$<>"n" TH
    EN 780
800 IF S$="J" OR S$="j" THEN CALL CLEAR :: GOTO 330
810 CALL CLEAR
820 END
830 REM Subrutin för att leta efter TEXT-file på flop
    py
840 PRINT "LETAR EFTER TEXT-FILE.": :
850 OPEN #3:"DSK1.",INPUT ,RELATIVE,INTERNAL
860 INPUT #3:A$,L,L,L
870 FOR N=1 TO 127 :: INPUT #3:A$,L,L,L
880 REM Finns TEXT-file?
890 IF A$=("TEXT"&STR$(INFILE))THEN 990
900 NEXT N
910 REM Inga fler TEXT-filer
920 CALL CLEAR :: IF OK THEN 950
930 PRINT "Det finns ingen TEXT-file": :"som är inmat
    ad med pro-": :
940 PRINT "gram P:TEXTIN.": : : :: GOTO 970
950 PRINT "Det finns inte fler TEXT-": :"filer som är
     inmatade med": :
960 PRINT "program P:TEXTIN.": : :
970 CLOSE #3 :: GOTO 780
980 REM TEXT-file finns
990 CLOSE #3 :: PRINT A$: : : :: OK=1
1000 RETURN
1010 REM Subprogram för att ta bort föÄ- och efterstäl
     lda blanktecken
1020 SUB TRIMMA(A$)
1030 REM Ta bort förstflllda blanktecken
1040 IF SEG$(A$,1,1)<>" " THEN 1050 ELSE A$=SEG$(A$,2,
     255):: GOTO 1040
1050 L=LEN(A$):: IF L=0 THEN 1090
1060 REM Ta bort efterställda blanktecken
1070 IF SEG$(A$,L-1,1)<>" " THEN 1090 ELSE A$=SEG$(A$,
     1,L-1):: L=L-1
1080 IF L>1 THEN 1070
1090 SUBEND
```

■

BREV

Jakobsberg 1984-03-25

Till Nittinians redaktion

Som nybliven ägare till en TI 99/4A har jag gått med i Programbiten - även retroaktivt för 1983 - och därigenom fått detta års nummer av Nittinian. En trevlig tidning i mitt tycke med en hel del både nyttig och roande information.

I nr 2/1983 förekom bl a en listning för "One Check", ett trevligt solitärspel. Fler läsare har kanske hittat ett par fel i det listade programmet, som gör det ganska lätt att få bort alla pjäser (utom den sista förstås).

Det första felet ligger i programrad 640. Första termen säger Noll=K+2, skall givetvis vara Bokst. O=K+2. Vidare är alla villkoren OR-ade med +, vilket tillåter såväl horisontella och vertikala hopp som en del andra lustigheter. Därigenom är det ingen större konst att rensa bordet från pjäser. Rätta till detta genom att sätta in ett AND-villkor mellan de vertikala och horisontella förflyttningarna på följande sätt:

```
640 IF (( O=K+2)+(O=K-2))*((P=L+2)+(P=L-2)) THEN 660
```

Ytterligare ett fel är att en tom ruta kan överhoppas, men programmet tar ändå bort en (obefintlig)pjäs i rad 700. Det hela kan då sluta med ett negativt antal pjäser på slutet (rad 800). Denna brist kan avhjälpas exvis med följande:

```
635 IF B$((K+O)/2,(L+P)/2)<>"mn" THEN 600
```

samt ändra i rad 610 till THEN 635

Nu blir spelet betydligt svårare, och tyvärr kan man också råka ut för låsningar. Om man i "JUMP FROM"-positionen i rad 560 väljer en pjäs som det inte går att hoppa med, blir man låst i "TO"-positionen i rad 600. Detta kan klaras med följande räddningsfunktion:

```
605 IF B$(K,L)=9 THEN 560
```

En nedtryckning av tangent 9 i "TO"-läge ger då återhopp till "JUMP FROM"-läge, och man kan välja en bättre pjäs.

Så ett par förslag till förbättringar. Man kan med en enkel åtgärd snofsa till grafiken ytterligare. Eftersom varje ruta bildas av fyra tecken i kvadrat, kan man för de fyllda rutorna bilda en figur som liknar en pjäs i ett Damspel. För de undre två tecknen kan man definiera ett nytt par C$(I,J) och utnyttja exvis standardtecknen "kl" för dessa. Mitt förslag är följande:


```
125 C$(I,J)="ee"
300 PRINT C$(I,J)
```

Tillför raderna 375, 425, 475 och 525 med innehållet C$(I,J)="kl"

Här kan man faktiskt reducera inknappningen genom att först definiera pjäsrutorna och sedan tomrutorna:

```
120 B$(I,J)="mn"
125 C$(I,J)="kl"
---
350 FOR I=3 TO 6
360 FOR J=3 TO 6
370 B$(I,J)="ee"
375 C$(I,J)="ee"
380 NEXT J
390 NEXT I
```

Raderna 400-540 behövs då inte.

Och så pjäsfigurerna:

```
715 CALL CHAR(107,"0103070F1F1F3F3F")
716 CALL CHAR(108,"80C0E0F0F8F8FCFC")
720 CALL CHAR(109,"0101070707030101")
730 CALL CHAR(110,"8080E0E0E0C08080")
```

Många irriterar sig säkert på den ständiga omritningen efter varje drag. Inbitna spelare koncentrerar sig djupt på att se flera drag i förväg och tappar lätt tråden vid omritningen. Helst vill man naturligtvis, att bilden skall stå stilla och att endast de flyttade pjäserna skall ändras - och ändras ögonblickligen. Detta kan man direkt åstadkomma i TI Extended Basic med hjälp av DISPLAY- och ACCEPT-funktionerna. Jag medsänder programkassett med en sådan version, där jag dessutom tillagt en del anvisningstext samt ett omstartförfarande.

Samma effekter kan med lite knep och knåp åstadkommas i TI Basic. För den som vill krama ut lite extra ur sin TI Basic kan det vara en roande övning. Pröva t ex Björn Gustavssons förslag till simulering av DISPLAY AT i nr 1/1983 av Nittinian (sid 13). ACCEPT AT kan hjälpligt simuleras med lämpligt utnyttjande av CALL KEY-funktionen. Lycka till!

På andra sidan av den medsända kassetten har jag bifogat programmet "Mönstergrafik" skrivet i TI Extended Basic, som kan användas för att ge uppslag till mattmönster etc. Det är svårt att sluta plocka fram nya färg- och mönsterkombinationer, då man en gång börjat köra det programmet.

Programmet ritar en kvadrat med 17x17 teckenpositioner. Dessa är uppdelade i fyra kvadrater 8x8 plus två korsande symmetriaxlar. Nio tecken ges var sin uppsättning teckenmönster och teckenfärg med hjälp av slumptal. Bakgrundsfärg kan väljas och gäller då för hela "mattan". Beroende på hur de nio tecknen kombineras kan man få mönstertyper med fem olika valbara symmetrier.

Datorn tar tid på sig att räkna fram de olika mönstren. Därför har jag uppe till vänster på skärmen infört en klocka, som visar att programmet arbetar. Vid 9 börjar mönstret ritas.

Jag hoppas dessa kommentarer plus REM-satserna är tillfyllest.

Programmet kan enkelt skrivas om i TI Basic, om DISPLAY-satserna ersätts med PRINT-satser och ACCEPT-satserna med INPUT samt om omstartsfunktionerna slopas och ersätts med BREAK/RUN. För att mönstret skall stå kvar måste man då emellertid avsluta programmet med:

(Programrad) GOTO (samma programrad)

Med bästa hälsningar och på återseende i Nittinian!

Kjell Mellberg

Jakobsberg 1984-10-11

Till redaktören för Programbiten/nittinian.

Under sommarhalvåret har hemdatoraktiviteten legat tämligen i ide för min del. Nu börjar emellertid uppvaknandet igen, och då jag fick tidningsnumret 84-2 i min hand nyligen kom jag att tänka på att jag har en oavklarad dialog med redaktionen (den förra ?).

I slutet av mars skrev jag ett brev till tidningen med lite rättelser och förbättringsförslag till en programlistning i ett tidigare nummer. Samtidigt skickade jag också in en kassett med ett grafikprogram. Jag har emellertid inte sett röken av denna insändare vare sig i tidningen eller brevledes. Inte heller finns grafikprogrammet med i programbanken och inte har jag fått tillbaka kassetten.

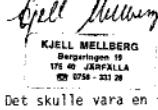
För den händelse brevet skulle ha hamnat på avvägar medsänder jag en ny kopia. Om kassetten går att hitta och grafikprogrammet "Mönstergrafik" accepteras för programbanken skulle jag i utbyte vilja ha:

```
01901001E...WORD PROCESSOR 1
01991001E...ANCESTRAL FILER
11901011S...STAPELDIAGRAM
```

Som jag skrev i förra brevet gick jag med i Programbiten retroaktivt för 1983 för att få detta års tidningsnummer. Jag fick också dessa utom det sista numret som kom ut i efterhand och som jag inte visste om förrän jag hade ett telefonsamtal med Lars Hedlund i våras. Han lovade att stöta på redaktionen för att jag skulle få detta nummer, men ingenting har hänt. Det gäller alltså nittinian 83-4/5 som jag skulle vara tacksam för att få så här i efterhand.

Jag inser att Du har det kämpigt på fritiden med det här och att saker och ting lätt kan falla mellan stolarna, men jag hoppas ändå att vi får bättre kontakt den här gången. Om Du är intresserad av mer TI 99-material så har jag väl en del småsaker som kan vara både till nytta och nöje.

Med bästa hälsningar och hopp om en fortsatt bra tidning!



P.S. Det skulle vara en fördel om programbanken kunde förses med lite mer kommentarer. Programnamnen är f. n. ganska anonyma. D.S.

Undanflykt på Mellbergs brev.

Tyvärr är det så att brev till Programbiten kan komma bort i hanteringen vid skickandet till "handläggare" inom föreningen. Vi har inte fört någon ankomsliggare för inkommande brev och till vem och när det är gjort. Ibland samlar man på sig några brev till styrelsemöten för att dela ut dem med eller utan "föredragning" och ibland har då inte vederbörande mottagare varit närvarande varför breven följer med hem och glöms till nästa möte när de kommer fram - i bästa fall.
Det händer ju även att handläggaren har mycket att göra med fördröjningar till följd.

Beträffande programbanken blev det problem när förre pb-ansvarige fick jobb i annat land. Det har tagit tid för Björn M att gå igenom banken tillsammans med tre eller fyra medhjälpare vilket snart ger resultat.
/Claes S.

FÖRENINGENs FORTH

och

TI FORTH

Denna beskrivning över programspråket
TI FORTH
erhållen från Maurice Swinnen, USA
kan köpas genom:

FÖRENINGEN PROGRAMBITEN, POSTGIRO: 19 83 00 - 6

för MEDLEMMAR	för ICKE MEDLEMMAR
190 SEK	270 SEK

Föreningen har även programmet på diskett, vilken kostar:

för MEDLEMMAR	för ICKE MEDLEMMAR
60 SEK	100 SEK

För beskrivning och diskett tillsammans:

för MEDLEMMAR	för ICKE MEDLEMMAR
250 SEK	370 SEK

Föreningen har även en egen FORTH framtagen av Björn Gustavsson, vilken inklusive kortfattad beskrivning kostar:

för MEDLEMMAR	för ICKE MEDLEMMAR
250 SEK	370 SEK

Vid SAMTIDIG beställning av TI FORTH och Föreningens FORTH

för MEDLEMMAR	för ICKE MEDLEMMAR
400 SEK	600 SEK

Medlem i FÖRENINGEN PROGRAMBITEN blir man enklast genom att betala in 120 SEK på POSTGIRO: 19 83 00 - 6.

Reservation mot prisändringar på grund av materialfördyrningar utanför vår kontroll.

Stockholm 1984-06-20

Lista BASICprogram med BASICprogram

Av Bo Nordlin
Artikeln skriven av Göran Nygren

Detta listprogram använder vi inom redaktionen till att lista ut alla Ti-Basic och Extended-Basic program. Givetvis har envar av oss modifierat och byggt ut det för att tillfredställa de egna behoven (framförallt att förbättra programmet och visa hur smarta vi är) och vår utrustning, speciellt de olika skrivare vi har.

Själv har jag en CANON PW-80 och den har exempelvis ingen pappersfram-matning s.k. "formfeed". Så jag har konstruerat en subrutin vilken matar fram en ny sida. SUB FORMFEED(VXL).

Programmet är alltså utbyggbart. Visa gärna dina lösningar det blir tips till oss. För att kunna lista program måste du först spara det i DIS/VAR 80 format genom att skriva LIST "DSK1.XXXXX". Kör nu programmet och lista ut dina program på en skrivare. Vad du kanske först måste göra är att ändra på filöppningen till skivaren för att det skall passa din skrivare.

Du måste ta reda på om din skrivare kan skriva med 12 TPT (Tecken per tum) eller 10 TPT. Vi skriver alltid med 12 TPT och radlängden är 55 tecken för 12 TPT (11,5 cm). För 10 TPT är radlängden 46 tecken. Detta är viktigt för att listningarna skall få rätt format och radlängd för montering i Programbiten.

Lycka till med dina programlistningar.

Ös på med mer bidrag (börjar vi bli tjatiga ?).

```
 10 REM LISTPROGRAM FÖR PB.
 20 REM
 30 REM AV BO NORDLIN
 40 REM
100 CALL CLEAR :: INPUT "PRINTER
    SPECIFIKATION:":DEVICE$
105 REM HÄR KAN DU SPECIFISERA DIN SKRIVARE
110 OPEN #2:DEVICE$
105 REM KONTROLLKODER FÖR DIN SKRIVARE
120 ! PRINT #2:CHR$(27);"F";

140 RL=44
150 RES1=10
160 DIM SL$(5)
170 SL$(1)="    "
180 SL$(2)="   "
190 SL$(3)="  "
200 SL$(4)=" "
210 SL$(5)=""
220 DISPLAY AT(8,0)ERASE ALL:"NY SIDA? J" :: ACCEPT A
    T(8,10)SIZE(-1)VALIDATE("JNjn")BEEP:A$
230 IF A$="J" OR A$="j" THEN PRINT #2:CHR$(12)
240 DISPLAY AT(8,0)ERASE ALL:"TECKEN/RAD? ";RL+6
250 ACCEPT AT(8,14)VALIDATE(DIGIT)SIZE(-2)BEEP:RL
260 RL=RL-6
270 IF RES1=0 THEN A$="N" ELSE A$="J"
280 DISPLAY AT(9,0):"RESEQUENCE?  ";A$
290 ACCEPT AT(9,14)VALIDATE("JNjn")SIZE(-1)BEEP:A$
300 IF A$="N" OR A$="n" THEN RES1=0 :: GOTO 330
310 DISPLAY AT(10,0):"INTERVALL?  ";RES1
320 ACCEPT AT(10,14)VALIDATE(DIGIT)SIZE(-3)BEEP:RES1
330 DISPLAY AT(11,0):"FILNAMN      DSK2.NAMN" :: ACCE
    PT AT(11,14)SIZE(-15)BEEP:NAMN$
340 OPEN #1:NAMN$
350 LINPUT #1:RAD$
360 IF EOF(1)=1 THEN CLOSE #1 :: GOTO 220
370 LINPUT #1:RAD$
380 IF LEN(RAD$)=80 THEN FLAG=1 ELSE FLAG=0
390 A=POS(RAD$," ",1)
400 SLASK$=SEG$(RAD$,1,A-1)
410 RADNRL=LEN(SLASK$)
420 RADNR=VAL(SLASK$)
430 PRINT #2:"     ";SL$(RADNRL);SLASK$;" ";
440 RAD$=SEG$(RAD$,RADNRL+2,80)
450 RL2=LEN(RAD$):: GOTO 480
460 RL2=LEN(RAD$)
470 PRINT #2:"          ";
480 IF RL>=RL2 THEN 520
490 PRINT #2:SEG$(RAD$,1,RL)
500 RAD$=SEG$(RAD$,RL+1,80)
510 GOTO 460
520 IF FLAG=0 THEN PRINT #2:RAD$ :: GOTO 360
530 IF EOF(1)=1 THEN CLOSE #1 :: GOTO 220
540 LINPUT #1:RAD1$
550 IF LEN(RAD1$)=80 THEN FLAG=1 ELSE FLAG=0
560 A=POS(RAD1$," ",1)
570 IF A<2 OR A>6 THEN 740
580 SLASK$=SEG$(RAD1$,1,A-1)
590 SLASK1$=SLASK$
600 FLG=0
610 FOR F=1 TO A-1
620 A=ASC(SLASK$)
630 IF A<48 OR A>57 THEN FLG=1
640 SLASK$=SEG$(SLASK$,2,9)
650 NEXT F
660 IF FLG=1 THEN 740
670 A=VAL(SLASK1$)
680 IF A>32767 OR A<=RADNR THEN 740
690 IF RES1=0 THEN 710
700 IF A<>RADNR+RES1 THEN 740
710 CALL CLEAR :: PRINT RAD$ :: PRINT :: PRINT RAD1$
720 DISPLAY AT(4,0)BEEP:"NY RAD? " :: ACCEPT AT(4,8)V
    ALIDATE("JNjn")SIZE(1):A$
730 IF A$="J" OR A$="j" THEN PRINT #2:RAD$ :: RAD$=RA
    D1$ :: GOTO 380
740 RAD$=RAD$&RAD1$
750 RL2=LEN(RAD$)
760 GOTO 480
```

■

FÖRENINGENS TILLBEHÖRSFÖRSÄLJNING

Följande finns att köpa för medlemmar genom att motsvarande belopp sätts in på postgiro 19 83 00 - 6.

Användartips med Mini Memory	60:-
FORTH Olika versioner, se annons på annan plats i tidningen (sid 37)	
Nittinian T-tröja	40:-
99'er Magazine nr 12/82 nr 1-5,7-9/83 (per styck)	20:-
Nittinian, årgång 1983	80:-
Astronomical Formulae for Calculators	Slut

Programbiten, årgång 1983	Kalkylatorer	80:-
Programbiten, årgång 1982	-"-	80:-
Programbiten, årgång 1981	-"-	60:-
Programbiten, årgång 1980	-"-	60:-
Programbiten, årgång 1978/79	-"-	60:-
Programbiten, fem årgångar 1978-1983		280:-
Katalog med belgiska och engelska program för räknare TI-57, TI-58, TI-59		20:-
Föreningens programmeringsblanketter (TI-59), olika typer, block om 50 blanketter (se pb 83-1 sidan 30)		11:-
Patenthandlingar TI-59		25:-
40 st tomma magnetkort med plånbok		150:-
Tom magnetkortsplånbok		10:-

PROGRAMBANKEN

```
          91  ** SPEL
02911002E --X Prickskytte med kanon mot en
              kyckling !
02911003E --- Startrek, textadventure
02911006E --- Othello.
02911007E --- Robotjakt.
02911008E --- 15-spel.
02911009E --- Tärningsspel.
02911011E --- Keno
03911013E -+D Starguard
03911014E --- Miner
03911015E --C Yahtzee
03911016E --- Backgammon
03911017E --X 3D Tic-tac-toe
03911018E --X Not one
03911019E --X Datorpoker
03911024E --- Fyra i rad (luffarschack)
04911025E --- Startrek 2
04911026E --X Hjälp kycklingen över vägen
04911027E --- Katapult
01911028H --- Car driver
01911029H --- Animals
06911032E --- Vem skjuter den sista roboten ?
06911034H --X L-game
07911035H --X Wari
07911036H --- Ta dig fram i en osynlig labyrint.
07911037  --X Eliza
07911039H --X Teckenjakt
07911040H --X Black jack
07911041  --X Space battle
07911043  --- Enarmad bandit
08911039E --X Sprite-jakt
08911046E --X Damspel.
08911049E --X Swords & sorcery, textadventure
08911051E --X Ägg-fångst
09911052E --- Deep space
09911054H --- Planetary lander.
09911056H --X Lunar lander.
09911057F --- Towers of Hanoi.
09911058E --X Breakout, bollspel squashtyp.
09911059H --J Ritprogram i olika färger.
09911060H -JX Treasure hunt, labyrintspel.
10911061H --X Kermit
10911062H -JX Find the gun
10911063E --- Matematikspel.
10911065E --- Camel, adventureritt i öknen
10911008E --- Colorfraktions, matematikspel
06912004E --- Slalom
06912005E --- Killer
09912008F --X Asteriod (tal).
09912009  --- Battlestar
11911055S --X Hängning
11917001S -JX Inkräktarna
11917002S -JX Päron
```

BLI MEDLEM NU !

Sätt in 120 skr på PG 430 01 59-3
19 83 00-6

DATAVISION MODUL
Modulen gör att du kan kommunicera med DATAVISION eller liknande baser.
Utskrift för printer finns tillgänglig, sparning av skärmar på kassett eller diskett.
Nödvändig utrustning:
RS232 interface
Televerkets modem eller liknade
Modulen kommer med instruktioner om hur du kopplar upp dig med Televerkets modem.

Pris: 565:-

16 KRAM CMOS MODUL
utan hölje. Modulen kan simulera ett ROM, inladdning sker från T.ex FORTH vilken medför att minimum utrustning är X-basic, MINIMENORY, eller ED/ASM och 32 K minnesutökning. Ram'et är batteribackupat så programmet ligger kvar minst 2 veckor. Denna modul används i princip för att testa ut hur egna program i modulform fungerar innan man bränner prommar vilket blir klart jobbigare.

Pris: 875:-

Säljes

Personal Record Keeping Modul (PRK).

Programming Aids I, kassett (passar till TI-Basic), nyttiga programmeringsrutiner ex. Display At i TI-Basic.

Graphing Packages, kassett (plotting).

Teach Yourself TI-Basic, kassett.

Teach Yourself Extended-Basic, kassett.

Personal Financial Aids.

Editor/Assembler manual.

Plus div. program ex. administrativa program.

Hans Wickström
tel: 018-146417 el. 165446

PROGRAMBANKEN

Här visas en lista på de tillgängliga programmen i PROGRAMBANKEN. Programmen kan köpas! Priset är uppdelat i två delar, dels en startkostnad som täcker kostnaden för porto och mediet, dels en kopieringsavgift för varje program.

KASSETT	Startkostnad	35:-
	Kopieringsavgift	15:-
DISKETT	Startkostnad	55:-
	Kopieringsavgift	10:-

Som förut får Du tre program i utbyte när Du skickar in ett program DU SJÄLV HAR GJORT!
Ange om Ditt program är i TI-basic , X-basic eller i FORTH. Beskriv vilken utrustning som behövs enligt vidstående kodförklaring.

Programkategori:

14 = Ekonomi	78 = Astronomi
20 = Regression, Kurvanpassning	90 = Programhjälpmedel
21 = Statistik, Varians	91 = Spel
29 = Statistik, Sannolikhet	92 = Utbildning
30 = Linjär algebra	96 = Musik
39 = Allmän matematik	97 = Demo
65 = Elektronik	99 = Övrigt

KODFÖRKLARING

De fyra första siffrorna är interna beteckningar.
Femte siffran anger utsprungsland:
1 = England, USA 2 = Holland
3 = Sverige
De tre sista siffrorna är ett löpnummer.
Bokstaven som finns sist anger vilket språk som används i programmet:
E = Engelska F = Franska
H = Holländska S = Svenska
Eventuella bokstäver efter numret talar om vilken utrustning som krävs:
C = CALL FILES (1) måste användas om man har disk.
D = Diskettenhet E = Minnesexpansion
F = FORTH J = Joystick
P = Printer X = Extended Basic
Ett + anger att programmet är delat i flera delar.

```
          39  ** ALLMÄN MATEMATIK
01391001E --- Integral, derivata, andragradsekv.
              reella rötter, cp analys, defini-
              era chars, 3-D plot, fakultet och
              primtalstest
09391003H --- Differential eq.

          78  ** ASTRONOMI
01781001E --- Beräkning av geostationära sate-
              liters positioner.

          14  ** EKONOMI
07141001H --X Annuities

          65  ** ELEKTRONIK
01651001E --- Beräkning av komponentvärden för
              resistiv parallellkoppling, kond-
              densator i serie, resonans för
              spole och kondensator, omvandling
              frekvens - våglängd, ohm's lag,
              uträkning av antennlängd

          30  ** LINJÄR ALGEBRA
06301001E --- Matris invertering/multiplikation

          96  ** MUSIK
08961001E --X Killing me softly
05962001  --C A Strauss walz
05962002  --- Never on a sunday
05962003  --X Berceuse
05962004  -XC Beethoven # 5
05962005  --X Serenade
05962006  +DX Amazing graze
05962007  --- Beethoven opus 27
05962008  +DX Time in a bottle
05962009  +DX Light up my life
05962010  --N Bummble-boogie
06962011  -+D Fiddler on the roof
07962013  --- Looking through you
08962014  --X Chopin op 11 # 3
06972001  --- The pink panther
06972006  --- Let me call you sweetheart
```

```
          90  ** PRAKTISKA PROGRAMHJÄLPMEDEL
01901001E --P Word-processor I
01901002E --- Adressregister
06901003E --X Sortering och utskrift från flera
              disketter av upp till 300 program
06901004E --- Diskkatalog med utskrift
07901005H --- Morse
08901006E -XE Utskrift av titelsida för ex.
              listningar
10901007H --- Lotto
10901008E --- Very large charachters
10901010E --C Ordbehandling för TP-printer.
11901011S --- Stapeldiagram
07902002  --P Banner, stor text utskriven på
              printer
07971003H --- Demonstration av olika tecken-
              storlekar

          20  ** REGRESSION, KURVANPASSNING
01201001E --- Uträkning av en linjes ekvation
              efter ett antal punkter fördelade
              kring en linje
01201002E --- Uträkning av riktningskoeff.

          29  ** STATISTIK, SANNOLIKHET
09291001H --- Statistiska kombinationer etc.

          21  ** STATISTIK , VARIANS
06211001E --C Beräkning av standardavvikelser

          92  ** UTBILDNING
04921001E --- Enkla räkneövn. med de fyra
              räknesätten
04921002E --- Kemifrågor
04921003E --X Relative IQ-test
04921004E --- Algebra
04921005E --- Projectile problems, beräkn.-spel
01921006H --X Arvsanlag
07921007E --- Träning i dåtid (imperfekt)
10921008E --- Räkneträning med bråkdelar

          99  ** ÖVRIGT
01991001E --X Släktforskningsregister
01991002H --- Utskrift av månadskalender
09991003F --- Biorythm.
01992001  --- Månkalender
```