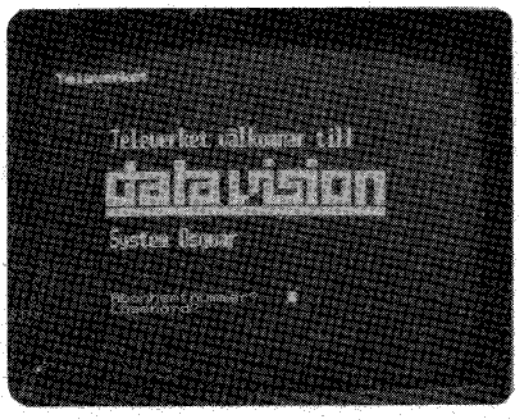
PROGRAM nittinian BITEN

FORTH spalten

REDAKTÖR: Lars-Erik Svahn

FORIT



Innehåll

ISSN 0281-1146

Redaktören

I detta nummer har vi försökt få med program som berör BASIC, FORTH och ASSEMBLER. Vi är medvetna om att det borde vara mer om BASIC. Vi i redaktionen kan inte skriva alla program och artiklar själva utan vi är beroende av att DU sänder in dina program och tips. Även om Du inte tycker att Du klarar av att skriva egna program kan Du komma med förslag om vilka typer av program Du vill läsa om.

För Dig som vill köpa tillbehör själv från USA finns det möjlighet att beställa kataloger. Jag har själv köpt från följande två:

TENEX COMPUTER EXPRESS
P.O. BOX 6578
SOUTH BEND, IN 46660
USA

TEXCOMP
P.O. BOX 33084
GRANADA HILL, CA 91344
USA

Du bör alltid sända med flygpost och skicka med 3 dollar i sedlar som Du kan köpa på banken.
Katalogerna är mycket omfattande och är på 50 respektive 28 sidor. I stort sett finns allt att köpa till mycket låga priser. Vad sägs om USD 4.95 för spel som Munchman eller Parsec.
Editor/Assembler kostar USD 38. Det finns även gott om program från fristående firmor.

Förutom USA tror jag vi bör följa utvecklingen i Västtyskland där det finns ganska många TI-99/4A.
Den bästa tidningen jag sett i Västtyskland är COMPUTER KONTAKT. Prenumeration kostar DM 54 till utlandet. Adressen är:

COMPUTER-KONTAKT
POSTFACH 1550
D-7518 BRETTEN
VÄSTTYSKLAND

Postgiro: Karlsruhe 43423-756

Varje nummer innehåller 10 sidor om TI-99/4A. Normalt finns ett BASIC-program och ett kort Assemblerprogram samt många programmeringstricks och rescensioner av program och hårdvara. Bl a har beskrivits Extended Basic II Plus-modul som innehåller ett stort antal grafikkommandon och PEEK och POKE till VDP förutom normal XB.
Det finns även ett tyskt expansionssystem CPS 99 med Diskdrive DSDD, RAM 32K, RS232/PIO för DM 1698. Fördelen med Västtyskland är att de har 220V/50Hz och PAL till skillnad från USA.

Jan Alexandersson

Lämnad till tryck 1985-10-08

Medlemsträff

Lördag 7 december klockan 10.00

Plats: KASTANJEVÄGEN 2
STAFFANSTORP

(I närheten av Lund)

Upplysningar erhålls av: SVEN LUNDGREN
TFN 046- 25 64 21

Medarrangör: ANDERS PERSSON !

I REDAKTIONEN

Redaktör	Jan Alexandersson
Utmaningsredaktör	Anders Persson
Forth-redaktör	Lars-Erik Swahn
Programförmedlare	Göran Nygren
Allt-i-allo	Claes S
Nya rubriker	Börje Häll

Detta nummer av PB hand-assemblerat av Claes + Jan

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 - Nittinian 1984, 1 - 5 -"- 100 SEK

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

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

För kommersiellt bruk gäller följande:

FÖRENINGENs TILLBEHÖRs-FÖRSÄLJNING

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

För TI-99/4A:

Användartips med Mini Memory			60:-
PB FORTH	Kassett (Föreningens FORTH)		250:-
PB FORTH	Diskett (Föreningens FORTH)		250:-
PB och TI FORTH + TIFManual SAMTIDIG beställning (600 för ICKE medlem)!!			400:-
TI FORTH	Diskett		60:-
TI FORTH	Manual		190:-
PB FORTH Källkod (3 disketter)		(*)	120:-
TI FORTH Källkod (3 disketter)		(**)	120:-
GAME FORTH	Diskett		60:-
FORTH&99	(4 disketter)	(*)	160:-

(*) Säljes endast till den som tidigare köpt PB FORTH
(**) -"- -"- " " " -"- " TI FORTH

MULTIPLAN/TI WRITER	Diskett	60:-
Super Debugger	Diskett	60:-
DATAVISION	Modul	565:-
16 K RAM CMOS	Modul	875:-
Nittinian T-tröja		40:-
99'er Magazine nr 12/82, nr 1-5,7-9/83 (per styck)		20:-

Nittinian, årgång 1983		80:-
Programbiten - Nittinian årgång 1984		100:-
Programbiten, årgång 1983	Kalkylatorer	60:-
Programbiten, årgång 1982	-"-	60:-
Programbiten, årgång 1981	-"-	60:-
Programbiten, årgång 1980	-"-	60:-
Programbiten, årgång 1978/79	-"-	60:-
Programbiten, fem årgångar 1978-1983		200:-
Programbiten, sex årgångar 1978-1984		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) per block		11:-
Patenthandlingar TI-59		25:-
40 st tomma magnetkort med plånbok		150:-
Tom magnetkortsplånbok		10:-

ORDFÖRANDEN har ORDET

Jag försöker följa med vad som händer på olika sätt, bl a genom att läsa tidningar, och då särskilt dem som specifikt tar upp TI 99/4A. Det är uppenbart att under 1984, sedan TI gjorde 99-an föräldralös - man talar i USA om "orphan computers" - har den fungerande marknaden inneburit att fler och bättre program än TI någonsin gjorde har utvecklats och distribuerats. Särskilt de amerikanska tidningarna ger en bild av en alldeles klar tendens. Den innebär

- att spelprogrammens dominans är bruten och att spelmodulerna har svårt att finna köpare. Det är t o m så att de som har stora lager av spelmoduler har slumpat bort dem för några tior och nu har ett problem med hur resten skall skrotas"

- att modulbundna nyttoprogram kommer i förbättrade versioner på skiva. Det gäller TI Writer, Multiplan, Terminal Emulator, Disk Manager m fl. De erbjuds i prislägen kring några hundra kronor.

- att avancerade nyttoprogram utvecklas och finns tillgängliga på skiva. De utvecklas dels professionellt av företag och saluförs till priser på några hundra kronor, dels hobbymässigt av folk som låter dem spridas med mun-till-mun-metoden som s k "free-ware". En del distribuerar sina program som free-ware av ideella skäl, andra därför att det kan vara ganska dyrbart och jobbigt att marknadsföra och distribuera program och ta ett ansvar för dem. Free-ware behöver minsann inte vara sämre att döma av de tester jag sett redovisade" Det handlar om program för databashantering, ordbehandling, kalkylering, datakommunikation, skivhantering (kopiering, katalogisering, felsökning/reparation mm), avlusning av program osv.

- att avancerad hårdvara utvecklas. Det finns Disk Controllers för DS/DD (dubbelsidiga, tätpackade skivor) som kan hantera fyra skivenheter, 128K minnesexpansioner med RAM-disk, CP/M-kort (som är en helt egen dator baserad på Z80-processorn och som förvandlar 99-an till en terminal till Z80-datorn i expansionsboxen"), 80-kolumners-kort, små helt tysta expansionsenheter med minnesexpansion, RS232, Disk Controller mm inbyggt, ritbord med penna, där pennans rörelser ger grafik direkt på skärmen, som sedan kan sparas osv.

Vissa av de program föreningen distribuerar - programbanken, TI Forth 3.0, de nya versionerna av TI Writer och Multiplan mm - är just "free-ware". Vi tar betalt för i stort sett skivan, portot och kopieringskostnaden. Vi försöker skaffa mera sådant. Känner Du till lämpliga och fria program - tala om det för t ex programbankiren eller någon annan i styrelsen. Vårt standardpris är ju nu 60:- kronor per skiva.

De professionella programmen kostar i allmänhet 20-40 USA-dollar. Hårdvaruprylar är dyrare - 100-400 USA-dollar är vanliga priser. Lägg till en nolla för det ungefärliga priset i kronor.

Den här utvecklingen gör 99-an till en nyttodator också. Det understryks av dels en del resonemang på insändarsidorna, där folk redovisar just erfarenheter av nyttoanvändningen - för ordbehandling och kommunikation, t ex - dels att det erbjuds programpaket för det lilla företagets bokföring, fakturering, lagerredovisning mm. Se för övrigt notisen om Paul Bergehamn och hans faktureringsprogram i denna tidning"

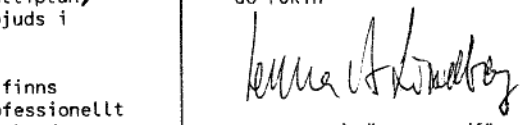
Jag har just avlåtit beställningar till USA på bl a RAM Disk Emulator till mitt 128K-kort, expansionsplatta för modulporten, som tillåter tre moduler på plats samtidigt och Terminal Emulator IV, som tillåter högre kommunikationshastigheter, datalagring i minnesexpansionen mm. Jag återkommer och berättar hur det går. Vid ett tidigare tillfälle lovade jag f ö att berätta om 128K-kortet och dess användning. Sanningen är att jag haft dålig användning av kortet p g a vissa begränsningar i dess DSR och mina krav på att kunna använda det med FORTH eller TI-Writer, vilket tydligen inte var konstruktörens avsikt. Med Disk Emulator - man får en ny DSKx att anropa, som blir blixtrande snabb - bör jag få mera att berätta.

I förra numret nämnde jag att det dykt upp andra 99-föreningar med annonser i Mikrodatorn. Jag har fått kontakt med en, som nog föredrar att vara med oss. Jag fortsätter och återkommer i ärendet.

När Du fått det här numret gissar jag att vi är i början av oktober 1985. Det bör då vara känt i Sverige om det under sommaren i USA anonserats någon storebror till 99-an, som ryktena aviserat. I skrivande stund är jag okunnig om detta.

Däremot vet jag att vi planerar en allmän medlemsträff vid Lucia-tiden i Stockholm. Närmare bestämt ses vi den 14 december, en lördag, på Militärhögskolan som vanligt, med start kl 14. Programmet är inte skisserat när jag skriver detta, men Du kan ju vika tiden t v.

Go FORTH"



Lennart Lindberg, ordförande.

PROGRAMBANKEN

RAPPORT FRÅN PROGRAMBANKIREN.

Äntligen har jag hunnit med att ta ett ordentligt tag i programbanken.
Under sommarens lopp har jag lagt in etthundrasex program, 106 st ". Det har tagit sin tid och varit svårt. Ibland har problemet varit vilken kategori programmet skall läggas under och om programmet överhuvudtaget skall vara med. Svåra val, för jag ville inte ta bort något program. En del av programmen är inte ordentligt testkörda, annat än att de startar och att det går att mata in data. Så det kan finnas logiska fel i programmen som inte syns. Ifall du hittar sådana fel skriv till oss och tala om detta och vi rättar felet eller i värsta fall stryker programmet. Som du ser är det mest spel som kommit in men det har också kommit in en hel del nyttoprogram, de flesta från tidningen. I tidningen har du också riktiga förklaringar till programmen. Tyvärr är det en del program vilka publicerats i tidningen som inte kommit med, anledningen är att jag inte har programmen på ett magnetmedia och då hade jag blivit tvungen att knappa in dessa program. Dessa skall givetvis också in i programbanken men någon måste knappa in dem .
Vill kanske du hjälpa till med att knappa in dem?
Själv studerar jag på heltid och som du nog förstår tar skolan en hel del tid och övrig tid vill jag arbeta lite med egna program samt syssla med min musik. Det vore mycket givande om någon ville knappa in dessa program och samtidigt bli aktiv i föreningen vilket är mycket roligt och givande, det kan jag verkligen lova dig. Jag har mycket roligt med de andra aktiva medlemmarna. De har gett och ger mig massor av tips, program och erfarenheter. Ibland är dessa kunskaper ovärderliga för min förståelse av 99:an.

Observera att programbanken fått två nya kategorier. Registerprogram nr.10 och Ordbehandling nr.12. Vi har börjat få ett antal program under dessa programkategorier. Jag hoppas att du hittar någon eller några godbitar ur programbanken och beställer program från oss, kanske du blir utmanad att skriva egna program.
Programbankiren önskar er en härlig och givande höst och vinter med tillfälle till programmering och lek med 99:an.

Göran Nygren. ■

UTMANINGEN!

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

Skarpögda iakttagare kan konstatera att Utmaningen den här gången inte innehåller något färdigt program alls och därigenom har uppnått ett nytt lågvattenmärke. Innehållet är dock lite mera blandat än förra gången. Alltid något att glädja sig över"

INPUT eller inte INPUT...

I PB 85-2, sidan 26, finns det några funderingar kring varför datorn inte vill lagra text på kassett ordentligt. Delar av texten tycks försvinna.

Som alla nog vet kan man tilldela flera olika variabler ett värde med endast en INPUT sats. De olika värdena skiljs åt med kommatecken. Om man då verkligen vill ha med ett kommatecken i inmatad text måste texten omges av citationstecken.

HEJ,KALLE är alltså inte det samma som "HEJ,KALLE". Det första exemplet består av två ord, HEJ och KALLE, det andra av frasen HEJ,KALLE.

Antag nu att ditt program exekverat en INPUT TEXT$. Du matade in "HEJ,KALLE", med fnuttar (populärt namn på citationstecken, kortare att skriva) före och efter. TEXT$ innehåller alltså HEJ,KALLE.

Om du sedan gör så här:

```
OPEN #1:"CS1", SEQUENTIAL, DISPLAY, OUTPUT, FIXED 128
PRINT #1:TEXT$
CLOSE #1
```

blir resultatet att TEXT$ hamnar på bandet. TEXT$ är ju HEJ,KALLE, vilket INTE är det samma som "HEJ,KALLE". Du har helt enkelt gjort av med fnuttarna"
Vid inläsning från filen hittar datorn HEJ,KALLE, och eftersom kommat säger att den första inmatningen slutar kommer du inte åt KALLE. Inte utan en INPUT TEXT1$,TEXT2$, men så kan man ju inte göra när man inte alltid vet hur många kommatecken det faktiskt finns.

Varför fungerar det med INTERNAL då? Jo, om du använder INTERNAL lagras texten som CHR$(9)&HEJ,KALLE. Koden för det första tecknet talar om hur många tecken som ingår. Datorn håller alltså reda på att hela HEJ,KALLE är en text.

Det går att använda DISPLAY också, och få det rätt. En metod är att använda Extended BASIC. Då finns det en instruktion som heter LINPUT, vilket betyder LINE INPUT. LINPUT tar emot hela raden, oberoende av om det finns kommatecken eller ej. Du behöver alltså inte skriva några fnuttar, varken i filen eller på något annat ställe.

Ett annat sätt, om du inte har X-BASIC, är att tvinga ut fnuttar i filen. Genom att skriva

```
PRINT #1:""""&TEXT$&""""
```

får du fnuttar både före och efter texten. Då fungerar även INPUT.

Experimentera gärna med det här, särskilt om du har möjlighet att använda både INPUT och LINPUT.

I vanlig ordning gäller det goda rådet "Läs bruksanvisningen" även här. På sidan 129 behandlas just PRINT på DISPLAY filer.

Tips: Om du har endast TI-BASIC, använd då filer av INTERNAL typ för sådan information som datorn ska läsa igen. Till exempel text eller siffror på kassett. Använd DISPLAY om du ska läsa resultatet, exempelvis vid utskrift på skrivare.

Om du däremot har Extended BASIC kan du med fördel använda DISPLAY typ för filer som innehåller text. All inläsing från filen ska göras med LINPUT.

FEMBITSPUSSEL

Sven-Arne Wallin i Älmhult har roat sig med det så kallade fembitspusslet eller pentominos. Pusslet består av tolv bitar, vilka utgör de olika former som man kan bilda med fem kvadrater som hänger ihop med sidorna mot varandra. De bildar alltså dessa figurer:

```
 RR  ZZ       L   U U   SS  W    PP   X   I  V   TTT
RR    ZZZ     L   UUU   S   WW   PP  XXX  I  V    T
 R         Y  L       SS    WW  P    X   I  VVV  T
         YYYY LL                        I
                                        I
```

Bokstäverna är så valda att personer med god fantasi ska kunna se att bitarna liknar bokstäverna sedda ur någon lämplig vinkel.

Med de här bitarna kan man lägga ihop rektanglar av olika storlek på en mängd olika sätt. Följande varianter är möjliga:

```
20 x 3 :     2 olika sätt
15 x 4 :   368 olika sätt
12 x 5 :  1010 olika sätt
10 x 6 :  2339 olika sätt
```

Genom att räkna även speglingar av lösningar som unika får man ännu fler varianter, men det har jag inte gjort.

Ett exempel, med de kodbokstäver som jag använt ovan, är det här (20 x 3):

```
UUXPPPLLLLRTTTWWSVVV
UXXXPPLZZRRRTWWYSSSV
UUXIIIIIZZZRTWYYYYSV
```

När du har läst så här långt har du väl redan genomskådat fortsättningen, nämligen hur får man datorn att lösa det här pusslet?

Ja, en sak har jag ju redan visat, nämligen hur resultatet kan presenteras. Kommen så långt kan det ju vara bra att veta"

Hur attackerar man då ett sådant här problem? Jag tar en kort genomgång av ett någorlunda passande sätt. Det finns säkert personer som har andra åsikter om hur det ska gå till, men de åsikterna lovar jag att vederbörligen redovisa när jag får ta del av dem.

För det första får man representera den rektangel som bitarna ska passa inuti med ett numeriskt fält. Lämpligen får siffran noll motsvara en tom liten kvadrat, -1 en kvadrat utanför den använda ytan och använda kvadrater får ett tal som anger vilken bit som finns där. Lämpligen användes ett fält med endast en dimension, då det knappast är realistiskt att försöka generera samtliga lösningar med något annat än ett assemblerprogram. Däremot kan man ju prova algoritmen med exempelvis BASIC.

Bitarna måste också lagras på något sätt. Genom att tilldela alla bitarna en bas-kvadrat nederst till vänster och lagra övriga kvadraters positioner relativt baskvadraten kan man få ett tämligen bra kodningssätt. Nu kan ju bitarna vändas eller orienteras på åtta olika sätt, varför tabellen måste innehålla åtta varianter av varje bit. Index i tabellen blir alltså bitnummer, orientering, kvadrat (det finns ju fyra utöver basrutan) och dimension (X eller Y). Värdet i tabellen ska ange hur långt från basrutan som kvadraten finns.

När det sedan gäller att leta efter lösningar får man börja med att bestämma i vilken ordning bitarna ska provas. Sedan letar man upp en tom ruta i rektangeln och undersöker om biten som står i tur passar där. Om den inte gör det utnyttjar man en ny orientering av biten tills den passar eller tills alla åtta sätten testats. Om det gick att få in biten letar man upp en ny tom ruta och börjar därifrån med nästa bit. Annars får man ta bort den förra biten och se om det går att vända den på något annat sätt.

Grundprincipen är alltså att hålla på så länge det går med en viss uppställning och sedan backa och ta bort tidigare bitar när det inte går längre. Om du tänker försöka kan du säkert ha god nytta av tidningen Byte, november 1979.

UPPSKRÄMT DISKKONTROLLKORT

Det är med disketter ungefär som med ryggsäckar. Det spelar ingen roll hur stora de är, man får ändå inte plats med allt som man vill ha på eller i dem. Jag tycker att det känns trångt, trots att två dubbelsidiga enheter ger fyra gånger så stor kapacitet i boxen som en TI originaldrive.

Nu finns det ju diskkontrollkort som ger möjlighet att använda dubbel densitet, och därigenom 360 kbytes per disk. Efter att ha konstaterat att det blev billigare att dubblera utrymmet på det sättet än genom att dubblera antalet disketter, köpte jag ett sådant kort, tillverkat av Cor Comp. Jag antar att fler personer funderar på det här, och därför skriver jag lite om hur det ser ut och fungerar.

Det man får är ett kretskort i en låda av plåt, en diskett och en bruksanvisning på 90 sidor i ungefär A5 format. Kortet har samma sorts kontakter som originalet, men innan man stoppar in det i boxen kan man utnyttja finessen att steghastigheten för huvudena är ställbar. De flesta moderna drivverk behöver bara 6 ms för att flytta mellan två spår. Genom att berätta detta för kortet sker sökning efter exempelvis filnamn snabbare. Hastigheten kan programmeras individuellt för varje drive.

Efter inkoppling uppför datorn sig lite annorlunda än normalt. Vid uppstart tar nämligen det här kortet över och visar en egen titelbild. Från den kan man starta den Disk Manager som hör till kortet, TI-BASIC eller någon modul som sitter i. Man kan också komma till en bild som ersätter den vanliga med färgbalkarna och därifrån till modullistan.

När man väl har startat BASIC, eller vad man nu vill, fungerar det precis som vanligt. Enda skillnaden är att disketterna rymmer dubbelt så mycket. Jag har provat med BASIC, Extended BASIC, Multiplan, TI-Writer, Forth, Logo, Pascal, Companion och flera andra program utan några problem. Genom en speciell teknik kan Pascal starta om så önskas, trots att diskkontrollkortet tar över kontrollen när systemet startas. Det går även att använda den vanliga Disk Manager 2 modulen, och det är tur det. Det är nämligen INTE möjligt att utnyttja den Manager som medföljer om man inte har minnesexpansionen. Av outgrundlig anledning står det, såvitt jag kan se, inte någonting om detta i bruksanvisningen" Klart underkänt, särskilt med tanke på att datorn hänger sig och informationen på disketten med Managern förstörs om man försöker ladda programmet utan minnesexpansionen"

Jag kan visserligen tänka mig att de flesta som köper det här kortet redan har byggt ut minnet, men det är i alla fall idiotiskt att inte nämna något om detta i bruksanvisningen.

När den väl har startat är Disk Managern dock full med finesser. Vad man vill göra väljer man från en meny. Det första man vill göra är antagligen att tala om för programmet vilka drivenheter man har (antal spår, sidor och packning), standard skrivarnamn, färgen på skärmen och en del annat. Denna information lagras sedan i programmet, så att den automatiskt används även nästa gång Managern laddas.

Utöver detta finns disk-, fil- och testrutiner. Diskrutinerna innefattar katalog, kopiering (backup), namnändring och initialisering av disketter. Programmet genererar hela katalogen på en gång, även om den inte får plats på skärmen. Sedan kan man bläddra sida för sida med piltangenterna. Det går även att få utskrift på skrivare.

Kopiering sker utan den omflyttning av filerna som sker med Disk Manager modulen. Detta för att det ska gå så snabbt som möjligt. Den andra varianten kan dock åstadkommas den också.

Vid initialisering av en diskett kan man även lagra Managern på disketten. Det går alltså bra att ha den på de disketter man vanligen använder, eller på allihop om man så önskar. Kortet klarar fyra olika format, SS/SD, DS/SD, SS/DD och DS/DD, varav de två första är helt kompatibla med de som Texas kort använder.

Filrutinerna är smart gjorda. Genom att integrera alla funktionerna kan man göra upp en lista på alla filer som ska kopieras, flyttas, raderas och ändras namn eller raderingsskydd på. Programmet exekverar sedan hela listan i ett svep, medan man själv gör något annat. Inget ständigt frågande om till exempel den här filen ska kopieras eller ej alltså.

Dessutom finns det en möjlighet att starta (inte skriva) assemblerprogram som normalt kräver Editor/Assemblermodulen. Vissa program fungerar inte som avsett, men många går. De som inte fungerar är i regel de som förutsätter något speciellt som gäller enbart när E/A modulen sitter i. Ett särskilt laddprogram för TI-Forth medföljer, vilket tillåter start från Managern och även med Mini Memory.
Testen finns i olika varianter, både destruktiva och läs-enbart varianten.

Detta var alltså de funktioner som Manangern har. Men det är inte allt. Kortet ger också tillgång till en del roliga subprogram som kan anropas från BASIC eller X-BASIC. Tack vare detta kan man åstadkomma ungefär samma saker från TI-BASIC som med Mini Memory, och ändå ha någon annan modul i datorn. Ett exempel är kombinationen Text to Speech (Terminal Emulator II) och PEEK och POKE. Följande subprogram finns:

MPEEK som läser från CPU RAM.
MPOKE som skriver till CPU RAM.
VPEEK som läser från VDP RAM.
VPOKE som skriver till VDP RAM.
WRTRG som skriver till VDP register.
MOVEM som flyttar hela block inom eller mellan VDP och CPU RAM.
EXEC som kör assemblerprogram på en viss adress, inte med ett visst namn. Motsvarar Exxxx i Easy Bug eller liknande program.
MGR som laddar in Managern utan att återvända till titelbilden.

Dessutom måste man använda instruktionen DELETE "LD-CMDS" för att kunna anropa rutinerna ovan från Extended BASIC.

Anropen sker med CALL MPEEK etc. från BASIC och med CALL LINK ("MPEEK") från X-BASIC. Syntaxen är något underlig när det gäller X-BASIC, men det fungerar.

Intressant är att POKE och PEEK fungerar med både tal och text. Det går alltså bra att ladda en strängvariabel i minnet och även läsa vad som finns direkt som text. I de fall då man vill flytta text till eller från bilden i BASIC kan VPEEK resp. VPOKE automatiskt lägga till eller dra ifrån det offset som behövs.

Olika exempel på hur funktionerna kan användas finns med på den skiva som medföljer. Dessutom är bruksanvisningen (engelska) bra, förutom att det inte står något om minnesexpansionen.

STIM 99-i Halmstad säljer sådana här tingestar i Sverige. Leveranstiden var lång, men priset lägre än ursprungsbudet på grund av sjunkande dollarkurs.

Jaha, ska jag nu säga att du gott kan köpa ett sådant här kort eller inte? Om man använder datorn mycket och hanterar stora datamängder är det naturligtvis ett gott val. Att köpa en massa disketter är ju inte gratis det heller. Dessutom är kortet billigare än en Winchester, än så länge... Men om du inte använder din 99:a till så stora arbete är det nog bättre att köpa ett vanligt kort, kanske ett begagnat av någon som byter upp sig"

FEL, FEL, FEL...

I förra Utmaningen (85-2, sid. 7) hade jag ett program som var avsett som tidsfördriv. Tyvärr fördrev det ännu mer tid än avsett, då det smög sig in några fel i listan. På raden efter labeln INLOP laddas R1 med något som inte ska vara noll. Hexadecimalt ska värdet vara >3020, vilket är ASCII för tecknen noll respektive blank. Citationstecknen fanns inte med på det typhjul som användes... Någon som har hört något sådant förut?

Sista raden blev inte fel. Den valde att försvinna i stället. Raden efter JMP RESTRT ska innehålla labeln LEAVE och inget mer. På förekommen anledning vill jag även påpeka att programmet visst går att köra med Mini Memory och Line-by-line assembler, förutsatt att namnen kortas ned och rättelserna ovan införes.

DISKETTRUTINER

Jag har tidigare efterlyst upplysningar om kan belysa (ha-ha) de subprogram som finns på kontrollkortet till skivenheterna. Tony Rogvall hittade motsvarigheten till CALL FILES i listningen av källkoden till TI-Forth. På skärm 54 ordnas två buffertar för filhantering. Motsvarande går lika bra att göra från assembler. Bra om man skriver assemblerprogram som utnyttjar både disketter och bitmap på samma gång. ■

Rättning av program

av Börje Häll

Programmet P:TEXTIN i PB 84-3 skrevs för att medlemmarna skulle kunna sända in artiklar mm till redaktionen på kassett eller floppy. Programmet P:TILLCOMP skrevs för att redaktionen skulle kunna omvandla en file från P:TEXTIN-format till Companion-format (Companion är ett ordbehandlingsprogram). Allt detta för att underlätta för redaktionen vid redigering av inkommet material. Companion kräver Extended BASIC varför P:TILLCOMP skrevs för den modulen. I Extended BASIC finns LINPUT, som har använts vid inläsning från kassett/floppy. Linput finns inte i BASIC varför Göran Nygren måste använda INPUT när han skrev P:TEXTUT. Det har visat sig att det inte går att få med kommatecken (,) vid utskrift av text, (med P:TEXTUT) som har skrivits in med program P:TXETIN. Det beror på att BASIC uppfattar kommatecken som skiljetecken mellan värden, som ska tilldelas variabler, när värdena läses in från kassett/floppy. Detta är inte specifikt för TI-BASIC utan gäller antagligen för alla BASIC-dialekter. Ett sätt att komma ifrån denna "felaktighet" i BASIC är att inleda och avsluta texten med tre (3) citationstecken """. Om därför programet P:TEXTIN ändras enligt:

```
410 PRINT "sluta med tre(3) citations"
411 PRINT "tecken "&CHR$(34) &CHR$(34)&CHR$(34)&"."
```

så kommer programmet att läsa in texten med kommatecken. ■

Kommandon i Basic och Extended Basic

av Jan Alexandersson

BASIC	EXTENDED BASIC
KOMMANDON (normalt utan radnummer)	
NEW	NEW
NUM	NUM
RES	RES
LIST bryta	LIST bryta
-	stoppa/forsätta
RUN kommando	RUN kommando
-	med radnummer
-	CS1
-	SIZE
BYE	BYE
TRACE	TRACE
UNTRACE	UNTRACE
BREAK	BREAK
UNBREAK	UNBREAK
CON	CON
INSTRUKTIONER (normalt med radnummer)	
REM extra " "	REM utan " "
DEF	DEF
DIM 3 dimensioner	DIM 7 dimensioner
OPTION BASE	OPTION BASE
END	END
RANDOMIZE enkel	RANDOMIZE bättre
IF THEN ELSE	IF THEN ELSE
-	AND
-	OR
-	NOT
-	XOR
relationer	relationer
radnummer	radnummer
-	instruktion
FOR TO STEP	FOR TO STEP
NEXT	NEXT
GOTO	GOTO
ON GOTO	ON GOTO
GOSUB	GOSUB
ON GOSUB	ON GOSUB
RETURN med GOSUB	RETURN med GOSUB
-	med radnummer
-	NEXT
-	CALL
-	SUB
-	SUBEND
-	SUBEXIT
-	ON BREAK
-	STOP
-	NEXT
-	ON WARNING
-	PRINT
-	STOP
-	NEXT
-	ON ERROR
-	STOP
-	radnummer
NUMERISKA FUNKTIONER	
RND 5291877823	RND 8225408469
INT	INT
ABS	ABS
SGN	SGN
SQR	SQR
LOG	LOG
EXP	EXP
SIN	SIN
COS	COS
TAN	TAN
ATN	ATN
-	PI
-	MAX
-	MIN
TAB	TAB
STRÄNGFUNKTIONER	
CHR$	CHR$
ASC	ASC
STR$	STR$
VAL	VAL
SEG$	SEG$
-	RPT$
POS	POS
LEN	LEN

I/O KONSOL

INPUT	INPUT ej (CTRL)128-131
-	LINPUT
-	ACCEPT
-	AT
-	VALIDATE
-	UALPHA
-	DIGIT
-	NUMERIC
-	BEEP
-	ERASE ALL
-	SIZE
PRINT ::	PRINT : :
suddar HCHAR	suddar ej HCHAR
-	PRINT USING
-	DISPLAY
-	AT
-	BEEP
-	ERASE ALL
-	SIZE
-	DISPLAY USING
-	AT
-	BEEP
-	ERASE ALL
-	SIZE
-	IMAGE
READ	READ
DATA	DATA
RESTORE	RESTORE

I/O KASSETT

SAVE	SAVE
CS1	CS1
CS2	CS2
Y	Y
N	N
R	R
C	C
E	E
-	PROTECTED
OLD	OLD
CS1	CS1
R	R
E	E
OPEN #	OPEN #
CS1	CS1
CS2	CS2
INPUT	INPUT
OUTPUT	OUTPUT
INTERNAL	INTERNAL
DISPLAY	DISPLAY
FIXED	FIXED
CLOSE #	CLOSE #
INPUT #	INPUT #
-	LINPUT #
PRINT #	PRINT #
-	PRINT # USING
RESTORE #	RESTORE #

CALL INSTRUKTIONER

CLEAR	CLEAR
SCREEN cyan	SCREEN cyan
ljusgrön	cyan
CHAR	CHAR fyra tecken
	multitilldelning
COLOR set 1-16	COLOR set 0-14
	multitilldelning
HCHAR	HCHAR
VCHAR	VCHAR
KEY delat IF K+1=1	KEY delat IF K=0
JOYST	JOYST
GCHAR	GCHAR
SOUND	SOUND
-	CHARPAT
-	CHARSET
-	VERSION
-	PEEK en adress
-	ERR
-	SPRITE #
-	PATTERN #
-	COLOR #
-	LOCATE #
-	MOTION #
-	POSITION #
-	MAGNIFY
-	DISTANCE #
-	COINC #
-	#
-	ALL
-	DELSPRITE
-	#
-	ALL

TANGENTER MM

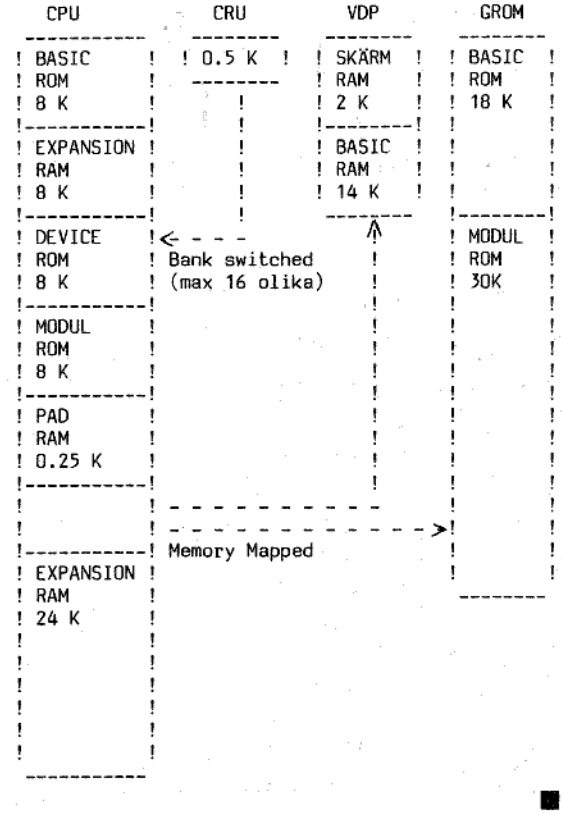
A=2	A,B,C=2
A=B<C	A=B<C
-	(REDO)
(DEL)	(DEL)
(INS)	(INS)
(ERASE)	(ERASE)
(CLEAR)	(CLEAR)
(QUIT)	(QUIT)
(ENTER)	(ENTER)
upp	upp
ned	ned
höger	höger
vänster	vänster
radnummer ned	radnummer ned
-	:: multiinstruktion
-	" tailremark
-	"'p- prescan stop
-	"'p+ prescan start

Tabellen över de olika BASIC-kommandona försöker visa samhörande kommandon. Jag har endast tagit med de kommandon som är helt nödvändiga och som kan användas med enkel kassettbandspelare. Jag har ej tagit med GO, LET, EDIT, DISPLAY(i BASIC), PERMANENT och STOP.

Jag har även försökt få med de viktigaste skilnaderna mellan BASIC och EXTENDED BASIC.

AND, OR, NOT och XOR kan även användas för operationer mellan stora tal på bit-nivå trots att detta ej har beskrivits i tabellen.

TI-99/4A MINNESKARTA I BASIC



Konsten att lära sig FORTH eller mitt första FORTH-program

Bo Carleö

Den traditionella - och något tråkiga - modellen att lära in t ex ett nytt språk är vanligen att följa någon mer eller mindre pedagogiskt upplagd stegvis introduktion.

Lär dig ord för ord och pröva deras funktion i enkla sammanhang.

Metoden fungerar bra så länge man har kvar nyfikenheten och lusten att lära.
Vi har alla kommit i kontakt med det här metodiken i skolan och vet att den är effektiv bara om man kan leva sig in i och förstå det praktiska ändamålet.

Jag föredrar en annan och roligare metod.

Min modell är att, så fort man lärt in språkets mest elementära delar, huvudstupa kasta sig in i ett försök att tackla ett programprojekt.

Programmet ska helst omfatta så många skiftande rutiner man kan komma på från början och ha en grundstomme som tillåter adderande av idéer som kommer under arbetets gång.

Man måste från början ordentligt tänka igenom sin programidé och gärna med papper och penna skissa upp en stomme och hur de olika byggbitarna kan infogas. Man måste ha klart för sig **vad** man vill att programmet skall kunna utföra, även om man inte har den blekaste aning om **hur** det ska gå till.

Sedan börjar man med den del som verkar roligast.

Den stora fördelen med den här modellen är att man hela tiden har en klar motivation att undersöka och lära sig nya och gåtfulla rutiner och foga samman dem till en fungerande enhet.

Ambitionen att få programmet färdigt gör också att man bara inte kan nöja sig med att ge tappt inför de svårigheter man kommer att stöta på.

Som vid all inlärning fordras ett flitigt studerande av manualer och handböcker (Brodie's böcker kan varmt rekommenderas).
Det blir bara så mycket mer meningsfullt att leta efter uppgifter och användbara rutiner när man har behovet klart för sig.

I mitt program ville jag testa 99:ans möjligheter till bit-grafik, som i mina tidigare språk (Basic, X-basic och Logo) varit tämligen svåråtkomliga.

Jag satte alltså glatt igång med ett enkelt CAD-program under arbetsnamnet FOCAD.
Under arbetets gång kom projektet att glida över mot ett ritprogram typ PC-paint eller Macpaint varför det ganska snart döptes om till FORIT.

Grunden till CAD-programmet finns i alla fall och FOCAD kommer kanske till så småningom.

Första steget i arbetet var att definiera en sprite som cursor, styrbar med joystick (eller med pil- tangenterna **E S D X** resp **W R Z C** för diagonaler).

För att kunna använda sprites i bit map mode måste först Sprite Attribute Table flyttas till >3800 (**' 3800 SATR**) och Sprite Descriptor Table flyttas till >3880 (**3800 SSDT** eller **' 3880 SPDTAB**).

För att kunna blinka växlar cursorspriten mellan två olika teckendefinitioner och färger.

Joysticksvaret översätts till en förändring av x- och y-koordinat och cursorn flyttas till en ny xy- position. Varje gång cursorn flyttas lämnar den en punkt (tänd bit i cursorcentrum) efter sig.
En kontrollrutin **CHECK** hindrar att man rör sig utanför ritytan.

Kommen så långt hade jag en fungerande grundversion med vilken jag kunde rita linjer på skärmen.

Något som överraskade mig när jag arbetade med FORTH var enkelheten och snabbheten. Man kan direkt testa olika versioner av rutiner och prova sig fram till den bästa. Hittills hade programmet krävt mindre än två timmars arbete.

FORTH:s exekveringshastighet var imponerande men ställde också till en del bekymmer då det visade sig att det ibland gick **för** fort. Försök själv att trycka på **‹FIRE›** och släppa knappen på 1/10000 sekund!

Jag fick då skriva ordet **WAIT** som fördröjer med 1/10 sekund. **WAIT** består av en dubbel loop varav den inre snurrar 15400 varv/sek. Värdet provade jag mig fram till och det stämmer bra för båda mina 99:or. Jag testade också samma rutin i X-basic och fann att FORTH är ungefär 60 gånger snabbare.

Nästa punkt på önskelistan var ett menysystem.

Hur skriver man text i bit map mode? Enligt TI:s manual är det nämligen inte möjligt.

Det enklaste sättet är naturligtvis med **SPLIT**, men det är ju att undvika problemet eftersom det fält man då skriver på inte ligger i grafikmode.
Dessutom försvinner en stor del av ritytan.
Det finns andra sätt varav rutinen **CLINE** i mitt tycke är elegantast. **CLINE** skriver text i bit map mode med 64 tecken/rad.

Jag offrade rad 24 till meny och började bygga upp ett system bestående av en huvudmeny och 5 sub-menyer som kan nås via huvudmenyn.
När man under ritandet trycker på joystickens **‹FIRE›** eller **‹control Q›** tänds huvudmenyn och cursorn förstoras och hoppar till menyraden.
Därpå flyttar man cursorn till önskat val och trycker **‹FIRE›**.

Menyrutinerna är skrivna så att det är mycket lätt att hänga på olika ritrutiner.

Nu kunde jag börja fylla på grundprogrammet med valmöjligheter.

Konsten att lära sig FORTH eller mitt första FORTH-program

Att kunna välja färg var en självklarhet och då behövdes en palett för att kunna se alla färger.
Den byggs upp av en vertikal rad av 16 sprites i högra kanten av bildytan. Sprites går snabbt att tända och släcka och förstör inte det som är ritat.
När paletten är tänd finns cursorn där och kan då flyttas fram till önskad färg.

Samtidigt som man byter färg ändras menyradens färg till den aktuella kombinationen (förgrund/bakgrund).

Nästa steg var att definiera tre olika penntyper.
FORIT innehåller hittills hel linje, prickad linje och spruta. Dessutom kan man radera samt förflytta pennan utan att rita någonting.

Andra rutiner kom till - och fortsätter att komma - efter önskemål från användarna (9 och 11 år).

Så finns t ex en speglingsrutin, där man kan välja att spegla horisontellt, vertikalt eller diagonalt runt skärmens centrum. Även kombinationer av olika typer av speglingar kan väljas.

En autofunktion låter datorn själv slumpmässigt välja riktning och längd på linjen och alltså rita själv.

En slumprutin på färgmenyn får datorn att själv välja färg med slumpmässiga intervall.

En nodlinjerutin ritar linjer mellan valda punkter. Tryck en gång på <FIRE> för en ny nod. Två tryck utan att flytta cursorn ger retur till menyn.

Alla funktioner (utom nodlinje, rektangel och radier där man bara kan välja färg) kan fritt kombineras med varandra.

Av de olika delfunktionerna var det bara en som krävde någon större tankemöda, nämligen printerdumprutinen **SCREENDUMP**. Uppgiften är tyvärr mer komplicerad än att bara leta rätt på skärmens teckendefinitioner och sända dem till skrivaren.

Hur hittar man för det första rätt i VDP-minnet?

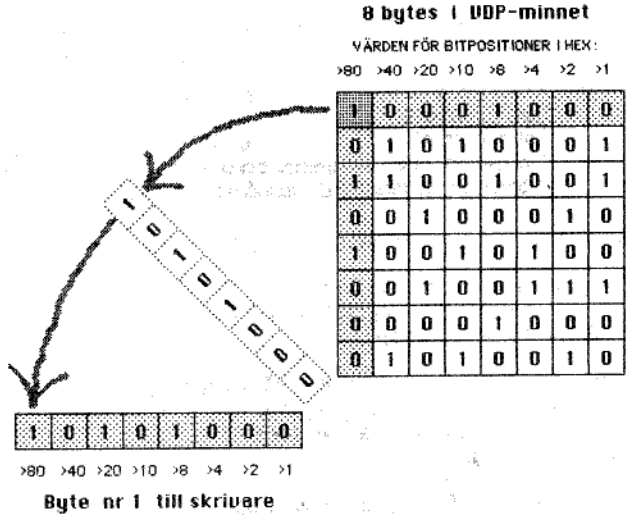
Screen Image Table (**SIT**) som börjar vid >1800 innehåller en byte för var och en av de 768 skärmpositionerna. **SIT** är uppdelad i tre delar på vardera 256 bytes, motsvarande en likadan delning av skärmen. De första 256 bytes står för skärmens översta tredjedel osv.
SIT innehåller inga teckendefinitioner utan endast adressoffsets till den tabell där de egentliga tecknen lagras, nämligen Pattern Descriptor Table (**PDT**).

PDT börjar vid >2000 och innehåller 8 bytes för varje unik skärmposition (totalt 6144 bytes). **PDT** är liksom **SIT** uppdelad i tre delar om vardera 2048 bytes. Det hela fungerar så att **SIT**:s första tredjedel lämnar offsets till **PDT**:s första tredjedel osv. I var tredjedel av **PDT** är varje tecken bara beskrivet en gång. Flera olika **SIT**-bytes kan alltså hänvisa till samma **PDT**-address.

Från VDP-minnet kan man läsa bytes med hjälp av **VSBR** (VDP Single Byte Read).

Nu när man kan hitta rätt tecken uppenbarar sig nästa och något knivigare problem.

Teckendefinitionen för varje skärmposition är alltså uppbyggd av 8 bytes. 99:datorn lagrar dessa horisontellt medan skrivare i allmänhet vill ha sina bytes serverade vertikalt. Undantag finns, eftersom det inte finns någon industristandard för grafisk representation. De vanligaste skrivarna fungerar dock på det här viset, t ex Epson och Gemini.



Man måste alltså göra en konvertering av tecken-matrisen (8*8 bitar).
Min rutin för detta använder en dubbel loop med 8*8 varv samt en mask för uppbyggnad av den byte som skall sändas till skrivaren. Man måste testa var och en av de 64 bitar som matrisen består av. Hur går det till att testa en bit?

Det finns många sätt, men det snabbaste och mest eleganta sättet jag kommit på att undersöka en bit i en byte går till på följande sätt.
Den byte som skall testas upptar de 8 positionerna längst till höger i en 16-bitarscell. (Cell = 2 bytes) Denna cell utsätter jag för funktionen **SRC** (Shift Right Circular).

Vad som då händer är att den bit som befinner sig längst till höger flyttas längst till vänster och att återstående bitar flyttar ett steg åt höger. Biten längst till vänster har en alldeles särskild betydelse för en cell genom att den tolkas som en teckenbit. Jag testar alltså bara om cellen är positiv eller negativ och vet då om biten var 0 eller 1.
(positivt = 0 negativt = 1)

Jag börjar med att testa en bit i byte nr 1 och fortsätter därefter med motsvarande bit i byte nr 2 osv. När jag kommit ned till byte nr 8 gör loopen ett hopp och börjar om från början i ytterligare 7 varv. På detta vis kan jag testa alla 64 bitarna i matrisen.

Resultaten av testerna används för att bygga upp de bytes som skall sändas till skrivaren. För uppbyggnaden används en mask som har värdet >80 i VDP-byte nr 1, >40 i byte nr 2, >20 i byte nr 3 osv. Värdet halveras alltså för varje steg nedåt i matrisen för att till slut bli 1 för byte nr 8. Maskvärdet adderas till skrivarbyten endast om den testade biten är 1.

När de 8 bytes som VDP-matrisen består av konverterats, sänds de till skrivaren som då snällt accepterar dem.

Konsten att lära sig FORTH eller mitt första FORTH-program

SCREENDUMP innehåller ett antal styrkoder för min Gemini-skrivare. Epson använder vad jag vet samma koder. Konsultera din manual och ändra koderna om du har en annan skrivare.

Styrkoder:

Esc "A" 7 (> 1B 41 7)	ändrar radavstånd (inget mellanrum)
Esc "K" 0 1 (> 1B 4B 0 1)	ställer skrivaren i Bit Map Mode. (256 bytes sändes = 1 rad)
Esc "2" (> 1B 32)	återställer radavståndet.

Menyer och kommandon

Cursorn flyttas med joystick eller pil/diagonalknapparna. Kommando: <FIRE> eller <control Q>.

Huvudmeny:

Figur	Flytta till figurmeny
Spegel	Flytta till spegelmeny
Färg	Flytta till färgmeny
Penna	Flytta till pennmeny
Stopp	Flytta till stoppmeny
Retur	Retur till bild

Figurmeny:

Linje	Ritar räta linjer mellan valda punkter (tryck på <FIRE> en gång för ny nod) (2 tryck på <FIRE> = tillbaka till meny)
Rekt	Ritar rektangel med 2 valda diagonal-punkter (<FIRE> lika Linje)
Radier	Nodlinjerutin liknande Linje men med en fast utgångspunkt
Auto	Datorn ritar själv
Sprut	Hela skärmen sprutas med 256 punkter
Retur	Retur till huvudmeny

Spegelmeny:

Hor	Horisontell spegling
Vert	Vertikal spegling
Diag	Diagonal spegling
Alla	Både horisontell och vertikal spegling
Retur	Retur till huvudmeny

Färgmeny:

Skärm	Ändrar skärmfärg (palett)
Bakgrund	Ändrar bakgrundsfärg (palett)
Förgrund	Ändrar förgrundsfärg (palett)
Total	Ändrar allt tidigare ritat till aktuell färg
Slump	Ändrar slumpmässigt färg med slumpmässigt valda intervall
Retur	Retur till huvudmeny

Pennmeny:

Spruta	Sprutar bred och smal stråle runt cursorn
Prick	Ritar prickad linje
Radera	Cursorn raderar allt den rör sig över
Hel	Ritar heldragen linje
Flytta	Flyttar cursorn utan att rita något
Retur	Retur till huvudmeny

Stoppmeny:

Plotta	Dumpar skärmen till skrivare
Sluta	Avslutar programmet
Rensa	Raderar allt tidigare ritat
Retur	Retur till huvudmeny

Konsten att lära sig FORTH eller mitt första FORTH-program

På alla menyer kan man lätt koppla på nya rutiner genom att programmet är modulärt uppbyggt och konstruerat för att kunna fungera som ett basverktyg för försök med egna och andras grafiska rutiner.

Försök själv med egna tilläggsrutiner!
Förslag: cirkel, polygon, kurva genom 3 punkter, parallella linjer, dumpning till disk (för användande i andra grafiska program tex).
Försök även att förändra slumptalsintervallen i **AUTO**, **SPRAY1** och **SPRAY2** och se vad det får för verkan.

Zoomning och förflyttning är också möjliga rutiner men betydligt mer komplicerade genom att den ritade bilden måste hanteras i en separat matris och från denna uppdateras till **SIT** och **PDT**, och har man löst dessa algoritmer har man nog passerat gränsen till FOCAD.

Talområdet för koordinatvärden är för övrigt större än vad FORIT just nu har glädje av. En X- eller Y-koordinat kan nu rymmas i en byte och alltså skulle ett koordinatpar mycket väl kunna packas in i samma variabel (= 1 cell). Att vid behov dela upp cellen i två bytes är en enkel operation.

Med separata koordinatvariabler som nu kan den verkliga bilden teoretiskt vara betydligt större än vad man kan se på skärmen. ($2^{16} * 2^{16}$ pixels stor eller 91178 gånger större än FORIT:s bildyta). I relation till den skala på bilden jag får på min 14" bildskärm ger det en rityta på 69 * 69 meter!
Skall man då kunna hantera dot- och färginformation på samma sätt som i FORIT blir minnesbehovet för stort även för 99:an (~ 1100 Mbyte), men med vektorbaserad grafik och en enda färg bör det gå.

Man bör nog hålla fast vid heltalsaritmetiken. Går man över till flyttal är ritytan och precisionen i praktiken obegränsad men programmet kommer att bli olidligt långsamt. Även en dator som IBM AT med separat flyttalsprocessor blir i dessa sammanhang långsam och PC/XT rent snigelartad.

När jag hade ett fungerande program kunde jag inte avhålla mig från att skriva en demonstrationsrutin.

Man startar programmet genom att skriva FORIT.
Då sprutas först en stjärnhimmel fram som bakgrund till FORIT:s logotype som ritas 2 gånger något förskjutna i 2 blå nyanser.
Därefter går man in i den egentliga demorutinen.
Datorn ritar själv med både horisontell och vertikal spegling, med slumpfärger och med slumpmässigt valda linjetyper.

När man sett sig mätt på 99:ans fantasifulla och ständigt skiftande mönster kan man avbryta genom att trycka på någon tangent. Trycker man sedan på **<P>** får man bilden utskriven på sin skrivare och trycker man på någon annan tangent kommer man in i huvudprogrammet.

Rättning av *FORIT*

```
SKÄRM #131 Rad 2
      Står: ... 0 VARIABLE YCOR
Skall stå : ... 0 VARIABLE YCORS
SCR # 131
2  Ø VARIABLE XCORS    Ø VARIABLE YCORS    \ Sparade k.
```

```
SKÄRM #132 Rad 6
      Står: 10 0 DO DF B * 4 + ....
Skall stå : 10 0 DO DF I B * 4 + ...
SCR # 132
0  10 Ø DO  DF I B * 4 + Ø 18  I D +  SPRITE  LOOP ;
```

Konsten att lära sig FORTH eller mitt första FORTH-program

FORTH

rogram

FORIT är skrivet i fi-FORTH version 4.0 men fungerar lika bra i version 3.0.

Har du inte redan i ditt system orden **UNDER+**, **2DROP**, **2DUP**, **-!**, ***!**, ****, **FH**, **THRU**, så kan du skriva följande definitioner på en egen skärm som du laddar in före FORIT. Se för övrigt Lennart Lindbergs artikel i Programbiten 84/5. Där finns många ord som du kan ha nytta av.

```
: UNDER+      ( x y n--x+n y )
  ROT + SWAP ;

: 2DUP  ( n1 n2--n1 n2 n1 n2 )
  OVER OVER ;

: 2DROP   ( n1 n2-- )
  DROP DROP ;

: *!        ( n address-- )
  DUP @ ROT * SWAP ! ;
: -!        ( n address-- )
  DUP @ ROT - SWAP ! ;

: \         ( -- )  ( skip rest of line )
  IN @ 64 / 1+ 64 * IN ! ; IMMEDIATE

: FH        ( offset--offset block )  ( relative loading )
  BLK @ + ;

: THRU      ( lo_offset hi_offset-- )  ( relative loading )
  1+ SWAP DO I LOAD LOOP ;
```

Innan man laddar in FORIT måste elektivet **-64SUPPORT** och (om du vill kunna skriva ut bilden) **-PRINT** laddas in.

Ett tips:
Elektiven tar ganska lång tid att ladda in eftersom de måste kompileras. Jag brukar arbeta med ett standardurval av elektiv (**-64SUPPORT**, **-PRINT**, **-COPY**, **-DUMP**, **-BSAVE**) som normalt tar ca 2 min att ladda.

Jag har då sparat hela standardurvalet i binär form vilket gör att det hela laddas in på mindre än 6 sek.
Jag har fört in ordet **-BSEL** (**B**inary **SEL**ect) på FORTH:s huvudmeny, vilket gör att jag bara behöver skriva **-BSEL** för att ladda in mina favoritelektiv.

Jag brukar även spara färdiga programversioner binärt. Forit tar på så sätt ca 3 sek att ladda in.
Använd orden **BSAVE** och **BLOAD** och studera manualen.
Koppla dock inte (som det står i manualen) **BSAVE** till ordet **TASK** utan haka på det sista ordet i den föregående ordlistan.
Kolla vilket det är med hjälp av **WORDS** eller **VLIST**.

Jag hoppas att du får lika roligt med FORIT som jag hade under skrivandet. Kommer du på några nya rutiner eller förslag till förbättringar, så sänd gärna in dem till föreningen så att vi alla kan ta del av dem!
Kanske någon kan prestera en assemblerkodad version av **SCREENDUMP**?
Se inte FORIT som ett avslutat och färdigt program utan som ett verktyg att leka vidare med.

Med vänliga hälsningar *Bo Carleö*

```
SCR # 130
0   \ FORIT                 Laddskärm
1
2   1 FH LOAD               \ Variabler och reset-rutiner
3   2 FH LOAD               \ Initiering
4   3 FH LOAD               \ Joystick- och gränskontroll
5   4 FH 5 FH THRU          \ Cursor-position och kommando
6   6 FH LOAD               \ Sprutrutiner
7   7 FH 9 FH THRU          \ Dot- och ritrutiner
8   10 FH 11 FH THRU        \ Screendump
9   12 FH 14 FH THRU        \ Demorutiner
10  15 FH LOAD              \ Texter
11  16 FH 18 FH THRU        \ Menyer
12  19 FH LOAD              \ Huvudrutiner
13
14
15
```

```
SCR # 131
0   BASE->R HEX      \ FORIT  Variabler och resetrutiner
1   Ø VARIABLE XCOR      Ø VARIABLE YCOR      \ Aktuella k.
2   Ø VARIABLE XCORS     Ø VARIABLE YCOR      \ Sparade k.
3   Ø VARIABLE COMMAND   Ø VARIABLE FIRE      \ Kom.flaggor
4   Ø VARIABLE MIRROR    Ø VARIABLE DOTLINE   \ Flaggor
5   Ø VARIABLE SPRAYLINE Ø VARIABLE PENMOVE   \ Flaggor
6   Ø VARIABLE RNDCOL                         \ Flagga
7   Ø VARIABLE S1        Ø VARIABLE S2        \ Teckenvar.
8   : ØFLAGS    ( -- ) \ Nollställning kommandoflaggor
9       Ø COMMAND !   Ø FIRE ! ;
10  : RESET     ( --t ) \ Nollställning variabler, färger,koord
11      1 SCREEN   Ø MIRROR !   Ø DOTLINE !   Ø SPRAYLINE !
12      Ø PENMOVE !   Ø RNDCOL !   DRAW   2Ø DCOLOR !
13      7C XCOR !   5A YCOR !   7C XCORS !   5A YCORS !
14      ØFLAGS   1 ;
15  R->BASE
```

```
SCR # 132
0   BASE->R HEX        \ FORIT  Initiering
1   : INIT      ( -- )   \ Initiering grafikmode, spritedef.
2       GRAPHICS2  3800 ' SATR  3800 SSDT
3       0050 0050 0000 0000  16 SPCHAR
4       0020 5020 0000 0000  17 SPCHAR
5       FFFF FFFF FF00 0000  18 SPCHAR  RESET
6   10 Ø DO  DF B * 4 + Ø 18  I D +  SPRITE  LOOP ;
7   : BLINK  ( t--f / f--t )  \ Cursorblinkning
8       IF  Ø  17 Ø  SPRPAT  D Ø  SPRCOL  ELSE
9           1  16 Ø  SPRPAT  F Ø  SPRCOL  ENDIF ;
10  : TEXTCOL      ( color-- )  \ Färg textrad
11      1700 100  ROT  VFILL ;
12
13
14
15  R->BASE
```

```
SCR # 133
0   BASE->R HEX      \ FORIT  Joystick- och gränskontroll
1   : WAIT          ( n-- )    \ 1 WAIT = 0.1 sek fördröjning
2       Ø DO  6Ø4 Ø DO  LOOP  LOOP ;
3   : ?FIRE          ( n-- )    \ FIRE eller control Q nedtryckt?
4       12 = IF  FIRE @  1+  2 MIN  FIRE !  ENDIF ;
5   : JOY            ( --flag )  \ Kontroll av joystick eller piltang.
6       1 JOYST  -DUP  IF  8Ø -  7C /  YCOR +!  ENDIF
7                -DUP  IF  8Ø -  -7C /  XCOR +!  ENDIF
8       ?FIRE  FIRE @  1 =  IF  3 WAIT  1 JOYST  2DROP
9       ?FIRE  ENDIF  FIRE @ ;
10  : CHECK          ( -- )      \ Gränskontroll
11      XCOR @ DUP  8 < IF  8 XCOR !  -1 S1 *!  DROP ELSE
12                  FØ > IF FØ XCOR !  -1 S1 *!  ENDIF ENDIF
13      YCOR @ DUP  Ø < IF  Ø YCOR !  -1 S2 *!  DROP ELSE
14                  B4 > IF B4 YCOR !  -1 S2 *!  ENDIF ENDIF ;
15  R->BASE
```

Konsten att lära sig FORTH eller mitt första FORTH-program

KONSTEN ATT LÄRA SIG FORTH
eller
MITT FÖRSTA FORTH-PROGRAM

SCR * 134
```
0  BASE->R HEX      \ FORIT     Cursorpos. och kommando
1  : CURSOR         ( -- )      \ Initiering cursor
2     7C 5A F 16 Ø SPRITE ;
3  : >CURSOR        ( -- )      \ Placerar cursor
4     XCOR @ YCOR @ Ø SPRPUT BLINK ;
5  : >MENUCURSOR ( -- )         \ Placerar menycursor
6     XCOR @ B7 Ø SPRPUT BLINK ;
7  : >PALETTECURSOR ( -- ) \ Placerar palettcursor
8     DF YCOR @ Ø SPRPUT BLINK ;
9  : >COMMAND    ( --com ) \ Cursorposition -> Kommando
10    XCOR @ 23 - 24 / 1+ DUP COMMAND ! ;
11 : >COLOR      ( --color ) \ Cursorposition -> Färg
12    YCOR @ 2 - B / ;
13
14
15 R->BASE
```

SCR * 135
```
0  BASE->R HEX      \ FORIT     Cursor-kommando
1  : MENUCOMMAND    ( --com )   \ Menykommando
2     ØFLAGS DD XCOR ! 1 MAGNIFY BEGIN >MENUCURSOR
3     JOY SWAP BLINK SWAP UNTIL ØFLAGS >COMMAND ;
4  : COLORCOMMAND ( -- ) \ Färgkommando
5     COMMAND @ CASE
6  3 OF >COLOR 1Ø * DCOLOR @ 1Ø MOD + DCOLOR ! ENDOF
7  2 OF >COLOR DCOLOR @ 1Ø / 1Ø * SWAP + DCOLOR ! ENDOF
8  1 OF >COLOR SCREEN ENDOF ENDCASE DCOLOR @ TEXTCOL ;
9  : PALETTE        ( -- )      \ Visar palett, palettkommando
10    1Ø Ø DO I ! D + SPRCOL LOOP 6 YCOR !
11    BEGIN >PALETTECURSOR JOY UNTIL COLORCOMMAND
12    1Ø Ø DO Ø ! D + SPRCOL LOOP ØFLAGS ;
13
14
15 R->BASE
```

SCR * 136
```
0  BASE->R HEX      \ FORIT     Sprutrutiner
1  : SPRAY1         ( cor--cor+rnd ) \ Smal spruta
2     2DUP 4 RND 2 - + 4 RND 2 - UNDER+ DOT ;
3  : SPRAY2         ( cor--cor+rnd ) \ Bred spruta
4     C RND 6 - + C RND 6 - UNDER+ DOT ;
5  : ?SPRAY         ( cor-- ) \ Sprut- eller dotrutin
6     SPRAYLINE @ IF SPRAY1 SPRAY2 ELSE DOT ENDIF ;
7  : >RNDDOT        ( -- )      \ Slumpvis punkt på skärmen
8     E9 RND 8 + B5 RND DOT ;
9  : >RNDCOL        ( -- )      \ Slumpvis färg
10    E RND 2+ 1Ø * DUP TEXTCOL DCOLOR ! ;
11 : SCREENSPRAY ( -- )        \ Sprutar över hela skärmen
12    RANDOMIZE 1ØØ Ø DO RNDCOL @
13    IF >RNDCOL ENDIF >RNDDOT LOOP ;
14
15 R->BASE
```

SCR * 137
```
0  BASE->R HEX      \ FORIT     Dotrutiner
1  : !COR           ( -- )      \ Sparar koordinater
2     XCOR @ XCORS ! YCOR @ YCORS ! ;
3  : @COR           ( -- )      \ Hämtar och byter koordinater
4     XCORS @ XCOR ! YCORS @ YCOR ! ;
5  : >DOT           ( -- )      \ Ritar punkt
6     XCOR @ 2+ YCOR @ 2+ ?SPRAY ;
7  : >DOTH          ( -- )      \ Ritar punkt horisontellt speglad
8     F8 XCOR @ - 2+ YCOR @ 2+ ?SPRAY ;
9  : >DOTV          ( -- )      \ Ritar punkt vertikalt speglad
10    XCOR @ 2+ B4 YCOR @ - 2+ ?SPRAY ;
11 : >DOTD          ( -- )      \ Ritar punkt diagonalt speglad
12    F8 XCOR @ - 2+ B4 YCOR @ - 2+ ?SPRAY ;
13 : >DOTA          ( -- )      \ Speglar alla riktningar
14    >DOTH >DOTV >DOTD ;
15 R->BASE
```

SCR * 138
```
0  BASE->R HEX      \ FORIT     Ritrutiner
1  : DOTDRAW        ( -- )      \ Huvudritrutin
2     RNDCOL @ -DUP IF 24 RND 6 + >
3     IF >RNDCOL 1 RNDCOL ! ELSE 1 RNDCOL +!
4     ENDIF ENDIF
5     DOTLINE @ IF DMODE @ IF DRAW ELSE UNDRAW
6     ENDIF ENDIF
7     >DOT MIRROR @ -DUP IF CASE
8     1 OF >DOTH ENDOF 2 OF >DOTV ENDOF
9     3 OF >DOTD ENDOF 4 OF >DOTA ENDOF
10    ENDCASE ENDIF ;
11 : NODE           ( --flag )
12    BEGIN UNDRAW >CURSOR JOY CHECK -DUP UNTIL BEEP ;
13 : NODEDRAW       ( -- )
14    XCORS @ 2+ YCORS @ 2+ XCOR @ 2+ YCOR @ 2+ LINE ;
15 R->BASE
```

SCR * 139
```
0  BASE->R HEX      \ FORIT     Ritrutiner
1  : RECTANGLE      ( -- )      \ Rektangelrutin
2     @COR NODE DRAW 1 = IF ØFLAGS
3     XCORS @ 2+ YCORS @ 2+ OVER YCOR @ 2+ LINE
4     XCORS @ 2+ YCOR @ 2+ XCOR @ 2+ OVER LINE
5     XCOR @ 2+ YCOR @ 2+ OVER YCORS @ 2+ LINE
6     XCOR @ 2+ YCORS @ 2+ XCORS @ 2+ OVER LINE
7     ENDIF ØFLAGS ;
8  : NODELINE       ( -- )      \ Nodlinjerutin
9     @COR NODE DRAW 1 = IF ØFLAGS NODEDRAW
10    !COR MYSELF ENDIF ØFLAGS ;
11 : NODEBEAM       ( -- )      \ Radierutin
12    NODE DRAW 1 = IF ØFLAGS NODEDRAW MYSELF
13    ENDIF ØFLAGS ;
14
15 R->BASE
```

SCR * 140
```
0  BASE->R HEX      \ FORIT     Screendump
1  Ø VARIABLE POINTER       Ø VARIABLE BYTES E ALLOT
2  : BYTE      ( n--address )
3     2 * BYTES + ;
4  : ADDRESS        ( pointer--PDT_address )
5     DUP VSBR 8 * SWAP
6     18ØØ - 1ØØ / 8ØØ * 2ØØØ + + ;
7  : MATRIX         ( -- )      \ Matriskonvertering
8     8 Ø DO Ø 8Ø 8 Ø DO
9     I BYTE @ 1 SRC DUP I BYTE !
10    Ø< IF DUP UNDER+ ENDIF
11    2 / LOOP DROP LOOP
12    8 Ø DO EMIT LOOP ;
13
14
15 R->BASE
```

SCR * 141
```
0  BASE->R HEX      \ FORIT     Screendump
1  : SCREENDUMP ( rows-- ) \ Dumpar antal rader till skrivare
2     SWCH 18ØØ POINTER !
3     1B EMIT 41 EMIT 7 EMIT          \ Esc "A" 7
4     Ø DO
5     1B EMIT 4B EMIT Ø EMIT 1 EMIT   \ Esc "K" 0 1
6     2Ø Ø DO 8 Ø DO
7     POINTER @ ADDRESS ! + VSBR ! BYTE ! LOOP
8     MATRIX 1 POINTER +! LOOP CR LOOP
9     1B EMIT 32 EMIT UNSWCH ;        \ Esc "2"
10
11
12
13
14
15 R->BASE
```

Konsten att lära sig FORTH
eller
mitt första FORTH-program

Konsten att lära sig FORTH
eller
mitt första FORTH-program

SCR # 142

```
0  BASE->R HEX     \ FORIT    Demorutiner
1  : DIRECTION     ( dir-- )   \ Riktning
2       CASE   1 OF   1 S1 !   Ø S2 !   ENDOF
3              2 OF   1 S1 !   1 S2 !   ENDOF \       7
4              3 OF   Ø S1 !   1 S2 !   ENDOF \   6       8
5              4 OF  -1 S1 !   1 S2 !   ENDOF \ 5<    •    >1
6              5 OF  -1 S1 !   Ø S2 !   ENDOF \       4       2
7              6 OF  -1 S1 !  -1 S2 !   ENDOF \         3
8              7 OF   Ø S1 !  -1 S2 !   ENDOF
9              8 OF   1 S1 !  -1 S2 !   ENDOF   ENDCASE ;
10 : +COR          ( -- )       \ Koordinatändring beroende
11      S1 @  XCOR +!   S2 @  YCOR +! ;      \ av riktning
12 : FORWARD       ( dir length-- )   \ Ritar linje
13      DIRECTION   Ø DO   >DOT   +COR   LOOP ;
14
15 R->BASE
```

SCR # 143

```
0  BASE->R HEX     \ FORIT    Demorutiner
1  : MFW           ( n-- )     \ Multipel FORWARD
2       Ø DO   FORWARD   LOOP ;
3  : LOGOCHAR      ( -- )      \ Definition av logotype
4       DRAW  ØA 5   1E 3   14 5   ØA 3   14 1   14 3   2C 5   ØA 3
5       36 1   46 7   ØA   MFW   28 XCOR +!   -5 YCOR +!
6       5 6   19 5   5 4   1E 3   5 2   19 1   5 8   1E 7
7       8 MFW  ØA XCOR +!  -7 YCOR +!  2 6   ØB 5   2 4   1Ø 3
8       2 2   ØB 1   2 8   1Ø 7   8 MFW   24 XCOR +!   ØC YCOR +!
9       ØA 5   18 3   5 4   ØE 5   ØA 3   ØE 1   5 8   5 3   ØA 1
10      28 7   ØA   MFW   27 XCOR +!   ØA 5   28 3   ØA 1   28 7
11      4 MFW   -2E YCOR +!   ØA 5   ØA 3   ØA 1   ØA 7
12      4 MFW   19 XCOR +!   2E YCOR +!   ØA 5   2E 3   ØA 5
13      ØA 3   ØA 1   ØE 3   ØA 1   ØE 7   5 1   ØA 7   5 5   2E 7
14      ØC MFW ;
15 R->BASE
```

SCR # 144

```
0  BASE->R HEX     \ FORIT    Demorutiner mm
1  : LOGOTYPE      ( -- )      \ Ritar logotype
2       Ø DELSPR
3       4Ø DCOLOR !   24 XCOR !   72 YCOR !   LOGOCHAR
4       5Ø DCOLOR !   26 XCOR !   74 YCOR !   LOGOCHAR ;
5  : AUTO          ( -- )      \ Datorgenererad ritning
6       8 RND 1+   DIRECTION   24 RND   4 + Ø   DO   DOTDRAW
7       JOY   DROP   CHECK   +COR   LOOP ;
8  : AUTODRAW      ( -- )
9       @COR  Ø MAGNIFY   BEGIN   AUTO   FIRE @   UNTIL   !COR ;
10 : WRITE      ( address-- )   \ Skriver menyrad
11      2+   3D   17   CLINE   DCOLOR @   TEXTCOL ;
12 : SOLID          ( -- )
13      DRAW   Ø DOTLINE !   Ø SPRAYLINE !   Ø PENMOVE ! ;
14
15 R->BASE
```

SCR # 145

```
0  : TEXTØ          ( -- )                                    ."
1                                                             ";
2  : TEXT1          ( -- )                                    ."
3  MENY   Figur   Spegel   Färg   Penna   Stopp   Retur " ;
4  : TEXT2          ( -- )                                    ."
5  FÄRG   Skärm   Bakgr   Förgr   Total   Slump   Retur " ;
6  : TEXT3          ( -- )                                    ."
7  PENNA   Spruta   Prick   Radera   Hel   Flytta   Retur " ;
8  : TEXT4          ( -- )                                    ."
9  FIGUR   Linje   Rekt   Radier   Auto   Sprut   Retur " ;
10 : TEXT5          ( -- )                                    ."
11 SPEGEL   Hori   Vert   Diag   Alla   Retur " ;
12 : TEXT6          ( -- )                                    ."
13 STOPP   Dumpa   Sluta   Rensa   Retur " ;
14 : TEXT7          ( -- )                                    ."
15                    BO CARLEO   1985                        ";
```

SCR # 146

```
0  BASE->R HEX     \ FORIT    Submenyer
1  : COLORMENU     ( -- )      \ Färgmeny
2       ' TEXT2   WRITE   Ø RNDCOL !   MENUCOMMAND   CASE
3       1 OF   PALETTE   MYSELF   ENDOF
4       2 OF   PALETTE   MYSELF   ENDOF
5       3 OF   PALETTE   MYSELF   ENDOF
6       4 OF   Ø 17ØØ   DCOLOR @   VFILL   ENDOF
7       5 OF   1 RNDCOL !   ENDOF   ENDCASE ;
8  : PENMENU       ( -- )      \ Pennmeny
9       ' TEXT3   WRITE   SOLID   MENUCOMMAND   CASE
10      1 OF   DRAW   1 SPRAYLINE !   Ø PENMOVE !   ENDOF
11      2 OF   DRAW   1 DOTLINE !   Ø PENMOVE !   ENDOF
12      3 OF   UNDRAW   Ø DOTLINE !   Ø PENMOVE !   ENDOF
13      4 OF   SOLID   ENDOF
14      5 OF   1 PENMOVE !   ENDOF   ENDCASE ;
15 R->BASE
```

SCR # 147

```
0  BASE->R HEX     \ FORIT    Submenyer
1  : SHAPEMENU     ( -- )      \ Figurmeny
2       TEXT4   WRITE   MENUCOMMAND   CASE
3  1 OF   ØFLAGS  Ø MAGNIFY   NODELINE   2 WAIT   MYSELF   ENDOF
4  2 OF   ØFLAGS  Ø MAGNIFY   RECTANGLE   2 WAIT   MYSELF   ENDOF
5  3 OF   ØFLAGS  Ø MAGNIFY   @COR   NODEBEAM   2 WAIT   MYSELF
6     ENDOF
7  4 OF   AUTODRAW   ENDOF
8  5 OF   SCREENSPRAY   MYSELF   ENDOF   ENDCASE ;
9  : MIRRORMENU    ( -- )      \ Spegelmeny
10      ' TEXT5   WRITE   Ø MIRROR !   MENUCOMMAND   -DUP   CASE
11      1 OF   MIRROR !   ENDOF      2 OF   MIRROR !   ENDOF
12      3 OF   MIRROR !   ENDOF      4 OF   MIRROR !   ENDOF
13      5 OF   DROP   MYSELF   ENDOF
14      6 OF   DROP   ENDOF   ENDCASE ;
15 R->BASE
```

SCR # 148

```
0  BASE->R HEX     \ FORIT    Sub- och huvudmeny
1  : STOPMENU      ( -- )      \ Stoppmeny
2       ' TEXT6   WRITE   MENUCOMMAND   CASE
3       1 OF   MYSELF   ENDOF      2 OF   MYSELF   ENDOF
4       3 OF   17 SCREENDUMP   ENDOF
5       4 OF   SPLIT2   ABORT   ENDOF
6       5 OF   RESET   INIT   CURSOR   ENDOF   ENDCASE ;
7  : MAINMENU      ( -- )      \ Huvudmeny
8       ' TEXT1   WRITE   MENUCOMMAND   CASE
9       1 OF   SHAPEMENU   MYSELF   ENDOF
10      2 OF   MIRRORMENU   MYSELF   ENDOF
11      3 OF   COLORMENU   MYSELF   ENDOF
12      4 OF   PENMENU   MYSELF   ENDOF
13      5 OF   STOPMENU   MYSELF   ENDOF
14      6 OF   ' TEXTØ   WRITE   ØFLAGS   Ø MAGNIFY
15        ENDOF   ENDCASE                              R->BASE
```

SCR # 149

```
0  BASE->R HEX     \ FORIT    Huvudrutiner
1  : DEMO          ( -- )      \ Demonstrationsrutin
2       RANDOMIZE   7C XCOR !   5A YCOR !
3       4 MIRROR !   BEGIN   2 RND   DOTLINE !
4       >RNDCOL   AUTO   ?KEY   UNTIL   6 WAIT
5       KEY   5Ø =   IF   17 SCREENDUMP   ENDIF ;
6  : TITLE          ( -- )      \ Textrad
7       ' TEXT7   WRITE ;
8  : MAIN           ( -- )      \ Huvudrutin
9       BEGIN   JOY   CHECK   IF   !COR   Ø PENMOVE !
10      MAINMENU   @COR   ENDIF   >CURSOR   PENMOVE @
11      Ø=   IF   DOTDRAW   ENDIF   AGAIN ;
12 : FORIT          ( -- )      \ Startrutin
13      INIT   1 RNDCOL !   SCREENSPRAY   LOGOTYPE
14      FØ DCOLOR !   TITLE   1Ø WAIT   INIT   TITLE   DEMO
15      INIT   CURSOR   MAIN ;                          R->BASE
```

TI 59
Bestämda integraler

Av Lennart Felländer

Programmet beräknar bestämda integraler med hjälp av biblioteksprogram 09 i standardmodulen (MASTER 1), men där svarets noggrannhet, d.v.s. antalet korrekta decimaler bestämmes i förväg. (Max 9 decimaler, dock beroende av antalet siffror som kan visas i displayen). Denna noggrannhetsbestämning har uppnåtts genom att jämföra dessa önskade decimaler vid två successiva beräkningar, den senare med dubbelt så många delintervall som vid den föregående beräkningen. Om någon eller några av decimalerna därvid skiljer sig åt, fördubblar programmet antalet delintervall och ny beräkning göres med förnyad jämförelse med föregående beräkning, till dess att decimalerna överensstämmer.
Då beräkningsresultatet inte ändrar sig vid en fördubbling av delintervallen, kan man förutsätta att svaret är korrekt.

Programmet ordnar även automatiskt så att integralens övre gräns blir större än den undre. Om så ej skulle vara fallet skiftas gränserna och byts tecken på integralens funktion. Detta förutsättes av biblioteksprogram 09.

Programmet knappas in under normaluppdelning, 6 2nd Op 17 enligt listning och kan spelas in på magnetkort i block 1.

Programmet startar med 2, respektive 4 delintervall som successivt fördubblas och ju större noggrannhet som fordras, desto fler blir delintervallen och ju längre tid tar det för maskinen att beräkna integralen.
Några tidsexempel anges i slutet.

När en integral skall beräknas måste först dess funktion programmeras in i huvudprogrammet. Man trycker ned tangenten E varvid maskinen övergår i programmeringsläge enligt ett knep som tidigare publicerats i programbiten. Se programsteg 128-134 i programlistan.
Programmet över integralens funktion avslutas med två högerparenteser samt med INV SBR, LRN. Givetvis kan fler parenteser förekomma om funktionen skulle fordra detta, men alltid med en av högerparenteserna sist före INV SBR.
Likhetstecken får ej användas.

x tages ur RCL 00.
Exemplet nedan är taget ur ML-09 i standarsbibliotekets bruksanvisning:

$$I = \int_0^{\pi/2} \frac{1}{\cos x + 2}\, dx$$

Program: E

Rcl 00	Obs!
2nd cos	Vid trigonometriska integraler
+	måste gränserna vara i radian-
2	form för att biblioteksprogram
)	skall kunna användas.
1/x	
INV SBR	Tryck 2nd RAD före körning.
LRN	

Efter att integralens funktion inprogrammerats i huvudprogrammet, förfares sålunda:

Undre gränsen lägges in i displayen och A tryckes.
Övre gränsen lägges in i displayen och B tryckes.
Antalet önskade decimaler lägges in i displayen och C tryckes, varvid programmet startar.

I exemplet:

2nd Rad	
0	Undre gräns
A	
2nd Pi	
:	
2	
=	Övre gräns
B	
4	Antal decimaler
C	Start

I exemplet ovan med 4 decimaler fordras 16/32 delintervall och 2 minuter 40 sekunder åtgår för svaret:

I = 0.6046

Bestämda integraler.

För att få svaret korrekt med 9 decimaler fordras 128/256 delintervall och 19 minuter 15 sekunder åtgår för svaret:

I = 0.604599788

Detta kan jämföras med det beräknade exemplet i ML-09, där svaret blev:

I = 0.604998903

beräknat med 2 delintervall och endast 3 decimaler riktiga.

Programmet är direktadresserat och använder register 00 - 12, varvid register 01 - 05 är reserverade för biblioteksprogram 09. Om integralens funktion inte ryms inom programsteg 479, kan indelningen ändras ända till 2 2nd op 17, varvid programeringsutrymmet utökas ända till steg 799 och register 13-19 kan användas.

Lennart Felländer

```
000  65  ×
001  43  RCL
002  11   11
003  65  ×
004  01   1
005  00   0
006  95  =
007  59  INT
008  48  EXC
009  12   12
010  75  -
011  43  RCL
012  12   12
013  95  =
014  67  EQ
015  00   00
016  23   23
017  02   2
018  49  PRD
019  10   10
020  61  GTO
021  01   01
022  05   05
023  43  RCL
024  12   12
025  85  +
026  69  OP
027  10   10
028  65  ×
029  05   5
030  95  =
031  55  ÷
032  01   1
033  00   0
034  95  =
035  59  INT
036  55  ÷
037  43  RCL
038  11   11
039  95  =
040  42  STO
041  12   12
042  99  PRT
043  98  ADV
044  91  R/S
045  76  LBL
046  11   A
047  42  STO
048  06   06
049  91  R/S
050  76  LBL
051  12   B
052  42  STO
053  07   07
054  91  R/S
055  76  LBL
056  13   C
057  42  STO
058  08   08
059  66  PAU
060  22  INV
061  28  LOG
062  52  EE
063  22  INV
064  52  EE
065  42  STO
066  11   11
067  02   2
068  42  STO
069  10   10
070  43  RCL
071  07   07
072  75  -
073  43  RCL
074  06   06
075  95  =
076  77  GE
077  00   00
078  92   92
079  01   1
080  94  +/-
081  42  STO
082  09   09
083  43  RCL
084  06   06
085  48  EXC
086  07   07
087  42  STO
088  06   06
089  61  GTO
090  00   00
091  95   95
092  01   1
093  42  STO
094  09   09
095  43  RCL
096  06   06
097  36  PGM
098  09   09
099  11   A
100  43  RCL
101  07   07
102  36  PGM
103  09   09
104  12   B
105  43  RCL
106  10   10
107  36  PGM
108  09   09
109  13   C
110  36  PGM
111  09   09
112  14   D
113  81  RST
114  76  LBL
115  16  A'
116  42  STO
117  00   00
118  53   (
119  43  RCL
120  09   09
121  65  ×
122  53   (
123  61  GTO
124  01   01
125  29   29
126  76  LBL
127  15   E
128  42  STO
129  31   31
130  51  BST
131  51  BST
132  56  DEL
133  41  SST
134  31  LRN
135  00   0
136  00   0
```

List 28

Detta listprogram har använts för TRAIN i Programbiten 85-2. Det har även använts för att lista sig själv. Det kan vara användbart för listning av program för nybörjare eller program som har långa PRINT-satser som sträcker sig över flera skärmrader. Jag har sett avskräckande exempel på listningar som är mycket svårlästa när snygg grafik skall PRINT-as på skärmen i början av ett spelprogram.
I övriga fall kommer vi att använda det PRINT-program som fanns i Programbiten 84-4 vilket ger mer lättläst text för normala program. Det kan dock tänkas att List28 kan användas när vi listar program med korta rader i BASIC. Vi kan då få in fyra kolumner per sida.
/Red.

Handhavande.

Lista först Ditt program från minnet till disk med kommandot:

```
LIST "DSKx.AAAAA"
```

där x är "disk nr" och "AAAAA" är filnamnet Du ger listfilen. LIST28 läser sedan filen och skriver ut den på printer.

```
100 REM Program LIST28 av
      Lennart Lindberg
110 REM Titelsida**********
      120-160
120 CALL CLEAR
130 DISPLAY AT(6,2):"PROGRAM
LISTA 28 TKN BREDD"
140 DISPLAY AT(9,3):"    Len
nart Lindberg"
150 DISPLAY AT(22,8):"EXTEND
ED BASIC"
160 DISPLAY AT(24,4):"valfri
 tangent startar"
170 REM Fortsett***********
      Rensa skermen
      180-200
180 CALL KEY(0,KEY,ST)
190 IF ST=0 THEN 180
200 CALL CLEAR
210 REM TI EXTENDED BASIC***
      ONLY
220 REM Instruktioner*******
      In/ut-definitioner
      Oeppna filer
      230-320
230 PRINT : :"          LIS
T28": :"laeser en fil, lista
d till":"disk, och skriver u
t den paa":"printer med 28 t
kn bredd":
240 PRINT : :"INPUT DSKx.NAM
N      "
250 INPUT "INPUT ":I$
260 IF I$="" THEN STOP
270 OPEN #1:I$,INPUT ! 80-tk
ns infil
280 PRINT "PRINTER : PIO ell
er                   RS232..
.."
290 INPUT "PRINTER : ":O$
300 OPEN #2:O$,OUTPUT,VARIAB
LE 28 ! 28 tkns listfil*****
310 PRINT #2:CHR$(27);"l";CH
R$(5);! 5 tkns venstermargin
al**************************
320 PRINT "LISTAR INFIL ";I$
330 REM Les/skriv-slinga****
      340-410
340 IF EOF(1)THEN 420
350 LINPUT #1:L2$ ! Les t bu
ffert2**********************
360 L1$=L2$ ! Flytta buf2 t
buf1************************
370 IF LEN(L2$)=80 AND EOF(1
)=0 THEN LINPUT #1:L2$ :: L1
$=L1$&L2$ :: GOTO 370
380 REM Rad 370 leser en pos
    t och legger den till in
    nehallet i buf1 om den n
    yss lesta post er exakt
    80 tkn long
390 REM De 80 tkn antages in
    nebera att denna post er
    de 80 foersta tkn av en
    long programrad= >80 tkn
400 PRINT #2:L1$ ! Printa fr
 buf1**********************
410 GOTO 340
420 PRINT "INFIL LISTAD KLAR
T"
430 INPUT "DELETE INFIL (J/N
)? ":D$
440 IF D$="J" THEN CLOSE #1:
DELETE :: GOTO 460
450 CLOSE #1
460 CLOSE #2
470 INPUT "EN LISTA TILL (J/
N)? ":X$
480 IF X$="J" THEN 240
490 END
```

Sortering av strängar

av Börje Häll

I tidningen COMPUTE" september 1984 frågade en läsare varför hans Comodore 64 tog så lång tid på sig vid sortering av strängar. Svaret var att datorn måste göra vad som kallades "garbage collection" och det förorsakades av att strängarna flyttades vid sortering. När nya strängar skapades försvinner inte de gamla utan de finns kvar i minnet och tar upp plats. När minnet är fullt måste datorn avbryta sorteringen och rensa bort alla gamla strängar i minnet.

Tipset i COMPUTE" var att i stället för att sortera strängar så bör man sortera ett index för strängarna. Detta fick mig att undra om samma sak gäller för TI-99. Jag knappade in programmet, för sortering av index, som fanns i tidningen, med de ändringar som måste göras för att programmet ska fungera i 99'an samt ändringar för att fylla upp minnet så mycket som möjligt med strängar, som ska sorteras, och körde det. Sorteringstiden blev 3 min 48 sek. Jag provade även med 1 statement/rad och då blev tiden 4 min 24 sek varför man i X-BASIC tydligen bör ha så många statements/rad som möjligt för att snabba upp körningen.

För att få reda på om "garbage collection" även sker i 99'an ändrade jag även programmet så att det sorterade strängarna i stället för index. Körtiden blev då 37 min 40 sek vilket måste betyda att 99'an uppför sig på samma sätt som Comodore 64 vid sortering av strängar.

Jag ändrade även programmet så att det gick att köra i BASIC. Körtiden för sortering av index blev då 3 min 36 sek (körtiden för att skapa strängarna är ca 7 min). Jag provade aldrig med sortering av strängar i BASIC.

```
100 ! Sortering index
110 ! X-BASIC
120 CALL CLEAR
130 PRINT "Skapar strengar."
140 OPTION BASE 1
150 DIM A$(100),A(100)
160 FOR I=1 TO 100
170 A$(I)=RPT$(CHR$(INT(RND
    *10)+65),115)
180 A(I)=I
190 NEXT I
200 PRINT "Sorterar."
210 CALL SOUND(100,1000,1)
220 FOR J=99 TO 1 STEP -1
    :: FOR K=1 TO J:: X=A(K)
    :: Y=A(K+1) :: IF A$(X)>
    A$(Y)THEN A(K+1)=X :: A(
    K)=Y
230 NEXT K :: NEXT J
240 CALL SOUND (100,1000,1)
250 FOR I=1 TO 100
260 PRINT A$(A(I))
270 NEXT I
280 END
```

```
100 ! Sortering strengar
110 ! X-BASIC
120 CALL CLEAR
130 PRINT "Skapar strengar."
140 OPTION BASE 1
150 DIM A$(100)
160 FOR I=1 TO 100
170 A$(I)=RPT$(CHR$(INT(RND
    *10)+65),120)
180 NEXT I
190 CALL CLEAR
200 PRINT "Sorterar."
210 CALL SOUND(100,1000,1)
220 FOR J=99 TO 1 STEP -1
    :: FOR K=1 TO J :: IF
    A$(K)>A$(K+1) THEN B$=
    A$(K) :: A$(K)=A$(K+1)::
    A$(K+1)=B$
230 NEXT K :: NEXT J
240 CALL SOUND (100,1000,1)
250 FOR I=1 TO 100
260 PRINT A$(A(I))
270 NEXT I
280 END
```

```
100 CALL CLEAR
110 PRINT "Skapar strengar."
120 OPTION BASE 1
130 DIM A$(100),A(100)
140 FOR I=1 TO 100
150 B$=CHR$(INT(RND*10)+65)
160 FOR J=1 TO 115
170 A$(I)=A$(I)&B$
180 NEXT J
190 A(I)=I
200 NEXT I
210 PRINT "Sorterar."
220 CALL SOUND(100,1000,1)
230 FOR J=99 TO 1 STEP -1
240 FOR K=1 TO J
250 X=A(K)
260 Y=A(K+1)
270 IF A$(X)<A$(Y)THEN 300
280 A(K+1)=X
290 A(K)=Y
300 NEXT K
310 NEXT J
320 CALL SOUND(100,1000,1)
330 FOR I=1 TO 100
340 PRINT A$(A(I))
350 NEXT I
360 END
```

■

DATAINMATNING

AV SVEN-ERIK WIND

Jag var på höstens medlemsträffar i Stockholm. De är jättenyttiga för oss, och jag hoppas att de upprepas. Jag jobbar mycket med att genom regelbundna avläsningar kartlägga bl a. elförbrukningen hemma. Det blir en hel del data som skall in till och ut ur 99'an. Eftersom jag bara arbetar med kasettminne, tar det en massa tid. Därför tände jag verkligen på Tony Rogvall's föredrag om effektiv packning av data till ett bandspelarminne. Med hjälp av Tony och en tidskrift som Peter Odelryd är vänlig och lånar mig, har jag fått dubbelt så snabb datalagring mot tidigare. (Gäller förstås båda riktningarna")
Listningarna är utdrag ur ett större program, det förklarar de höga radnumren. Jag undviker helst att ändra antal siffror i radnumreringen om det inte är SUB-program eller SUB-rutiner jag särskilt vill markera. Där har vi 2-intervallet för radnumren.

```
624 REM INMATNING AV UPPGIFTER TILL BANDMINNE
626 REM
628 IF (N/4)=INT(N/4) THEN M=N-3 :: GOTO 634
630 M=N
632 REM
634 OPEN #2:'CS2',INTERNAL,OUTPUT,FIXED 192
636 PRINT #2:Q,M,N,DAT(Q),A(Q),B(Q),C(Q),D(Q)
638 FOR I=1 TO M
640 FOR J=1 TO 3
642 PRINT #2:DAT(I),A(I),B(I),C(I),D(I),
644 I=I+1
646 NEXT J
648 PRINT #2:DAT(I),A(I),B(I),C(I),D(I)
650 NEXT I
652 CLOSE #2
654 REM
670 REM FRAMTAGNING AV LAGRADE UPPGIFTER TILL
CENTRALENHETEN
672 REM
674 OPEN #3:'CS1',INTERNAL,INPUT,FIXED 192
676 INPUT #3:Q0,M,N,DAT(Q),A(Q),B(Q),C(Q),D(Q)
678 REM
680 IF Q0<>0 THEN 698
682 FOR I=1 TO M
684 FOR J=1 TO 3
686 INPUT #3:DAT(I),A(I),B(I),C(I),D(I),
688 I=I+1
690 NEXT J
692 INPUT #3:DAT(I),A(I),B(I),C(I),D(I)
694 NEXT I
696 REM
698 CLOSE #3
```

Beskrivning av programmet.

Först av allt måste jag veta hur många data som gick att lagra på varje segment. Ett segment rymmer max 192 bytes. I mitt fall behövde jag lagra 5 siffergrupper för varje avläsningsperiod.
Varje siffergrupp tar upp 9 bytes. Detta ger 5*9=45 ; 192/45>4 . Alltså kan jag lagra fyra perioder på varje segment.
Hur kan jag lära datorn att alltid lagra data på detta sättet? - Naturligtvis" Ett kommatecken efter sista datum i perioden.
Nu gällde det att finna algoritmen, för att 99'an skulle mata ut rätt antal segment till bandspelaren, oberoende om jag läst in 2,5,6,7 eller 8 perioder. Rad 628 och 630 ger svaret. Rad 636 skriver in Q(kan användas som kontrollvariabel), M(som är styrvariabeln för hur många segment som skall skrivas), N(antalet avläsningsperioder), DAT(Q)datum, A(Q), B(Q), C(Q), D(Q) är avlästa siffergrupper. Rad 638-650 är loopen för lagringen av hela datamängden.
Rad 640-646 loop som styr antal inläsningar på varje segment. Observera att variabeln 'I' räknas upp med ett för varje gång loopen genomlöps. Rad 642 kommatecknet efter sista datavariabeln gör att inläsningen

→→→

Textmode med Mini Memory

av Jan Alexandersson

I PROGRAMBITEN 84-04 (sid 25 och 31) fanns listade några program jag gjort. De handlade om TEXTMODE med MINI-MEMORY. Beskrivning till programmen saknades, men kommer här istället.

TEXTMODE är en egenskap hos den sk videoprocessorn i 99-an och gör att man kan skriva 40 tecken per rad på skärmen mot 32 i GRAFIKMODE, som är standardläget i vanlig BASIC. I GRAFIKMODE är ett tecken 8 pixels brett på skärmen medan det i TEXTMODE reduceras till 6 pixels.

Mina program bygger på en artikel i PROGRAMBITEN 84-01 (sid 9) av Tony Rogvall: "VDP REGISTER FRÅN BASIC". Han visar där att VDP-register kan nås med CALL POKEV. Tonys program innehåller dock några skrivfel och kan förbättras på några punkter.
Skrivfelen finns på raderna 200 och 350, som bör skrivas som följer:

```
200 CALL LOAD(-31788,240)
350 CALL LOAD(-31788,224)
```

Så till förbättringarna.

Som **Tonys program** nu är skrivet får man ett antal konstiga tecken på skärmen under startskedet och ett vitt streck kvar även senare. Detta kan ändras genom att rad 160 flyttas och blir rad 90.

Det är mycket stor risk för dubbeltryckning av bokstäver under körningen. Risken elimineras genom en ändring i rad 240, som bör skrivas som följer:

```
240 IF S<1 THEN 230
```

De normala färgerna vid TEXTMODE är vit text på blå bakgrund. Om man vill ha detta skall rad 180 ändras:

```
180 CALL POKEV(-30731,0)
```

Uppläggningen av **mina program** följer nedan.

Om man vill använda TEXTMODE i BASIC mer allmänt måste man skapa ett antal SUB-rutiner. Följande tre rutiner behövs:

- Initiering av TEXTMODE
- Avslutning av TEXTMODE(initiering av GRAFIKMODE)
- Skriva till TEXTMODE

Det är lämpligt att i rutinerna välja radnummer som är mycket större än de som vanligen används i normala program.

Subrutinerna anropas från BASIC med "GOSUB radnummer".

Programbeskrivning:

```
9010      Initiera TEXTMODE: F0 till VDP-reg 1
9020      Kopia till VDP-reg 1
9040-9060 Rensar nedre delen av skärmen(fylls med CH32)
          För att snabba upp programmet tas 10 värden
          åt gången (32 + offset 96 = 128 )
9070      Vit text på blå bakgrund: F4 till VDP-reg 7
9120      Initiera GRAFIKMODE: E0 till VDP-reg 1
9130      Kopia av VDP-reg 1
9150-9180 Återställer färgtabell till svart på
          transparent
9220      24 rader med 40 kolumner
```

Programmet med SUB-rutinerna finns listat i PROGRAMBITEN 84-04 sid 25 längst ned till höger.

SUB-programmet för PRINT anropas från huvudprogrammet med:

```
RAD=...
KOL=...
TEXT$="..."
GOSUB 9200
```

Ett enkelt demonstationsprogram fanns listat i PROGRAMBITEN 84-04 sid 31. Detta kan köras med MINI-MEMORY eller EDITOR/ASSEMBLER.

Ett mer praktiskt och användbart program kan fås genom att skriva om DISK CATALOG-programmet på sid 41 i manualen för DISK MEMORY SYSTEM. För att snabba upp programmet skapas först en hel rad som en TEXT$ i minnet innan den skrivs. Det krävs då att allt görs om till textsträng och läggs samman (konkateneras) med &. Numeriska variabler görs till strängar med STR$. För att simulera TAB används SEG$(SPACE$,1,antal) för att fixa lämpligt antal mellanslag så att efterföljande text alltid kommer lika oberoende av textens längd.

I stället för att markera skrivskyddat program med Y används * som placeras mellan sektorantal och programtyp för att spara kolumner: CHR$(32-10*(A<0))
Om A>=0 fås (A<0)=0 och hela uttrycket ger CHR$(32).
Om A<0 dvs programmet är skrivskyddat fås (A<0)=-1 och hela uttrycket ger CHR$(42) dvs *.
På motsvarande sätt skrivs filernas längd K ut om ABS(A)<>5 men ej vid PROGRAM-fil. För att inte strängen som skall sågas av skall vara kortare än tre tecken läggs SPACE$ till för säkerhets skull.
Obs att rad 360 behöver editeras flera gånger så att radlängden 112 kan överskridas. Skriv in en korrekt rad tills editorn säger stopp. Tryck sedan på (ENTER) och hämta därefter tillbaka raden för editering.

Programmet avslutas med (ENTER). Använd aldrig (CLEAR) eftersom då övergången till GRAFIKMODE blir felaktig. Programmet visar upp till 42 program samtidigt på skärmen vilket bör räcka för alla normala fall.

Det här katalogprogrammet, som fanns listat i PROGRAMBITEN 84-04 sid 25, kan köras med MINI-MEMORY eller EDITOR/ASSEMBLER i modulporten. ■

➔➔➔

fortsätter på samma segment. Efter inläsning av tre perioder går programmet vidare till rad 648 som ger inläsning av fjärde perioden. Segmentet är nu fullt och nästa inläsning sker på ett nytt segment.
Det fungerar på liknande sätt vid inläsning från bandspelare till dator. Variabeln 'N' anger hur många avläsningar som har gjorts. Jag har här endast använt numeriska variabler för datalagringen. Skall man ha strängvariabler måste kanske hänsyn tas till dimensioneringen av strängarna, men det finns beskrivet i BEGINNERS GUIDE.

Med vänlig datorhälsning ■

Sven-Erik Wind tel 026-191733
Takaregatan 21
80238 Gävle

■

Följande lilla program i EXTENDED BASIC suddar ut hela programmet i VDP-RAM. D et är alltså möjligt att använda något som liknar NEW i kommando-mode. Det kan INTE användas tillsammans med expansionsminne 32k. Programmet kommer från den tyska tidningen COMPUTER KONTAKT.

```
32000 SUB NEW
32001 ON ERROR 32003
32002 RUN ""
32003 END
32004 SUBEND
```

■

FORTH spalten

REDAKTÖR: Lars-Erik Svahn
Mellanbergsv. 23
13545 TYRESÖ
tel: 08-742 61 07

ORDLISTAN

När ett FORTH-program kompileras, skapas först ett utrymme i minnet för rutinens mm, ett programhuvud. Ett program huvud är uppdelat i 3 delar: länkfältet, namnfältet och kodfältet. Direkt efter programhuvudet börjar parameterfältet, där själva programmet skrivs. Adresserna till dessa fält brukar förkortas lfa, nfa,cfa och pfa. I Texas Instruments fig-FORTH (dvs kärnan i TI-FORTH, PB-FORTH och Forth&99) inleds programhuvudet med länkfältet (2 byte), följt av namnfältet (max 32 byte) och kodfältet (2 byte). På nästa adress (pfa=cfa+2) börjar koden för det kompilerade programmet.
I FORTH översätts inte programmen till maskinkod (inte normalt, men det finns sk makro-FORTH, där så sker), utan det som skrivs i pfa är: kodfältsadresser för de ord som ingår i definitionen samt div data. De kodfältsadresser som skrivs vid kompileringen, hänvisar till andra ords programhuvuden och parameterfält, där hela historien upprepas på nytt" Ett FORTH-ord (dvs en kolondefinition) bildar på detta sätt ett träd av referenser, med de primitiva FORTH-orden som löv. Primitiverna är (naturligtvis) skrivna i maskinkod.
En av dessa primitiver är den inre interpretorn, som vanligen brukar ha namnet NEXT. I TI:s fig-FORTH är denna rutin namnlös (utan programhuvud) och börjar på adress hex 8334, en adress i 99:ans snabbaste minne. Det är NEXT som tillsammans med returstacken (för returadresserna) och parameterstacken (för parametrarna), nystar upp trädet av referenser och ser till att de rätta primitiverna exekveras, i rätt ordning.
När ett nytt ord kompileras, börjar alltid kompileringen vid den adress som finns i variabeln DP (dictionary pointer). Denna adress läser man lättast med FORTH-ordet HERE (-- a).
När ett ord ska tolkas, från källkod (ascii kod), söker den yttre interpretorn (med namnet INTERPRET) i ordlistan, dvs listan av kompilerade ord (programhuvuden + parameterfält). INTERRPRET jämför då med namnen i namnfälten, och exekverar rätt ord, om det finns med i ordlistan. INTERRPRET börjar att söka i slutet av ordlistan, på den namnfältsadress som ges av LATEST (-- a).
Om namnet på denna adress inte överensstämmer med det sökta, så läses innehållet i länkfältet (adressen ovanför), ty där finns adressen till näst sista ordets namnfält etc.
I kodfältet finns:
för kolondefinitioner, adressen till den inre interpretorn (hex 8334);
för assemblerdefinitioner, adressen där maskinkoden börjar;
för ord definierade med hjälp av DOES> (tex variabler och konstanter), adressen till ett speciellt kodfält inne i parameterfältet för det definierande ordet (tex i den kompilerade koden för ordet VARIABLE).

Några viktiga definitioner:

DP (-- adr)
: variabel innehållande adressen till första fria adressen i ordlistan.

HERE (-- adr)
: lägger innehållet i variabeln DP på stacken.

LATEST (-- adr)
: ger namnfältsadressen till senast kompilerade ordet (i rådande vokabulär).

CURRENT (-- adr)
: USER-variabeln som pekar på en pekare till LATEST (beror av vald vokabulär, t ex FORTH, EDITOR1 etc).

CONTEXT (-- adr)
: USER-variabel som pekar på en pekare som pekar på den nfa, vilken nya definitioner ska länkas till (via lfa. vokabulär beroende, ändras genom att t ex skriva FORTH DEFINITIONS etc).

LFA (-- adr)
: beräknar ett ords länkfätsadress från dess parameterfältsadress.

NFA (-- adr)
: beräknar namnfältsadressen från parameterfältsadressen.

CFA (-- adr)
: beräknar kodfältsadressen från parameterfältsadressen.

PFA (-- adr)
: beräknar parameterfältsadressen från namnfältsadressen.

ID. (-- adr)
: skriver namnet på rutinen med namnfältsadress nfa. Ex: ' FORGET NFA ID. skriver just "FORGET".

DET VIRTUELLA MINNET

Med ordet BLOCK (n -- adr) kan man läsa skärm nr "n" från massminnet till adress "adr" i cpu-minnet (observera att det är BLOCK som bestämmer VAR data hamnar). Varje skärm innehåller 1024 byte, och det finns 5 olika cpu-buffertar att läsa till. Den totala buffertarean är på 5*(1024+2) byte. Adressen innan den som bes av block (själva buffertadressen) finns nämligen 2 byte med information om skärmnummer (15 bitar) och uppdatering (den mest signifikanta biten). Om en buffert är markerad för uppdatering (görs med ordet UPDATE), kommer den att skrivas till massminnet, tex när Du beordrar FLUSH (annars raderas den).
De skärmar som finns i buffertminnet är ordnade i en kö, så att om en 6:e skärm läses från massminnet, så övertar den bufferten från den minst aktuella skärmen (denna skärm raderas eller skrivs till massminnet, beroende på uppdateringsbiten). I varje FORTH-editor (värd namnet) sker denna uppdatering automatiskt, när man editerar på skärmen.

HUVUDLÖSA FORTH-PROGRAM

När Du programmerar i FORTH, ska Du sträva efter att göra korta rutiner. Idealet är korta rutiner som anropar korta rutiner etc. Då blir programmen lättare att läsa och avlusa. En FORTH-program består därför av relativt många definitioner.
Utifrån, behöver Du kanske bara kunna nå ett fåtal av dessa ord (eller kanske bara det sista, huvudrutinen).

Några viktiga definitioner:

BLOCK (n -- adr)
: överför skärm "n" från massminne till buffertarean (på "adr").

UPPDATE (n --)
: raderar skärm nr "n" (en raderad buffert med det numret görs iordning och markeras för uppdatering).

FLUSH
: tömmer buffertminnet. Uppdaterar eller raderar.

EMPTY-BUFFERS
: tömmer buffertminnet och raderar alla buffertar.

FIRST (-- adr)
: ger adressen till buffertareans början.

LIMIT (-- adr)
: ger adressen till första bytet efter buffertarean.

Programhuvudena till de övriga orden används bara vid kompileringen, och är sedan onödiga.
Nedan följer några ord som löser detta problem. Det är kommandon för att kompilera vissa utvalda ord huvudlöst. Sådana rutiner skall laddas med HL-LOAD och avslutas med HEAD-OFF. De huvudlösa rutinerna definieras med :: istället för : .
HL-LOAD gör FLUSH och lånar tillfälligt en buffert från det virtuella minnet,innan själva laddningen pöbörjas. I denna buffert kompileras tillfälliga huvuden, som länkas bort från ordlistan när man beordrar HEAD-OFF.

Ex: 60 HL-LOAD

```
scr ñ60

0 :: TEST1 ." TEST" ;
1 : TEST TEST1 ;
2 HEAD-OFF ;F
3
```

Efter HEAD-OFF kan inte TEST1 anropas i nya definitioner, men kan nås indirekt via TEST.

FORTH-SKÆRMAR

```
scr #50

 0 ( Huvudløsa program 1/3, fig-FORTH. L-E. S. )
 1 1 VARIABLE HSCR  ( ant buff før temp huvuden )
 2 ' FORGET CFA @ CONSTANT 'NEXT
 3 ( adressen till inre intprn )
 4 : ADDBUF ( n -- )  ( ændrar antalet buffertar )
 5     FLUSH B/BUF 4 + * LIMIT$ +! ;
 6     ( B/BUFF ant bytes/buff )
 7 : HL-LOAD ( scr --) ( laddar huvudløsa )
 8     HSCR @ MINUS ADDBUF LIMIT 2+ LIMIT ! LATEST R# !
 9                                          LOAD ;
10 : TCREATE  ( skapar temporæra huvuden )
11     HERE LIMIT @ DP !
12     <BUILDS IMMEDIATE DUP , HERE LIMIT ! DP !
13     DOES> @ , ;
14     ( adressen LIMIT anvænds som variabel )
15 -->
16

scr #51

 0 ( Huvudløsa.. 2/3 )
 1
 2 : ::  ( ersætter : før huvudløsa )
 3     !CSP CURRENT @ CONTEXT ! TCREATE 'NEXT , A ;
 4
 5 : NEWLF  ( korrigerar lænkfæltet )
 6     PAD @ PFA LFA ! ;  ( PAD anvænds som variabel )
 7
 8 : LINK ( k m -- k n )  ( lænkar bort temp huvud )
 9     LIMIT OVER U< OVER LIMIT @ U< AND NOT
10     IF PAD @ IF DUP NEWLF
11              ELSE DUP CURRENT @ !
12              THEN DUP PAD !
13     THEN PFA LFA @ ;
14 -->
15

scr #52

 0 ( Huvudløsa.. 3/3 )
 1
 2 : HEAD-OFF  ( avslutar block med temp huvuden )
 3     0 PAD ! R# @ LATEST
 4     BEGIN OVER OVER = NOT
 5     WHILE LINK
 6     REPEAT PAD @ IF NEWLF
 7                  ELSE DROP
 8                  THEN HSCR @ ADDBUF EMPTY-BUFFERS
 9                                          DROP ;
10 ;S Kællkoden måste finnas i massminne vid
11 kompileringens børjan.
12    Vokabulær får inte ændras innan HEAD-OFF,
13    nær skærmar laddas med HL-LOAD.
14
15
```

SIG-FORTH

Nu har lite drygt 20 medlemmar anmält sitt intresse för SIG-FORTH. Det borde vara fler" Hitills har 2 utskick levererats, innehållande bla:
* källkod till programmet MASKEN av Micael Dahlquist;
* datablad till den supersnabba FORTH-processorn NC4000;
* presentationer av SIG-FORTH-medlemmarna;
* hur man laddar TI-FORTH från PB-FORTH;
* hur man styr drivens motor via CRU;
* information om FIG, FORTH Interest Group i USA;
* information om TI-FORTH International Center i USA.

Nästa SIG-FORTH-utskick kommer i mitten av september. Du får utskicken till självkostnadspris: 30 kr/år (minst 6 utskick). Pengarna ska Du sätta in på vår distributörs postgirokonto: Peter Odelryd pg 52 20 80-1.

TI-FORTH DEMO

Ifrån USA har det kommit en demoskiva, bla innehållande ett särklassigt musikprogram: "Lilla" fugan i F-moll, av J. S. Bach.
Denna kan nu köpas från föreningen i en något utvidgad version innehållande:
* musikprogrammet ovan;
* grafikdemonstration;
* spelet Breakforth;
* The Game of Life;
* kopieringsprogrammet Three Pass Copy (perfekt för kopiering av FORTH-skivor);
* Information om Demoskivan, FORTH, SIG-FORTH och föreningen.

Någon gång i höst kommer troligen ytterligare en demoskiva, med program gjorda av medlemar i föreningen. OBS Du behöver inte ha eller kunna FORTH för att kunna köra demoprogrammen, men Du måste ha drive, expansion och XB, Minimemory, E/A eller TIWriter modulen.

Come FORTH"

■

Prator

Det är inte så enkelt att komma åt alla funktioner i pratorn första gången. Hur får man den att säga de långa uttrycken i ordlistan? Jo, du måste sätta nummertecken runt det hela, t.ex.:

```
CALL SAY("ñTEXAS INSTRUMENTSñ")
```

Jag har även gjort ett kort program för att enkelt kunna testa orden i orlistan hopsatta till hela meningar. Efter varje ord trycker du på (ENTER). När meningen är klar skriv punkt följt av (ENTER). Pratorn uttalar sedan hela meningen. Du avslutar programmet med (CLEAR).

```
100 REM MULTISAY XB SS
110 REM JAN ALEXANDERSSON
120 REM VERSION 1985-05-23
130 CALL CLEAR
140 DIM A$(100),X$(100)
150 FOR I=1 TO 100
160 ACCEPT A$(I):: IF A$(I)="" THEN 160
170 IF A$(I)="." THEN M=I-1 :: GOTO 200
180 CALL SPGET(A$(I),X$(I))
190 NEXT I
200 FOR I=1 TO M
210 CALL SAY(,X$(I))
220 NEXT I
230 GOTO 130
```

■

EN NY FORTH-BOK

av Lennart Lindberg

De flesta böcker om FORTH jag hittar i hyllorna hos bokhandlare börjar från början med FORTHs elementa. Jag har berättat om några sådana tidigare. Sedan träffade jag på Leo Brodies andra FORTH-bok - Thinking FORTH. Den boken tar läsaren ett gott stycke vidare. Den rescenserades av Björn Gustavsson i Programbiten nummer 85-2. I mitt tycke är Björn kanske kritisk i överkant - jag tycker boken är väldigt trevlig och lärorik. Jag medger förstås att Björn är en åtskilligt kunnigare programmerare än jag är och kanske därför har lättare att hitta Brodies misstag.

Nu har jag hittat en bok till som inte startar från noll utan förutsätter att läsaren har passerat ett par FORTH-klasser och därför kan tillgodogöra sig överkursen. Den heter FORTH Tools and Applications av Gary Feierbach och Paul Thomas, Reston Publishing Company, Reston, Virginia, 1985, drygt 300:- kr i svensk bokhandel.

Den här boken är mycket matnyttig. Den ligger på ungefär samma svårighetsgrad och utvecklingsnivå som Thinking Forth men tar ett litet annorlunda grepp. Det fins en hel del färdiga rutiner och kluriga ord för allt möjligt man har nytta av. Se bara kapitelrubrikerna:

- Designing in Forth
- Documentation Techniques
- Debugging Techniques
- Programming Techniques
- General Utilities
- Selecting, Sorting and Searching

De mera omfattande rutiner som beskrivs handlar om utsökning, sortering och sökning i register, om ett sätt att passa in FORTH i CP/M, om dataöverföring mellan datorer, om flyttalsrutiner t ex. Det finns rutiner för att göra referenslistor på de FORTH-ord man själv skrivit, för dumpningar och ordlistesökningar mm. Rutinerna är skrivna utan anpassning till en speciell maskin. Det innebär att man måste utforma en del grundord själv för den maskin man just har.

Själv är jag speciellt förtjust i två avsnitt i boken:

- det ena är en liten rutin på åtta korta rader för att åstadkomma en återställningsbar patch (patch=en ändring i redan kompilerad kod). Med dess hjälp kan man göra ändringar i redan kompilerade ord långt ned i kärnan i ordlistan - ändringar som får effekt på alla ord som sedan använder grundordet trots att de är kompilerade när ändringen görs" Tänk efter bara vad man kan åstadkomma genom att ändra på t ex KEY och EMIT "
- det andra är en liten förteckning på tio punkter över de vanligaste felen en FORTH-programmerare gör. Och det gör jag ju. Det är bara så skönt att ha dem definierade" Själva slarvfelen verkar legitimerade genom att de finns analytiskt beskrivna, tycker jag.

Det är emellertid ett krux med boken - den handlar om FORTH-79. Alla FORTH-varianter jag känner till för 99-an tillhör FIG-FORTH-familjen. Det krävs alltså en del översättande. Nu skall problemen med detta inte överdrivas. Det handlar om ungefär samma saker som är skillnaden mellan FIG-FORTH och polyFORTH, vilket senare är den variant som Brodie använder i sina exempel i Starting FORTH. Översättningshjälp för detta finns ju i dokumentationen till såväl TI FORTH 3.0 som till föreningens FORTH. Skulle någon inte vara nöjd med detta finns en speciell översättningsbok: FORTH-79 Standard Conversion av Robert Smith, Mountain View Press, 1981.

Slutligen: boken innehåller mycket källkod till FORTH-rutiner. Att sitta och skriva av koden kan vara tröttsamt. Det finns hjälp även för detta. Författaren tilhandahåller nämligen på beställning källkod på skiva i CP/M-format och PC-DOS-format. Jag tror inte 99-an kan läsa detta direkt, men det borde gå att överföra t ex med hjälp av speciella översättningsmaskiner för skivor, som nu finns i bruk i Sverige, vet jag. Bla används de för att vid datasättning av texter kunna ta emot skivor från författare med olika ordbehandlare. ■

COMPUTE! BÖCKER

AV JAN ALEXANDERSSON

Det har kommit ut flera ny böcker från COMPUTE" som enbart behandlar TI-99/4A. Följande finns:

C.Regena: Programmer's Reference Guide to the TI-99/4A (1983) Pris USD 14.95

C.Regena: COMPUTE"'s First Book of TI Games (1983) Pris USD 12.95

R.Herold: COMPUTE"'s Guide to TI-99/4A Sound & Graphics (1984) Pris USD 12.95

COMPUTE"'s TI Collection Volume One (1984) Pris USD 12.95

C.Flynn: COMPUTE"'s Guide to Extended BASIC Home Applications (1984) Pris USD 12.95

B.Flynn: 33 Programs for the TI-99/4A (1984) Pris USD 12.95

P.Lottrup: COMPUTE"'S Beginners Guide to Assembley Language on the TI-99/4A (1985) Pris USD 14.95

Creating Arcade Games on the TI-99/4A Pris USD 12.95

TI Games for Kids Pris USD 12.95

Dessa böcker kan vara svåra att hitta i Sverige. De två första har jag köpt i Sverige och jag tror att de fortfarande finns att få tag på. Det går att köpa alla böcker direkt från COMPUTE" i USA. Addressen dit är(använd flygpost för brev):

COMPUTE" Books
P.O. Box 5406
Greensboro, NC 27403
USA

Till ovanstående pris skall läggas frakt med USD 2.00 per bok för ytpost eller USD 5.00 per bok för flygpost. Det finns två sätt att sända pengar till USA. Betalningen kan överföras med hjälp av bank. Tala med din egen bank om hur detta kan ordnas. Själv har jag dock använt kreditkort. COMPUTE" accepterar VISA, MASTER CARD eller AMERICAN EXPRESS. På din beställning måste då anges:
-Typ av kreditkort
-Nummer
-Exp.Date (sissta giltighetsdag)
-Namnteckning
-Adress till kreditkorstägaren(annan leveransadress accepteras ej)
-Flygpost eller ytpost
-Namn på beställda böcker

Till dessa kostnader kommer svensk tull och moms som tas ut av posten innan du får paketet. Kostnaden för detta blev i mitt fall 26 %.

Alla dessa böcker verkar var mycket bra och är klart köpvärda även om jag inte hunnit med att titta så noga i alla.

SOUND AND GRAPHICS

Denna är endast avsedd för Extended BASIC. Även de enklare exemplen är skrivna i detta språk. Det finns 36 olika program med varierande längd som tillsammans är 90 kbytes. Det finns följande kapitel:

1.Sound and Graphics
2.Introduction to Graphics
3.Sprites
4.Advanced Sprite-Handling Techniques
5.Sound
6.Speech Syntesis
7.Putting i All Together

TI GAMES

I "TI Games" som är sammanställd av C.Regena finns 29 bra spelprogram varav 7 stycken är för Extended BASIC.

För att du skall veta vilka program som är värda att knappa in, kan du titta i följande tabell:

BASIC-SPEL:

HIDDEN MAZE	osynlig labyrint		*
SUPERCASE	katt och rätta i labyrint		*
MARBLE HUNT	jaga punkter (ej labyrint)		-
CLOSEOUT	jaga punkter i flera husvåningar		*
TIC-TAC-TOE	tre i rad		*
BOGGLER	liknar luffarschack, två spelare	(8303)	*
FLIP FLOP	liknar BOGGLER, men datorn spelar		*
COLORCODES	liknar Master Mind		*
COPYCAT	repetera toner		**
GRID	leta hemlig koordinat		*
RIVER TRIP	manövrera i kurvig bana		*
ACROSS	manövrera under scrollning		-
ASTROSTORM	manövrera under scrollning	(8306)	-
GOBLIN	manövrera förbi hinder	(8307)	**
JUMPING JACK	hoppa över hål	(8305)	*
AIRDEFENCE	skjuta sönder fallande bomb	(8304)	**

EXTENDED BASIC-SPEL:

BALLONS	panga ballonger		-
DIAMOND DROP	fånga fallande diamanter	(8309)	**
TRAPSHOOT	skjuta lerduvor	(8303)	***
ROCKET DUEL	skjuta rymdskepp, två spelare		-
MYSTERY SPELL	gissa hemligt ord	(8309)	***
DEVASTATOR	skjuta ned "invaders"	(8408)	***
MOSAIC PUZZLE	femtonspel	(8310)	***

För de spel som tidigare publicerats i COMPUTE" anges årtal+månad inom parentes. Jag har använt följande poängsättning:

- \- ej värt att knappa in
- * hyfsat
- ** bra
- *** mycket bra

GUIDE TO ASSEMBLY LANGUAGE

Detta är en mycket bra nybörjarbok om Assembler för TI-99/4A. Exemplen i boken är skräddarsydda för Mini Memory men kan ganska lätt skrivas om till Editor/Assembler. Följande kapitel finns:

1. THE FIRST STEP
2. DIRECTIVES AND YOUR FIRST PROGRAMS
3. MORE PROGRAMING POWER
4. THE NEXT FEW STEPS
5. KEYBOARD AND JOYSTICKS
   - INPUT from Assembler
   - Moving with KEY
   - Moving with JOYSTICK
6. UTILITIES,MATEMATICS AND SCROLLING
   - Scrolling up, down or right
7. BASIC ANND ASSEMBLY LANGUAGE
   - Passing string and number to or from AL
8. CHARACTER DEFINITIONS AND COLOR CHANGES
   - Redefined cursor
   - Color changing
   - Bouncing ball
9. CREATING SPRITES
   - Sprite MAGNIFY
   - Sprite motion pixel by pixel
   - Automatic motion
   - Checking coincidences
   - Vanishing sprites
10. GENERATING SOUNDS
    - Multiple AL tones
    - Generating noise
11. GRAPHICS MODES ON THE TI
    - Screen switching
    - Text mode
    - Multicolor mode
    - Box draw
    - Bit Map bouncer
    - Bit Map draw
    - Bit Map back to graphics
12. ASSEMBLEY LANGUAGE PROGRAMING TECHNIQUES
    - Hex to decimal
    - Random numbers

TI COLLECTION ONE

Denna innehåller ett stort antal program som tidigare har publicerats i tidningen COMPUTE". Det finns både spel och tillämpningar. Innehåll:

1.GETTING STARTED
- TI Features
- Write your own games
- Easy editing (83-03)
- All about the charecter set (83-11)

2.THE BASICS
- TI BASIC one-liners (83-05)
- CALL KEY-hints
- All sorts of BASIC Sorts
- Searching Algoritms
- Transfering Variables in TI Extended BASIC
- Computer Visuals
- Using a printer (83-06)

3.APPLICATIONS
- Mailing List (83-07)
- Statistics for nonstatistican (84-07)
- TI-Calc
- Financial Interests
- A Mini Data Base Management System
- TI Word Processor (83-12)

4.RECREATION
- Trap
- Duck Leader
- Freeway 2000
- The Chase
- Thinking
- Bowling Camp
- Worm of Bremer (84-04)

5.SOUND AND GRAPHICS
- TI Graphics made easy (83-03)
- Animating TI Display without sprites
- SuperFont (85-06)
- Sound Maker
- Sound Shaper (84-03)
- The Mozart Machine (84-01)

6.SPRITES
- A beginners guide to sprites
- Sprite Editor (83-09)
- Runway 180 (83-10)

7.UTILITIES
- TI Disk Deleter
- Master Disk Directory

I avsnittet om One-liners har författaren helt i onödan skrivit tal med 10 decimaler när ju TI-99/4A jobbar med 14 decimaler. Definitionerna bör skriva så här:

```
DEF LOG10(X)=LOG(X)/LOG(10)

DEF LOG2(X)=LOG(X)/LOG(2)

DEF RAD(X)=X/(360/(8*ATN(1)))  grader till radianer

DEF DEG(X)=X*360/(8*ATN(1))    radianer till grader

DEF ACS(X)=2*ATN(1)-ASN(X)
```

EXTENDED BASIC HOME APPLICATIONS

Den innehåller följande:

1.Introduction
2.Extended BASIC Techniques
3.File Management
  Build name list
  Update name list
  Load relative file
  Update relative file
  Load indexed file
  Uppdate indexed file
  Backup relative file
  Hex dump program
4.Electronic Spreadsheets
5.Computer Graphics
6.Electronic Card File
7.Appointment calender
8.Putting it all together

33 PROGRAMS FOR THE TI-99/4A

Den innehåller följande:

1.MONEY MANAGEMENT
  Effective Yield on an Investment
  Treasury bill yields
  IRA Planner
  Municipal Bond Buyer
  Loan Payment
  Mortage Payment

2.BASICS FOR BUSINESS
  Net Present Value of cash flow
  Internal rate of Return
  Least-Squared Forecasting
  Time-series Forecasting
  Computer cash register

3.GAMES
  Brer rabit
  Rings and poles
  Matches
  Vanilla cookie

4.CURVE-FITTING ROUTINES
  Simple Correlation Coefficients
  Simple Least Squares
  Multiple Linear-Regression analysis
  t-Curve critical value
  General-Form Curve fitter

5.MATRIX MANIPULATIONS
  Matrix addition and subtraction
  Matrix multiplication
  Sweet and simple matrix inversion
  Determinant of matrix
  Matrix inversion using Gauss-Jordan
  with Complete Pivoting

6.SIMPLE STATISTICS
  Mean, Variance and Standard Deviation
  Relative and Cumulative Frequencies
  Frequency plot

7.NUMERICAL ANALYSIS
  Sort
  Random number generator
  Random number tester
  Numerical integration
  Derivate

GUIDE TO THE TI-99/4A

I Nittinian 83-4 finns en rescension av "Guide to the TI-99/4A". Detta är en bra bok. Den är en mycket tjock(358 sidor) med massor av enkla programmerings- tricks avsedda för vanlig BASIC. Det finns även många längre program.

■

==========

SÄLJES

TI 99/4A
Bandspelare, kabel
Schackmodul
Basic för nybörjare
Div. egna program

Pris för allt: 1500:-

Thomas Hedren Tel: 0470-71532
Pl 395
360 32 Gemla

==========

BYTES - SÄLJES

Jag byter bort eller säljer modulerna:

CONGO BONGO MICROSURGEON
PARSEC INDOOR SOCCER.

Vid byte vill jag ha t ex:

Adventure Moon patrol
Q-bert Miner 2049er
Buck Rogers Zork I, II, III
Pole position.

(Jag säljer modulerna till högstbjudande)

Jonas Finze Tel: 08-867628

==========

SÄLJES

TI 99/4A
med Bandspelarsladd och Joysticks
samt manual.
På band: Basic för nybörjare.
På modul:
Registerhantering, Parsec,
TI invaders, Munchman
Number magic, Mind challengers.

Pris: 1600:-

Gun-Inger Fraser Tel 046-209839

Var god ring efter kl 18.00

==========

KÖPES

Programmen Tippsy (B3ic)
  Tips (B2iu)

till TI-59:an

skriv och meddela vad Du har samt pris till:

Conny Bonet
Nyponstigen 44
152 00 Strängnäs

==========

SÄLJES

TI/99 Expansionsbox (utan enheter) 700:-

Byggbeskrivning med Layout till expansionsboxen 85:-

Roland Pettersson 031-822910

==========

LT-SUPPORT

I Programbiten 85-1 och 85-2 finns en beskrivning av LT-WRITER och LT-SUPPORT. För att den sis ta artikeln med LT-SUPPORT skall bli lättare att förstå kommer vi här med källkoden till LT-SUPPORT. Den ska dock INTE knappas in och assembleras som den står utan för detta krävs ytterligare ett program, som ej publiceras. För dig som skall använda programmet gäller det fortfarande att kontakta Lennart Thelander.

```
**********************
*
* SUPPORT PROGRAMS FOR LT-WRITER.
* CONTAINS A STRING HANDLING PACKAGE AND AN INTERRUPT DRIVEN ROUTINE WHICH GIVES
* AN ADAPTABLE CURSOR.
*
* STRING HANDLING:
*
* CALL LINK("STINIT",START)
* INITIALIZES THE STRING STORAGE AREA. START GIVES THE FIRST BYTE IN THE BUFFER.
*
* CALL LINK("STCLR",START)
* CLEARS THE STORAGE AREA. START IS GIVEN THE SAME VALUE AS WITH STINIT.
* STINIT ALSO CLEARS THE STRINGS, BUT STINIT CAN BE USED ONCE ONLY IN A PROGRAM.
*
* CALL LINK("STFREE",START,FREE)
* GIVES THE FIRST UNUSED BYTE (START), AND THE NUMBER OF BYTES FREE (FREE).
*
* CALL LINK("STSTO",ADDRESS,STRING$)
* STORES THE STRING$ AT ADDRESS. ADDRESS IS UPDATED TO POINT TO THE NEXT ADDRESS
* AVAILABLE.
*
* CALL LINK("STRCL",ADDRESS,STRING$)
* RECALLS THE STRING AT ADDRESS INTO STRING$. ADDRESS IS UPDATED TO POINT TO THE
* NEXT STRING IN THE BUFFER, OR, IF THE READ STRING WAS THE LAST, TO THE NEXT
* UNUSED BYTE IN THE BUFFER.
*
* CALL LINK("STSAVE",FILENAME$)
* SAVES THE ENTIRE BUFFER ON FILENAME$.
*
* CALL LINK("STLOAD",FILENAME$)
* READS IN THE DATA FROM FILENAME$. THE DATA IS MERGED WITH THE STRINGS CURRENT-
* LY IN THE BUFFER.
*
* CURSOR CONTROL:
*
* CALL LINK("FLIP")
* ENABLES THE ADAPTING CURSOR.
*
* CALL LINK("NOFLIP")
* DISABLES THE ADAPTING CURSOR. RESTORES THE CURSOR TO THE DEFINITION IT HAD
* WHEN FLIP WAS EXECUTED.
*
* A-DATA 841121
*
**********************
*
* WORKSPACE REGISTERS  (STRWS)
*
* R0  PARAMETER PASSING
* R1  PARAMETER PASSING
* R2  PARAMETER PASSING
* R3  PARAMETER PASSING
* R4  PARAMETER PASSING, USED BY DIVIDE IN CIRCUL
* R5  USED BY DIVIDE IN CIRCUL
* R6
* R7  BUFFER MODE FLAG
* R8  SAVED VDP WORD FROM FILE BYTE COUNT
* R9  PAB SIZE
* R10 CURRENT STRING SIZE, READ STRING SIZE
* R11 SUBROUTINE RETURN LINKAGE
* R12 SAVED RETURN LINK (IN SETPAB)
* R13 FIRST BYTE NOT CURRENTLY USED
* R14 FIRST BYTE AVAILABLE          (FSTLOW)
* R15 FIRST BYTE NOT AVAILABLE      (LSTLOW)
*
**********************

PAD    EQU  >8300
PABADR EQU  >831C
FAC    EQU  >834A
NMLPNT EQU  >8356
DSRMOD EQU  >836D
SP     EQU  >8373
STATUS EQU  >837C
GPLWS  EQU  >83E0
GPL11  EQU  >83F6

VDPPAB EQU  >1000       VDP RAM LOCATION OF PAB FOR LOAD AND SAVE
BYTCNT EQU  >1080       FIRST BYTE OF FILE BUFFER, HOLDS FILE BYTE COUNT
VDPBUF EQU  BYTCNT+2    FILE DATA BUFFER

CIF    EQU  >20         EXTENDED BASIC ROM TABLE ENTRY
CFI    EQU  >12B8
NUMASG EQU  >2008
NUMREF EQU  >200C
STRASG EQU  >2010
STRREF EQU  >2014
XMLLNK EQU  >2018
VSBW   EQU  >2020
VMBW   EQU  >2024
VSBR   EQU  >2028
VMBR   EQU  >202C
ERR    EQU  >2034
UTILWS EQU  >2038
VSWR   EQU  >23CA       VDP SINGLE WORD READ. UTILITY USED BY STRREF AND OTHERS
*                       READS ONE WORD FROM VDP ADDRESS IN R3 TO R1.
VSWW   EQU  >23E6       THE OPPOSITE OF VSWR. VDP ADDRESS SHOULD BE IN R4.
*                       NOTE: THE ADDRESS IS DESTROYED" (OR1 R4,>4000)

ERRMEM EQU  >0B00       MEMORY FULL ERROR
ERRNPP EQU  >1D00       NO PROGRAM PRESENT ERROR
ERRIO  EQU  >2400       I/O ERROR

GRMRD  EQU  >9800
GRMRA  EQU  >9802
GRMWA  EQU  >9C02

EQUMSK DATA >2000
IOERMS DATA >E000
BUFSIZ DATA 0           CPU RAM BUFFER SIZE
BUFPNT DATA 0           CPU RAM BUFFER POINTER

FSTLOW EQU  >2002       FIRST FREE ADDRESS IN LOW MEMORY
LSTLOW EQU  >2004       LAST FREE ADDRESS IN LOW MEMORY

STRWS  BSS  >20         WS FOR THIS PROGRAM
DSRWS  BSS  >20         WS FOR DSRLNK

MXFLLN EQU  26          MAXIMAL FILE NAME LENGTH

STPAB  DATA 0,BYTCNT,0,0,>6000     PAB WITH SCREEN OFFSET IN PLACE
FLNAME BSS  MXFLLN      SPACE FOR THE FILENAME
MAXLEN BYTE MXFLLN      INDICATES MAXIMAL NAME LENGTH
SAVPAB BSS  10+MXFLLN   SAVEAREA FOR VDP MEMORY AT PAB LOCATION
       EVEN

PIOCOD EQU  STPAB       ADRESSES OF THE PARTS OF THE PAB
PSTAT  EQU  STPAB+1     MOST OF THEM AREN'T USED IN THIS PROGRAM
PBUFA  EQU  STPAB+2
PRECL  EQU  STPAB+4
PCHCNT EQU  STPAB+5
PRECNO EQU  STPAB+6
PSCOFF EQU  STPAB+8
PFLLEN EQU  STPAB+9

LOAD   DATA >0500
SAVE   DATA >0600

       DEF  STINIT,STFREE,STCLR,STSTO,STRCL,STSAVE,STLOAD

**************************
*
* DSRLNK ROUTINE TO BE USED IN ASSEMBLY PROGRAMS CALLED BY EXTENDED BASIC.
* APPEARS EXACTLY LIKE THE ED/AS DSRLNK TO THE CALLING PROGRAM.
* BLWP 'DSRLNK
* DATA >8         (FOR NORMAL DSR CALL)
* IN CASE OF AN ERROR, THE EQUAL BIT IS SET UPON RETURN.
*
* NOTE: THIS ROUTINE IS AN IMPROVEMENT OVER THE ED/AS DSRLNK, SINCE THIS ONE
* ALSO CHECKS WHETHER A GPL DSR OR A ML DSR IS TO BE USED, THEREBY RELIEVING
* THE CALLING PROGRAM OF THAT TASK. THANKS TO B. GUSTAVSSON.
*
* THE NAME LENGTH POINTER MUST BE STORED '>8356 BY THE CALLING PROGRAM.
* ALSO, IN CASE OF AN ERROR THAT IS SUPPOSED TO BE REPORTED BY THE ERR ROUTINE,
* THE CALLING PROGRAM MUST STORE THE PAB POINTER '>831C.
*
* A-DATA 841011
*
**************************

DSRLNK DATA DSRWS,DSRPGM      DSRLINK ENTRY VECTOR

SAVGRM DATA 0           TO SAVE GROM ADDRESS IN
HIGROM BYTE >00         GROM TABLE ADDRESS OF LINK ROUTINE
LOGROM BYTE >10
DSRPGM MOVB 'GRMRA,'SAVGRM   SAVE GROM ADDRESS
       MOV  *R14+,R0         GET DSR MODE   (* TAKES TIME *)
       MOVB 'GRMRA,'SAVGRM+1
       DEC  'SAVGRM          CORRECT GROM ADDRESS
       MOVB 'HIGROM,'GRMWA   SET UP NEW GROM ADDRESS
       MOV  'GPL11,R10       SAVE GPL RETURN ADDRESS (* TAKES TIME *)
       MOVB 'LOGROM,'GRMWA

       SWPB R0           DSR MODE
       MOVB R0,'DSRMOD
       MOV  'NMLPNT,R3
       MOVB 'GRMRD,R0    CALCULATE GROM ADDRESS OF ROUTINE
       ANDI R0,>1F00
       SWPB R0
       MOVB 'GRMRD,R0
       SWPB R0
       INCT R0

       MOVB 'SP,R2       PREPARE PUSH ON GPL RETURN STACK
       SRL  R2,8
       AI   R2,PAD
       INCT R2

       LI   R1,>AA0C     PUSH >AA0C ON RETURN STACK
       MOV  R1,*R2
       SWPB R2
       MOVB R2,'SP       WRITE STACK POINTER BACK

       MOVB R0,'GRMWA    SET UP NEW GROM ADDRESS
       SWPB R0
       MOVB R0,'GRMWA
```

```
       MOV  '>3FE4,R2    >3FE4 HAPPENS TO BE A GOOD ADDRESS...
       LI   R1,ENDIO
       MOV  R1,'>3FE4
       LWPI GPLWS
       LI   R11,>0070
       B    *R11         BRANCH TO GPL INTERPRETER

ENDIO  LWPI DSRWS
       MOV  R2,'>3FE4    RESTORE '>3FE4
       AI   R3,-8
       MOV  R3,R0
       BLWP 'VSBR        READ PAB STATUS BYTE
       SZC  'EQUMSK,R15  ASSUME NO ERRORS

       CZC  'IOERMS,R1   CHECK ERROR CODE IN PAB STATUS
       JNE  ERROR        IF NOT ZERO THEN THERE IS AN ERROR

       MOVB 'STATUS,R2   ZERO. MUST CHECK STATUS BYTE
       CZC  'EQUMSK,R2
       JEQ  NOERR        IF ZERO THEN NOERROR

ERROR  SRL  R1,5         STORE ERROR CODE IN MSB OF R0 (CALLING WS)
       CLR  *R13
       MOVB R1,*R13
       SOC  'EQUMSK,R15  SET EQUAL BIT IN ST

NOERR  MOVB 'SAVGRM,'GRMWA    RESTORE GROM ADDRESS
       MOV  R10,'GPL11      RESTORE GPL RETURN ADDRESS
       MOVB 'SAVGRM+1,'GRMWA
       RTWP

*************************
*
* PARAMETER PASSING TO BASIC WITH INTEGER/FLOATING CONVERSION.
*

GETNUM BLWP 'NUMREF      GET PARAMETER
       BLWP 'XMLLNK      CONVERT TO INTEGER
       DATA CFI
       B    *R11

*********

PUTNUM BLWP 'XMLLNK      CONVERT PARAMETER TO FLOATING
       DATA CIF
       BLWP 'NUMASG      ASSGIN PARAMETER
       B    *R11

**********************
*
* CALCULATES CPU RAM BUFFER SIZE AND LOCATION.
*
FINSIZ MOV  '>8386,R2    END OF BUFFER AREA, ACCORDING TO BASIC INTERPRETER
       LI   R0,LAST      START OF BUFFER AREA ?
       LI   R1,>A000     FIGURE OUT IF THIS PROGRAM IS IN 24K RAM
       C    R0,R1
       JH   IN24K
       MOV  R1,R0
IN24K  S    R0,R2        CALCULATE BUFFER SIZE
       AI   R2,-100      SAFETY LIMIT, SINCE '>8386 ISN'T EXACT
       MOV  R2,'BUFSIZ
       JGT  BUFOK
       LI   R0,ERRNPP    ABORT IF SIZE IS NEGATIVE
       BLWP 'ERR
BUFOK  MOV  R0,'BUFPNT   BUFFER POINTER
       B    *R11

********
*
* INITIALIZES THE STRING STORAGE SPACE IN THE 8-K RAM PART.
*

STINIT LWPI STRWS
       MOV  'FSTLOW,R14  FIRST AVAILABLE ADDRESS
       MOV  'LSTLOW,R15  LAST AVAILABLE ADDRESS
       MOV  R15,'FSTLOW  INDICATE NO SPACE TO THE LOADER
       BL   'FINSIZ      FIND CPU RAM BUFFER SIZE AND LOCATION

CLSTRT MOV  R14,R13      INIT FIRST NOT USED BYTE POINTER
PUSTRT MOV  R13,'FAC
       CLR  R0
       LI   R1,1
       BL   'PUTNUM      ASSIGN TO START PARAMETER
       LWPI GPLWS
       B    *R11

*******
*
* GIVES THE FIRST BYTE NOT CURRENTLY USED AND THE NUMBER OF BYTES FREE.
*

STFREE LWPI STRWS
       MOV  R15,'FAC     LAST AVAILABLE BYTE
       S    R13,'FAC     SUBTRACT FIRST NOT CURRENTLY USED TO GIVE FREE SPACE.
       CLR  R0
       LI   R1,2
       BL   'PUTNUM
       B    'PUSTRT      USE COMMON PUT-START PROGRAM

*******
*
* STCLR FREES THE STRING STORAGE AREA
*

STCLR  LWPI STRWS
       BL   'FINSIZ      FIND CPU RAM BUFFER SIZE AND LOCATION
       B    'CLSTRT

******************
*
* READS THE FILE NAME USED BY STSAVE AND STLOAD.
*

GFLNAM CLR  R0
       LI   R1,1
       LI   R2,PFLLEN    POINTER TO THE FILENAME LENGTH BYTE IN THE PAB
       MOVB 'MAXLEN,*R2  MAXIMAL FILENAME LENGTH
       BLWP 'STRREF
       MOVB 'PFLLEN,R9   GET ACTUAL FILENAME LENGTH
       SRL  R9,8
       AI   R9,10        CALCULATE PAB SIZE
       B    *R11

**********
*
* LOADS THE PAB, DOES THE I/O, RESTORES THE PAB AREA
*

SETPAB MOV  R11,R12      SAVE RETURN LINK
       LI   R3,BYTCNT    ADDRESS OF VDP FILE BUFFER
       MOV  R3,R4
       BL   'VSWR
       MOV  R1,R8        SAVE IN R8
       MOV  R10,R1       NUMBER OF BYTES TO STORE
       BL   'VSWW        WRITE TO FILE BUFFER

       LI   R0,VDPPAB    SAVE DATA IN PAB AREA
       LI   R1,SAVPAB
       MOV  R9,R2        PAB SIZE
       BLWP 'VMBR
       LI   R1,STPAB     INSTALL PAB
       BLWP 'VMBW

       MOV  R0,R3        COPY OF NAMELENGTH POINTER
       AI   R0,9
       MOV  R0,'NMLPNT
       BLWP 'DSRLNK
       DATA 8
       JEQ  BADIO

       LI   R0,VDPPAB    DESTROYED BY DSR CALL
       LI   R1,SAVPAB    BYTE COUNT ISN'T CHANGED
       BLWP 'VMBW        RESTORE PAB AREA
       B    *R12         SAVED RETURN LINKAGE

BADIO  MOV  R3,'PABADR   DOESN'T GIVE THE CORRECT CODE.
       LI   R0,ERRIO
       BLWP 'ERR

*********
*
* MOVES ONE PART OF THE FILE THROUGH THE BUFFER.
* USED WHEN THE CPU RAM BUFFER ISN'T LARGE ENOUGH TO KEEP ALL THE DATA.
*

MOVBUF DATA DSRWS,MOVPGM    * USES SAME WS AS DSRLNK

MOVPGM MOV  *R13,R0       FETCH VDP BUFFER POINTER
       MOV  'BUFPNT,R1    FETCH CPU RAM BUFFER POINTER
       MOV  R1,R3         COPY BUFFER POINTER
       MOV  '6(R13),R2    FETCH BYTE COUNT
       BLWP 'VMBR         FILE BUFFER TO CPU RAM BUFFER

       MOV  '2(R13),R1    FETCH STRING ADDRESS
       BLWP 'VMBW         STORE IN FILE BUFFER

       A    R2,*R13       UPDATE POINTERS WITH BYTE COUNT
       A    R2,'2(R13)
*
* THE CPU RAM BUFFER IS MOVED TO THE STRING AREA BYTEWISE. THIS IS DUE TO THE
* BYTE ORIENTED DATA STRUCTURE USED IN THE STRING STORAGE AREA.
*
MOVLOP MOVB *R3+,*R1+
       DEC  R2
       JNE  MOVLOP
       RTWP

*******
*
* CIRCULATES THE ENTIRE FILE DATA THROUGH THE BUFFER, ONE PIECE AT A TIME.
*

STPOIN DATA 0             POINTER TO STRING STORAGE AREA USED

CIRCUL CLR  R4            SET UP FOR DIVIDE
       MOV  R10,R5        BYTECOUNT
       MOV  'BUFSIZ,R3    USED BY MOVBUF
       DIV  R3,R4         NUMBER OF BYTES/BUFFER SIZE
*
* NOW R4 HOLD THE NUMBER OF FULL CIRCULATIONS THAT ARE REQUIRED.
* R5 HOLDS THE NUMBER OF BYTES LEFT OVER FOR THE LAST CIRCULATION
*
       LI   R0,VDPBUF     POINTER TO FILE BUFFER
       MOV  'STPOIN,R1    STRING POINTER

CIRLOP BLWP 'MOVBUF       MOVE USING FULL BUFFER
       DEC  R4
       JNE  CIRLOP
       MOV  R5,R3         SURPLUS BYTE COUNT
       BLWP 'MOVBUF
       B    *R11

********
*
* SAVES THE CURRENTLY USED STORAGE AREA
*

STSAVE LWPI STRWS
       BL   'GFLNAM       READ FILENAME, CALCULATE PAB SIZE
       MOV  'SAVE,'PIOCOD      I/O OPCODE
       MOV  R13,R10       CALCULATE CURRENTLY USED SPACE
       S    R14,R10
       JEQ  RETURN        TO AVOID ERROR IN CASE OF NO BYTES AT ALL
       MOV  R10,'PRECNO   STORE IN PAB
       INCT 'PRECNO       ALLOW SPACE FOR BYTE COUNT IN THE FILE

       C    R10,'BUFSIZ   CAN THE CPU RAM BUFFER TAKE THE FILE BUFFER?
IF1    JLE  ELSE1

       MOV  R14,'STPOIN   NO, MOVE IN SMALLER PIECES
       BL   'CIRCUL
       CLR  R7            BUFFER MODE FLAG FALSE
       JMP  ENDIF1

ELSE1  LI   R0,VDPBUF     YES, MOVE ALL AT ONCE
       MOV  'BUFPNT,R1
       MOV  R10,R2
       BLWP 'VMBR         VDP TO BUFFER
       MOV  R14,R1        STRINGS TO FILE BUFFER
       BLWP 'VMBW
       SETO R7            BUFFER MODE FLAG TRUE

ENDIF1 BL   'SETPAB       SET UP PAB, DO I/O, RESTORE PAB
       LI   R4,BYTCNT     RESTORE BYTECOUNT
       MOV  R8,R1
       BL   'VSWW
```

```
       MOV  R7,R7        BUFFER MODE FLAG
IF2    JNE  ELSE2

       BL   'CIRCUL      IF FALSE, MOVE IN SMALLER PIECES
       JMP  RETURN

ELSE2  LI   R0,VDPBUF    ELSE MOVE ALL AT ONCE
       MOV  'BUFPNT,R1
       MOV  R10,R2
       BLWP 'VMBW

RETURN CLR  R0           TO BASIC
       MOVB R0,'STATUS   CLEAR GPL STATUS BYTE
       LWPI GPLWS
       B    *R11

**********
*
* LOADS STRINGS FROM A FILE INTO MEMORY. SEVERAL FILES CAN BE MERGED, IF ENOUGH
* SPACE IS AVAILABLE.
*

STLOAD LWPI STRWS
       BL   'GFLNAM      READ FILENAME, CALCULATE PAB SIZE
       MOV  'LOAD,'PIOCOD     I/O OPCODE
       MOV  R15,R10      CALCULATE AVAILABLE SPACE
       S    R13,R10
       MOV  R10,'PRECNO  STORE IN PAB
       INCT 'PRECNO      ALLOW FOR BYTE COUNT

       LI   R0,VDPBUF    USED IN BOTH THEN AND ELSE PARTS
       MOV  R10,R2
       C    R10,'BUFSIZ  COMPARE AVAILABLE SPACE WITH BUFFER SIZE
IF3    JLE  ELSE3
*
* BUFFER NOT LARGE ENOUGH
*
       MOV  R13,R1       SAVE FILE BUFFER IN STRING STORAGE AREA
       BLWP 'VMBR
       CLR  R7           BUFFER MODE FLAG FALSE
       JMP  ENDIF3
*
* BUFFER LARGE ENOUGH
*
ELSE3  MOV  'BUFPNT,R1   SAVE FILE BUFFER IN CPU RAM BUFFER
       BLWP 'VMBR
       SETO R7           BUFFER MODE FLAG TRUE

ENDIF3 BL   'SETPAB      SET UP PAB, DO I/O, RESTORE PAB
       LI   R3,BYTCNT    FILE BUFFER ADDRESS
       MOV  R3,R4
       BL   'VSWR        READ BYTECOUNT
       MOV  R1,R10
       MOV  R8,R1        RESTORE BYTECOUNT
       BL   'VSWW

       MOV  R7,R7        BUFFER MODE FLAG
IF4    JEQ  ELSE4

       LI   R0,VDPBUF    IF TRUE, THE FILE BUFFER IS IN THE CPU RAM BUFFER
       MOV  R13,R1       READ DATA TO STRING AREA
       MOV  R10,R2
       BLWP 'VMBR
       MOV  'BUFPNT,R1   RESTORE THE FILE BUFFER
       MOV  'PRECNO,R2   NUMBER OF BYTES INITIALLY RESERVED FOR THE FILE BUFFER
       DECT R2           ONLY THE DATA AREA SHOULD BE RESTORED FROM THE BUFFER
       BLWP 'VMBW
       JMP  ENDIF4
*
* IF THE BUFFER MODE FLAG IS FALSE, THE STRING STORAGE AREA HOLDS THE DATA TO BE
* RESTORED IN THE FILE BUFFER.
* NOW IT'S NECESSARY TO DETERMINE IF ALL OF THE READ DATA CAN BE TEMPORARILY
* STORED IN THE CPU RAM BUFFER, OR IF READ DATA HAS TO BE MOVED TO THE STRING
* SPACE IN SMALLER PIECES.
*
ELSE4  C    R10,'BUFSIZ  COMPARE READ NUMBER OF BYTES WITH BUFFER SIZE
       JH   ELSE5
       LI   R0,VDPBUF    R10<=BUFSIZE. BUFFER CAN HOLD ALL DATA READ IN
       MOV  'BUFPNT,R1
       MOV  R10,R2       MOVE READ DATA TO RAM BUFFER
       BLWP 'VMBR

       MOV  R13,R1       RESTORE FILE BUFFER
       MOV  'PRECNO,R2   NUMBER OF BYTES INITIALLY  RESERVED FOR THE FILE BUFFER
       DECT R2           ONLY THE DATA AREA IS RESTORED HERE
       BLWP 'VMBW

       MOV  'BUFPNT,R0   MOVE RAM BUFFER TO STRING SPACE
       MOV  R10,R2

LOOP1  MOVB *R0+,*R1+    BYTE MOVE, SINCE THE STRING STORAGE IS BYTE ORIENTED
       DEC  R2
       JNE  LOOP1
       JMP  ENDIF4

ELSE5  MOV  R13,'STPOIN  R10>BUFSIZE. BUFFER CAN'T TAKE ALL OF THE READ DATA
       BL   'CIRCUL      CIRCULATE READ DATA INTO PLACE
       MOV  R15,R2       CALCULATE NUMBER OF SURPLUS BYTES IN STRING AREA
       S    R1,R2        POINTERS ARE LEFT READY BY CIRCUL
       BLWP 'VMBW

*ENDIF5 TOO
ENDIF4 A    R10,R13
       B    'RETURN

**************************
*
* STRCL READS A STRING FROM MEMORY AT THE ADDRESS SPECIFIED.
* AFTER RECALLING, THE ADDRESS IS MODIFIED TO POINT TO THE NEXT STRING.
*

STRCL  LWPI STRWS
       CLR  R0
       LI   R1,1
       BL   'GETNUM      GET ADDRESS
       MOV  'FAC,R2
       MOVB *R2,R3       GET STRING LENGTH
       SRL  R3,8
       INC  R1           2ND PARAMETER
       BLWP 'STRASG
       DEC  R1           1ST PARAMETER
       A    R3,R2
       INC  R2           ADDRESS+LENGTH+1
       MOV  R2,'FAC
       BL   'PUTNUM
       B    'RETURN

********
*
* STSTO IS THE OPPOSITE OF STRCL. IT STORES A STRING AT THE ADDRESS PROVIDED.
* AFTER STORAGE THE ADDRESS IS MODIFIED TO POINT TO THE NEXT FREE BYTE.
*

STSTO  LWPI STRWS
       CLR  R0
       LI   R1,1
       BL   'GETNUM      GET ADDRESS
       MOV  'FAC,R2

       MOV  R15,R3
       S    R13,R3       CALCULATE FREE SPACE
       CI   R3,2         SMALLEST POSSIBLE STRING
       JL   MEMFUL       MEMORY FULL ERROR
       CI   R3,>FF       LONGEST STRING
       JLE  LTHFF        IF LESS THAN >FF, KEEP LENGTH AS IT IS
       LI   R3,>FF       ELSE SET IT TO >FF
LTHFF  SWPB R3
       MOVB R3,*R2       STORE MAX LENGTH IN LENGTH BYTE

       INC  R1           2ND PARAMETER
       BLWP 'STRREF      AUTOMATICALLY ABORTS IF NOT ENOUGH SPACE
       MOVB *R2,R3       FETCH ACTUAL LENGTH
       SRL  R3,8
       A    R3,R2
       INC  R2           ADDRESS+LENGTH+1
       C    R2,R13       CHECK IF OUTSIDE CURRENTLY USED AREA
       JLE  NOTMOR
       MOV  R2,R13       IF SO, UPDATE CURRENTLY USED POINTER
NOTMOR DEC  R1           1ST PARAMETER
       MOV  R2,'FAC
       BL   'PUTNUM      NEW ADDRESS BACK
       B    'RETURN      TO BASIC

MEMFUL LI   R0,ERRMEM    MEMORY FULL ERROR
       BLWP 'ERR
**********************
*
* ADAPTABLE CURSOR DEFINITION PROGRAM.
*

FLIPWS BSS  >20
FLWS4  EQU  FLIPWS+8

CURCOD BYTE 96+30         CURSOR CHARACTER CODE
SPCOD  BYTE 96+32         SPACE CHARACTER CODE
CURDEF EQU  1008          ADDRESS OF CURSOR DEFINITION
CHR    EQU  >8301         THE LOCATION IN PAD WHERE A CHARACTER IS STORED WHILE
*                         THE CURSOR OCCUPIES THE CHARACTER'S POSITION.
INTLNK EQU  >83C4         USER DEFINED INTERRUPT POINTER

NRMCUR BSS  8             NORMAL CURSOR DEFINITION

       DEF  FLIP,NOFLIP

FLIP   LWPI FLIPWS
       CLR  R0
       C    'INTLNK,R0   ALREADY IN FLIP MODE?
       JNE  OUT
       LI   R15,>FEFE    CURSOR DEFINITION MASK
       LI   R0,CURDEF    SAVE NORMAL CURSOR DEFINITION
       LI   R1,NRMCUR
       LI   R2,8
       BLWP 'VMBR
       LI   R0,ADAPT
       MOV  R0,'INTLNK   ENABLE INTERRUPT
       JMP  OUT

NOFLIP LWPI FLIPWS
       CLR  R0
       C    'INTLNK,R0   ALREADY IN NOFLIP MODE?
       JEQ  OUT
       CLR  'INTLNK      DISABLE INTERRUPT
       JMP  SPACE        RESTORE NORMAL CURSOR

ADAPT  LWPI FLIPWS
       CB   'CHR,'CURCOD IS THE CURSOR ON SCREEN?
       JEQ  OUT          NO, SO NO USE TO REDEFINE IT

       CB   'CHR,'SPCOD  IS THE CURSOR WHERE A SPACE SHOULD BE?
       JEQ  SPACE        YES. SPECIAL CASE

       CLR  R0
       MOVB 'CHR,R0      READ CHARACTER CODE
       SRL  R0,5         CALCULATE DEFINITION ADDRESS
       LI   R1,FLWS4     STORE DEFINITION IN R4-R7
       LI   R2,8
       BLWP 'VMBR        READ DEFINITION

       XOR  R15,R4       MAKE THE CURSOR AN INVERTED COPY OF THE CHARACTER IT
       XOR  R15,R5       REPLACED
       XOR  R15,R6
       XOR  R15,R7

       LI   R0,CURDEF    ADDRESS OF CURSOR DEFINITION
       BLWP 'VMBW        WRITE NEW DEFINITION OF THE CURSOR

OUT    LWPI GPLWS
       B    *R11

SPACE  LI   R0,CURDEF    THE SPACE IS A SPECIAL CASE, IN ORDER TO AVOID A LARGE
       LI   R1,NRMCUR    CURSOR WHEN IT ISN'T NEEDED
       LI   R2,8
       BLWP 'VMBW
       JMP  OUT

**********

LAST
```

■

OHMS LAG

Vi visar OHMS LAG som ett exempel på ett enkelt och bra strukturerat program./RED.

```
100 REM********************
110 REM* OHMS LAG         *
120 REM* WATTS LAG        *
130 REM* SKRIVET AV       *
140 REM* GÖRAN NYGREN     *
150 REM* UPPINGEGRÄND 10 *
160 REM* 163 61 SPÅNGA   *
170 REM* TEL:08-7615432  *
180 REM* TI-BASIC        *
190 REM* VERSION 1.0      *
200 REM********************
210 REM
220 REM*******************
230 REM* SVENSKA TECKEN *
240 REM*******************
250 REM
260 GOSUB 1920
270 REM
280 REM**************
290 REM* HUVUDMENY *
300 REM**************
310 REM
320 REM*******************
330 REM* NOLLSTÄLL ALLT *
340 REM*******************
350 REM
360 CHOICE=0
370 I=0
380 P=0
390 R=0
400 U=0
410 CALL CLEAR
420 PRINT "HUVUD MENY": : : :
430 PRINT "DETTA PROGRAM RÄKNAR UT": :
440 PRINT "OHM, VOLT, AMPERE, WATT.": : : :
450 PRINT "1. OHMS LAG (U/R*I)": :
460 PRINT "2. WATTS LAG (P/U*I)": :
470 PRINT "3. AVSLUTA": : : :
480 INPUT "DITT VAL:":CHOICE
490 IF (CHOICE<1)+(CHOICE>3)THEN 510
500 ON CHOICE GOSUB 610,1170,520
510 GOTO 360
520 INPUT "ÄR DU SÄKER (J/N):":REGRET$
530 IF REGRET$="J" THEN 550
540 GOTO 360
550 END
560 REM
570 REM************
580 REM* OHMS LAG *
590 REM************
600 REM
610 CALL CLEAR
620 PRINT "OHMS LAG (U/R*I)": : : :
630 PRINT "MATA IN LÄMPLIG SIFFRA": :
640 PRINT "FÖR ATT VÄLJA DET DU VILL": :
650 PRINT "HA SVAR PÅ": : : :
660 PRINT "1. RÄKNAR VOLT": :
670 PRINT "2. RÄKNAR AMPERE": :
680 PRINT "3. RÄKNAR OHM": :
690 PRINT "4. HUVUD MENY": : : :
700 INPUT "DITT VAL:":CHOICE
710 IF (CHOICE<1)+(CHOICE>4)THEN 740
720 IF CHOICE=4 THEN 750
730 ON CHOICE GOSUB 810,930,1050
740 GOTO 610
750 RETURN
760 REM
770 REM**************
780 REM* RÄKNA VOLT *
790 REM**************
800 REM
810 CALL CLEAR
820 INPUT "AKTUELL OHM:":R
830 INPUT "AKTUELL AMPERE:":I
840 U=R*I
850 PRINT : :"VOLT BLIR";U;"V": :
860 GOSUB 1730
870 RETURN
880 REM
890 REM****************
900 REM* RÄKNA AMPERE *
910 REM****************
920 REM
930 CALL CLEAR
940 INPUT "AKTUELL VOLT:":U
950 INPUT "AKTUELL OHM:":R
960 I=U/R
970 PRINT : :"AMPERE BLIR";I;"A": :
980 GOSUB 1730
990 RETURN
1000 REM
1010 REM**************
1020 REM* RÄKNA OHM *
1030 REM**************
1040 REM
1050 CALL CLEAR
1060 INPUT "AKTUELL VOLT:":U
1070 INPUT "AKTUELL AMPERE:":I
1080 R=U/I
1090 PRINT : :"OHM BLIR";R;"OHM": :
1100 GOSUB 1730
1110 RETURN
1120 REM
1130 REM**************
1140 REM* WATTS LAG *
1150 REM**************
1160 REM
1170 CALL CLEAR
1180 PRINT "WATTS LAG (P/U*I)": : :
1190 PRINT "MATA IN LÄMPLIG SIFFRA": :
1200 PRINT "FÖR ATT VÄLJA DET DU VILL": :
1210 PRINT "HA SVAR PÅ": : : :
1220 PRINT "1. RÄKNAR WATT": :
1230 PRINT "2. RÄKNAR VOLT": :
1240 PRINT "3. RÄKNAR AMPERE": :
1250 PRINT "4. HUVUD MENY": : : :
1260 INPUT "DITT VAL:":CHOICE
1270 IF (CHOICE<1)+(CHOICE>4)THEN 1300
1280 IF CHOICE=4 THEN 1310
1290 ON CHOICE GOSUB 1370,1480,1600
1300 GOTO 1170
1310 RETURN
1320 REM
1330 REM**************
1340 REM* RÄKNA WATT *
1350 REM**************
1360 REM
1370 CALL CLEAR
1380 INPUT "AKTUELL VOLT:":U
1390 INPUT "AKTUELL AMPERE:":I
1400 P=U*I
1410 PRINT : :"WATT BLIR";P;"W": :
1420 GOSUB 1730
1430 RETURN
1440 REM
1450 REM**************
1460 REM* RÄKNA VOLT *
1470 REM**************
1480 CALL CLEAR
1490 INPUT "AKTUELL WATT:":P
1500 INPUT "AKTUELL AMPERE:":I
1510 U=P/I
1520 PRINT : :"VOLT BLIR";U;"V": :
1530 GOSUB 1730
1540 RETURN
1550 REM
1560 REM****************
1570 REM* RÄKNA AMPERE *
1580 REM****************
1590 REM
1600 CALL CLEAR
1610 INPUT "AKTUELL WATT:":P
1620 INPUT "AKTUELL VOLT:":U
1630 I=P/U
1640 PRINT : :"AMPERE BLIR";I;"A": :
1650 GOSUB 1730
1660 RETURN
1670 REM
1680 REM****************
1690 REM* TID ATT LÄSA *
1700 REM* AV SVARET    *
1710 REM****************
1720 REM
1730 PRINT : :"TRYCK ENTER"
1740 CALL KEY(3,K,S)
1750 IF S=0 THEN 1740
1760 REM
1770 REM*******************
1780 REM* NOLLSTÄLL ALLT *
1790 REM*******************
1800 REM
1810 CHOICE=0
1820 I=0
1830 P=0
1840 R=0
1850 U=0
1860 RETURN
1870 REM
1880 REM*******************
1890 REM* SVENSKA TECKEN *
1900 REM*******************
1910 REM
1920 CALL CHAR(91,"004400384470444")
1930 CALL CHAR(92,"002800384444438")
1940 CALL CHAR(93,"001000384470444")
1950 RETURN
```

DATAVISION MED TEXVISION

Av Michael Öhman

Kärt barn har många namn. Teledata, videotex och viewdata är några exempel. Televerket har kallat deras nya tjänst DATAVISION.
Datavision är ett enkelt sätt att få och föra ut information, skicka meddelanden och göra beställningar. Allt sker på ett standardiserat sätt. Datavision är en dubbelriktad kommunikation.

Detta var ett litet utdrag ur Televerkets broschur om Datavision.

Allt som behövs är ett modem, en TI99/4a, ett RS232 och ett terminalprogram för att ta fram den information du söker.

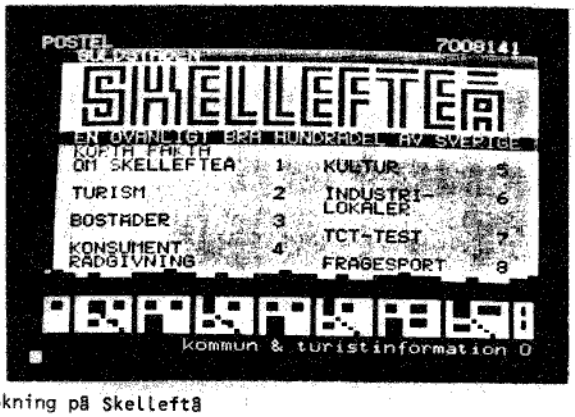
Terminalprogrammet för att kunna kommunicara med Datavision har fått namnet Texvision. Programmet tar emot 1200 BPS(Bitär per sekund) och sänder 75 BPS vilket motsvarar en sändningshastighet av c:a 7,5 tecken per sekund och mottagning med 120.

I Datavision sänder man över bilder med 40 kolumner och 24 rader. Varje position skall kunna ha sin egen för och bakgrundsfärg. 8 färger kan maximalt användas. För att kunna göra detta på en TI99/4a måste man gå in i högupplösningsmode, s.k. BIT-MAP-MODE. Inte ens då är färgupplösningen helt tillräcklig, men ofta används inte flera olika färger bredvid varandra. I och med att man går in i högupplösningsmode uppstår ett nytt problem: programmet skall kunna tömma skärmen 120 ggr per sekund och det är 13 Kbytes den skall radera. Detta möjliggörs med buffring av inkommande data.

Vad kan man nu göra i Datavision ?

Här följer några urval av bilder:

Inloggningsbild



Sökning på Skellefteå

Sökning av internationella kök i Stockholm

Sökning på bilar

Genom sökning av ord t.ex. "Restauranger" fås en bild med val över länder (f.n. endast Skandinavien). Sverige väljs och ännu ett val kommer upp:
städer i Sverige. Efter val av stockholm kommer en meny upp över vilken typ av restaurang vi letar efter. Jag väljer internationella kök. En ny bild blir synbar (se. bild 2) härifrån kan jag sedan gå in på de olika restaurangerna och se vad de har för utbud. I framtiden kommer man även att kunna beställa bord osv.

Texvision har möjlighet att spara bilderna på diskett för de som har ett disksystem . För en snabb lagring av bilder har ett nytt diskhanteringssystem utvecklats. Bildnamnet får vara högst 16 tecken långt. Ett grundnamn ges varpå namnet kommer att öka på sig självt varje gång man spar en bild. Detta för att inte behöva skriva dit bildnamnet varje gång. När man sedan i lugn och ro gått ur Datavision kan man efteråt ändra bildnamnen.

För de som har en skrivare ansluten till en seriell utgång kan en hårdkopia av texten på skärmen skrivas ut. En meny finns att tillgå för att ställa parametrarna för skrivare.

Det mesta finns att tillgå i Datavisionen såsom banker, förtag av alla de slag, nyhetsbyråer, tidningar, myndigheter, datorklubbar, mindre baser och mycket mer.

Hela telefonkatalogen är till exempel inlagd.

Nödvändig utrustning:

TI99/4a
RS232 (finns som fristående)
Modem 1200/75 BPS (kan hyras billigt på televerket)

Valfri utrustning: Expansionbox
Diskkontroller
Diskdrive

■

PROGRAMBANKEN

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 datamediet, 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 skrivet i TI-Basic, Extended-Basic, Assembler, Pascal eller i Forth. Beskriv noggrant vilken utrustning som behövs (även i själva programmet), hur man startar upp och vad som behöver göras för att programmet skall fungera. Skriv även vilken modul som erfordras till programmet. Se KODFÖRKLARING"

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

10 ** REGISTERPROGRAM
01101002E --- Adressregister
11103069 --- Filläsning på Kassett
11103072 --K Snabbfil till Personal Record Keeping modulen
14103045 -DX Mini-Reg, Utman.84/3
15103092 -DX Tel-Ad, Telefonadresser

12 ** ORDBEHANDLING
10121010E --C Ordbehandling för TP-printer.
01121001E --P Word-processor
11123064 --- P:TEXTIN
11123065 --- P:TEXTUT, Lista P:TEXTIN filer
14123039 -PM Nova-Verba

14 ** EKONOMI
07141001H --X Annuities

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

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

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

30 ** LINJÄR ALGEBRA
06301001E --- Matris invertering/multiplikation
14303050 --X Lösning av linjära ekvationssystem med Gauss-eliminationsmetod

39 ** ALLMÄN MATEMATIK
01391001E --- Integral, derivata, andragradsekv. reella rötter, cp analys, definiera chars, 3-D plot, fakultet och primtalstest
09391003H --- Differential eq.
11393071 --X Rita sinuskurva
14393051 --X Math-Pac, Integration, Newton-Raphson, Intervallhalvering, Romberg-integration

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

78 ** ASTRONOMI
01781001E --- Beräkning av geostationära satelliters positioner.
15783000 --- Trippelpunkten på 99:an, Tyngdpunkten till en sfärisk triangel

90 ** PRAKTISKA PROGRAMHJÄLPMEDEL
06901003E --X Sortering och utskrift från flera disketter av upp till 300 program
06901004E --- Diskkatalog med utskrift
08901006E -XE Utskrift av titelsida för ex. listningar
10901008E --- Very large charachters
11901011S --- Stapeldiagram
07902002 --P Banner, stor text utskriven på printer
11903063 DPX Lista Basicprogram med Basicprogram
11903066 --X PGM LIST59, Lista TI59 program på TI99/4A
11903067 --- Screen-Saver, spara skärmen
11903068 --- Testbild för Din Dator
11903075 -DX Lista DIS/VAR Filer
12901054 -JX Color-Draw, Ritprogram med färger
12903055 --- Screen-Ram
13903024 --X Grafik Exklusive, Kassettbaserat
13903022 -DX Fil-Data-Editor, för DIS/VAR80
13901057 --X Cursor, ändra utseendet på cursorn
13901058 --- Ensa-komma
13903060 -DX List28, Lista program 28tkn/bredd
13903061 -PX ASCII-koder för olika tecken-uppsättningar
13903062 -DX P:TILLCOMP, överför P:TEXTIN filer till Companion format
14903034 --- Teckendefinierare
14903042 --- Kassettinnehåll
14902052 --X Färg-Editor
14901053 --X Talkonvertering, Hex-Dec-Binärt
15903005 --- Baskonvertering, Hex-Dec-Bin
15903098 -JX Tommy Erikssons Sprite-Maker
15903094 -JX Seppo Huikuris Sprite-Maker
15903076 DPX Listprogram, Listar Basicprogram
15903077 -SX Trunkering, skapa egna ord med trunkering
15903088 -EX List-Size, listar storleken på dina program
15901089 -DX Merge/Read, listar DIS/VAR163 filer
15901090 -EX Mg/Peeker, listar minnes-adresser
15903093 -DX Char-Adapt, Teckenändring, lagras som DIS/VAR80 filer
15903095 DEX Load-Assembler-Bas, omformar Assembler program till Extended-Basic program
15903001 EAD Disassembler, en fullständig disassemblerare
15903002 EAD K:INPUT, subprogram för inmatning av data från tangentbord
16903006 -DM Diskkatalog i Textmode

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 --X 3D Tic-tac-toe
03911017E --- Black-Jack II
03911018E --X Not one
03911019E --X Datorpoker
03911024E --- Fyra i rad (luffarschack)
04911025E --- Startrek
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
12911001S --- One Check, brädspel
14911037E --X One Check II
12913002 --J Jockes Hostility
12913004 --- Giljotinen, ordlek
12913007 --- Tjugoett
12913008 --- Totto
12913009 --- Octopus
12913010 --- Tipsservice
12913012 --- The Hunt of Aladin
12913013 --- Bowling
12913014 --- Bilrally i Norge
12913015 --- Masken
12913016 --- Djungelfight
12913017 JXB Snake, Ormen
12913018 JXB Zombie
13913019 --J Kami-Kaze
13913020 --X Ubåt
13913023 --- Jumping Jack
13913024 --- Astrostorm
13913025 --- Goblin
13913026 --- Identify the States
14913035 --X Skroting
14913036 -JX Starport 99
14913040 --- Blockman
14913044 --X Mastermind
14911046E --X Pricka flygplan
14912047H --X Rymdslaget
14911048E --X Spacegame
15913079/1 --X Adventurespel, Gör dina egna Adventure-spel, (1-9)
15913080/2 --X Adventurespel
15913081/3 --X Adventurespel
15913082/4 --X Adventurespel
15913083/5 --X Adventurespel
15913084/6 --X Adventurespel
15913085/7 --X Adventurespel
15913086/8 --X Adventurespel
15913087/9 --X Adventurespel

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 --- Räknetränning med bråkdelar
11923074 --X Fransk glosträning
13923031 --X Byggnadsmekanik
13923029 --X Huvudräkning
13923030 --X Glosträning, kassettbaserad

96 ** MUSIK
08961001E --X Killing me softly
05962001 --C A Strauss walz
05962002 --- Never on a sunday
05962003 --X Berceuse
05962004 -XC Beethoven ñ
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 Bumble-boogie
06962011 --D Fiddler on the roof
07962013 --- Looking through you
08962014 --X Chopin op 11 ñ
06962001 --- The pink panther
06962006 --- Let me call you sweetheart
12963056 --- Tremolo
14963033 --- TI-Keyboard
14963043 --- When the Saints ____

97 ** DEMO
07971003H --- Demonstration av olika tecken-storlekar
11973070 -EX Demoladdning av Assembler program
13973059 --- Korsord
14973038 --X Mönstergrafik
15973097 --- Snabbare grafik med Ti-Basic
15973099 --X Doppler-effekt
15971091 -EX Mg/Text, Textmode
15973003 --M TextMode i TI-Basic
15973004 --M Auto-Sprite i TI-Basic

99 ** ÖVRIGT
01991001E --X Släktforskningsregister
01991002H --- Utskrift av månadskalender
09991003F --- Biorythm.
01992001 --- Månkalender
11993073 -PX Tipsrad
13993028 --X Biorytm II
13993021 --P Etiketter
14993032 --X Biorytm III
14993041 --- Resultatlista, Slalom
14993049 --X Tips-system
15993078 -EX Automatisk uppringning
15993096 --- Dagräkning med almanacka
07991005H --- Morse
10991007H --- Lotto

KODFÖRKLARING

De fyra första siffrorna är interna beteckningar (diskettnummer, programkategori).
Femte siffran anger ursprungsland:

1 = England, USA
2 = Holland
3 = Sverige
4 = Frankrike

De tre sista siffrorna är ett internt löpnummer. Bokstaven som finns sist anger vilket språk som används i programmet.

E = Engelska
F = Franska
H = Holländska
S = Svenska

Eventuella bokstäver efter programnumret och bokstaven talar om vilken utrustning som krävs för att kunna använda programmen som de är:

C = CALL FILES (1) måste användas om man har diskettenhet.
D = Diskettenhet
E = Minnesexpansion
F = Forth
A = Editor/Assembler
M = MiniMemory
X = Extended-Basic
P = Printer
S = Speech Synthesizer
J = Joystick
(pratorn)
T = Terminal Emulator II
K = Personal Record Keeping

Ett + anger att programmet är uppdelat i flera delar.