PROGRAM

nittinian

BITEN

86-87

FORTH spalten

REDAKTÖR: Lars-Erik Svann

HELI-KOP-TER

TRYCK NAGON TANGENT

STJÄRN KIKERI

19W/86

Innehåll

ISSN 0281-1146

Redaktören. . .

Detta nummer av Programbiten har vi gjort något tunnare än tidigare. Claes har nämligen räknat ut att det är så mycket billigare att trycka 24 sidor mot tidigarte 28 sidor att vi kan ha råd med ett helt extra nummer. Enligt nuvarande planer kommer vi att jobba med följande tider när det gäller manusstopp:

PB 86-3	31 mars
PB 86-4	31 maj
PB 86-5	30 september
PB 86-6	15 november

Sänd in dina bidrag i god tid, speciellt om du tror att redaktören måste bearbeta ditt program eller artikel. Du skall dock försöka göra den så färdig att vi kan trycka den utan ändringar. Skicka även med en beskrivning av hur programmet skall användas och hur det är strukturerat. Du bör speciellt påpeka de rader där du gjort något smart. Tänk på att vi trycker med en spaltbredd på 55 tecken. (Vid 12 tecken per tum.) Spaltbredd vid montering = 115 mm.

Ditt program blir speciellt användbart för övriga medlemmar om du skriver i grund-BASIC och gör att filer kan lagras på både kassett och diskett. Om du använder expansionsminne 32k bör du kanske göra två versioner så att alla får maximal nytta av ditt program.

Om programmet använder printer bör du skriva en REM-sats i början där du hänvisar till vilka rader som kan behöva ändras. Printrar kan ju ha mycket olika kontrollkoder. Förklara även noga vad de olika kontrollkoderna förväntas göra.

I början av ditt program bör du ha ett antal REM-satser som talar om följande:

```
REM programmets namn
REM ditt namn
REM nödvändiga moduler och tillbehör
REM rader som kan behöva ändras av användaren
REM kort beskrivning av vad programmet gör
REM hänvisning till böcker eller tidningar
REM (lämna en tom REM-sats för PB-nummer)
```

Gör alltid RESEQUENCE av ditt program innan du skickar in det.

Till dig som har TI-59 skall jag säga att vi har inte glömt bort dig. Det är dock viktigt att du kommer in med bidrag. Det skulle vara speciellt trevligt om vi kunde få bra program med listningar både för TI-59 och TI-99.

TELEFONTID

Förenigen har telefontid tisdagar kl. 18-20 telefon 08-761 51 62.

PROGRAMFÖRMEDLING

Vi har fått tag i några FREEWARE som vi hoppas presentera i nästa nummer. Vi håller även på och sammanställer en flexskiva med alla publicerade program för 1985. Eventuellt kommer den att distribueras på samma sätt som FORTH. Vi tittar också på möjligheten att överföra det till kassett.

Vi har även andra sakar på gång till PB 86-3 t.ex. SL-DATABAS för XB+EM, HARDCOPY till printer, svenska tecken till editorn i EDITOR/ASSEMBLER, sammanställning av enkäten till medlemmarna och minst två spel. Vi har dock plats för mycket mer så vi väntar på ditt bidrag.

FREEWARE		Beskrivning kommer i nr 86-3.
NEATLIST	60 SEK	
DISK MANAGER	60 SEK	
SCREEN DUMP	60 SEK	

I REDAKTIONEN

Redaktör --> 86-2	Jan Alexandersson
Redaktör 86-3 -->	Peter Odelryd
Utmaningsredaktör	Anders Persson
Forth-redaktör	Lars-Erik Swahn
Programförmedlare	Börje Häll
Allt-i-allo	Claes S

Föreningens och redaktionens adress:

```
Föreningen Programbiten        ++++++++++++++++++++++++++
c/o Schibler                   +   DATAINSPEKTIONENs   +
Wahlbergsgatan 6  1 tr ned     +     LICENSNUMMER      +
121 46  Johanneshov            +                       +
                               +       82100488        +
Postgiro 19 83 00 - 6          ++++++++++++++++++++++++++
```

Medlemsavgift för 1986 är .....120 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:-
Programbiten - Nittinian 1985, TI-59+99		100:-
Programbiten - Nittinian 1984, TI-59+99		80:-
Nittinian, årgång 1983 Datorn 99/4A		60:-

För TI-59 mm

Programbiten, årgång 1983	Kalkylatorer	40:-
Programbiten, årgång 1982	-"-	40:-
Programbiten, årgång 1981	-"-	40:-
Programbiten, årgång 1980	-"-	40:-
Programbiten, årgång 1978/79	-"-	40:-
Programbiten, fem årgångar 1978-1983		160:-
Programbiten, sex årgångar 1978-1984		220:-
Programbiten, sju årgångar 1978-1985		300:-
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		SLUT
Tom magnetkortsplånbok		10:-

ORDFÖRANDEN har ORDET

NY STYRELSE

Vid årsmötet 8 februari i valdes följande ledamöter:

Ordförande	Jan Alexandersson
Kassör	Mikael Nordlin
Redaktör	Peter Odelryd
Sekreterare	Hans Wickström
Ledamöter	Claes Schibler (allt-i-alllo)
	Michael Öhman
	Börje Häll (programförmedlare)
	Tony Rogvall
	Anders Törnkvist

Årsmötesprotokoll och övriga beslut kommer att publiceras i nästa nummer. I praktiken betyder detta att Peter Odelryd börjar som redaktör fr.o.m. PB 86-3. Du som varit med i SIG-FORTH vet att han har skött utskicken där tillsammans med Lars-Erik Svahn.

GRUND-BASIC

Till dig som endast har TI-99/4A med grund-BASIC och kanske tycker att du fått ut för litet av tidningen tänkte jag göra ett försök med att vi träffas hemma hos mig i små grupper på c:a 6 personer. Du som är intresserad av detta kan skriva till mig. Min adress är:

Jan Alexandersson
Springarvägen 5, 3 tr
142 00 TRÅNGSUND

Tel. 08-771 05 69

Min tanke var att träffas en måndagkväll mellan 18 och 21. Vi skulle då kunna titta på hur hur mycket som kan göras med BASIC. Jag tänkte att vi skulle hålla på med både spel, grafik och nyttoprogram för registerhantering m.m. Även du som tycker att du kan väldigt litet är välkommen. Det kommer att fiinas möjlighet till att kopiera program så tag med kassett eller flexskiva.

Även du som kan mycket mer men gärna vill hjälpa medlemmarna med grund-BASIC kan höra av dig.

Om många hör av sig kommer vi att träffas flera måndagar i den takt som jag orkar med. Jag hör av mig till dig om datum senare.

Till dig som bor långt från Stockholm (Trångsund ligger 20 min söderut med pendeltåg) kan jag göra följande förslag. Om du sätter in 20:- på mitt postgirokonto 61 68 86-8 så sänder jag en 20 min kassett med BASIC-program. Ange om du vill ha spel, nyttoprogram eller blandat. (för 25:- får du samma sak på en flexskiva). Om många hör av sig får du ha tålamod med att det kan ta en viss tid.

Jan Alexandersson

Lämnad till tryck 1986-03-14

◀ 24

PROGRAMBANKEN

Börje Häll har ändrat kategorier på en del program, varför de återfinns under annan rubrik än i tidigare programbankslistor.

/csch

91 SPEL

06911034H	--X L-game
06912004E	--- Slalom
06912005E	--- Killer
07911035H	--X Wari
07911036H	--- Ta dig fram i en osynlig labyrint
07911037H	--X Eliza
07911040H	--X Black jack
07911041H	--X Space battle
07911043H	--- Enarmad bandit
07911045E	--- Labyrint. 2 spelare
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	--- Breakout, bollspel squashtyp
09911059H	--J Ritprogram i olika färger
09911060H	-JX Treasure hunt, labyrintspel
09912008F	--X Asteriod (tal)
09912009E	--- Battlestar
10911061H	--X Kermit
10911062H	-JX Find the gun
10911063E	--- Matematikspel
10911065E	--- Camel, adventureritt i öknen
11911055S	--X Hängning
11917001S	-JX Inkräktarna
11917002S	-JX Päron
11917003S	-JX Masken
12911001S	--X One Check, brädspel
12913002	-JX Jockes Hostility
12913004	--X Giljotinen, ordlek
12913007	--- Tjugoett
12913009E	--- Octopus
12913012	--- The Hunt of Aladin
12913013	--- Bowling
12913014	--- Bilrally i Norge
12913015	--- Masken II
12913016	--- Djungelfight
12913017	-JX Snake, Ormen
12913018	-JX Zombie
13913019	--J Kami-Kaze
13913020	-JX Ubåt
13913023	--- Jumping Jack
13913024E	--- Astrostorm
13913025	--- Goblin
13913026E	--- Identify the States
14911037E	--X One Check II
14911046E	--X Pricka flygplan
14911048E	--X Spacegame
14913035	--X Skroting
14913036	-JX Starport 99
14913040E	--J Blockman
14913044	--X Mastermind
15913079	--X Adventurespel, Gör dina egna Adventurespel
16913001	--- Helikopter
16913002	--X Lander, labyrintspel
16913003	--X Dart
16913005	--J Ormar, (liknar Masken)
16913008	-JX NUC, Atombombsspel. Försvara staden

▶ 24

Att hitta på en skiva

av Anders Persson, efter inspiration från Lennart Lindberg.

Många har säkert av och till undrat över hur data är lagrade på en diskett. I manualen Disk Memory System ges visserligen vissa upplysningar om sektorer mm, men det är mycket allmänt hållet. En del kan man kanske sluta sig till av program-exemplet för att läsa katalogen på skivan. För den som vill kunna hantera filerna på skivan på andra sätt än DSR förutsätter finns emellertid inte mycket att hämta. (DSR står för Device Service Routine, och är i det här fallet de program på skivkontrollkortet som sköter om filhanteringen.) Det visar sig också att TI inte har lämnat ut mycket upplysningar om detta. De har tydligen betraktat det som affärshemligheter. Den attityden var för övrigt troligen vad som fick 99:an att bli en dålig affär för dem - det blev för svårt för andra att utveckla program för maskinen.

Nu finns det faktiskt en del dokumentation i alla fall. Genom att studera källkoden till Forth kan man komma underfund med en del. Den koden har Texas släppt ut till allmänt betraktande, nu när datorn ändå har gått åt pepparn till, för dem. Dessutom finns det ett dokument som heter "Texas Instruments Terminal Emulator Protocol Manual". Det skrevs 1981, men har såvitt jag vet inte sålts till användare av 99:an. Avsikten med den här manualen är att beskriva hur TE-modulen kommunicerar med andra datorer, så att man ska kunna skriva speciell programvara för de datorer Texasen ska "prata" med. Dessa program ska kunna utnyttja de specialfunktioner som TE-modulen kan ge. En av dessa funktioner är överföring av filer. Det är därför som filformatet beskrivs i "Protocol Manual".

Dessutom finns det förstås folk som har luskat ut en del. En sådan person är Bryan Wilcutt från en användarförening i USA. Han har skrivit en snutt om vad han har hittat. Den har sedan förbättrats av Mike Walker från Brevards User's Group i Palm Bay, Florida. Nedan kommer en översättning av deras beskrivning, med en del rättelser och tillägg som jag har gjort.

En skiva består, logiskt sett, av ett antal sektorer. Det kan finnas 360 sektorer (SS/SD), 720 sektorer (DS/SD eller SS/DD) eller 1440 sektorer (DS/DD). Varje sektor rymmer 256 bytes. Sektorerna numreras från 0 till antal-1. I verkligheten ligger sektorerna i olika spår på skivan. Vanligtvis innehåller en skiva 40 spår med nio sektorer i varje (SS/SD). Dubbelsidiga skivor har 80 spår (2*40), medan dubbel densitet ger 18 sektorer i varje spår.

SS = Single Sided
DS = Double Sided
SD = Single Density
DD = Double Density

De två första sektorerna (sektor 0 och 1) är speciella. De innehåller alltid en viss skivinformation (se nedan). Övriga sektorer innehåller antingen information OM en fil eller också data som finns INUTI en fil.

Sektor 0 - Disk information (Skivinformation)

Adress	Innehåll
0000-0009	Skivans namn. Högst 10 tecken.
000A-000B	Totalt antal sektorer på skivan. (>0168=360, >02D0=720, >05A0=1440)
000C	Antalet sektorer/spår. (9=SD, 18=DD)
000D-000F	'DSK' (>44534B)
0010	>20 (blank) eller >50 ('P'). P betyder skydd mot kopiering med Disk Manager.
0011	antalet spår per sida (normalt >28=40)
0012-0013	>0101=SS/SD, >0201=DS/SD, >0102=SS/DD, >0202=DS/DD
0038-	Bit-karta över använda sektorer.

I den här kartan representerar varje bit en sektor på disketten. En etta markerar en använd sektor, en nolla en ledig. Tag en byte i taget och läs den bakifrån för att få rätt värden.
Ett exempel: Antag att den första byten (vid >0038) innehåller >73. Skrivet binärt blir det 0111 0011. Vänd på det, och du får 1100 1110. Härav ser man att sektorerna 0, 1, 4, 5 och 6 är använda, medan 2, 3 och 7 är lediga.
Sedan fortsätter det på samma sätt. Hur många bitar som är intressanta beror på hur mycket disketten rymmer. Utrymmet är tillräckligt för 1600 sektorer, vilket är lite mer än de 1440 som en DS/DD diskett behöver.

Sektor 1 - Directory Link (Katalog pekare)

Här finns 16-bitars pekare till var på disketten som de olika filernas beskrivningar finns. Det finns plats för 128 pekare, men eftersom listans slut markeras med en pekare som är noll, blir det 127 stycken kvar till filerna. Därav begränsningen i antal filer på en diskett. Genom att läsa av filnamnen i den ordning som de här pekarna anger, får man namnen i alfabetisk ordning. (ASCII ordning är kanske bättre, eftersom även siffror och en del andra tecken får finnas i filnamn.) Varje gång en ny fil läggs till en diskett, sorteras alltså dess pekare in på rätt plats här.

Sektor 2 till 359 - 719 - 1439 beroende på diskett

Varje fil har ett huvud, som innehåller filens namn, typ, längd osv samt en svans, där den information som filen innehåller finns placerad. Pekarna i sektor ett talar om var filernas huvuden finns, medan respektive huvud pekar till de sektorer på disketten som härbärgerar själva filen. När datorn ska leta reda på en fil med ett visst namn, måste den via katalogpekarna läsa in sektorer med filhuvuden och kontrollera om det är rätt namn. Detta innebär att diskdrivens huvud - vilket är en mekanisk anordning och INTE det samma som ett filhuvud - måste placeras över rätt spår på disketten. För att undvika en massa flyttande av huvudet har TI valt att reservera sektorerna 2->21, dvs 2-33, för i första hand filhuvuden. När den första filen lagras på en diskett placeras dess huvud i sektor 2, och dess data i sektor >22.
Detta ger plats till 32 filnamn nära skivans sektor med diskinformation. Om fler än 32 filer lagras på en diskett, sprids filhuvudena ut över disketten i första tillgängliga sektor. Omvänt, om stora filer lagras, kommer deras sista sektorer att placeras i eventuellt oanvänt utrymme i de här sektorerna.

File headers (Filhuvuden)

Adress	Innehåll
0000-0009	Filens namn. Högst 10 tecken.
000C	Filens typ. >00=Display Fixed >01=Program >02=Internal Fixed >80=Display Variable >82=Internal Variable Det filskydd som kan skapas med Disk Manager markeras genom att 8 adderas till dessa tal.
000D	Minsta antal poster (records) per sektor. Om filen har fast postlängd (fixed) är detta just det antal poster som en sektor innehåller. Om längden är variabel, anges det antal poster av maxlängd som får plats.
000E-000F	Antal sektorer som filen upptar. Katalogprogram anger ytterligare en, eftersom även huvudet räknas.
0010	Offset för filslutet i sista sektorn. Gäller för filer av typen program och variable. Denna byte pekar på den första oanvända byte i den sista av de sektorer som filen upptar. I en variable fil finns alltid sektorsslutsmarkeringen, >FF, där.

0011 Postlängden. Är alltså 80 för en DIS/VAR 80 fil osv.
0012-0013 Antalet poster i filen, om den är av typ fixed. Värdet lagras i två bytes som är omkastade. >1234 lagras som >3412.
För variable filer anges samma värde som finns vid >000E->000F (se ovan).
001C Pekare till filens svans.

När en fil lagras på skivan utnyttjas det första lediga utrymme som finns att tillgå. Om detta uppstod då en liten fil raderades, kan det tänkas att det inte är stort nog för en ny fil som ska skrivas in. I så fall delar datorn den nya filen, och fortsätter lagringen i nästa lediga utrymme. Detta fortsätter tills hela filen lagrats. Eftersom filen då kan bestå av flera olika block, måste det finnas en pekare till varje block. Dessutom behövs en uppgift om hur många sektorer som ingår i blocket. Fenomenet kallas extern fragmentering, och ger sig främst till känna på disketter som ofta ändrar innehåll. Skivor som används till exempelvis ordbehandling har i regel ett synnerligen transient innehåll. Gamla filer raderas, nya tillkommer och befintliga ändrar storlek. Med tiden resulterar detta i att de flesta filerna blir utspridda på flera olika block på skivan. Därigenom blir åtkomsten av informationen långsammare, eftersom det behövs mer flyttande mellan olika spår. Genom att kopiera diskettens innehåll med Disk Managerns Backup Disk ändras filernas placering, varvid de åter kommer att hänga ihop. Dessutom går det fortare att hitta ett filnamn, eftersom även katalogen sorteras om i bokstavsordning.

Varje blockpekare är tre bytes lång. Den andra byten delas i två hexadecimala siffror. Placera den sista siffran före byte ett, och den första efter byte tre. Då får du två 12-bitars tal, där det första talar om vilken sektor som filblocket finns i, och den andra vilket nummer den sista av filens sektorer som finns i det här blocket har. Sektornummerna inom filen börjar med noll.
Exempel: >014325 ger >301 och >254, vilket innebär att blocket börjar i sektor >301 på disketten och fortsätter till och med sektor nummer >254 inom filen räknat. Det motsvarar alltså sektor >555 på disketten.

Exempel igen: En >26 sektorer lång fil kan ha blockpekarna >22E000145102. Av detta ser man att filen börjar i sektor >22, och att den sista sektorn i det blocket är >E. Sedan fortsätter filen i sektor >114, där den håller på till och med sektor >25 i filen. Sedan finns det inte fler sektorer, eftersom även nollan räknas.

DATAFORMAT.

Att kunna hitta filerna på en diskett är dock bara halva hemligheten. För att kunna ändra något på ett vettigt sätt måste man också veta hur filerna ser ut. Därför kastar vi oss nu raskt över till filformaten.

För det första allokeras utrymme till filer endast som hela sektorer. (Allokering betyder ungefär "tilldelning av utrymme".) En fil som upptar 257 bytes behöver alltså två sektorer (512 bytes), medan en på 256 bytes kan klara sig med en sektor (256 bytes). Det är lätt att inse att detta fenomen, kallat intern fragmentering, gör att vi (åtminstone i teorin) förlorar i genomsnitt 128 bytes per fil.

FILTYPER

Det finns tre huvudtyper av filer, nämligen FIXED, VARIABLE och PROGRAM. Dessutom kan innehållet i de två första typerna vara antingen DISPLAY eller INTERNAL.

Fixed filer har en fast postlängd. Alla posterna i filen har den längd som specificeras när filen skapas. Så många hela poster som möjligt lagras i varje sektor. Därför är det olämpligt att välja en postlängd på 129 bytes, eftersom det då bara får plats en post i varje sektor. 127 bytes slösas alltså bort vid varje post.

Variable filer har däremot en variabel postlängd, något som läsare med detektiva anlag troligen redan har genomskådat. Detta innebär att det bara finns en övre gräns för hur lång en post får vara. Däremot lagras inte fler bytes än vad som verkligen behövs. För att datorn ska kunna veta hur lång en viss post är, inleds den med en byte som talar om längden.
Även här lagras hela poster i en och samma sektor. Om det finns 26 lediga bytes när en post om 54 bytes ska lagras, skrivs en markering för sektorslut (>FF) i den första byten av de lediga 26. Posten lagras i början av nästa sektor.

Program filer innehåller en kopia av en viss del av datorns minne. De är inte uppdelade i poster, utan består av ett enda block. Detta sprids ut över så många sektorer som behövs, med 256 bytes i varje. Outnyttjat utrymme finns endast i den sista sektorn, i de fall då filens längd inte är jämt delbar med 256. Filkatalogen innehåller en pekare till den första byten som inte är använd i denna sektor. Det är lätt att förstå att denna filtyp är den utrymmesmässigt mest ekonomiska, eftersom intern fragmentering endast förekommer i den sista sektorn.

DATATYPER

Inom varje post i en fil kan data lagras på olika sätt. BASIC ger alternativen DISPLAY och INTERNAL.

Display innebär att posterna i filen ser ut precis som de skulle göra om motsvarande data skrivits ut på en skrivare. Text lagras som den står, tal omvandlas till ASCII-tecken. På disketter utnyttjas detta format huvudsakligen till textfiler för ordbehandlare. Dessutom används det normalt vid överföring av text till skrivare, plottrar och andra datorer, men det har ju inte med disketter att göra.

Internal betyder att data lagras på disketten på samma sätt som det lagras i en variabel i BASIC. Textsträngar föregås av en byte som talar om hur många tecken som finns i strängen. Tal lagras i det interna flyttalsformat som BASIC använder, samt med en längdindikator först. Tal i internt format är dock alltid åtta bytes långa.
Observera att den här längdindikatorn inte har något att göra med den som finns först i alla poster med variabel längd. I en fil av typen Internal/Variable börjar alla poster med två längdindikatorer, en som talar om hur lång posten är och en som talar om längden på det första elementet. Inget hindrar att en post innehåller flera element.

PROGRAM

För att ha någon glädje av det här måste du också ha tillgång till något program som låter dig ändra innehållet i sektorerna. Forth går att använda, eftersom Forth har en alldeles egen åsikt om hur man ska hantera information på diskett. I stället för att lagra program i filer, ser Forth hela disketten som en lång rad av sektorer. Vilken sektor som helst kan läsas, i godtycklig ordning. Egentligen är det fyra sektorer som läses, eftersom uppdelningen göres i block om 1024 bytes. Men det är ju inte svårt att räkna ut vilket block en viss sektor finns i.
Någon editor i Forth kan utnyttjas till att läsa och ändra innehållet med. Problemet är att editorn är tänkt till att redigera text med, varför det blir besvärligt när en byte inte innehåller en ASCII-kod för något snällt, skrivbart tecken.
DISK-PATCH heter ett program som Texas själva gjort. Det kan visa en sektor i taget på skärmen, antingen som text eller som hexadecimala koder. Med hjälp av programmets fullskärmseditor kan man ändra godtyckliga bytes och sedan skriva tillbaka alltihop till skivan igen.
DISK FIXER från Navarone har ingen fullskärmseditor, men kan å andra sidan exempelvis ge utskrift av en sektor och leta upp en viss text på en diskett.
Till p-systemet finns UTILITY:PATCH.CODE. Det programmet har både en skärmeditor och avancerade utskriftsmöjligheter.

Lagring på kassettband

av Anders Persson, efter inspiration från Lennart Lindberg

I anslutning till en annan artikel som behandlar lagring av data på diskett, tittade vi efter något som handlar om lagring på kassett. Lennart hittade en artikel i den belgiska TI Software Exchange Newsletter av TIC-AZUR CLUB - vad nu det är för något. Den som skrivit artikeln är uppenbarligen tekniskt intresserad, och säger ungefär så här.

De signaler som sänds till kassettbandspelaren är komponerade av en serie binärkodade tecken (nollor och ettor). När data - datafiler eller programfiler - lagras på bandet representeras dessa tecken av signaler som gör det möjligt för datorn att avgöra om varje enskild signal skall tolkas som 1 eller 0 när data läses tillbaka. Så är det för alla datorer. Det här åstadkommes genom att signalerna har en bestämd utsträckning i tiden (frekvens) och en bestämd styrka och polaritet (amplitud).

Om man mäter signalerna som 99-an använder - med t. ex. ett oscilloskop - finner man att

- en nolla representeras av en puls med en längd av c:a 0.74 millisekunder

- en etta representeras av två pulser om 0.37 millisekunder

Det är alltså fråga om en frekvensmodulerad signal. När jag undersökte hur datorns bandlagring fungerar (se Utmaningen 85-2), konstaterade jag bland annat att timern i PSI:n används för att hålla rätt frekvens. PSI (Programmable Systems Interface) eller TMS 9901 är en krets som sitter i 99:an och används till diverse I/O, såsom tangentbord och band.

Signalens informationsinnehåll är c:a 1400 bitar/sekund eller 175 bytes/sekund eller 10 Kbytes/minut.

Om vi studerar förloppet när ett program registreras (med "SAVE CS1") kan vi iakttaga följande.

Först hör vi ett högt pipande ljud under mer än fyra sekunder, vilket motsvarar sändning av 768 bytes med värdet 00 (hex). Det här pipet använder datorns kretsar vid inläsningen, för att avgöra signalens nivå. Därefter kommer en byte med värdet FF och två likadana med värdet N, där N är det antal 64-bytes grupper som utgör det meddelande som skall sändas. Det antalet beror förstås på storleken på det program som vi vill lagra. Efter denna inledning kommer N grupper om 74 bytes, där varje grupp sänds två gånger.

En 74-bytes grupp är uppbyggd av

- åtta bytes med värdet 00
- en byte med värdet FF
- en 64-bytes grupp med själva meddelandet
- en byte med en kontrollsumma

Kontrollsumman räknas ut under registreringen och placeras på bandet. Vid läsningen beräknas den igen och jämförs med den registrerade. Genom att allt lagras två gånger finns det en möjlighet att korrigera fel. Men om båda registreringarna läses med något fel får vi meddelandet "ERROR DETECTED IN DATA".

Den här proceduren tar tid och gör skrivningen långsam. Vi måste ju skriva 148 bytes för varje grupp om 64 bytes som vi vill lagra. Samma gäller vid läsningen. Den effektiva hastigheten blir 600 bitar/sekund eller 4.5 Kbytes/minut.

Här nedan följer ett program i assembler som ger en möjlighet att kopiera kassettband med program eller filer på. Det behövs förstås två kassettbandspelare, en som spelar in och en som spelar av. Sätt bara igång programmet. Med Mini Memory görs det med "EASY BUG" och E7D00. Programmet stannar när man trycker på = tangenten och kopieringen avbrytes.

99/4 ASSEMBLER

```
0000 04CC        CLR  R12   Bas adress för CRU
0002 1E12        SBZ  >12   Initiera tangentbordsavläsning
0004 1E13        SBZ  >13
0006 1E14        SBZ  >14
0008 1F1B  B0    TB   >1B   Läs kassett
000A 1304        JEQ  UN    Till UN om en etta
000C 1E19        SBZ  >19   Sänd en nolla
000E 1F03        TB   3     Titta efter nedtryckt =
0010 13FB        JEQ  B0    Nej, fortsätt
0012 045B        B    *R11  Återhopp
0014 1D19  UN    SBO  >19   Sänd en etta
0016 1F03        TB   3     Titta efter nedtryckt =
0018 13F7        JEQ  B0    Nej, fortsätt
001A 045B        B    *R11  Återhopp
                 END
```

Den i assembler övade läsaren inser nog att finessen med det här är att datorn läser och skriver samtidigt, varvid bandkopiering går dubbelt så fort" Priset för det är datasäkerhet, eller snarare brist på sådan. Det finns inte längre någon kontroll av att överföringen är felfri. Kolumnen som börjar med 04CC innehåller maskinkoden för programmet. Det vore en trevlig utmaning (Nu är vi där igen") för någon att med utnyttjande av den skriva ett program i BASIC eller X-BASIC, som lägger maskinkoden i data-satser och på så sätt får det här körbart med enklaste utrustning.

Den här beskrivningen lämnar väl en del övrigt att önska. Se även artikeln Närkontakt med kassettbandspelaren i Utmaningen 85-2, om du vill veta lite mer om hur man själv kan göra kassettrutiner. Dessutom finns det säkert läsare som har egna erfarenheter eller problem att dela med sig av. Gör det! Sitt inte där och ruva på Din bit! ■

Dessutom har det på senare tid dykt upp ett antal mer avancerade program. Dessa tillåter mer sofistikerade tricks, typ reparation av skadade filer utan att användaren behöver veta exakt vad han håller på med, även om det säkert underlättar. Viss inbyggd intelligens, alltså. Andra funktioner finns också i somliga program, såsom kontroll av drivenheternas varvtal och läsning av hela spår, inklusive ID och data poster, CRC osv. ID kommer före varje datasektor, och innehåller bland annat spår- och sektornummer. CRC betyder Cyclic Redundancy Check, vilket innebär en kontrollsumma som beräknas på lagrad information. Om summan inte stämmer när sektorn läses senare, vet datorn att något är galet på disketten.

De här programmen heter Advanced Diagnostics, Explorer, DISK+AID mm.

STJÄRN KIKERI

19VAV36

av Arne Wennberg, Landskrona

Programspråk: TI-BASIC

Detta är ett program för astronomiskt intresserade. Man kan med hjälp av det beräkna en stjärnas eller annan himlakropps höjd och bäring från vilken punkt på jorden som helst och vid vilken tid som helst. Man kan också lätt beräkna tiden för meridianpassage eller passage av sanna horisonten eller vertikalen.

En förutsättning är att man känner himlakroppens SHA (Sidereal Hour Angle) och deklination för den aktuella tidpunkten. Dessa värden kan hämtas från en astronomisk eller nautisk almanacka.

Aktuella värden för 37 st av de större stjärnorna finns dock redan inmatade i programmet. Dessa torde utan större fel kunna användas inom en 10-årsperiod.

För att få programmet enkelt har vissa approximeringar gjorts vid beräkningarna. Resultatet anges därför endast i hela grader. Det hindrar inte att slå in mera noggranna värden i form av decimaler av grader om så önskas.

PROGRAMBESKRIVNING:

100-290	Definitioner mm.
300	Går till subprogram rad 2260 för titelbild. Vidare från rader 2430, 2460 och 2490 till subprogram för musik.
310-1080	Programmet tar emot latitud, longitud, tidsmeridian, stjärnans namn (eller deklination och SHA), månad, timmar och minuter.
1090-1510	Här beräknas höjden. I rad 1120 går programmet gosub till 1940 där vårdagjämningspunktens timvinkel beräknas. I rad 1320 gosub till 1600 för beräkning av azimut. Den förvandlas till bäring.
1520-1590	Här finns en slutvinjett.
1600-1930	Underprogram för azimut.
1940-2250	Vårdagjämningspunktens timvinkel.
2260-2520	Titelbild.
2530-2820	Instruktioner.
2830-3110	Här prövas om data finns på namngiven stjärna. I datasatserna finns: stjärnans namn, SHA, deklinationens namn, deklination.
3120-3380	Musik.

Följande rader kan uteslutas och programmet fungerar ändå:
220-300, 1530-1590, 2260-2520 och 3120-3380.

När programmet köres visas först titelbild med lämplig musik. Därefter kommer instruktion. Programmet är självinstruerande men ett exempel kanske underlättar.

Halley's komet lyser som starkast i början av april. Den 8:e är SHA=115.9 och deklinationen S 47.1 grader.

Kan man se kometen i Sverige då?

Pröva från södra Sverige. Slå in N 55.3, O 13.2 (positionen för Trelleborg) och Sveriges tidsmeridian, som är O 15 grader.

Datorn frågar då efter stjärnans namn. Svara med Halley's komet. Nu undersöker datorn om denna stjärna finns i programmet. Om så är fallet visas SHA och deklination på skärmen, annars frågas efter dessa värden. Slå in värdena. Så frågas efter månad och datum. Slå in den 4:e månaden och den 8:e dagen.

Vidare frågas efter timmar och minuter. Slå på måfå in 6 tim och 0 min. Datorn svarar med att räkna fram bäring=209, höjd=-18 grader. Eftersom bäringen är större än 180 (syd) har meridianpassagen ägt rum. Tryck T och försök med kl 4. Datorn svarar med bäring =189, höjd=-13. Också detta var lite för sent. Kl 3 ger bäring=178, höjd=-12 och kl 3 tim och 10 min ger bäring=180 och höjd=-12. Bingo"

Kometen passerar alltså meridianen kl 0310 och är då 12 grader under horisonten. Vi kan inte se den. Gör gärna om beräkningen från en sydligare position"

På samma sätt kan också upp- och nedgång beräknas.

För att göra beräkningar för solen, månen, planeterna eller för stjärnor som inte finns i programmet måste man ha tillgång till en almanacka. I en timvinkelalmanacka finns i regel inte solens och månens SHA tabellerat. SHA kan då beräknas genom att för samma tidpunkt ta: Himlakroppens timvinkel minskat med vårdagjämningspunktens timvinkel.

Jag har en annan version av programmet, där man kan slå in två tidpunkter och få bäring och höjd med visst intervall beräknade och snyggt utskrivna på printer. Den räknar också noggrannare och ger höjden med två decimaler. Även den är i TI-BASIC.

Om någon är intresserad, skicka mig namn och adress och trettio kronor så kommer ett band på posten.

Min adress är: St Norregatan 91, 261 33 LANDSKRONA.

LYCKA TILL MED BERÄKNINGARNA" ➔➔

```
100 REM ****************
110 REM * STJÄRNKIKERI *
120 REM *ETT PROGRAM AV*
130 REM *ARNE WENNBERG *
140 REM ****************
150 DIM STJ$(40),SHA1(40),DEKL$(40),DEKL(40)
160 CALL CLEAR
170 CALL SCREEN(5)
180 CALL COLOR(9,5,5)
190 CALL CHAR(91,"00280038447C4444")
200 CALL CHAR(92,"0028007C4444447C")
210 CALL CHAR(93,"00382838447C4444")
220 CALL CHAR(94,"C3C7666637361E1E")
230 CALL CHAR(95,"C3E36666EC6C7878")
240 CALL CHAR(97,"000103070F1F3F7F")
250 CALL CHAR(98,"0080C0E0F0F8FCFE")
260 CALL CHAR(99,"FEFCF8F0E0C08000")
270 CALL CHAR(100,"7F3F1F0F07030100")
280 CALL CHAR(101,"FFFFFFFFFFFFFFFF")
290 CALL CHAR(104,"")
300 GOTO 2260
310 REM  *** INPUT ***
320 CALL CLEAR
330 CALL SCREEN(4)
340 PRINT "DIN LATITUD?": :
350 INPUT "NAMN N/S:     ":LAT$
360 PRINT
370 IF LAT$="N" THEN 390 ELSE 380
380 IF LAT$="S" THEN 390 ELSE 350
390 INPUT "GRADER:       ":LAT
400 PRINT
410 IF LAT>90 THEN 390
420 IF LAT$="S" THEN 430 ELSE 440
430 LAT=-LAT
440 PRINT
450 PRINT "DIN LONGITUD?"
460 PRINT
470 INPUT "NAMN O/V:     ":LON$
480 PRINT
490 INPUT "GRADER:       ":LON
500 IF LON>180 THEN 480
510 IF LON$="O" THEN 540 ELSE 520
520 IF LON$="V" THEN 530 ELSE 460
530 LON=-LON
540 PRINT : :
550 PRINT "TIDSMERIDIAN?"
560 PRINT
570 INPUT "NAMN O/V:     ":TIDMER$
580 PRINT
590 INPUT "GRADER:       ":TIDMER
600 IF TIDMER>180 THEN 580
610 IF TIDMER$="O" THEN 640 ELSE 620
620 IF TIDMER$="V" THEN 630 ELSE 560
630 TIDMER=-TIDMER
640 PRINT : :
650 DL=LON-TIDMER
660 DT=DL/15
670 CALL SCREEN(4)
680 INPUT "STJÄRNA:      ":S$
690 GOTO 2330
700 PRINT
710 INPUT "SHA:          ":SHA
720 IF SHA>360 THEN 730 ELSE 750
730 PRINT :"FÖR HÖGT VÄRDE""
740 GOTO 700
750 PRINT
760 INPUT "DEKL (N/S):   ":D$
770 IF D$="N" THEN 810 ELSE 780
780 IF D$="S" THEN 810 ELSE 790
790 PRINT :"N ELLER S""
800 GOTO 750
810 PRINT
820 INPUT "GRADER:       ":DK
830 IF DK>90 THEN 840 ELSE 860
840 PRINT :"INTE ÖVER 90""
850 GOTO 810
860 PRINT
870 INPUT "MÅNAD:        ":M
880 IF M>12 THEN 890 ELSE 910
890 PRINT :"SÅ MÅNGA MÅNADER FINNS INTE""
900 GOTO 860
910 PRINT
920 INPUT "DATUM:        ":DA
930 IF DA>31 THEN 940 ELSE 970
940 PRINT :"FÖR MÅNGA DAGAR""
950 GOTO 910
960 CALL SCREEN(4)
970 PRINT
980 INPUT "TIMMAR:       ":TI
990 IF TI>24 THEN 1000 ELSE 1020
1000 PRINT :"FÖR MÅNGA TIMMAR""
1010 GOTO 970
1020 TI=TI+DT
1030 PRINT
1040 INPUT "MINUTER:      ":MI
1050 IF MI>60 THEN 1060 ELSE 1080
1060 PRINT :"LÄR DEJ KLOCKAN FÖRST""
1070 GOTO 1030
1080 PRINT : :
1090 REM  *** BER HÖJD ***
1100 IF D$="S" THEN 1110 ELSE 1120
1110 DK=-DK
1120 GOSUB 1940
1130 ST=VT+SHA
1140 IF ST>360 THEN 1150 ELSE 1160
1150 ST=ST-360
1160 IF ST>180 THEN 1170 ELSE 1200
1170 ST=360-ST
1180 TV$="O"
1190 GOTO 1210
1200 TV$="V"
1210 TR=ST*(4*ATN(1))/180
1220 FIR=LAT*(4*ATN(1))/180
1230 DR=DK*(4*ATN(1))/180
1240 XX=SIN(FIR)*SIN(DR)+COS(FIR)*COS(DR)*COS(TR)
1250 DEF ARCSIN(X)=ATN(X/SQR(1-X*X))
1260 HR=ARCSIN(XX)
1270 HG=HR*(180/(4*ATN(1)))
1280 DELT=HG-INT(HG)
1290 HG=INT(HG)
1300 IF DELT>0.5 THEN 1310 ELSE 1320
1310 HG=HG+1
1320 GOSUB 1600
1330 PRINT "HÖJDEN ÄR:   ";HG;" GRADER"
1340 PRINT
1350 PRINT
1360 PRINT
1370 PRINT
1380 PRINT "TRYCK  I, P, S, T ELLER F"
1390 PRINT
1400 PRINT
1410 PRINT
1420 DO=0
1430 IF D$="S" THEN 1440 ELSE 1450
1440 DK=-DK
1450 CALL KEY(0,K,S)
1460 S=0
1470 IF K=83 THEN 670
1480 IF K=80 THEN 320
1490 IF K=73 THEN 2530
1500 IF K=84 THEN 960
1510 IF K=70 THEN 1520 ELSE 1450
1520 CALL CLEAR
1530 PRINT "          S L U T"
1540 PRINT : : : :
1550 PRINT "          19^_86"
1560 PRINT : : : : : : :
1570 FOR I=1 TO 1000
1580 NEXT I
1590 END
1600 REM  *** AZIMUT ***
1610 IF TR=0 THEN 1620 ELSE 1640
1620 AZ=180
1630 GOTO 1720
1640 IF TR=4*(ATN(1))THEN 1650 ELSE 1670
1650 AZ=0
1660 GOTO 1720
1670 AA=COS(FIR)*TAN(DR)/SIN(TR)-SIN(FIR)/TAN(TR)
1680 DEF ARCCOT(X)=(4*ATN(1))/2+ATN(-X)
1690 AZ=ARCCOT(AA)
1700 AZ=AZ*(180/(4*ATN(1)))
1710 DELTA=AZ-INT(AZ)
1720 AZ=INT(AZ)
1730 IF DELTA>0.5 THEN 1740 ELSE 1750
1740 AZ=AZ+1
1750 IF TV$="V" THEN 1760 ELSE 1780
1760 BG=360-AZ
1770 GOTO 1800
1780 IF TV$="O" THEN 1790 ELSE 1750
1790 BG=AZ
1800 PRINT
1810 IF LAT=90 THEN 1820 ELSE 1840
1820 BG=180
1830 GOTO 1860
1840 IF LAT=-90 THEN 1850 ELSE 1860
1850 BG=0
1860 PRINT
1870 PRINT
```

```
1880 CALL SCREEN(11)
1890 PRINT "STJÄRNA:      ";S$
1900 PRINT
1910 PRINT "BÄRINGEN ÄR: ";BG;" GRADER"
1920 PRINT
1930 RETURN
1940 REM  *** TIMV ARIES ***
1950 DIM MD(12)
1960 MD(1)=31
1970 MD(2)=28
1980 MD(3)=31
1990 MD(4)=30
2000 MD(5)=31
2010 MD(6)=30
2020 MD(7)=31
2030 MD(8)=31
2040 MD(9)=30
2050 MD(10)=31
2060 MD(11)=30
2070 MD(12)=31
2080 MVAL=M-1
2090 FOR I=1 TO MVAL
2100 DO=DO+MD(I)
2110 NEXT I
2120 DO=DO+DA
2130 DV=DO-79.5
2140 IF DV<0 THEN 2150 ELSE 2160
2150 DV=DV+365
2160 MST=(TI+MI/60)*15
2170 IF MST<180 THEN 2180 ELSE 2200
2180 MST=MST+180
2190 GOTO 2220
2200 IF MST>=180 THEN 2210 ELSE 2220
2210 MST=MST-180
2220 VT=MST+((360/365)*DV)
2230 IF VT>360 THEN 2240 ELSE 2250
2240 VT=VT-360
2250 RETURN
2260 REM  *** TITELBILD ***
2270 DATA hhhhhhhhhhhhhhhhhhhhhhhhhhhh,hhhhhhhhhhhhhhehhehhhhhhhhh,hhhhhhhhhhhhhhhhhhhhhhhhhhhh
2280 DATA aeebheeeeehhehaeebheeebhehhe,ebhehhhehhhhehehhehehehehe,debhhhhehhhhehehhehehheebe
2290 DATA hdebhhhehhhheheeeeheeechedee,ehdehhhehhehehehhehedebhehde,deechhhehhdechehhehehdehehhe
2300 DATA hhhhhhhhhhhhhhhhhhhhhhhhhhhh,hhhhhhhhhhhhhhhhhhhhhhhhhh,hhhhhhhhhhhhhhhhhhhhhhhhhhhh
2310 DATA ehaechhehhehaecheeeeheeebhhe,eaechhhehheaechehhhhehhehhe,eeehhhhehheeehhhehhhhehhehhe
2320 DATA eeebhhhehheeebhheeehheeechhe,ehdebhhehhehdebhehhhhedebhhe,ehhdehhehhehhdeheeeehehdehhe
2330 DATA hhhhhhhhhhhhhhhhhhhhhhhhhhhh,hhhhhhhhhhhhhhhhhhhhhhhhhh,hhhhhhhhhhhhhhhhhhhhhhhhhhhh
2340 DATA hhhhhhhhhhhh19^ 86hhhhhhhhhh,hhhhhhhhhhhhhhhhhhhhhhhhhh,hhhhhhhhhhhhhhhhhhhhhhhhhhhh
2350 CALL SCREEN(5)
2360 FOR I=1 TO 24
2370 READ BILD$
2380 PRINT BILD$
2390 NEXT I
2400 CALL SCREEN(12)
2410 FL=FL+1
2420 CALL COLOR(9,5,12)
2430 GOSUB 3170
2440 CALL SCREEN(2)
2450 CALL COLOR(9,16,2)
2460 GOSUB 3230
2470 CALL SCREEN(10)
2480 CALL COLOR(9,13,10)
2490 GOSUB 3290
2500 CALL KEY(0,K,S)
2510 IF S<>0 THEN 310
2520 IF FL<2 THEN 2400
2530 REM  *** INSTRUKTIONER ***
2540 CALL CLEAR
2550 PRINT ">I N S T R U K T I O N E R<"
2560 PRINT "**************************"
2570 PRINT : :
2580 PRINT :"SLÅ IN FÖLJANDE VÄRDEN:"
2590 PRINT :"LATITUD      N EL. S   0-90"
2600 PRINT "LONGITUD     O EL. V  0-180"
2610 PRINT "TIDSMERIDIAN O EL. V  0-180"
2620 PRINT :"STJÄRNANS NAMN"
2630 PRINT "SIDERIAL HOUR ANGLE   0-360"
2640 PRINT "DEKLINATION  N EL. S   0-90"
2650 PRINT :"MÅNAD                 1-12"
2660 PRINT "DATUM                  1-31"
2670 PRINT "TIMMAR                 0-24"
2680 PRINT "MINUTER                0-60"
2690 PRINT : :"TRYCK VALFRI TANGENT""
2700 CALL KEY(0,K,S)
2710 IF S=0 THEN 2700 ELSE 2720
2720 CALL CLEAR
2730 PRINT "NÄR BERÄKNING UTFÖRTS:"
2740 PRINT : :"TRYCK I FÖR INSTRUKTIONER."
2750 PRINT :"TRYCK P FÖR NY POSITION."
2760 PRINT :"TRYCK S FÖR NY STJÄRNA."
2770 PRINT :"TRYCK T FÖR NY TID."
2780 PRINT :"TRYCK F FÖR SLUT."
2790 PRINT : : : : : :"TRYCK VALFRI TANGENT""
2800 PRINT : : :
2810 CALL KEY(0,K,S)
2820 IF S=0 THEN 2810 ELSE 310
2830 REM  *** STJÄRNOR ***
2840 RESTORE 3040
2850 FOR I=1 TO 37
2860 READ SJ$,SH,DK$,DKL
2870 STJ$(I)=SJ$
2880 SHA1(I)=SH
2890 DEKL$(I)=DK$
2900 DEKL(I)=DKL
2910 NEXT I
2920 FOR I=1 TO 37
2930 IF S$=STJ$(I)THEN 2960
2940 NEXT I
2950 GOTO 700
2960 SHA=SHA1(I)
2970 DK=DEKL(I)
2980 D$=DEKL$(I)
2990 PRINT :"SHA:          ";SHA
3000 PRINT
3010 PRINT "DEKL (N/S):    ";D$
3020 PRINT :"GRADER:       ";DK
3030 GOTO 860
3040 DATA MARKAB,14.0,N,15.1,FORMALHAUT,15.8,S,29.7,DENEB,49.8,N,45.2,PEACOCK,53.9,S,56.6,ALTAIR,62.5,N,8.8
3050 DATA VEGA,80.9,N,38.8,KAUS AUSTRALIS,84.2,S,34.4,ANTARES,112.8,S,26.4,RIGIL KENT,140.3,S,60.8,ARCTURUS,146.3,N,19.3
3060 DATA RASALHAGUE,96.4,N,12.6,ALPHECCA,126.5,N,26.8,HADAR,149.3,S,60.3,MIMOSA,168.3,S,59.6
3070 DATA ALKAID,153.3,N,49.4,SPICA,158.9,S,11.1,ALIOTH,166.6,N,56.0,GACRUX,172.4,S,57.0,ACRUX,173.5,S,63
3080 DATA DUBHE,194.3,N,61.8,REGULUS,208.1,N,12,POLLUX,243.9,N,28.1,PROCYON,245.4,N,5.3,CASTOR,246.6,N,31.9
3090 DATA ADHARA,255.5,S,28.9,SIRIUS,258.9,S,16.7,ELNATH,278.6,N,28.6,POLARIS,325.5,N,89.2,CAPH,357.9,N,59.1
3100 DATA CANOPUS,264.1,S,52.7,BETELGEUSE,271.4,N,7.4,BELLATRIX,278.9,N,6.3,CAPELLA,281.1,N,46.0
3110 DATA RIGEL,281.5,S,8.2,ALDEBARAN,291.2,N,16.5,ACHERNAR,335.7,S,57.3,ALPHERATZ,358.1,N,29.0
3120 REM  *** MUSIK ***
3130 DATA 400,196,400,392,400,349,400,330,400,277,400,294,800,440
3140 DATA 400,247,400,494,400,440,400,392,400,370,400,392,800,523
3150 DATA 400,587,400,523,400,494,400,440,400,392,400,349,400,330,400,294,800,440,800,247,1200,392,400,30000
3160 DATA 400,587,400,523,400,494,400,440,400,392,400,349,400,330,400,294,800,440,800,247,1200,262,400,30000
3170 RESTORE 3130
3180 FOR I=1 TO 7
3190 READ D,F
3200 CALL SOUND(D,F,10)
3210 NEXT I
3220 RETURN
3230 RESTORE 3140
3240 FOR I=1 TO 7
3250 READ D,F
3260 CALL SOUND(D,F,10)
3270 NEXT I
3280 RETURN
3290 IF FL=1 THEN 3300 ELSE 3320
3300 RESTORE 3150
3310 GOTO 3340
3320 IF FL=2 THEN 3330 ELSE 3340
3330 RESTORE 3160
3340 FOR I=1 TO 12
3350 READ D,F
3360 CALL SOUND(D,F,10)
3370 NEXT I
3380 RETURN
```

FORTH spalten

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

En lustig detalj med FORTH (i varje fall med figFORTH) är, att när Du har startat FORTH men inte kör något speciellt program och kursorn blinkar i kommando-läge - så är det ordet QUIT som exekverar! Eller, närmare bestämt, det är KEY i EXPECT i QUERY i QUIT, som exekverar.

En annan märklig detalj är att det är ordet QUIT som skriver ut alla "ok" på skärmen! QUIT är ju annars det ord Du kan använda för att gå ut ur rutiner utan att utskriften "ok" dyker upp, eller hur?
Visste Du inte detta redan kan kanske en förklaring vara på sin plats? Du kan fortsätta att läsa här om Du tycker det. Du kan också titta i Starting Forth, kap 9 - t ex på sid 228.

Låt mig emellertid först redogöra för ett par termer och begrepp som kan vålla problem - inte minst genom svårigheter att hitta bra svenska ord vid översättningen från engelska.
Först gäller det de olika status FORTH kan befinna sig i:
- "execution" (svenska exekvering, körning)
- "compilation" (svenska kompilering)
Status markeras av variabeln STATE, som är=0 vid exekvering och=icke-noll (i TI FORTH=192) vid kompilering. STATE ändras av de ord i kärnan, som ändrar status. Vid status kompilering lagras nya ord i ordlistan. Vid status exekvering utförs ord.
Sedan vill jag också klara ut
- "parse" (svenska analysera, dela upp)
- "interpret" (svenska tyda, tolka, översätta)
När FORTH läser ord från indataströmmen krävs en signal som säger var indata skall hämtas. Den signalen finns i variabeln BLK. När BLK=0 hämtas data från tangentbordet. När BLK=icke-noll hämtas data från den FORTH-skärm vars nummer står i BLK.
Indataströmmen hanteras genom dels ord för "parse", dels ord för "interpret". Med "parse" förstås att indataströmmen delas upp i enskilda ord. Det görs genom att orden är skilda åt av en eller flera blanka, vilket utnyttjas vid "parse". Varje enskilt ord tolkas därefter - "interpret". Mera om detta nedan.
Så här ser QUIT ut:

```
: QUIT
    0 BLK !          Ö data skall tas fr
                     Ö tg-bordet!
    ÄCOMPILEÅ Ä      Ö sätt status exekvering
    BEGIN            Ö starta en obest. slinga
      RP!            Ö töm returstacken
      CR             Ö byt rad
      QUERY          Ö hämta upp t 80 tkn
                     Ö fr inbuffer
      INTERPRET      Ö analysera och
                     Ö tolka en in-rad
      STATE § NOT    Ö sant om exekvering
      IF ." ok"      Ö sänd ok om exekvering
                     Ö som kvitto på lyckad INTERPRET
      THEN
    AGAIN ;          Ö en "evig" slinga!
```

Det är QUIT som är FORTHs parser. Med det menar jag det program som med hjälp av tolken (interpretatorn) kommunicerar med användaren. Att beordra QUIT är att anropa parsern, logiskt nog.

Man kan även ha alternativa parser-program. Det kanske kan vara lämpligt i en del spel eller kanske i vissa nyttoprogram?
Så här t ex:

```
: PARSER1 BEGIN QUERY INTERPRET AGAIN ;
```

PARSER1 är den enklast tänkbara parsern. Den allvarligaste nackdelen med den är att den kan ge överfull returstack! Skärmtexten ser inte heller vidare snygg ut.
Har Du flera parser-program kan Du lämpligen anropa dem från den parser Du just befinner Dig i.
I ett kommande nummer av Programbiten/Nittinian hoppas jag kunna visa hur Du kan göra en intelligent parser med hjälp av en del ideer från AI-området (AI=artificiell intelligens). Den parsern är tänkt att användas till äventyrsspel (som Du då får göra själv förstås).

INTERPRET står för den yttre interpretatorn i FORTH. Den kallas också för "text interpreter". Det finns en inre interpretator - "address interpreter" - som hanterar exekveringen av de enskilda orden. Den berör jag inte vidare här.
INTERPRET är också byggt på en "evig" slinga. Det kan vara lite svårt att förstå hur programflödet någonsin kommer ut ur den slingan. Låt oss titta på det!

```
: INTERPRET
    BEGIN            Ö starta en obest. slinga
    -FIND            Ö sök i ordlistan
      IF             Ö hittat! yttergren sann
        STATE § <    Ö sann vid kompilering
        IF           Ö innergren sann
          CFA ,      Ö lägg en pekare t
                     Ö ordets kodfält
        ELSE         Ö innergren falsk
          CFA EXECUTE Ö exekvera ordet
        THEN         Ö avsluta innergren
          ?STACK     Ö stackkontroll?
      ELSE           Ö ej ord! yttergren falsk
        HERE NUMBER  Ö konv t ett tal
        DPL § 1+     Ö punkt i talet?
        IF           Ö ja! innergren sann
         ÄCOMPILEÅ   Ö om kompilering lägg
         DLITERAL    Ö D-talet i ordlistan
        ELSE         Ö ingen punkt innergren falsk
         DROP        Ö kasta del av talet
         ÄCOMPILEÅ   Ö om kompilering lägg
         LITERAL     Ö enkel-talet i ordl.
        THEN         Ö avsluta innergren
         ?STACK      Ö stackkontroll
      THEN           Ö avsluta yttergren
    AGAIN ;          Ö upprepa slingan till
                     Ö ett fel eller tills
                     Ö "null" exekveras
```

Det normala sättet att komma ur slingan är via EXECUTE.
I figFORTH finns det nämligen ett onämnbart ord, vars namn består av ascii-tecknet för 0, och som ser till att programmet lämnar INTERPRET! Ordet brukar kallas "null" eller X. "Null" är ju just den kod som alltid placeras sist i indataflödet av EXPECT för att markera slutet på indataraden.
Interpretatorn hittar då till sist detta ord, och när det exekverar lämnar programflödet den "eviga" slingan.
Så här definieras det "onämnbara" (här kallat X):

```
: X
   BLK §           Ö indata fr tg-bord?
    IF             Ö från skärm!
       1 BLK +!    Ö nästa block
       0 IN !      Ö börja från början
       BLK § B/SCR Ö är en hel skärm
       MOD 0=      Ö klar?
       IF ?EXEC    Ö kompil. ger fel!
          R> DROP  Ö utgång! (skärm)
       THEN        Ö
    ELSE           Ö från tg-bord!
       R> DROP     Ö utgång! (tg-bord)
    THEN
  ; IMMEDIATE      Ö exekveras alltid!

DECIMAL
128 ' X NFA 1+ C!  Ö rätt namn!
```

Förklaringen till att 128 (hex 80) skall lagras i namnfältet i stället för 0 gavs i förra FORTH-spalten (första och sista byte i namnfältet har en flagga ettställd för att TRAVERSE ska kunna söka sig igenom namnfältet).

Ett annat ord som kan vara lämpligt att ta upp i detta sammanhang, är --> . Detta är ju den rutin som länkar ihop skärmar vid laddning. Laddningen sköts just av interpretatorn.

```
: -->
      ?LOADING
      0 IN !
      B/SCR BLK § OVER MOD - BLK +!
    ; IMMEDIATE
```

Ordet testar att laddning pågår, nollställer IN och adderar 1 till BLK. Allt "strul" med B/SCR (B/SCR ger i TI FORTH talet 1 på stacken, vilket betyder att det går ett block per skärm) har med portabilitet att göra. Det här är en av finesserna med att programmera i FORTH! Det finns FORTH-system där varje skärm består av flera block.

GO FORTHWITH! ■

P:KOPIERA

av Sven-Erik Wind, Gävle

Jag läste Lennart Lindbergs intressanta artikel i Programbiten nr 84-3 sid 15 om strukturerade program. Rätt använt kan det ge överskådliga program, som det är lätt att göra tillägg i utan att råka ut för 'spagettisyndromet'. Efterföljande program visar hur jag uppfattat tekniken. Det är ingen renodlad JSP, men jag söker mig fram. Programmet kompletterar P:TEXTIN och P:TEXTUT med P:KOPIERA.

Det är tänkt för dig, som vill ha en egen kopia av det du skickar in till föreningen. Programmet fungerar både för text utan och med kommatering. SKICKA ALLTID IN ORGINALINSPELNINGEN för jag vet inte hur en ordbehandlare reagerar på kopian. P:TEXTUT skriver i alla fall ut den felfritt på skärmen.

```
100 REM P:KOPIERA
110 REM ++++++++++++++++++
120 REM + Sven-Erik Wind +
130 REM + Takaregatan 21 +
140 REM + 80238 Gefle    +
150 REM + tel 026-191733 +
160 REM ++++++++++++++++++
170 REM
180 REM Kopiering band till band av P:TEXTIN-filer.
190 REM *********** SEKVENS
200 CALL CLEAR
210 PRINT "Ladda bandet med P:TEXTIN-filerna till ban
    dspelare 1."
220 PRINT
230 PRINT "Ladda bandspelare 2 med bandet du skall ko
    piera till."
240 OPEN #1:"CS1",SEQUENTIAL,DISPLAY ,INPUT ,FIXED 12
    8
250 REM
260 OPEN #2:"CS2",SEQUENTIAL,DISPLAY ,OUTPUT,FIXED 12
    8
270 REM
280 CALL CLEAR
290 PRINT TAB(8);"P:KOPIERING"
300 REM ------------------
310 REM Matar in en textrad till datorn
320 GOSUB 450
330 REM
340 REM Skriver textrad till bandspelare 2
350 GOSUB 490
360 IF TEXT$="" THEN 390
370 GOTO 320
380 REM
390 CLOSE #1
400 CLOSE #2
410 END
420 REM
430 REM ------------------
440 REM Inmatning av textrad till datorn
450 INPUT #1:TEXT$
460 RETURN
470 REM
480 REM Skriver textrad till bandspelare 2
490 PRINT #2:""""&TEXT$&""""
500 RETURN
510 REM
520 REM ********************
```

■

TI 59

Roots of a polynomial - up to degree 6x

av Robert Prins

This program calculate all roots, real and complex, of up to a 62 nd degree polynomial in one variable with real coefficients, using the Lin - Barstow method.

The instructions for the program are:

	Procedure	Enter	Press	Display
1	Enter degree of polynomial	n	A	n
2	Enter coefficients of xün, repeat until all coefficients have been entered.	a1	R/S	n-1
		:	:	:
	(To correct a wrong entry: Enter the power of x	:	:	:
	and press B, then reenter the coefficient)	an+1		-1
3	Enter the allowable error	e	C	e
4	Optional: Enter initial estimates for u and v	u	D	u
		v	D	v
5	Calculate u and v		E	u
	During the calculations, sussesive values of delta u and delta v are pause- displayed to monitor convergence.		R/S	v
6	Calculate first two roots		R/S	x1 or Re(x12)
	If the display is flashing the roots are complex		X:T	x2 or Im(x12)
7	Reduce degree of polynomial by 2 by dividing P(x) by x2 + ux + v		R/S	new degree
8	Optional: Recall the coefficients of the reduced polynomial		B'	a1
	The next coefficients can be recalled by pressing		R/S	a2
	repeatedly R/S, until a flashing zero appears		R/S	an+1
			R/S	"0"
9	Repeat steps 4 - 8 until all roots have been found.			

Notes:

1: Monotonic convergence of u and v seldom occurs.
2: If P(x) has multiple sets of equal root, convergence will almost certainly not occur, although it might seem so.
3: If the reduced degree is 1, only one root exists. It will appear almost at once after pressing E.

If yoy would like to have an example I would suggest you take the one from the EE - library. Starting with initial estimates of 0 for u and v, the coefficients of the 3rd degree polynomial are: 1, -13.6, 40.19 and -63.86

```
000  76 LBL   040  05  05   080  61 GTO   120  05  05   160  03  03   200  50 I×I   240  43 RCL   280  54  )   320  22 INV
001  11  A    041  43 RCL   081  01  01   121  54  )    161  65  ×    201  32 X:T   241  06  06   281  53  (   321  67  EQ
002  42 STO   042  06  06   082  02  02   122  82 HIR   162  43 RCL   202  53  (    242  92 RTN   282  32 X:T   322  03  03
003  01  01   043  92 RTN   083  75  -    123  06  06   163  03  03   203  53  (    243  02  2    283  85  +   323  16  16
004  76 LBL   044  69 DP    084  82 HIR   124  87 IFF   164  75  -    204  43 RCL   244  22 INV   284  61 GTO   324  53  (
005  12  B    045  25  25   085  18  18   125  01  01   165  82 HIR   205  03  03   245  44 SUM   285  02  02   325  43 RCL
006  42 STO   046  69 DP    086  82 HIR   126  02  02   166  17  17   206  75  -    246  01  01   286  65  65   326  08  08
007  03  03   047  26  26   087  07  07   127  87  87   167  33 X²   207  82 HIR   247  22 INV   287  72 ST*   327  55  ÷
008  94 +/-   048  76 LBL   088  65  ×    128  53  (    168  65  ×    208  15  15   248  49 PRD   288  04  04   328  43 RCL
009  42 STO   049  15  E    089  43 RCL   129  97 DSZ   169  43 RCL   209  65  ×    249  05  05   289  97 DSZ   329  07  07
010  04  04   050  22 INV   090  06  06   130  03  03   170  06  06   210  43 RCL   250  53  (    290  03  03   330  54  )
011  43 RCL   051  86 STF   091  75  -    131  00  00   171  54  )    211  06  06   251  43 RCL   291  01  01   331  94 +/-
012  01  01   052  01  01   092  82 HIR   132  83  83   172  82 HIR   212  65  ×    252  05  05   292  02  02   332  69 DP
013  44 SUM   053  43 RCL   093  14  14   133  43 RCL   173  04  04   213  82 HIR   253  33 X²   293  43 RCL   333  31  31
014  04  04   054  01  01   094  82 HIR   134  05  05   174  67  EQ   214  17  17   254  75  -    294  01  01   334  92 RTN
015  06  6    055  42 STO   095  08  08   135  65  ×    175  00  00   215  54  )    255  43 RCL   295  92 RTN   335  61 GTO
016  44 SUM   056  03  03   096  65  ×    136  53  (    176  44  44   216  55  ÷    256  06  06   296  76 LBL   336  03  03
017  04  04   057  69 DP    097  43 RCL   137  29 CP   177  53  (    217  82 HIR   257  54  )    297  17 B'   337  16  16
018  43 RCL   058  23  23   098  05  05   138  65  ×    178  53  (    218  14  14   258  53  (    298  43 RCL   338  76 LBL
019  03  03   059  87 IFF   099  54  )    139  82 HIR   179  82 HIR   219  54  )    259  29 CP   299  01  01   339  10 E'
020  92 RTN   060  01  01   100  82 HIR   140  15  15   180  16  16   220  44 SUM   260  77  GE   300  42 STO   340  47 CMS
021  69 DP    061  00  00   101  04  04   141  85  +    181  49 PRD   221  06  06   261  02  02   301  03  03   341  29 CP
022  24  24   062  68  68   102  69 DP    142  82 HIR   182  03  03   222  66 PAU   262  75  75   302  69 DP    342  00  0
023  69 DP    063  32 X:T   103  24  24   143  16  16   183  65  ×    223  50 I×I   263  34 ΓX   303  23  23   343  92 RTN
024  33  33   064  01  1    104  53  (    144  54  )    184  82 HIR   224  82 HIR   264  32 X:T   304  06  6    344  00  0
025  72 ST*   065  77  GE   105  73 RC*   145  82 HIR   185  17  17   225  04  04   265  43 RCL   305  42 STO   345  00  0
026  04  04   066  03  03   106  04  04   146  06  06   186  75  -    226  43 RCL   266  05  05   306  04  04
027  61 GTO   067  20  20   107  75  -    147  82 HIR   187  82 HIR   227  02  02   267  44 SUM   307  69 DP
028  00  00   068  00  0    108  82 HIR   148  17  17   188  15  15   228  32 X:T   268  05  05   308  24  24
029  18  18   069  82 HIR   109  15  15   149  85  +    189  65  ×    229  77  GE   269  54  )    309  73 RC*
030  76 LBL   070  04  04   110  65  ×    150  42 STO   190  82 HIR   230  00  00   270  94 +/-   310  04  04
031  13  C    071  82 HIR   111  43 RCL   151  03  03   191  18  18   231  53  53   271  92 RTN   311  92 RTN
032  42 STO   072  05  05   112  06  06   152  82 HIR   192  54  )    232  82 HIR   272  61 GTO   312  97 DSZ
033  02  02   073  82 HIR   113  75  -    153  18  18   193  55  ÷    233  14  14   273  00  00   313  03  03
034  92 RTN   074  06  06   114  82 HIR   154  54  )    194  82 HIR   234  77  GE   274  51  51   314  03  03
035  76 LBL   075  82 HIR   115  16  16   155  53  (    195  14  14   235  00  00   275  34 ΓX   315  07  07
036  14  D    076  08  08   116  82 HIR   156  82 HIR   196  54  )    236  53  53   276  75  -    316  00  0
037  48 EXC   077  06  6    117  05  05   157  08  08   197  44 SUM   237  43 RCL   277  32 X:T   317  69 DP
038  06  06   078  42 STO   118  65  ×    158  22 INV   198  05  05   238  05  05   278  43 RCL   318  68  68
039  48 EXC   079  04  04   119  43 RCL   159  44 SUM   199  66 PAU   239  92 RTN   279  05  05   319  92 RTN
```

■

TI 59

HISTOGRAM

av Robert Prins

This program prints after input of max 19 values and an eventual level - line a lenght wise histogram in Fast-Graphics mode.

The instructions for this program are:

	Procedure	Enter	Press	Display
1	Read in the program			
2	Create the two neccesary hex codes: 10 OP 17 CLR GTO 016 PGM 19 SBR 045 P/R LRN INS SST, up to step 032 INS LRN RST 6 OP 17			
3	Enter the number of bars (max:19)	n	A	n
4	Optional- nothing is done with it yet-: Enter the print code for the bar descriptor for the first bar (it will be indicated on the histogram as bar A). Up to 5 caracters may be entered using the TI-59 printcode	printcode	X:T	0
5	Enter the numeric value for the first bar	x	R/S	1
6	Repeat steps 4 and 5 for all bars			
7	Enter the value of a level line, for example the average. If a level line is not required enter a zero.	level	R/S	program starts

Notes:

1: No printout of the input values and their descriptors is provided - I simply can not think of a way to provide a satisfactory printout.

2: No information is printed about the scale on on the left side.

The way of interpolation of this scale goes like this:
1: The program tries to make the longest bars lenght as close as possible to 100 units.
2: A 100 unit bar can only mean 5, 10, 15 etc % if all values were to be converted to percentages.
3: The scale lenght is always adjusted to the nearest multiple of 10 units that is equal or greater than the lenght of the longest bar.

Here is an example:
Suppose the largest input is 162.3487(A) and the sum of all input values is 941.856(B). This means that A is =.. 17.24% of B. From this value we see that that we need a 20% 100 unit scale. On this scale 17.24% means 86(.18504) units, so that the top of the scale is 90 units. On this scale 1 unit (=2 dots, because h25 is placed at step 016) equals 162.-3487/86(.18504) =.. 1.88

```
000 92 RTN    040 99 PRT    080 03 03     120 13 C      160 42 STO
001 76 LBL    041 47 CMS    081 97 DSZ    121 01 1      161 12 12
002 11 A      042 32 X:T    082 01 01     122 42 STO    162 22 INV
003 61 GTO    043 02 2      083 00 00     123 08 08     163 59 INT
004 00 00     044 02 2      084 60 60     124 04 4      164 69 OP
005 39 39     045 42 STO    085 92 RTN    125 12 B      165 10 10
006 76 LBL    046 02 02     086 42 STO    126 69 OP     166 54 )
007 12 B      047 42 STO    087 20 20     127 05 05     167 59 INT
008 61 GTO    048 04 04     088 01 1      128 00 0      168 53 (
009 00 00     049 48 EXC    089 42 STO    129 92 RTN    169 48 EXC
010 20 20     050 06 06     090 21 21     130 73 RC*    170 12 12
011 76 LBL    051 32 X:T    091 01 1      131 04 04     171 65 ×
012 13 C      052 42 STO    092 03 3      132 22 INV    172 01 1
013 25 CLR    053 01 01     093 02 2      133 77 GE     173 00 0
014 69 OP     054 42 STO    094 48 EXC    134 01 01     174 42 STO
015 05 05     055 05 05     095 14 14     135 39 39     175 13 13
016 74 SM*    056 42 STO    096 12 B      136 32 X:T    176 55 ÷
017 90 90     057 10 10     097 02 2      137 02 2      177 43 RCL
018 90 LST    058 92 RTN    098 01 1      138 44 SUM    178 12 12
019 61 GTO    059 44 SUM    099 09 9      139 04 04     179 85 +
020 29 CP     060 11 11     100 42 STO    140 97 DSZ    180 22 INV
021 11 A      061 63 EX*    101 14 14     141 03 03     181 59 INT
022 70 RAD    062 02 02     102 43 RCL    142 01 01     182 69 OP
023 28 LOG    063 69 OP     103 13 13     143 32 32     183 10 10
024 01 1      064 22 22     104 48 EXC    144 22 INV    184 54 )
025 34 ΓX     065 53 (      105 01 01     145 44 SUM    185 69 OP
026 33 X²     066 32 X:T    106 12 B      146 06 06     186 00 00
027 35 1/X    067 55 ÷      107 13 C      147 69 OP     187 49 PRD
028 22 INV    068 01 1      108 97 DSZ    148 25 25     188 13 13
029 58 FIX    069 00 0      109 01 01     149 53 (      189 53 (
030 86 STF    070 22 INV    110 01 01     150 32 X:T    190 73 RC*
031 71 71     071 28 LOG    111 08 08     151 55 ÷      191 06 06
032 35 1/X    072 54 )      112 03 3      152 43 RCL    192 55 ÷
033 97 DSZ    073 72 ST*    113 01 1      153 11 11     193 43 RCL
034 01 01     074 02 02     114 44 SUM    154 65 ×      194 11 11
035 99 PRT    075 69 OP     115 14 14     155 02 2      195 65 ×
036 74 SM*    076 22 22     116 12 B      156 00 0      196 69 OP
037 72 72     077 69 OP     117 13 C      157 85 +      197 26 26
038 44 SUM    078 23 23     118 69 OP     158 56 DEL    198 02 2
039 00 00     079 43 RCL    119 00 00     159 56 DEL    199 00 0
```

```
200 00 0      240 59 INT    280 03 03     320 04 4      360 22 INV
201 00 0      241 65 ×      281 17 17     321 01 1      361 58 FIX
202 55 ÷      242 04 4      282 87 IFF    322 54 )      362 92 RTN
203 43 RCL    243 06 6      283 00 00     323 59 INT    363 43 RCL
204 12 12     244 85 +      284 03 03     324 44 SUM    364 21 21
205 85 +      245 02 2      285 20 20     325 07 07     365 22 INV
206 93 .      246 04 4      286 87 IFF    326 97 DSZ    366 67 EQ
207 05 5      247 54 )      287 01 01     327 04 04     367 03 03
208 54 )      248 42 STO    288 03 03     328 03 03     368 26 26
209 59 INT    249 07 07     289 14 14     329 39 39     369 06 6
210 74 SM*    250 01 1      290 43 RCL    330 05 5      370 04 4
211 06 06     251 42 STO    291 08 08     331 48 EXC    371 61 GTO
212 69 OP     252 02 02     292 85 +      332 04 04     372 03 03
213 26 26     253 43 RCL    293 53 (      333 48 EXC    373 24 24
214 97 DSZ    254 10 10     294 53 (      334 07 07     374 00 0
215 05 05     255 42 STO    295 69 OP     335 84 OP*
216 01 01     256 03 03     296 28 28     336 02 02
217 89 89     257 04 4      297 85 +      337 69 OP
218 92 RTN    258 42 STO    298 04 4      338 22 22
219 97 DSZ    259 04 04     299 02 2      339 02 2
220 05 05     260 02 2      300 54 )      340 44 SUM
221 02 02     261 03 3      301 55 ÷      341 06 06
222 32 32     262 42 STO    302 07 7      342 97 DSZ
223 01 1      263 06 06     303 93 .      343 03 03
224 00 0      264 43 RCL    304 08 8      344 02 02
225 42 STO    265 01 01     305 05 5      345 67 67
226 05 05     266 32 X:T    306 54 )      346 49 PRD
227 03 3      267 01 1      307 59 INT    347 04 04
228 07 7      268 00 0      308 65 ×      348 29 CP
229 61 GTO    269 00 0      309 02 2      349 00 0
230 02 02     270 49 PRD    310 93 .      350 48 EXC
231 48 48     271 07 07     311 05 5      351 07 07
232 53 (      272 73 RC*    312 85 +      352 67 EQ
233 53 (      273 06 06     313 93 .      353 03 03
234 43 RCL    274 22 INV    314 03 3      354 62 62
235 05 05     275 77 GE     315 07 7      355 58 FIX
236 55 ÷      276 03 03     316 93 .      356 40 IND
237 02 2      277 63 63     317 03 3      357 04 04
238 54 )      278 53 (      318 02 2      358 84 OP*
239 22 INV    279 67 EQ     319 93 .      359 02 02
```

■

Sätt musik till dina Extended Basicprogram

av Lars-Erik Svahn

I ett amerikanskt program har jag hittat en mycket enkel metod att programmera njutbar musik i Basic. Jag TROR att en viss Stephen Foster ligger bakom det ursprungliga programmet. Härmed är han nämnd.
Jag har bearbetat metoden en smula och gjort den ännu enklare att använda för alla som har Extended Basic. Det blev faktiskt ett utbyggbart litet operativsystem för musik.

Man börjar med att döpa ett tillräckligt antal variabler för frekvenser. Ge dem gärna namn efter de toner de representerar (ex: LET A0=110 etc).
Jag provade också att i stället för variabler använda konstanter, dvs DEFs (ex:DEF A0=110 etc). Detta blev dock avsevärt långsammare!

```
100 GOSUB 340 !INITIALIZE
110 !
120 FOR I=1 TO 2
130 CALL TRIO(2,A2,E2,C2)
140 CALL TRIO(2,B2,E2,G1)
150 CALL TRIO(2,C3,A1,E2)
160 CALL TRIO(2,D3,A2,F2)
170 CALL TRIO(4,E3,A2,C2)
180 CALL TRIO(4,C3,E2,A1)
190 CALL TRIO(4,B2,E2,D2)
200 CALL TRIO(4,A2,E2,C2)
210 CALL TRIO(2,B2,F§2,D1)
220 CALL TRIO(2,F§2,D1,B1)
230 CALL TRIO(2,C3,F§2,D1)
240 CALL TRIO(2,B2,G§2,D2)
250 CALL TRIO(4,A2,C2,A0)
260 CALL TRIO(4,E2,E1,E1)
270 CALL TRIO(4,C2,C1,C1)
280 CALL TRIO(4,A1,A0,A0)
290 NEXT I
300 CALL QUIET
310 STOP
320 !
330 !INITIALIZE
340 A0=110 :: A§0=117
350 B0=123
360 C1=131 :: C§1=139
370 D1=147 :: D§1=156
380 E1=165
390 F1=175 :: F§1=185
400 G1=196 :: G§1=208
410 A1=220 :: A§1=233
420 B1=247
430 C2=262 :: C§2=277
440 D2=294 :: D§2=311
450 E2=330
460 F2=349 :: F§2=370
470 G2=392 :: G§2=415
480 A2=440 :: A§2=466
490 B2=494
500 C3=523 :: C§3=554
510 D3=587 :: D§3=622
520 E3=659
530 F3=698 :: F§3=740
540 G3=784 :: G§3=831
550 A3=880 :: A§3=932
560 B3=988
570 PSE=20000
580 RETURN
590 !
600 SUB TRIO(T,P,H,C)
610 FOR A=0 TO 28 STEP 0.9*T :: CALL SOUND(-
    500,P,A,H,20,C,16):: NEXT A
620 SUBEND
630 !
640 SUB QUIET
650 CALL SOUND(1,110,29,110,29,110,29)
660 SUBEND
670 !
680 END
```

Ett SUB-program definieras för att generera tre toner med en viss längd. I själva musikprogrammet anropas detta subprogram med parametrarna (notvärde + tre tonvariabler).
Ex:
CALL TRIO(4,C2,G3,F1) ger ett ackord av kvartsnoterna C2, G3 och F1.

Den första tonen ljuder med avtonande klang (P som i piano). Den andra tonen ljuder med svag volym (H som i dragspel). Den tredje ljuder också med konstant, men något starkare volym (C som i klarinett).

Det finns en variabel PSE=20000, som simulerar en paus för någon eller några av tongeneratorerna.
Ex:
CALL TRIO(16,B2,F§2,PSE) ger två sextondelsnoter och en sextondelspaus. Tecknet § i F§2 står för förhöjd ton (F§2 är alltså tonen Fiss-2).

För att kunna tysta alla tongeneratorerna finns subprogrammet QUIET.

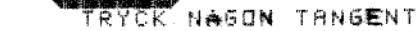
Detta är allt, men systemet är som sagt utbyggbart och avsett att användas till alla tänkbara program Du själv vill göra!

Musiken i första programmet (rad 120 till 310) härstammar troligtvis från ovan nämnda Stephen Foster.
Musiken i det andra programmet, skall föreställa inledningen till 'Iste confessor' av Palestrina.
Det är två vackra och stillsamma stycken. Men systemet fungerar bra även för musik med betydligt högre tempo!

```
100 CALL CLEAR
110 PRINT TAB(6);"Inledningen till":TAB(8);"
    Gloriasatsen": :TAB(6);"'Iste confessor'
    ": :TAB(7);"av  Palestrina": : : : : : :
120 GOSUB 400 !INITIALIZE
130 !
140 CALL TRIO(3/2,D2,G2,B1)
150 CALL TRIO(2,D2,A2,A1)
160 CALL TRIO(2,E2,G2,B1)
170 CALL TRIO(4,E2,G2,C2)
180 CALL TRIO(4,E2,E2,C2)
190 CALL TRIO(2,D2,F§2,A1)
200 CALL TRIO(2,C2,G2,G1)
210 CALL TRIO(2,D2,G2,A1)
220 CALL TRIO(2,D2,F§2,A1)
230 CALL TRIO(2,B1,G2,G1)
240 CALL TRIO(2,D2,G2,G1)
250 CALL TRIO(2,D2,F1,A1)
260 CALL TRIO(2,C2,F1,A1)
270 CALL TRIO(2,D2,F1,A1)
280 CALL TRIO(2,E2,E1,G1)
290 CALL TRIO(2,F2,D1,A1)
300 CALL TRIO(2,F2,D1,B1)
310 CALL TRIO(2,E2,C1,C2)
320 CALL TRIO(2,E2,C3,C2)
330 CALL TRIO(2,F2,A2,C2)
340 CALL TRIO(2,E2,C3,C2)
350 CALL TRIO(1,D2,B2,G1)
360 CALL QUIET
370 STOP
380 !
390 !INITIALIZE
400 A0=110 :: A§0=117
410 B0=123
420 C1=131 :: C§1=139
430 D1=147 :: D§1=156
440 E1=165
450 F1=175 :: F§1=185
460 G1=196 :: G§1=208
470 A1=220 :: A§1=233
480 B1=247
490 C2=262 :: C§2=277
500 D2=294 :: D§2=311
510 E2=330
520 F2=349 :: F§2=370
530 G2=392 :: G§2=415
540 A2=440 :: A§2=466
550 B2=494
560 C3=523 :: C§3=554
570 D3=587 :: D§3=622
580 E3=659
590 F3=698 :: F§3=740
600 G3=784 :: G§3=831
610 A3=880 :: A§3=932
620 B3=988
630 PSE=20000
640 RETURN
650 !
660 SUB TRIO(T,P,H,C)
670 FOR A=0 TO 28 STEP 0.9*T :: CALL SOUND(-
    500,P,A,H,20,C,16):: NEXT A
680 SUBEND
690 !
700 SUB QUIET
710 CALL SOUND(1,110,29,110,29,110,29)
720 SUBEND
730 !
740 END
```

■

HELI-KOPTER



av Martin Florin och Fredrik Nilsson

När titelbilden ritas upp och melodin startar trycker du på valfri tangent. Då skrivs det ut hur mycket olika saker är värda.
Du måste skjuta en "full" låda(F) för att inte få bensinbrist. Att bensinen håller på att ta slut märker man när det börjar pipa korta pip.(Det gör det redan från början, så detta är det första du måste göra./RED).
Spelet går ut på att ta sig igenom luftvärnseld och skjuta så mycket som möjligt. Raketer är farliga för de lyfter när man passerar dem, och följer efter en.
Kannoner skjuter uppåt slumpmässigt.
Du har klarat spelet när du passerat 15 skärmar. På 6-10:e skärmen slumpas det ut ballonger som går att skjuta. På 11-15:e skärmen slumpas det ut stenar som inte går att skjuta.

```
 10 REM ******************
 20 REM *   HELIKOPTER   *
 30 REM *                *
 40 REM *       AV       *
 50 REM * MARTIN FLORIN  *
 60 REM *       &        *
 70 REM *FREDRIK NILSSON *
 80 REM *      1985      *
 90 REM ******************
100 RANDOMIZE
110 CALL CLEAR
120 CALL SCREEN(6)
130 CALL CHAR(93,"00100038447C4444")
140 CALL CHAR(91,"00280038447C4444")
150 CALL CHAR(126,"0028007C4444447C")
160 CALL CHAR(96,"7F080FC9F91F143E")
170 CALL CHAR(41,"0103070F1F3F7FFF")
180 CALL CHAR(42,"FFFFFFFFFFFFFFFF")
190 CALL CHAR(43,"80C0E0F0F8FCFEFF")
200 CALL CHAR(104,"FFC1C1CFC7CFCFFF")
210 CALL CHAR(105,"03070E1C78F03020")
220 CALL CHAR(106,"1010101010101010")
230 CALL CHAR(107,"000000FF00000000")
240 CALL CHAR(108,"00183C7E7E7E99C3")
250 CALL CHAR(109,"00101038387CFE82")
260 CALL CHAR(110,"183C7E3C18001800")
270 CALL CHAR(112,"3C4299A5A599423C")
280 CALL CHAR(113,"003C425A5A423C00")
290 CALL CHAR(111,"00187EFFFF7E42E7")
300 CALL CHAR(44,"FF7F3F1F0F070301")
310 CALL CHAR(45,"FFFEFCF8F0E0C080")
320 CALL CHAR(97,"183C7E5A7E7E5A5E")
330 CALL CHAR(98,"000002FF93FFA5FD")
340 CALL CHAR(99,"0008483830EE0E0E")
350 CALL COLOR(10,11,6)
360 CALL COLOR(11,9,11)
370 CALL COLOR(2,13,6)
380 PRINT ")   + )**** +     +"
390 PRINT "*   * *-    *     *"
400 PRINT "*   * **    *     *"
410 PRINT "***** ****  *     * )**-"
420 PRINT "*   * *-    *     *"
430 PRINT "*   * **    *     *"
440 PRINT ",   - ,***- ,***- ,"
450 PRINT "   )  )- )**** )****+"
460 PRINT "   * )-  *- ,* *- ,*"
470 PRINT "   *)-   *   * *  )*"
480 PRINT "   **    *   * ****- )**-"
490 PRINT "   *,+   *   * *"
500 PRINT "   * ,+  ** )* *"
510 PRINT "   *  ,+ ,***- -"
520 PRINT ",*********-)**** )**** )****+"
530 PRINT "     *       *   *-    *- ,*"
540 PRINT "     ****+   *   **    *   *"
550 PRINT "**  )- ,*    *   ****  ****-"
560 PRINT ",**** )*     *   *-    * ,+"
570 PRINT "    ,***-    *   **    *  ,+"
580 PRINT "  + )-,+ )   *   ,***- ,   ,"
590 PRINT "  ,******-"
600 PRINT "    TRYCK NÅGON TANGENT"
610 GOSUB 3490
620 CALL CLEAR
630 PRINT " h = 0P    n = 5P   a = 30P "
640 PRINT " i = 100P  o = 10P  l = 50P"
650 PRINT " m = 100P  b = 25P  c = 100P"
660 PRINT "             UPP=A"
670 PRINT "                `kkkSKOTT=L"
680 PRINT "              NER=Z"
690 PRINT "            )***"
700 PRINT "           )-  ,+"
710 PRINT "         )- AV ,+"
720 PRINT "    )****-      ,****+"
730 PRINT "    * MARTIN&FREDRIK *"
740 PRINT "    *                *"
750 PRINT "    * FLORIN&NILSSON *"
760 PRINT "    *                *"
770 PRINT "    *      1985      *"
780 PRINT "    ,****************-"
790 PRINT "OM UNDER SPEL ETT PIPANDE":"LJUD UPPSTÅR S
    Å BETYDER":"DET ATT BENSINEN ÄR SLUT."
800 PRINT "FÖR ATT EJ DÖ SKJUT EN: h."
810 PRINT "    TRYCK NÅGON TANGENT": :"     TRYCK NER
     ALPHA LOCK"
820 RESTORE 3400
830 GOSUB 3490
840 CALL CLEAR
850 PRINT "      SVÅRIGHETSGRAD?": :"      1 SVÅRT 9
    LÄTT.": :
860 CALL KEY(0,K,S)
870 IF (K<49)+(K>57)THEN 860
874 SVA=K-48
875 P0=0
880 A=16
890 GU=4
900 B=2
910 C=1
920 D=1
930 BE=15
935 SK=0
936 CALL COLOR(2,13,6)
940 SK=SK+1
950 IF SK>5 THEN 960 ELSE 970
960 SDE=1
970 IF SK>10 THEN 980 ELSE 990
980 SDE=2
990 KA=0
1000 B=2
1010 CALL CLEAR
1020 ON SK GOSUB 2380,2530,2740,2960,3180,3790,3790,37
     90,3790,3790,3890,3890,3890,3890,3890,3990
1030 FOR T=1 TO 7
1040 PRINT "****************************"
1050 NEXT T
1060 CALL HCHAR(C,D,32)
```

```
1070 CALL HCHAR(A,B,96)
1080 C=A
1090 D=B
1100 BE=BE-1
1110 IF BE<11 THEN 1120 ELSE 1130
1120 CALL SOUND(-50,500,3)
1130 IF BE<0 THEN 2080
1140 CALL KEY(0,K,S)
1150 IF S=0 THEN 1220
1160 IF K=65 THEN 1190
1170 IF K=90 THEN 1210
1180 IF K=76 THEN 1510 ELSE 1220
1190 A=A-1
1200 GOTO 1220
1210 A=A+1
1220 B=B+1
1230 IF B>30 THEN 1240 ELSE 1280
1240 CALL CLEAR
1250 C=1
1260 D=1
1270 GOTO 940
1280 IF A<2 THEN 1290 ELSE 1300
1290 A=2
1300 CALL GCHAR(A,B,X)
1310 IF X<>32 THEN 2080
1320 IF D>F THEN 1330 ELSE 1440
1330 E=E-1
1340 F=F+1
1350 IF E<1 THEN 1360 ELSE 1390
1360 CALL HCHAR(E+1,F-1,32)
1370 F=32
1380 GOTO 1440
1390 CALL HCHAR(E+1,F-1,32)
1400 CALL HCHAR(E,F,105)
1410 IF E=C THEN 1420 ELSE 1440
1420 CALL HCHAR(E,F,105)
1430 GOTO 2080
1440 IF KA=0 THEN 1450 ELSE 1060
1450 IF INT(RND*SVA)+1=1 THEN 1460 ELSE 1060
1460 IF D=H THEN 1470 ELSE 1480
1470 TR=1
1480 CALL VCHAR(G,H,106,I)
1490 CALL VCHAR(G,H,32,I)
1500 IF TR=1 THEN 2080 ELSE 1060
1510 CALL GCHAR(C,D+1,X)
1520 CALL SOUND(-200,-6,3)
1530 IF (X=42)+(X=41)+(X=43)+(X=44)+(X=45)THEN 1220
1540 IF X=32 THEN 1570 ELSE 1550
1550 SH=1
1560 GOTO 1760
1570 CALL GCHAR(C,D+2,X)
1580 IF (X=42)+(X=41)+(X=43)+(X=44)+(X=45)THEN 1590 EL
     SE 1620
1590 CALL HCHAR(C,D+1,107)
1600 CALL HCHAR(C,D+1,107)
1610 GOTO 1220
1620 IF X=32 THEN 1650 ELSE 1630
1630 SH=2
1640 GOTO 1760
1650 CALL GCHAR(C,D+3,X)
1660 IF (X=42)+(X=41)+(X=43)+(X=44)+(X=45)THEN 1670 EL
     SE 1700
1670 CALL HCHAR(C,D+1,107,2)
1680 CALL HCHAR(C,D+1,32,2)
1690 GOTO 1220
1700 IF X=32 THEN 1730 ELSE 1710
1710 SH=3
1720 GOTO 1760
1730 CALL HCHAR(C,D+1,107,3)
1740 CALL HCHAR(C,D+1,32,3)
1750 GOTO 1220
1760 CALL HCHAR(C,D+1,107,SH)
1770 CALL SOUND(-100,-6,0,110,5)
1780 CALL HCHAR(C,D+1,32,SH)
1790 IF X=105 THEN 1800 ELSE 1830
1800 F=32
1810 PO=PO+100
1820 GOTO 1220
1830 IF X=109 THEN 1840 ELSE 1870
1840 KA=1
1850 PO=PO+100
1860 GOTO 1220
1870 IF X=104 THEN 1880 ELSE 1900
1880 BE=BE+40
1890 GOTO 1220
1900 IF X=108 THEN 1910 ELSE 1930
1910 PO=PO+50
1920 GOTO 1220
1930 IF X=99 THEN 1940 ELSE 1960
1940 PO=PO+100
1950 GOTO 1220
1960 IF X=110 THEN 1970 ELSE 1990
1970 PO=PO+5
1980 GOTO 1220
1990 IF X=98 THEN 2000 ELSE 2020
2000 PO=PO+25
2010 GOTO 1220
2020 IF X=97 THEN 2030 ELSE 2050
2030 PO=PO+30
2040 GOTO 1220
2050 IF X=111 THEN 2060 ELSE 2070
2060 PO=PO+10
2070 GOTO 1220
2080 FOR T=1 TO 10
2090 CALL HCHAR(C,D,112)
2100 CALL SOUND(-300,-6,0)
2110 CALL HCHAR(C,D,113)
2120 CALL SOUND(-300,-6,6)
2130 NEXT T
2140 C=1
2150 D=1
2160 TR=0
2170 IF SDE=0 THEN 2200
2180 IF SDE=1 THEN 2220
2190 IF SDE=2 THEN 2240 ELSE 2200
2200 SK=0
2210 GOTO 2250
2220 SK=5
2230 GOTO 2250
2240 SK=9
2250 A=16
2260 B=2
2270 GU=GU-1
2280 IF GU=0 THEN 2290 ELSE 930
2290 CALL CLEAR
2300 PRINT "DU FICK ";PO;"POÄNG"
2310 INPUT "VILL SPELA EN GÅNG TILL? J/N":SV$
2320 IF (SV$="J")+(SV$="JA")THEN 840
2330 IF (SV$="N")+(SV$="NEJ")THEN 2340 ELSE 2360
2340 CALL CLEAR
2350 END
2360 PRINT "SLUTA MUMMLA"!"
2370 GOTO 2310
2380 PRINT TAB(20);"m"
2390 PRINT TAB(19);")++"
2400 PRINT TAB(16);"i )****"
2410 PRINT TAB(15);")*******+   l"
2420 PRINT TAB(13);"h)**********+)***"
2430 PRINT TAB(7);")**********************"
2440 PRINT TAB(6);")**********************"
2450 PRINT TAB(5);")************************"
2460 PRINT TAB(4);")************************"
2470 E=10
2480 F=18
2490 G=1
2500 H=22
2510 I=7
2520 RETURN
2530 PRINT "                  )*******-"
2540 PRINT "                 )******-"
2550 PRINT "                 *******"
2560 PRINT "                ,*****-"
2570 PRINT "               b  ,***"
2580 PRINT "              )++"
2590 PRINT "         o     )***       c"
2600 PRINT "             )*****     )++"
2610 PRINT "            )*******  m)***"
2620 PRINT "            ***************"
2630 PRINT "+          )***************"
2640 PRINT "++       a)*****************"
2650 PRINT "***      )******************"
2660 PRINT "**** i h)*******************"
2670 PRINT "*****************************"
2680 E=15
2690 F=8
2700 G=7
2710 H=26
2720 I=3
2730 RETURN
2740 PRINT ",***********-  )*****-,****"
2750 PRINT " ,**********-  )*****-   ,**"
2760 PRINT "        ,**-  )****-      ,*"
2770 PRINT "         ,-  )***-         *"
2780 PRINT "            )***-       o  *"
2790 PRINT "            ***-           *"
2800 PRINT "            ,*-  c         *"
2810 PRINT "             o  )**        c"
2820 PRINT "             h)****       c*"
```

```
2830 PRINT "+b         *********       ***"
2840 PRINT "**+  i l  ,********       ***"
2850 PRINT "*********+   ,*****        ***"
2860 PRINT "**********+   ,***-        ***"
2870 PRINT "***********+   **-   m b )***"
2880 PRINT "************+        )********"
2890 PRINT "*************+  h a)*********"
2900 E=11
2910 F=8
2920 G=4
2930 H=23
2940 I=10
2950 RETURN
2960 PRINT "********-     ,*-  ,****- h,*"
2970 PRINT "********-      c     ,**-  oo*"
2980 PRINT "******-   lla*       ,-   ll*"
2990 PRINT "*****-    ****             ***"
3000 PRINT "**-       ,***             ***"
3010 PRINT "*-      h              c    ***"
3020 PRINT "-    o )*+ b         )*+   ***"
3030 PRINT "    o )******+      )***"
3040 PRINT "+    )********     ****  )***"
3050 PRINT "*+   ,********     ***- )****"
3060 PRINT "**+    h******-    **-  *****"
3070 PRINT "****+   ,****-     ,-   *****"
3080 PRINT "*****+                  *****"
3090 PRINT "******+          a     hh*****"
3100 PRINT "*******+     h m*+   hhh*****"
3110 PRINT "******** ic)******+hhhhh*****"
3120 E=16
3130 F=12
3140 G=2
3150 H=17
3160 I=13
3170 RETURN
3180 PRINT "********-                ,****"
3190 PRINT "*******-  bbb h          ,***"
3200 PRINT "******-  )*****  a        ,**"
3210 PRINT "*****-  )*********   l  n ,*"
3220 PRINT "**-  n  *********- ) *+    *"
3230 PRINT "*-  c   ,*******- )* *- o  *"
3240 PRINT "-  )***  ******- )** -     *"
3250 PRINT "  *****  hh,**- )*-    m n ,"
3260 PRINT "+  ,**-  hhl   )*-    )***+ n"
3270 PRINT "*+       )***- )*-    )*****"
3280 PRINT "***+    )***- )*-  c m*******"
3290 PRINT "****+   ,**-       **********"
3300 PRINT "*****+            b**********"
3310 PRINT "******+     a h l)***********"
3320 PRINT "*******+ im)*****************"
3330 PRINT "****************************"
3340 E=15
3350 F=11
3360 G=2
3370 H=22
3380 I=10
3390 RETURN
3400 DATA 300,330,262,300,349,262,300,330,247,300,294,
     247,300,262,220,300,294,220,300,262,196,300,220,1
     96
3410 DATA 300,294,175,150,294,175,150,330,175,300,294,
     185,300,262,185,300,247,196,300,220,262
3420 DATA 600,196,247
3430 DATA 300,196,196,300,247,196,300,294,175,150,196,
     175,150,247,165,150,294,165
3440 DATA 150,196,165,150,247,147,600,294,147
3450 DATA 300,196,262,300,262,262,300,330,247,150,196,
     247
3460 DATA 150,262,220,150,330,220,150,196,220,150,262,
     196,600,330,196
3470 DATA 300,262,131,300,262,165
3480 DATA 150,262,196,150,262,220,1200,262,30000
3490 FOR T=1 TO 16
3500 READ A,B,C
3510 CALL SOUND(A*1.35,B,0,C,0)
3520 CALL KEY(0,K,S)
3530 IF S<>0 THEN 3780
3540 NEXT T
3550 RESTORE 3400
3560 FOR T=1 TO 34
3570 READ A,B,C
3580 CALL SOUND(A*1.35,B,0,C,0)
3590 CALL KEY(0,K,S)
3600 IF S<>0 THEN 3780
3610 NEXT T
3620 RESTORE 3430
3630 FOR T=1 TO 9
3640 READ A,B,C
3650 CALL SOUND(A*1.35,B,0,C,0)
3660 CALL KEY(0,K,S)
3670 IF S<>0 THEN 3780
3680 NEXT T
3690 RESTORE 3470
3700 FOR T=1 TO 5
3710 READ A,B,C
3720 CALL SOUND(A*1.35,B,0,C,0)
3730 CALL KEY(0,K,S)
3740 IF S<>0 THEN 3780
3750 NEXT T
3760 RESTORE 3400
3770 GOTO 3490
3780 RETURN
3790 CALL COLOR(2,8,6)
3800 ON SK-5 GOSUB 2380,2530,2740,2960,3180
3810 FOR T=1 TO 9
3820 Q=INT(RND*16)+8
3830 W=INT(RND*15)+8
3840 CALL GCHAR(Q,W,QW)
3850 IF QW<>32 THEN 3820
3860 CALL HCHAR(Q,W,110)
3870 NEXT T
3880 RETURN
3890 CALL COLOR(2,2,6)
3900 ON SK-10 GOSUB 2380,2530,2740,2960,3180
3910 FOR T=1 TO 5
3920 Z=INT(RND*16)+8
3930 X=INT(RND*15)+8
3940 CALL GCHAR(Z,X,ZX)
3950 IF ZX<>32 THEN 3920
3960 CALL HCHAR(Z,X,42)
3970 NEXT T
3980 RETURN
3990 CALL CLEAR
4000 PRINT "    DU HAR KLARAT SPELET    ": : : : : : :
      : : : : : :
4010 PRINT "         DIN POÄNG:":"           ";PO
4020 PRINT "": : : :"   VILL DU SPELA IGEN J/N   "
4030 RESTORE 3400
4040 GOSUB 3400
4050 IF K=74 THEN 850
4060 CALL CLEAR
4070 END
```

Sprites i cirklar

```
100 "**********************
110 "*                     *
120 "* SPRITES I CIRKLAR *
130 "*   ULF BOURELIUS     *
140 "*    BÖRJE HÄLL       *
150 "*                     *
160 "**********************
170 CALL CLEAR
180 CALL SCREEN(1)
190 Y1=21 :: X1=128 :: G1=180
200 Y2=96 :: X2=32 :: G2=270
210 Y3=156 :: G3=0
220 X4=220 :: G4=90
230 CALL CHAR(100,"061D2E3D6E75A6FCCDAF793B3D161F07A0
    D85CF476AB5DCA6B2D2AF62E4AB080")
240 FOR I=1 TO 8
250 CALL SPRITE(#I,100,I+2,192,256)
260 NEXT I
270 CALL MAGNIFY(3)
280 A=PI/180
290 CALL LOCATE(#1,COS(G1*A)*20+Y1,SIN(G1*A)*20+X1)
300 CALL LOCATE(#2,COS(G2*A)*20+Y2,SIN(G2*A)*20+X2)
310 CALL LOCATE(#3,COS(G3*A)*20+Y3,SIN(G3*A)*20+X1)
320 CALL LOCATE(#4,COS(G4*A)*20+Y2,SIN(G4*A)*20+X4)
330 CALL LOCATE(#5,SIN(G2*A)*20+Y2,COS(G2*A)*20+X1)
340 CALL LOCATE(#6,SIN(G3*A)*20+Y2,COS(G3*A)*20+X1)
350 CALL LOCATE(#7,SIN(G4*A)*20+Y2,COS(G4*A)*20+X1)
360 CALL LOCATE(#8,SIN(G1*A)*20+Y2,COS(G1*A)*20+X1)
370 G1=G1+10 :: G2=G2+10 :: G3=G3+10 :: G4=G4+10
380 GOTO 290
```

Masken för Mini Memory

av Björn Landsberg, Hägersten

När du kör programmet dyker först fyra små ringar upp, en i taget utspridda över spelplanen. Sedan kommer en mask (den växer sakta och blir längre och längre) som du kan styra med två tangenter, H=höger och V=vänster.

När du träffar en grön ring med masken så får du poäng (det är bra), men det kommer också en upp en grön samt en svart överkorsad ring på spelplanen som man inte bör krocka med eftersom det blir två minuspoäng i så fall. När du krockar med de svarta får du förutom minuspoängen en till grön ring att försöka träffa.

Programmet kan köras från BASIC med Mini Memory eller Editor/Assembler.

```
  1 REM  .....MASKEN.......
  2 REM  MINIMEM-BASIC
  3 REM  B.LANDSBERG
  4 REM  08/972460
  5 REM  (BREV 85-10-07)
  6 REM  MASKEN STYRS MED "H" OCH "V"
  7 REM  ...................
100 DIM A(800)
110 DEF SLUMP=32*INT(RND*22)+INT(RND*30)+33
115 GOSUB 1530
120 GOSUB 1230
130 CALL CLEAR
140 GOSUB 1290
150 GOSUB 660
160 GOSUB 760
170 FOR I=1 TO 4
180 CALL SOUND(10,1760,5,880,0,440,5)
190 CALL POKEV(SLUMP,200)
200 FOR I1=1 TO 100
210 NEXT I1
220 NEXT I
230 FOR I=1 TO K6
240 CALL KEY(K7,T,S)
250 IF (T<>K8)*(T<>K9)THEN 270
260 GOSUB 1030
270 A(I)=A(I-K1)+V
280 CALL PEEKV(A(I),TKN)
290 IF TKN=K2 THEN 320
300 IF TKN>K3 THEN 440
310 GOSUB 1120
320 CALL POKEV(A(I*K4),K2,"",A(I),K5)
330 NEXT I
340 REM   ===============
350 REM    orm slut:
360 REM
370 CALL SOUND(500,-8,0,110,5,220,5,440,10)
380 CALL SOUND(10,1760,0)
390 CALL POKEV(129,162,165,171,172,161,167,161,178,14
    0,175,178,173,165,174,193,180,175,167,193,179,172
    ,181,180)
400 CALL POKEV(152,193,129)
410 REM  ===============
420 REM   KRASH
430 REM
440 CALL SOUND(500,110,2,-7,5)
450 CALL POKEV(193,182,169,172,172,193,164,181,193,17
    9,165,193,178,165,176,178,169,179,193,159)
460 FOR K=1 TO 300
470 NEXT K
480 CALL KEY(3,T,S)
490 IF S=0 THEN 480
500 IF T<>74 THEN 530
510 GOSUB 1590
520 GOTO 450
530 CALL POKEV(193,182,169,172,172,193,164,181,193,17
    9,172,181,180,161,193,179,176,165,172,161,193,159
    )
540 FOR K=1 TO 300
550 NEXT K
560 CALL KEY(3,T,S)
570 IF S=0 THEN 560
580 IF T<>74 THEN 120
590 FOR I=1 TO 500
600 NEXT I
605 GOSUB 1530
610 CALL CLEAR
620 END
630 REM   ===============
640 REM     SKÄRM:
650 REM
660 CALL HCHAR(1,1,97,767)
670 CALL VCHAR(2,1,121,22)
680 CALL VCHAR(2,32,121,22)
690 CALL HCHAR(1,2,120,30)
700 CALL HCHAR(24,2,120,30)
710 CALL POKEV(0,218,"",31,219,"",767,220,"",736,221)
720 RETURN
730 REM  ==============
740 REM   VARIABLER:
750 REM
760 RANDOMIZE
770 SL=9999*RND
780 RANDOMIZE SL
790 V=1
800 A(0)=321
810 K1=1
820 K2=193
830 K3=207
840 K4=.7
850 K5=208
860 K6=1200
870 K7=3
880 K8=72
890 K9=86
900 P=0
910 RETURN
920 REM  ===============
930 REM   POÄNG:
940 REM
950 P=P+3*(TKN=191)+1
960 IF P>=0 THEN 980
970 P=0
980 CALL POKEV(770,INT(P/10)+144,"",774,P-10*INT(P/10
    )+144,"",776,208)
990 RETURN
1000 REM  ===============
1010 REM    ÄNDRA V
1020 REM
1030 CALL SOUND(-10,220,0,223,0)
1040 IF T=86 THEN 1070
1050 V=-(V=-32)+32*(V=-1)+(V=32)-32*(V=1)
1060 RETURN
1070 V=(V=-32)+32*(V=1)-(V=32)-32*(V=-1)
1080 RETURN
1090 REM  ===============
1100 REM   MAT/PARASIT:
1110 REM
1120 GOSUB 950
1130 CALL POKEV(SLUMP,200)
1140 IF TKN=191 THEN 1180
1150 CALL SOUND(-10,1760,0)
1160 CALL POKEV(SLUMP,191)
1170 RETURN
1180 CALL SOUND(-10,110,0)
1190 RETURN
1200 REM  ===============
1210 REM  SPRITE-SKAPNING
1220 REM
1230 CALL POKEV(768,10,10,144,1,10,25,144,1,208)
1240 CALL POKEV(1920,0,0,"",1924,0,0,"",-32287,0)
1250 CALL LOAD(-31788,225,"",-31878,0)
1260 RETURN
```

Mini-ASMBAS för Mini Memory

av Björn Landsberg, Hägersten

Det här programmet är tänkt att användas med Mini Memory och kassettbandspelare. Det läser in t.ex. ett maskinkodprogram direkt från minnet och skriver ett BASIC-program som laddar maskinkoderna med hjälp av CALL LOAD-satser. Det nya BASIC-programmet ersätter det gamla och finns i minnet efter programkörningen. När du kör programmet frågar det efter första resp. sista adressen till maskinkoden och sedan om du vill avsluta.

Om du vill så kan du fortsätta med att 'översätta' andra delar av minnet också. Om du skriver LIST när programkörningen är avslutad så ser du att det ursprungliga programmet är borta men att det nya laddningsprogrammet finns i minnet.

För att förstå programmet bättre kan du läsa om Börje Hälls 'LOADASMBAS' i PB 84-1 och hur BASIC-program lagras i minnet på s.6 i PB 85-1 Innan du matar in programmet måste du ändra värdet på adresserna 8330 och 8332 (hex). De talar om var radnummertabellen börjar och slutar och programmet som du matar in kommer inte att hamna på någon högre adress än vad som anges av (8332). Om du skriver CALL LOAD(-31952,53,0,53,0) så blir det drygt 2.7 Kb lediga högst upp i minnet där det nya programmet hamnar (tack vare raderna 180-310).

Raderna 350-370 skriver den nya radnummertabellen. På rad 380 ändras pekarna så att det blir den nya radnr.tabellen (och det nya programmet) som gäller. Ps. om någon kom på hur man kan 'MERGE:a' med kassettband så skulle det här programet vara ännu nyttigare

OBS""" Skriv CALL LOAD(-31952,53,0,53,0) innan du knappar in programmet och försök inte att numrera om det med RES av MINIASMBAS för då kraschar datorn" Spara programmet innan du kör det"

```
  10 REM MINI-ASMBAS
 100 DIM RADADR(100)
 110 DEF HB(X)=INT(X/256)
 120 DEF LB(X)=X-256*HB(X)
 130 CALL CLEAR
 140 RAD=0
 150 ADR=16383
 160 LOAD$=CHR$(157)&CHR$(200)&CHR$(4)&"LOAD"&CHR$(183
     )
 170 INPUT "START-,SLUTADRESS :":MK,MKSTOP
 180 RAD$=LOAD$&CHR$(200)&CHR$(LEN(STR$(MK)))&STR$(MK)
 190 FOR MK=MK TO MKSTOP
 195 CALL PEEK(MK,X)
 200 RAD$=RAD$&CHR$(179)&CHR$(200)&CHR$(LEN(STR$(X)))&
     STR$(X)
 210 IF LEN(RAD$)>=150 THEN 230
 220 NEXT MK
 230 MK=MK+1
 240 RAD$=RAD$&CHR$(182)&CHR$(0)
 250 ADR=ADR-LEN(RAD$)
 260 RAD=RAD+1
 270 RADADR(RAD)=ADR+1
 280 FOR I=1 TO LEN(RAD$)
 290 CALL POKEV(ADR+I,ASC(SEG$(RAD$,I,1)))
 300 NEXT I
 310 IF MK<=MKSTOP THEN 180
 320 INPUT "VILL DU AVSLUTA ? (J/N)":X$
 330 IF (X$="N")+(X$="n")THEN 170
 340 IF (X$<>"J")*(X$<>"j")THEN 320
 350 FOR I=1 TO RAD
 360 CALL POKEV(ADR-4*I+1,HB(90+10*I),LB(90+10*I),HB(R
     ADADR(I)),LB(RADADR(I)))
 370 NEXT I
 380 CALL LOAD(-31952,HB(ADR-4*RAD+1),LB(ADR-4*RAD+1),
     HB(ADR),LB(ADR))
 390 END
```

Följande ändringar har provats av RED.:

Om du använder flexskiva måste du skriva CALL LOAD(-31952,45,0,45,0) före inknappningen av programmet, samt ändra rad 150:

```
 150 ADR=14295
```

Ändringar om du vill att programmet skall överföra "Last free Address" >701E och REF/DEF-tabellen till CALL LOAD-satser tillsammans med själva assemblerprogrammet:

```
175 GOSUB 180
177 GOTO 320
320 RETURN
325 MK=28702
330 MKSTOP=28703
335 GOSUB 180
340 CALL PEEK(28702,A,B)
345 MK=A*256+B
347 MKSTOP=32767
348 GOSUB 180
```

■

**

```
1270 REM  ===============
1280 REM   TECKENDEF:
1290 CALL SCREEN(6)
1300 CALL CHAR(97,"8100000000000081")
1310 CALL CHAR(120,"0000FF0000FF0000")
1320 CALL CHAR(121,"2424242424242424")
1330 CALL CHAR(122,"00003F2020272424")
1340 CALL CHAR(123,"0000FC0404E42424")
1350 CALL CHAR(124,"2424E40404FC0000")
1360 CALL CHAR(125,"24242720203F0000")
1370 CALL CHAR(95,"9966669999666699")
1380 CALL CHAR(104,"3C42818181423C")
1390 CALL CHAR(112,"8100000000000081")
1400 CALL COLOR(8,2,15)
1410 CALL COLOR(9,9,15)
1420 CALL COLOR(10,13,15)
1430 CALL COLOR(11,15,7)
1440 CALL COLOR(12,7,15)
1450 CALL COLOR(5,1,15)
1460 CALL COLOR(6,1,15)
1470 CALL COLOR(7,1,15)
1480 CALL COLOR(4,1,15)
1490 RETURN
```

```
1500 REM  ===============
1510 REM   ÅTERSTÄLLNING
1520 REM
1530 CALL LOAD(-31788,224)
1540 CALL LOAD(-31878,0)
1550 RETURN
1560 REM  =============
1570 REM   REPRIS:
1580 REM
1590 RANDOMIZE SL
1600 GOSUB 1230
1610 CALL CLEAR
1620 GOSUB 660
1630 P=0
1640 J=I-1
1650 FOR K=1 TO 4
1660 CALL POKEV(SLUMP,200)
1670 NEXT K
1680 FOR K=1 TO J
1690 CALL PEEKV(A(K),TKN)
1700 IF TKN=K2 THEN 1720
1710 GOSUB 1120
1720 CALL POKEV(A(K*K4),K2,"",A(K),K5)
1730 NEXT K
1740 FOR K=1 TO 500
1750 NEXT K
1760 RETURN
```

■

LOGO – ett försök till mänsklig anpassning

av Lars Skyttner

För mig har alltid högteknologin haft en tendens att leva sitt eget liv, att sakna mänsklig anknytning. Den har anpassat människan till sina egna förutsättningar och resulterat i de segmenterade själar som blivit så många i dagens samhälle.

Den datateknologi som framställt sådana monster av användarfientlighet som programmeringsspråket APL eller den torftiga fattigmansengelskan i BASIC, har aldrig lyckats att imponera på mig.

Andra pretentiösa försök av mera akademisk karaktär som PASCAL, har med sina hårda krav på inre strukturerad logik bara blivit något som liknar ett elegant självändamål.

Det är först när det användbara och praktiska fått en dos av kultur som begreppet människovänlig får någon verklig innebörd. Den som under sena kvällstimmar arbetat med programmering och fått det blinkande budskapet:"Fatal error 49 in 513", kan knappast undgå någon form av klubbslageffekt.

Betydligt behagligare är det att läsa LOGOs diskreta uppmaning:"Tell me how to do with...".

Exemplet visar en annan attityd, nämligen att här är det datorn som är anpassad efter vårt sätt att tänka, inte tvärt om.

Lyckligtvis är jag nu inte ensam om att uppleva den mentala torftigheten i vårt sätt att kommunicera med datorer. Ett ambitiöst projekt som startade 1969 med MITs laboratorium för artificiell intelligens har resulterat i datorspråket LOGO som ställer människan i stället för datorn i centrum. Utgångspunkten var LISP, som gärna används av forskare i artificiell intelligens. Resultatet har blivit något som till stora delar gör skäl för namnet användarvänlig.

I motsats till andra datorspråk som konstruerats av matematiker och ingenjörer har LOGO konstuerats av psykologer och pedagoger.

Delar av LOGO som finns i ett antal olika versioner har börjat att inkorporeras i andra språk som SMALLTALK och USCD-PASCAL. Utmärkande drag för LOGO är möjligheten att definiera procedurer med lokala variabler för att tillåta rekursion. Det är alltså möjligt att definiera nya kommandon och funktioner som kan användas på samma sätt som de sk primitiverna. Dessa är de ursprungliga ca 200 grundkommandona.

LOGO är ett interpreterande språk, vilket utgör grunden för den interaktiva användningen. Det har dessutom fullständiga liststrukturer och opererar på listor vars ingående enheter själv kan vara listor, listor över listor o. s. v.

Bakom LOGO ligger alltså en utvecklad pedagogisk id', ett system för att ge tidiga och enkla inkörsportar till programmering för nybörjare som saknar matematisk kunskap. Därvid har man använt sig av färger, musik och rörelser. Ett exempel är sköldpaddsgrafiken (Turtle-grafik), som visat sig fungera utmärkt som inlärningsinstrument. Sköldpaddan kan få's att vandra över bildskärme med olika kommandon eller med en joy-stick, och därvid utföra olika uppdrag. Med olika kommandon kan den arbeta synligt eller osynligt och färglägga sitt eget spår.

En annan möjlighet är Sprite-hanteringen. Man skapar då olika figurer, tecken eller symboler som kan få's att agera tillsammans eller ensamma. Allt med olika färger, hastighet och till och med under musikaccompanjemang. Det är till och med möjligt att skapa rörliga bilder av filmkaraktär.

LOGO är ett underbart språk för barn skriver Texas Instument i introduktionen av sitt nya LOGO-paket. Jag försäkrar att det är precis lika underbart för vuxna. Särskilt för den som yrkesmässigt terroriseras av enfärgade bildskärmar, kryptiska koder och fantasilösa tabeller.

Trots allt detta eller kanske därför, är inte LOGO att betrakta som en leksak för barn, utan som ett kraftfullt programmeringsspråk, med alla vanliga användningsmöjligheter.

LOGOs komplexitet och styrka har inneburit krav på minnesutrymme som gjorde det tillgängligt endast på större datorer under 70-talet. Sjunkande minneskostnader och den nya microdatortekniken har gjort LOGO tillgängligt på 48 Kb-system som APPLE-II och Texas 99.

Datatekniken har kommit för att stanna. De flesta samhällssektorer och skolan har redan "tagits över". Skall du ha en changs att påverka utvecklingen måste du lära dig tekniken. Gör det på det minst smärtsamma sättet, lär dig LOGO. ■

LT-WRITER
huvudprogrammet

av Lennart Thelander, Helsingborg

Vi listar här själva huvudprogrammet till LT-Writer som har beskrivits i PB 85-1,2,3,4.

```
100 GOTO 120 :: DIM A$(300):: CALL CLEAR :: CALL KEY
    :: CALL SCREEN :: CALL CHAR :: A,B,C,D,E,F,G,H=0
    :: B$,C$,D$,E$,F$,G$
110 CALL SOUND :: CALL LINK :: CALL INIT :: CALL LOAD
     :: DIM H$(300):: I,J,K,L,M,N,O,P,Q :: "!P-
120 CALL CLEAR :: PRINT "INITIALIZING": :"PLEASE WAIT
    "
130 ON WARNING NEXT :: ON BREAK NEXT :: CALL INIT
140 CALL LOAD(-31806,16):: CALL LOAD(9460,36,250,37,3
    2,0,0,"",9498,0,16,32,0,224,0,216,32)
150 CALL LOAD(9506,152,2,36,248,192,62,216,32,152,2,3
    6,249,6,32,36,248,216,32,37,26,156,2)
160 CALL LOAD(9528,194,160,131,246,216,32,37,27,156,2
    ,6,192,216,0,131,109,192,224,131,86,208,32)
170 CALL LOAD(9550,152,0,2,64,31,0,6,192,208,32,152,0
    ,6,192,5,192,208,160,131,115,9,130)
180 CALL LOAD(9572,2,34,131,0,5,194,2,1,170,12,196,12
    9,6,194,216,2,131,115,216,0,156,2)
190 CALL LOAD(9594,6,192,216,0,156,2,192,160,63,228,2
    ,1,37,150,200,1,63,228,2,224,131,224)
200 CALL LOAD(9616,2,11,0,112,4,91,2,224,36,250,200,2
    ,63,228,2,35,255,248,192,3,4,32)
210 CALL LOAD(9638,32,40,67,224,37,28,36,96,37,30,22,
    5,208,160,131,124,36,160,37,28,19,5)
220 CALL LOAD(9660,9,81,4,221,215,65,227,224,37,28,21
    6,32,36,248,156,2,200,10,131,246,216,32)
```

```
230 CALL LOAD(9682,36,249,156,2,3,128,5,0,10,80,0,0,3
    3,0,96,0,"",9726,0,29)
240 CALL LOAD(9728,38,44,38,66,0,0,0,0,0,0,0,0,"",977
    2,192,160,38,10,2,1)
250 CALL LOAD(9778,160,0,4,32,32,44,192,32,38,8,4,32,
    32,36,4,91,192,96,38,8,192,160)
260 CALL LOAD(9800,38,10,4,32,32,44,4,91,2,224,38,12,
    4,192,2,1,0,1,2,2,37,225)
270 CALL LOAD(9822,212,160,37,255,4,32,32,20,208,146,
    9,130,2,34,0,10,2,0,10,0,2,1)
280 CALL LOAD(9844,37,216,4,32,32,36,2,32,0,9,200,0,1
    31,86,4,32,36,244,0,8,22,16)
290 CALL LOAD(9866,2,0,10,0,2,1,38,18,2,2,0,2,2,3,0,1
    1,4,32,32,36,200,0)
300 CALL LOAD(9888,131,28,2,0,36,0,4,32,32,52,2,0,10,
    82,193,0,2,1,38,4,2,2)
310 CALL LOAD(9910,0,8,4,32,32,44,160,2,192,224,38,6,
    5,131,10,19,192,227,38,0,6,147)
320 CALL LOAD(9932,192,32,38,4,19,2,160,4,16,236,4,19
    2,216,0,131,124,2,224,131,224,4,91)
330 CALL LOAD(16376,83,85,80,79,82,84,38,80):: CALL L
    OAD(8196,63,248):: CALL LINK("SUPORT","DSK.LT_DIS
    K.LTSUPPORT"):: CALL CLEAR :: CALL SCREEN(4):: ON
     ERROR 1150
340 CALL LINK("STINIT",I):: D=D+1 :: GOTO 1060
350 CALL LINK("FLIP"):: C=MAX(C,D):: PRINT STR$(D)&">
    ";:: ACCEPT BEEP:E$ :: CALL LINK("NOFLIP"):: IF L
    EN(E$)=0 THEN GOTO 710 ELSE IF ASC(E$)<>131 THEN
    710
360 CALL LINK("FLIP"):: CALL LOAD(-31884,5):: LINPUT
    "CMD>":E$ :: IF E$>"Z" THEN E$=CHR$(ASC(E$)-32)&S
    EG$(E$,2,255)
370 CALL LOAD(-31884,3):: CALL LINK("NOFLIP"):: IF SE
    G$(E$,1,1)="C" THEN F$=SEG$(E$,2,1):: GOTO 570
380 IF SEG$(E$,1,1)="I" THEN 700
390 IF E$="P" THEN 1100
400 IF SEG$(E$,1,1)="P" THEN 660
410 IF SEG$(E$,1,1)="L" THEN F$=SEG$(E$,2,1):: GOTO 7
    20
420 IF E$="N" THEN 750
430 IF SEG$(E$,1,1)="M" THEN 760
440 IF E$="S" THEN 770
450 IF E$="R" THEN 850
460 IF E$="E" THEN 960
470 IF E$="G" THEN 1450
480 IF E$="W" THEN 970
490 IF SEG$(E$,1,1)="B" THEN 1240
500 IF E$="O" THEN 1260
510 IF E$="Z" THEN 1270
520 IF E$="F" THEN 1280
530 IF SEG$(E$,1,1)="D" THEN 1160
540 IF E$="H" THEN 1060
550 IF E$="Ñ" THEN 1130
560 PRINT "ILLEGAL COMMAND" :: GOTO 360
570 F=POS(E$,F$,1):: G=POS(E$,F$,F+1):: H=POS(E$,F$,G
    +1):: IF F=0 OR G=0 OR H=0 THEN PRINT "WRONG FORM
    AT" :: GOTO 360
580 B$=SEG$(E$,F+1,G-F-1):: C$=SEG$(E$,G+1,H-G-1):: D
    $=SEG$(E$,H+1,LEN(E$)-H):: IF D$="" THEN E=D :: G
    OTO 640 ELSE IF D$<>"ALL" AND D$<>"all" THEN E=VA
    L(D$):: GOTO 640
590 FOR E=1 TO C
600 F=POS(A$(E),B$,1):: IF F=0 THEN 630
610 PRINT STR$(E)&">";A$(E):: INPUT "CHANGE THIS Y/N?
    ":E$ :: IF E$<>"Y" THEN 630
620 A$(E)=SEG$(A$(E),1,F-1)&C$&SEG$(A$(E),F+LEN(B$),L
    EN(A$(E))-F+LEN(B$)):: PRINT STR$(E)&">";A$(E)::
    GOTO 600
630 NEXT E :: GOTO 360
640 D=E :: F=POS(A$(D),B$,1):: IF F=0 THEN PRINT "STR
    ING '";B$;"' NOT FOUND" :: GOTO 360
650 A$(D)=SEG$(A$(D),1,F-1)&C$&SEG$(A$(D),F+LEN(B$),L
    EN(A$(D))-F+LEN(B$)):: PRINT A$(D):: GOTO 360
660 F=POS(E$,",",1):: IF F=0 THEN D=VAL(SEG$(E$,2,LEN
    (E$)-1)):: PRINT A$(D):: GOTO 360
670 G=VAL(SEG$(E$,2,F-2)):: H=VAL(SEG$(E$,F+1,LEN(E$)
    -F+1)):: FOR E=G TO H :: PRINT A$(E):: CALL KEY(0
    ,A,B):: IF B=0 THEN 690
680 CALL KEY(0,A,B):: IF B<>1 THEN 680
690 NEXT E :: GOTO 360
700 PRINT "TYPE CTRL/C IN POS 1 TO EXITTYPE CTRL/E TO
     END FORMAT   TYPE CTRL/F TO START FORMAT" :: CAL
    L LOAD(-31884,5):: GOTO 350
710 FOR E=C TO D STEP -1 :: A$(E+1)=A$(E):: NEXT E ::
     A$(D)=E$ :: C=C+1 :: D=D+1 :: GOTO 350
720 B$=SEG$(E$,3,LEN(E$)-3):: FOR E=D+1 TO C :: GOSUB
     730 :: NEXT E :: FOR E=1 TO D :: GOSUB 730 :: NE
    XT E :: PRINT "NO LOCATION" :: GOTO 360
730 F=POS(A$(E),B$,1):: IF F=0 THEN RETURN
740 D=E :: GOTO 1140
750 D=D+1 :: D=MIN(D,300):: C=MAX(C,D):: GOTO 1140
760 D=VAL(SEG$(E$,2,LEN(E$)-1)):: D=MIN(D,300):: C=MA
    X(C,D):: GOTO 1140
770 PRINT "SAVE":"ENTER FILENAME:":"CS1" :: CALL LINK
    ("FLIP"):: CALL LOAD(-31884,3):: ACCEPT AT(23,1)S
    IZE(-28)BEEP:G$ :: CALL LINK("NOFLIP")
780 E=1 :: CALL LINK("STCLR",I):: CALL LINK("STSTO",I
    ,STR$(C))
790 CALL LINK("STSTO",I,A$(E)):: CALL LINK("STFREE",I
    ,N):: E=E+1 :: IF E>C THEN 840
800 IF N-10>LEN(A$(E))THEN 790
810 E$=CHR$(200)&CHR$(201)&CHR$(202):: CALL LINK("STS
    TO",I,E$):: CALL LINK("STSAVE",G$):: CALL LINK("S
    TCLR",I)
820 CALL LINK("STSTO",I,A$(E)):: E=E+1 :: IF E>C THEN
     830 ELSE 820
830 IF G$<>"CS1" AND G$<>"CS2" THEN G$=SEG$(G$,1,LEN(
    G$)-1)&CHR$(ASC(SEG$(G$,LEN(G$),255))+1)
840 CALL LINK("STSAVE",G$):: CALL LINK("STCLR",I):: G
    OTO 360
850 PRINT "READ":"ENTER FILENAME:":"CS1" :: CALL LINK
    ("FLIP"):: CALL LOAD(-31884,3):: ACCEPT AT(23,1)S
    IZE(-28)BEEP:G$ :: CALL LINK("NOFLIP")
860 PRINT "DO YOU WANT TO DELETE ALL  CURRENT LINES
    <Y/N>";:: INPUT B$ :: CALL LINK("STCLR",I):: E$=C
    HR$(200)&CHR$(201)&CHR$(202):: CALL LINK("STLOAD"
    ,G$):: CALL LINK("STRCL",I,C$):: F=VAL(C$)
870 IF B$<>"Y" THEN 920
880 FOR E=1 TO F :: CALL LINK("STRCL",I,A$(E)):: IF A
    $(E)<>E$ THEN 910
890 IF G$<>"CS1" AND G$<>"CS2" THEN G$=SEG$(G$,1,LEN(
    G$)-1)&CHR$(ASC(SEG$(G$,LEN(G$),255))+1)
900 CALL LINK("STCLR",I):: CALL LINK("STLOAD",G$):: C
    ALL LINK("STRCL",I,A$(E))
910 NEXT E :: CALL LINK("STCLR",I):: FOR E=F+1 TO C :
    : A$(E)="" :: NEXT E :: C=F :: GOTO 360
920 FOR E=1 TO F :: CALL LINK("STRCL",I,A$(E+C)):: IF
     A$(E+C)<>E$ THEN 950
930 IF G$<>"CS1" AND G$<>"CS2" THEN G$=SEG$(G$,1,LEN(
    G$)-1)&CHR$(ASC(SEG$(G$,LEN(G$),255))+1)
940 CALL LINK("STCLR",I):: CALL LINK("STLOAD",G$):: C
    ALL LINK("STRCL",I,A$(E+C))
950 NEXT E :: CALL LINK("STCLR",I):: C=C+F :: GOTO 36
    0
960 INPUT "DO YOU WANT TO EXIT Y/N? ":E$ :: IF E$<>"Y
    " THEN 360 ELSE CALL LINK("NOFLIP"):: CALL LOAD(-
    31803,36):: STOP
970 PRINT "WRITING TO FILE" :: CALL LINK("FLIP"):: CA
    LL LOAD(-31884,3):: PRINT "FILENAME:":"RS232.DA=8
    .BA=9600"
980 ACCEPT AT(23,1)SIZE(-28)BEEP:G$ :: IF SEG$(G$,1,5
    )="RS232" THEN INPUT "LEFT MARGIN:":O :: GOTO 100
    0
990 CALL LINK("NOFLIP"):: OPEN #1:G$ :: FOR E=1 TO C
    :: PRINT #1:A$(E):: NEXT E :: CLOSE #1 :: GOTO 36
    0
1000 CALL LINK("NOFLIP"):: OPEN #1:G$ :: FOR E=1 TO C
     :: CALL LINK("CODES",A$(E),B$)
1010 ON ERROR 1040
1020 PRINT #1:TAB(O);B$ :: CALL KEY(0,A,B):: IF A=2 TH
     EN 1030 :: ON ERROR 1150 :: NEXT E :: PRINT #1:CH
     R$(12)
1030 CLOSE #1 :: GOTO 360
1040 ON ERROR 1150 :: RETURN 1030
1060 CALL CLEAR :: PRINT "Bx<,y> buffer line x<thro y>
     O outputs buff before actlinZ zeroes buffer
           F formats text" :: PRINT "P print all line
     s":"G gets memory space"
1070 PRINT "Px<,y> print line x<thro y> L/text/ locate
     s text         C/text1/text2/(x?ALL) change    tex
     t1 to text2 on line x    or on all lines"
1080 PRINT "S saves all lines on CS1     R reads all li
     nes from CS1  Mx moves to line x           N next
     line"
1090 PRINT "I inserts before actline    E exits progra
     m            W writes to a file (printer)Dx<,y>
     del's line x<thro y>" :: PRINT "H help" :: PRINT
     "Ñ prints no of lines" :: GOTO 360
1100 FOR E=1 TO C :: PRINT STR$(E);">";A$(E):: CALL KE
     Y(0,A,B):: IF B=0 THEN 1120
1110 IF A=2 THEN 360 :: CALL KEY(0,A,B):: IF B<>1 THEN
      1110
1120 NEXT E :: GOTO 360
1130 PRINT "You have";C;"lines" :: GOTO 360
```

■

Input för Assembler

av Börje Häll

RÄTTELSE till INPUT ett subprogram i assembler, som var publicerat i PB 84-3.

Tyvär har jag inte uppmärksammat att det blivit ordentligt fel vid listningen av programmet när det publicerades. Det var först när jag fick ta del av Lennart Thelanders svar på enkäten som jag fick reda på detta fatala misstag.

Som man säger BÄTTRE SENT ÄN ALDRIG så kommer här hela artikeln förhoppningsvis felfri.

INPUT ett subprogram i assembler.

Input är ett subprogram för inmatning av data från tangentbordet. Det kan inte användas fristående utan måste anropas från ett annat program. I det anropande programmet måste REF INPUT finnas med. De tecken som kan matas in är blank tom ~ (ASCII 32-126). Inga kontroller för att förhindra skrivning utanför skärmen finns. Skärmadresserna är 0-767 i graphics mode och 0-959 i text mode. Det sker heller ingen kontroll att max antal tecken som får matas in är större än 0.

Förhoppningsvis förklarar kommentarerna på respektive rad hur subprogrammet fungerar.

Det som kanske behöver förtydligas är JHE på raderna 115 och 142. Talen i datorn lagras binärt med negativa tal i 2-komplementform. Det betyder att datorn behandlar tal i området -32768 tom 32767. JHE är en logisk jämförelse och det innebär att datorn vid jämförelse inte uppfattar tal i 2-komplementform som negativa. Det betyder i sin tur att datorn, vid logisk jämförelse, arbetar med talområdet 0 tom 65535.

På rad 120 ska hopp ske om antal inmatade tecken är lika med eller större än max antal tecken som får matas in. Varken max antal tecken eller antal tecken som har matats in kan vara negativt och det innebär att vi bara jämför positiva tal. Då kan JHE användas i den vanliga betydelsen av större än eller lika med. Motsvarande gäller för rad 147.

Om något eller båda av talen kan vara negativt så kan inte JHE användas utan man kan göra på följande sätt:

```
Antag: R3=R2
            R5=-5
            Hopp ska ske om R3>=R5


            C    R3,R5
            JEQ  A        * Inget hopp ty R3<>R5
            JGT  A        * Hopp ty R3>R5
```

Om JHE används så sker inget hopp eftersom datorn, vid logisk jämförelse, uppfattar 2 som mindre än -5.

```
 1 ************************************************************
 2 * Källkod: K:INPUT                  Använda register       *
 3 * Objektkod O:INPUT                 R0  VDP access         *
 4 *                                   R1  -"-                *
 5 * Börje Häll                        R2  Status             *
 6 * Vasavägen 101                     R3  Teckenräknare      *
 7 * 175 32  JÄRFÄLLA                  R4  Blinkräknare       *
 8 *                                   R5  Max antal tecken   *
 9 * 1984-04-23                           som får matas in   *
10 * Modifierat 1984-05-27             R14 Överföring av data *
11 *                                       och returadress    *
12 * Subprogram för inmatning                                 *
13 *                                                          *
14 * Antal inmatade tecken returneras                        *
15 *        i anropande programs R10                          *
16 *                                                          *
17 * Anrop: BLWP 'INPUT                                      *
18 *        DATA Startadress på skärmen                       *
19 *        DATA Max antal tecken                             *
20 *                                                          *
21 ************************************************************
```

```
22         IDT  'INPUT'
23 *--------------------------
24 * Ingångsadress
25 *
26         DEF  INPUT
27 *--------------------------
28 * Globala rutiner
29 *
30         REF  VSBW,KSCAN,VSBR,GPLLNK
31 *--------------------------
32 * Equates
33 *
34 KEY    EQU  >8375 * Här spars nertryckt tangent
35 KEYBRD EQU  >8374 * Vilket tangentbord som ska skannas
36 STATUS EQU  >837C
37 *--------------------------
38 * Konstanter
39 *
40 B0     BYTE 0
41 B127   BYTE 127   * För kontroll av godtagbart tecken
42 B31    BYTE 31    * -"-
43 B32    BYTE 32    * Blanktecken
44 ENTER  BYTE >0D   * För kontroll av ENTER-tangenten
45 PV     BYTE 8     * -"-             pil vänster-tangenten
46 D0     DATA 0     * -"-             flaggor
47 D1     DATA 1     * -"-
48 SET    DATA >2000 * -"-          om någon tangent har tryckts
                                    ner
49 *--------------------------
50 * Variabler
51 *
52 OLDKEY BYTE 0     * För att spara gammalt tecken
53         EVEN
54 BLINK  DATA 0     * 1=skriv tecken, 0=skriv kursorn
55 DLJUD  DATA 0     * 1=ljud har avgivits, 0=ljud har inte
                       avgivits
56 START  DATA 0     * Startposition på skärmen
57 STOPP  DATA 0     * Slutposition på skärmen
58 *--------------------------
59 * Workspace
60 *
61 INPREG BSS  >20   * Subprogrammets workspace
62 *--------------------------
63 * Programstart
64 *
65 INPUT  DATA INPREG        * Workspace pointer
66         DATA INP1         * Program counter
67 INP1   MOV  *R14+,'START  * Spar startposition
68         MOV  *R14+,'STOPP
69         MOV  'STOPP,R5    * Spar antal tecken som får matas
                               in
70         CLR  'BLINK       * 0-ställ blinkflaggan
71         CLR  'DLJUD       * 0-ställ ljudflaggan
72         MOVB 'B32,'OLDKEY * Blanktecken till OLDKEY
73         CLR  R3           * 0-ställ teckenräknaren
74         CLR  R4           * 0-ställ blinkräknaren
75         A    'START,'STOPP * Addera startpositionen till antal
                               tecken
76 *                         * för att få slutposition på
                               skärmen
77 *                         * OBS slutpositionen är 1 för
                               mycket
78         DEC  'STOPP       * Justera slutpositionen till rätt
                               värde
79         MOV  'START,R0    * Startpositionen till R0
80 INP2   C    'BLINK,'D1   * Ska tecken skrivas?
81         JEQ  INP3         * Ja
82         LI   R1,>1E00     * Nej, cursorn till R1
83         INC  R4           * Öka blinkräknaren
84         CI   R4,>100      * Har cursorn varit på tillräckligt
                               länge?
85         JLT  INP4         * Nej
86         INC  'BLINK       * Ja, 1-ställ flaggan för
                               tecken/cursor
87         CLR  R4           * 0-ställ blinkräknaren
88         JMP  INP4         * Hoppa och skriv cursorn
89 INP3   MOVB 'OLDKEY,R1    * Skriv tecknet
90         INC  R4           * Öka blinkräknaren
91         CI   R4,>100      * Har tecknet varit på tillräckligt
                               länge?
92         JLT  INP4         * Nej
93         DEC  'BLINK       * Ja, 0-ställ flaggan för
                               tecken/cursor
94         CLR  R4           * 0-ställ blinkräknaren
95 INP4   BLWP 'VSBW         * Skriv tecknet/cursorn på skärmen
96         LIMI 2            * För att ljud ska höras
97         LIMI 0
98         CLR  'KEYBRD      * Tangentbord 0 ska skannas
99         BLWP 'KSCAN       * Skanna tangentbord 0
100        MOVB 'STATUS,R2   * Statusbyte till R2
101        COC  'SET,R2      * Har en tangent tryckts ner?
102        JNE  INP2         * Nej
103        CB   'KEY,'ENTER  * Ja, har ENTER-tangenten tryckts
                               ner?
104        JEQ  INP11        * Ja
105 INP5   CB   'KEY,'PV     * Nej, har pil vänster tryckts ner?
106        JEQ  PILV         * Ja
107        CB   'KEY,'B31    * Nej, är teckenkoden för liten?
```

```
108         JGT   INP6          * Nej
109         BL    'LJUD         * Ja, hoopa till subrutin för
110 *                           * bad response tone
111         JMP   INP2
112 INP6    CB    'KEY,'B127    * Är teckenkoden för stor?
113         JLT   INP7          * Nej
114         BL    'LJUD         * Ja
115         JMP   INP2
116 *-------------------------
117 * Godtagbart tecken
118 *
119 INP7    C     R3,R5         * Har max antal tecken matats in?
120         JHE   INP8          * Ja
121         INC   R3            * Nej, öka teckenräknaren
122 INP8    MOVB  'KEY,R1       * Flytta tecknet till R1
123         BLWP  'VSBW         * Skriv tecknet på skärmen
124         INC   R0            * Öka skrivpositionen
125         C     R0,'STOPP     * Har max antal tecken skrivits?
126         JGT   INP9          * Ja
127         BLWP  'VSBR         * Nej, läs tecken från skärmen
128         MOVB  R1,'OLDKEY    * Spar tecknet
129         JMP   INP2
130 INP9    C     'DLJUD,'B0    * Har ljud avgivits
131         JEQ   INP10         * Ja
132         BL    'LJUD         * Nej
133 INP10   MOV   'B1,'DLJUD    * 1-ställ flaggan för ljud
134         MOV   'STOPP,R0     * Sista skrivpositionen till R0
135         BLWP  'VSBR         * Läs ett tecken
136         MOVB  R1,'OLDKEY    * Spar tecknet
137         BLWP  'VSBW         * Skriv tecknet
138         JMP   INP2
139 *-------------------------
140 * Pil vänster
141 *
142 PILV    MOVB  'B32,R1       * Blanktecken till R1
143         BLWP  'VSBW         * Skriv tecknet på skärmen
144         DEC   R3            * Minska teckenräknaren
145         DEC   R0            * Minska skrivadressen
146         C     R0,'START     * Förbi första skrivpositionen?
147         JHE   PILV1         * Nej
148         BL    'LJUD         * Ja
149         MOV   'START,R0     * Första skrivpositionen till R0
150         CLR   R3            * 0-ställ teckenräknaren
151 PILV1   BLWP  'VSBR         * Läs tecken från skärmen
152         MOVB  R1,'OLDKEY    * Spar tecknet
153         BLWP  'VSBW         * Skriv cursorn på skärmen
154         B     'INP2
155 *-------------------------
156 * Avsluta
157 *
158 INP11   MOVB  'OLDKEY,R1    * Gamla tecknet till R1
159         BLWP  'VSBW         * Skriv tecknet på skärmen
160         MOV   R3,'20(R13)   * Antal inmatade tecken till
161 *                           * anropande programs R10
162         RTWP                * Hoppa till anropande program
163 ****************************************
164 * Subrutin för att avge bad response tone *
165 *                                        *
166 * Anrop: BL    'LJUD                     *
167 ****************************************
168 LJUD    MOVB  'B0,'STATUS   * Status måste 0-ställas innan GPLLNK
                                  anropas
169         BLWP  'GPLLNK       * Ge ljud
170         DATA  >36           * Bad response tone
171         RT                  * Hoppa till anropande rutin
172         END
```

■

ANNONS

TI 99/4A grundenhet och tillbehör säljes

Expensionsbox, Disk controller-card, RS-232 interface, 32 kB minnes exp., matrisskrivare(Epson), Diskdrive, Extended basic, Joysticks, Personal record keeping, Multiplan, Video chess, Kassettkabel, Structural engeneering library, Math routine library, Electrical engeneering library, Teach yourself ex.basic, Basic för nybörjare, disketter med spel, nyttoprogram typ glosprogram, budget etc.
Manualer och instruktionsböcker för allt+ Assembler manual.

Säljes i delar eller som helhet till högst erbjudna pris.

Tel 0470-33579 (16.30-20.30)

Lars-Åke Ragnarsson
Syrenvägen 4
360 14 VÄCKELSÅNG

ANNONS

SÄLJES: 3 spel i X-BASIC på kasset

WCT	(3D-tennis. Mycket bra! 1-2 spelare)	35 kr
GOLF	(18 hål! 1-2 spelare, från ca 13 år Grafik)	35 kr
SPACIC	(Rymdspel, Action! 1 spelare)	30 kr
SAMTLIGA SPEL		85 kr

Porto tillkommer.

Anders Månsson
Ängeltoftag 42
216 22 MALMÖ
040-15 74 37

****

ANNONS: SÄLJES

TI 99/4A, Bandspelare, Kabel, Joystic, Spelmodul Parsec, Extended Basic.

Pris 1700:- eller högstbjudande

Mikael Neglen
Svanebäcksgränd 6
216 20 MALMÖ

Tel. 040-15 58 81

ANNONS SÄLJES

Texas Instruments expansionsbox med:
Flex cable-interface kort
RS232-kort
Mini-Memory + användartips på svenska
Extended Basic
Personal Report Generator
Personal Record Keeping
Music Maker (modul)
Mind Challenger (modul)

Nypris c:a 6500:- säljes för 3000:-
Christer Carlström
Tel. 0413-211 90

ANNONS

Hardware fra MYARC:

MYARC Mini-Box. 1 x True centronics, 1 x RS232, Disc Controller for 4 DS/DD disc drive. Kr 6000:- uten disc drive.

MYARC RS 232 kort for TI-boxen, dette kortet har eindel pluss i forhold till Texas sitt RS 232 kort. Kr 1700:-

MYARC Disc Control Card. Med dette kortet kan du bruke opptill 4 DD/DS. Det har inbygget nokre exstra basic komandoer gjer att ein slepp å bruke ED/AD modulen i enkelte tillfelle. Kr 2500:-

MYARC 128k Memory Expansion. Detta kortet har 32k RAM som ein må ha for å kjöre ED/AS, TI LOGO og andere. I tillegg har dette kortet 96k RAM DISC og det kan brukast som ein PRINTER BUFFERT. Kortet kan byggast ut till 512k RAM. Kr 2900:-

Fra Tysland kan eg tilby:

Monitor kabel Kr 150:-. Joystick adapter Kr 100:-. Joystick Kr 150:- ikkje Texas sine. Frittstående Centronics-Interface Kr 1200:-

Disk-COntroller (1-4 disks, SS/SD - DS/DD) 1 disk (360 kbyte) 32k RAM pluss strömforsyning Kr 4000:-

Gunnar Håland
Import for Texas Instruments
Boks 88
N-4220 Sandeid
Norge

Tlf. (047) 61 453 etter kl 18.00

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:- per kassett
	Kopieringsavgift	15:-
DISKETT	Startkostnad	55:- per diskett
	Kopieringsavgift	10:-

OBS Var noggrann när DU anger programnummret. Det är det jag går efter vid kopieringen

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 kör programmet och vad som behöver göras för att programmet skall fungera. Skriv även vilken modul som erfordras till programmet.

PROGRAMKATEGORIER

02 = Operativsystem
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

KODFÖRKLARING

De fyra första siffrorna är interna beteckningar (diskettnummer, programkategori).

Femte siffran anger ursprungsland:

1 = England, USA
2 = Holland, Belgien, Tyskland
3 = Sverige
7 = Finland

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

Dessutom kan ett D finnas allra sist i programnumret och det anger att datafile(r) ingår.

Eventuella bokstäver efter programnumret talar om vilken utrustning som krävs.

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 (pratorn)
J = Joystick
T = Terminal Emulator II
K = Personal Record Keeping

02 OPERATIV SYSTEM

10 REGISTERPROGRAM

01101002E ---- Adressregister
11103069 ---- Filläsning på Kassett
11103072 --K Snabbfil till Personal Record Keeping modulen
14103045 PDX Mini-Reg. Utmaningen 84-3
15103092 -DX Tel-Ad. Telefonadresser

12 ORDBEHANDLING

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

14 EKONOMI

07141001H --X Annuities
12143006 ---- Ränteberäkningar

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
01391002E ---- Lösning av ekvationssystem
01392001E ---- Division med valfritt antal decimaler i svaret
09391003H ---- Differential eq
11393071 ---- Rita sinuskurva
12393005 ---- Matrisberäkningar, lösning av ekvationssystem
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
16653004 ---- Ohmslag, räknar ut ampere, ohm, volt och watt

78 ASTRONOMI

01781001E ---- Beräkning av geostationära satelliters positioner
15783000 --X 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
07902002E --P Banner, stor text utskriven på TP-printer
08901006E PEX Utskrift av titelsida för ex listningar
08902003E ---- Stora tecken för 12x28 teckenskärm
10901008E ---- Very large charachters
11901011S ---- Stapeldiagram
11903063 PDX Lista Basicprogram med Basicprogram
11903066 -PX 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
13901057 -EX Cursor, ändra utseendet på cursorn
13901058 ---- Ensa-komma
13903022 -DX Fil-Data-Editor, för DIS/VAR80
13903027 --X Grafik Exklusive, Kassettbaserat
13903060 PDX List28, Lista program 28tkn/bredd
13903061 -PX ASCII-koder för olika teckenuppsättningar
13903062 -DX P:TILLCOMP, överför P:TEXTIN filer till Companion format
14901053 --X Talkonvertering, Hex-Dec-Binärt
14902052 --X Färg-Editor
14903034 ---- Teckendefinierare
14903042 ---- Kassettinnehåll
15901089E -DX Merge/Read, listar DIS/VAR163 filer
15901090E -EX Mg/Peeker, listar minnes-adresser
15903001 EAD Disassembler, en fullständig disassemblerare
15903002 EAD K:INPUT, subprogram för inmatning av data från tangentbord
15903005 ---- Baskonvertering, Hex-Dec-Bin
15903076 PDX Listprogram, Listar Basicprogram
15903077 -SX Trunkering, skapa egna ord med trunkering
15903088 -EX List-Size, listar storleken på dina program
15903093 -DX Char-Adapt, Teckenändring, Lagras som DIS/VAR 80 filer
15903094 -JX Seppo Huikuris Sprite-Maker
15903095 EDX Load-Assembler-Bas, omformar Assembler program till Extended-Basic program
15903098 -JX Tommy Erikssons Sprite-Maker
16903003 -EX Miniassembler. Utmaningen 85-2
16903006 -DM Diskkatalog i Textmode
16903011 EAD CSAVE Minnesdumpning till kassett Utmaningen 85-2

3

91 SPEL

01911028H ---- Car driver
01911029H ---- Animals
01911030H ---- Space Invaders
02911001E ---- Master Mind
02911002E --X Prickskytte med kanon mot en kyckling
02911003E ---- Startrek, textadventure
02911006E ---- Othello
02911007E ---- Robotjakt
02911008E ---- 15-spel
02911009E ---- Tärningsspel
02911011E ---- Keno
02911012ED --D Star Guard
03911013E ---- Miner
03911014E --C Yahtzee
03911015E ---- Backgammon
03911016E ---- 3D Tic-tac-toe
03911017E --X Black-Jack II
03911018E -SX Not one
03911019E --X Datorpoker
03911024E ---- Fyra i rad (luffarschack)
04911025E ---- Startrek
04911026E --X Hjälp kycklingen över vägen
04911027E ---- Katapult
04911031E ---- ISOLA. Isolera motståndaren
06911032E ---- Vem skjuter den sista roboten

3

92 UTBILDNING

01921006H --X Arvsanlag
04921001E ---- Enkla räkneövn. med de fyra räknesätten
04921002E ---- Kemifrågor
04921003E --X Relative IQ-test
04921004E ---- Algebra
04921005E ---- Projectile problems, beräkn.-spel
07921007E ---- Träning i dåtid (imperfekt)
10921008E ---- Räkneträning med bråkdelar
13923029 --X Huvudräkning
13923030 --X Glosträning, kassettbaserad
13923031 EDX Byggnadsmekanik
16923009 EDX Glosträning, diskbaserad

96 MUSIK

05962001 --C Johann Strauss. Tales from Vienna woods
05962002 ---- Never on a sunday
05962003 --X Berceuse
05962004 --X Beethoven. Symfoni nr 5
05962005 --X Serenade
05962006D -DX Amazing graze
05962007 --X Beethoven opus 27
05962008D -DX Time in a bottle
05962009D -DX You light up my life
05962010 ---- Bumble-boogie
06962001 ---- The pink panther
06962006 ---- Let me call you sweetheart
06962011D --D Fiddler on the roof
06962012 ---- Bach 3. menuet
07962013 ---- I'm looking through you
08961001E --X Killing me softly
08962014 --X Chopin op 11 ñ 3
12963056 ---- Tremolo
14963033 ---- TI-Keyboard
14963043E ---- When the Saints ...

97 DEMO

01971001H --C Demo av färger, grafik och ljud
06971002E ---- Kaleidoscope
06972004 --X Halloween (sprites)
06972005 --X Sprites
07971003H ---- Demonstration av olika teckenstorlekar
11973070 -EX Demoladdning av Assembler program
13973059 ---- Korsord
14973038 --X Mönstergrafik
15971091 -EX Mg/Text, Textmode
15973003 --M Textmode i TI-Basic
15973004 --M Auto-Sprite i TI-Basic
15973097 ---- Snabbare grafik med TI-Basic
15973099 --X Doppler-effekt
16973010 EAD Tapetest. Utmaningen 85-2

99 ÖVRIGT

01991001E --X Släktforskningsregister
01991002H ---- Utskrift av månadskalender
01992001E ---- Månkalender
07991005H ---- Morse
09991003F ---- Biorythm
10991007H ---- Lotto
11993073 --X Tipsrad
11997005S ---- Slumpning av LOTTO
12993008 ---- Toto
12993010 ---- Tipsservice
12993011 ---- Alkotest
13993021 -PX Etiketter
13993028 --X Biorytm II
14993032 --X Biorytm III
14993041 ---- Resultatlista, slalom
14993049 --X Tips-system
15993078 -EX Automatisk uppringning
15993096 ---- Dagräkning med almanacka