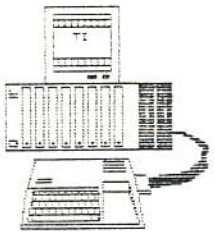
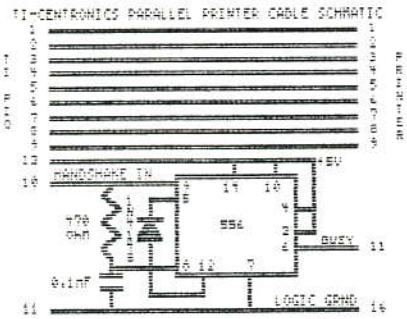
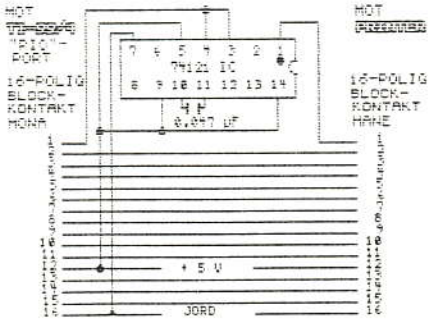
PROGRAM BITEN 90-2







ISSN 0281-1146

REDAKTÖREN

Föreningens årsmöte som hölls den 10 mars valde följande styrelse 1990:

Claes Schibler, ordförande
John Hanssen, kassör
Jan Alexandersson, redaktör
Åke Olsson, sekreterare
Börje Häll, programbankir
Mikael Nordlin
Peter Olsson
Per Rundquist

Vi återkommer med protokoll från årsmötet och med reviderade stadgar i nästa nummer av Programbiten.

PB 90-3 beräknas komma ut under våren 1990 vilket betyder att du bör sända in bidrag och tips inom 14 dagar efter det du fått PB 90-2.

Det finns nu underlag för 6 nummer i år genom att tidningen fylls med 3-4 återkommande artikelserier: McGoverns XB, Tigercub, Swedlow XB. Dessa ger 12 sidor i tidningen. Dessutom kanske det kommer med 3 sidor från Peter Walker så att PB kan ges ut med 6 nummer med vardera minst 12-15 sidor. Om det inte kommer något från medlemmarna kommer Programbiten att ges ut i detta skick.

Det har kommit in mera artiklar nu under 1990 än under hela föra året från följande medlemmar: L-H Andersen, Anders Persson, Bertil Stenfeldt, Arne Wennberg. Bo Arne Östborg har sänt in en kopia av Corcomps RS232/PIO-manual och skriver att Star NL-10 har fungerat bra med kortet. Från utlandet har nya artiklar kommit från McGovern, Tigercub och Swedlow. Dessa artiklar har kommit utan att föreningen beställt eller betalt för flexskivorna.

L-H Andersen, Blegdammen 2 IV, DK-8000 ÅRHUS C, Danmark är även intresserad av TI-74 och kan tänka sig att skriva regelbundet i Programbiten om TI-74. Du som vill läsa om detta sänd in önskemål till redaktören.

Sänd dina bidrag direkt till:
Jan Alexandersson, Springarvägen 5, 3tr, 142 61 TRÅNGSUND.

Redaktör: Jan Alexandersson
Utmanarredaktör: Anders Persson
Programbankir: Börje Häll

Föreningens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9 NB
S-121 46 JOHANNESHOV
Sverige

Postgiro 19 83 00-6
Medlemsavgiften för 1990 är 120:-

Datainspektionens licensnummer:
82100488

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

Övriga annonser kostar 200 kr för hel sida. För lösblad (kopieras av annonsören) som skickas med tidningen gäller 200 kr per blad. Föreningen förbehåller sig rätten att avböja annonser som ej hör ihop med föreningens verksamhet eller ej på ett seriöst sätt gäller försäljning av originalexemplar av program.

Föreningens tillbehörsförsäljning:
Följande tillbehör finns att köpa genom att motsvarande belopp insätts på postgiro 19 83 00-6 (porto ingår)

Användartips med Mini Memory	20:-
Nittinian T-tröja	40:-
99er mag. 12/82, 1-5,7-9/83(st)	40:-
Nittinian årgång 1983	50:-
Programbiten årgång 84-89(styck)	50:-
TI-Forth manual	100:-
Hel diskett ur programbanken(st)	30:-

Enstaka program 5:- st + startkostnad 15 kr per skiva eller kassett (1 program=20kr, 3 program=30 kr).

UTMANINGEN

av Anders Persson

Det var ett tag sedan som något syntes under rubriken Utmaningen. Jag har nästan glömt hur det stavas. Men även om ni kanske har trott något annat, har varken spalten eller dess redaktör gått under jorden. Med dessa låtsasvintrar har den inte ens varit under isen!

Om du har något problem, eller lösningen till ett gammalt, hör av dig till mig. Adressen är ny, vilket också är den huvudsakliga anledningen till den här lilla artikeln. Adr:

Anders Persson
Drottninggatan 35, 2 tr.
341 36 LJUNGBY
Telefon 0372-847 63

Vill ni ha något att fundera på, kan ni ju lämpligen tänka ut något som ökar intresset för föreningen Programbiten, i stället för tvärtom. Att antalet medlemmar minskar torde väl inte ha undgått någon. Det vore intressant med en diskussion om vad som kan göras för att ändra på den tendensen.

I dag finns det ju betydligt mer kapabla maskiner än 99:an, maskiner som dessutom inte kostar en förmögenhet. Det är knappast förvånande att de entusiaster, som är tillräckligt intresserade för att gå med i en förening av det här slaget, ger sig ut på jakt efter nya datorer att studera.

Trots detta tycker jag att vår Texas är en intressant skapelse. Jag håller med Lennart Thelander. Ingen annan dator har lockat till så många glada skratt. Det är naturligtvis en av anledningarna till att en dator, som i rimlighetens namn borde vara utdöd för länge sedan, fortfarande används över hela världen. Hur stort är i dag intresset för ABC 80?

Tyvärr kan inte föreningen vara hur liten som helst, om verksamheten ska kunna kvarstå. De ökade internationella kontakterna som tagits, mycket tack vare Jan Alexandersson, är ett steg på vägen mot en förnyelse. Finns det andra?

FAIRWARE R.A.GREEN

Nya Fairware-skivor från Canada kan fås från programbanken för 30 kr/st.

MACROASSEMBLER VERSION 8 (dec 1989)
(32 kB och EA, XB, TW eller FW)
Ryms på på 3 skivor. Tidigare version bör ej användas eftersom den kan assemblera fel. Detta gäller:

- SBO, SBZ och TB med negativt tal
- rad med längden noll kraschar

Helt nyskriven dokumentation:
- User's Guide (8 sid)
- Assembler Language Reference(37 s)
- Macro Descriptions & Writing(11 s)
- 9900 CPU Reference (10 sid)

Nytt macro-direktiv, $OPCODE, möjliggör definition av nya maskinkodsinstruktioner för t.ex. 9995.

LINKER VERSION 3 (okt 1988)
(32 kB och EA, XB, TW eller FW)
Ryms på 1 skiva. Det finns även en LIBRARIAN-skiva som ej ändrats.

TI WRITER VERSION 4.5
(32 kB och EA, XB, TW, MM eller FW)
Ryms på 1 skiva och har en förbättrad editor. Manual 8 sidor. Formateraren kan ställas in så att följande tecken kan användas fritt, vilket inte gäller TIW och FWB:

- & tecken 38: understrykning
- @ tecken 64: fet stil
- * tecken 42: adresslista
- ^ tecken 94: blanktecken

MULTIPLAN VERSION 4.0 (nov 1989)
(32 kB och modul Multiplan)
Det finns 2 olika skivor:
- Standard-versionen (1 skiva)
- GRAM-versionen (1 skiva)

1000 WORDS

(en bild säger mer än tusen ord) är en ny Fairware-skiva som kan beställas från programbanken. Det är skrivet av Norman Rokke och är ett program som omvandlar TI-Artist-bilder till ett format som kan skrivas ut av formatteraren i TI-Writer.

ATT KRYMPA PROGRAM

av Anders Persson

I ett antal artiklar i tidningen har på senare tid behandlats frågan om att krympa storleken på program, främst assemblerprogram. Med undantag för en artikel av Mikael Nordlin i PB 89-2, tycks den gemensamma nämnaren för dessa artiklar vara att ingen egentligen vet hur man gör.

Att lära sig själv

Datorer är intressanta apparater. För den som ger sig in på upptäcktsfärd i den lustiga värld som finns under skalet i apparaten, är det lätt att bli fascinerad av det man ser. Önskan att få datorn att dansa efter ens egen pipa leder ofta till att man provar sig fram efter egna idéer, så gott det går. När sen den förb... apparaten inte vill ta rättelse, blir idéerna allt vildare. Det ena lilla fixet följer det andra, tills man plötsligt snubblar över något som fungerar, eller åtminstone verkar fungera. En ahaupplevelse som får en att sväva på små moln en stund, eller i alla fall tills nästa bug dyker upp.

Problemet med den här närmast slumpartade strategin är att man i regel hittar lösningar som inte har någon annan bra egenskap än just den att de råkar fungera. Om man inte ställer större krav än så på sig själv, eller, som ofta är fallet för en nybörjare, inte vet vilka krav man borde ställa, blir slutsatsen uppenbar. Allt som fungerar är bra. Fortsättning följer, med upprepande av dåliga lösningar. "De fungerade ju förra gången."

Att bita sig i svansen

När ens samlade erfarenheter växer, gapar man över allt större uppgifter. Ju större programmen blir, desto värre blir det att hålla ordning i en kod, som redan från början är full av nedärvda fix. Antalet fix tenderar dessutom att öka proportionellt med kvadraten på kodens storlek, eller något åt det hållet i alla fall. Eftersom de flesta fix tar plats (bortsett från de som består av att man raderar allt som orsakar felmeddelanden), uppnår man till slut en övre gräns för hur stort ett program kan bli. Varje nytt fix minskar möjligheten att överblicka koden, vilket leder till att nya fix behövs, och så vidare. Knappast ett trevligt ekorrhjul att fnatta runt i. Det är bara att hoppas att datorns minne är litet, så att pinan blir kort.

Genom att man inte redan från början har en god och systematisk ordning i sitt program, har resultatet blivit det berömda spagettisyndromet. Programmet är så tillkrånglat att dess funktion kräver ständiga modifikationer, vilket med tiden leder till att minnet inte räcker till, varvid nya, mer eller mindre smarta, modifikationer behövs, och nya fel införs.

Att börja i rätt ände

Det gäller alltså att se till att man inte behöver ge sig på ett färdigt program, som inte längre får plats, och med knipsluga metoder eliminera enstaka bytes här och var. Detta oftast till priset av att programmet blir mer och mer obegripligt. Vad ska man då göra i stället?

Den bättre vägen heter planering, vilket är det samma som konsten att tänka efter före. Ett program som utsatts för planering under sin tillkomst, fungerar inte sämre för det. Dessutom blir det inte heller större. Detta beror helt enkelt på att det inte finns så mycket i ett sådant program, som inte är avsett för det som programmet faktiskt gör. Framför allt arbetar programmet med en datastruktur som på ett eller annat sätt är optimalt anpassad till uppgiften. En väl avpassad datastruktur är nog det mest värdefulla man kan skaffa sig. Motsatsen gäller också. Något mer tungarbetat än en dålig datastruktur får man leta efter.

Hur väljer man då rätt datastruktur? Ja, vad som är "rätt" kan naturligtvis oftast diskuteras. Dessutom är det så, att man inte alltid kan ha en datastruktur som är optimal för alla de operationer som man kan önska sig, men den kan vara en bra kompromiss. (Här ska det erkännas att jag är ute på tunn is över djupt vatten. Det påstås nämligen att det inte finns några bra kompromisser. En kompromiss är bara en lösning som gör alla inblandade parter lika mycket förbannade.)

Men nu får vi skaka av oss den prosaiska verkligheten, och se på några bra teorier för hur det kan gå till. Att välja en bra datastruktur är nämligen inte alltid så lätt.

Först, vad menar jag med datastruktur? Ja, som framgår av namnet är det strukturen på den information som man önskar behandla. Allt som finns i en dator är abstrakt. Vi skapar oss symboler för den verklighet vi vill manipulera på något sätt. Detta av den fullgoda anledningen att vi inte kan ta med oss verkligheten in i datorn. Ska vi skaffa oss ett register över bilar, blir det problem så snart vi försöker köra in Volvon i 99:an. Möjligen finns det någon liten japansk plåtburk som går in, men jag tvivlar på det. Vi får i stället representera bilen på något annat sätt, med en lämplig symbol. Hur den symbolen ska se ut, beror på vad vi vill göra med den. Vill vi skriva en lista på bilmärken kan vi lagra texten "VOLVO" som en textsträng. Men om vi vill räkna ut medelvärdet av bilarnas pris, behöver vi i stället lagra talet 175000.

Datastrukturen är alltså en fråga om vilken information vi lagrar i datorn. Men det är också i minst lika hög grad en fråga om hur delarna av denna information är knutna till varandra. Om vi först önskar en lista över olika bilmärken, och sedan, när vi funnit ett som vi är intresserade av, vill se vilka modeller som finns, och slutligen vilka varianter, har vi plötsligt nya krav på vår datastruktur. En hierarkisk struktur kan kanske vara lämplig. Den kan se ut så här.

Andra <------ VOLVO ------> märken

440	480	740
GL	ES	GL
GLT	Turbo	GLE
Turbo		GLT
		GLT 16V
		Turbo

Ser du vad strukturen liknar? Det är ett träd, fast vänt upp och ner. Trädstrukturer är vanliga i många program. Sambandet mellan de olika delarna av trädet kan exempelvis implementeras med hjälp av pekare.

Det har, under de egentligen ganska få år som datorer liknande dagens har funnits, skrivits massor om det här. Framför allt på senare tid. Läs gärna någon sådan bok. Om du inte har någon till hands, kan du försöka med Utmaningen i PB 84-4. Där finns ett resonemang kring uppbyggnaden av en datastruktur för ett registerprogram. Den kan fungera som en god illustration till ett tillvägagångssätt.

Nu är ett gott resonemang i den stilen naturligtvis inte en garanti för att man kommer fram till det bästa svaret. Men ju mer du försöker, desto mer lär du dig av erfarenheterna. Dessutom kan du ju lära dig av resultaten i böckerna också. Men bekymra dig inte bara om hur du ska lagra det du vill, utan även om vad du egentligen behöver lagra. Tänk på det engelska uttrycket "Information you use, data you store".

Söndra och härska

När man väl har bestämt sig för hur informationen ska lagras, återstår det som var problemet från början, nämligen vad man ska göra med den. Här gäller i hög grad att man inte ska ta sig allt vattnet över huvudet samtidigt. Dela upp uppgiften i smådelar. Delarna ska vara tillräckligt små för att du ska kunna överblicka varje del för sig. Kan du inte dela upp programmet så här, är det kris. Men det är inte ditt programmeringskunnande det är fel på. Du har bara inte förstått uppgiften i sig. Alla uppgifter består av mindre delar.

För att göra uppdelningen behöver du inte veta ett skvatt om programmering. Det behövs inte förrän delarna ska implementeras på något sätt.

Hur små delarna ska vara beror inte bara på uppgiften. Det beror även på programmerarens skicklighet. Ofta kan man se klart avgränsade delar i uppgiften, delar som dock är alltför stora i sig. Då får man dela delarna, eventuellt i flera steg, tills de är tillräckligt små för att man ska kunna hantera dem.

En annan sak att iakttaga är kommunikationen mellan delarna. Ju mindre information som behöver skickas mellan de olika delarna, desto bättre har du lyckats dela upp programmet i logiskt avgränsade delar. Detta minskar också risken för att en del ställer till med problem för en annan. Därmed minskar behovet av mindre välbetänkta fix, och följaktligen graden av elände.

Det är bra att börja med att rita lite skisser över de olika delarna, och försöka tänka sig vilken information som behöver sändas mellan dessa delar. Verkar det svårt? Ge då upp. I dag. Men fundera vidare på lediga stunder. I morgon är en ny dag. Kom ihåg att lite vändor före eliminerar mycket vända i slutänden.

När inget annat hjälper...

Nu är verkligheten tyvärr sådan att man långt ifrån alltid vet tillräckligt mycket om en uppgift, när man börjar på den. Eftersom man inte vet att man inte vet allt, går det ju inte heller att vänta på att vetandet ska komma. Man vet ju inte att man har något att vänta på. Men när man har skrivit på programmet en tid, brukar det bli uppenbart hur det skulle ha sett ut från början. Lite arg på sig själv konstaterar man att "om jag bara hade vetat detta innan, då skulle jag gjort det rätta".

Gör det då. Bara för att du har fått fan i båten, behöver du ju inte ro honom i land. Tippa honom i sjön i stället. Släng programmet i papperskorgen, och börja om, nu när du vet precis hur det ska gå till. Nu är han helt tokig, tänker du kanske. Slänga det som jag jobbat fram med sådan möda? Och alltihop!

Om du tänkte så, ska du tänka två minuter till, innan du läser vidare. Du hade ju redan från början delat upp programmet i olika delar. Gjorde du det någorlunda vettigt, och det brukar vara lättare, behöver du förhoppningsvis inte ändra på alla delarna nu. Släng de som besvärar dig, och behåll det som är bra. Resultatet blir betydligt bättre. Dessutom tar det antagligen inte längre tid heller, eftersom du med ett Alexanderhugg har befriat dig från en massa framtida trubbel.

Strukturerad programmering

Ett modernt uttryck. Många bokstäver (25 stycken faktiskt). Nästan som "hokus-pokus". Vad är det då? Ja, att det är betydligt mer än en snygg kod, med kommentarer med många asterisker i, det har du nog insett.

För det första: Tänk efter vilken information du behöver för att kunna lösa det problem som du har.

För det andra: Tänk efter hur de olika bitarna av information ska vara kopplade till varandra.

För det tredje: Tänk efter vilka olika delar som din uppgift kan separeras i, och hur dessa kommunicerar. Delarna kan även ha gemensamt intresse i operationer på informationen, operationer som kan bli självständiga underprogram. På så sätt sparar du på minnet i maskinen också.

För det fjärde: Skriv delarna. (Nu är det alltså dags att göra något.) Testa de enskilda modulerna så långt det är möjligt. Det brukar bli en del att fundera över när de ska trivas ihop ändå.

För det femte: Gläds åt ditt nya program, som både fungerar bra, eftersom det är väl genomtänkt, som är lätthanterligt när du vill lägga till något nytt, eftersom det är väl genomtänkt, och som inte behöver en massa dumma fix, eftersom det är väl genomtänkt.

ATT BYGGA UT DIN TI-99/4A

av Jan Alexandersson

Du bör först skaffa en Extended Basic-modul. Det finns även en förbättrad modul från Triton nämligen Super Extended Basic.

Nästa stora steg är att skaffa en expansionsbox med plats för 8 kort. Du kan försöka att få tag i en begagnad genom att annonsera i Programbiten. Den bör alltid bestyckas med minst fyra kort:
- Interface-kort till 99/4A
- Expansionsminne 32 kbytes
- Diskkontrollkort
- RS232/PIO-kort

DISKKONTROLLKORT

Det vanligaste är att ha ett TI original-kontrollkort med en fullhöjds Shugart 90 kbytes (360 sektorer) flexskiveenhet. Du bör allvarligt fundera på att byta ut skivenheten mot två halvhöjds dubbelsidiga enheter. En ny TEAC-flexskiveenhet (klarar både DS/SD och DS/DD) kostar nu 690 kr + moms i Sverige. Du kommer då att få 2x180 kbytes d.v.s. fyra gånger större kapacitet än tidigare. Om du har en fristående (standalone) kontrollenhet så klarar den endast enkelsidigt d.v.s. 90 kbytes är maximum även om du byter flexskiveenhet. Alla kontrollkort i expansionsboxen klarar dock dubbelsidigt.

Du behöver även en Y-kabel för kraft som enkelt kan tillverkas (med lödning) med delar som finns hos ELFA m.fl. De skruvhål som finns i expansionsboxen passar för den första halvhöjdaren som sätts till vänster. Den andra till höger kan t.ex. förankras genom att en tunn aluminiumplåt med fyra skruvhål används, som täcker båda enheterna med två skruvhål för vardera. Skruva fast plåten i de två hålen som finns i den högra flexskiveenheten. Skjut sedan in de två enheterna och skruva fast de två skruvarna till vänster så att box, aluminiumplåt och flexskiveenhet sitter stadigt.

För dataöverföring behöver du en flatkabel med kontakter till de två flexskiveenheterna. Du kan enkelt göra den själv utan lödning med hjälp av ett skruvstäd som pressar fast kontakterna på kabeln. I första hand bör du ha en gemensam kabel för de två enheterna som även anslutes till kontakten för interna enheter.

Du måste även ändra byglingen på de två flexskiveenheterna så att de blir DS0 och DS1 vilket motsvarar DSK1 och DSK2. Det kan finnas en resistor-pack som skall tas bort från den enhet som sitter närmast kontrollkortet, medan den som sitter längst bort behålles. TEAC använder större resistanser så dessa kan ej plockas bort, men de kan parallellkopplas från flera enheter utan problem.

DOUBLE DENSITY

Det finns även nya kontrollkort från Corcomp och Myarc som har möjlighet att styra 4 st DS/DD-enheter (1440 sektorer) så att du får 4x360 kbytes. För Corcomp-ägare finns det en modifierad PROM från MG. Myarc finns även i en variant med 720 kbytes DS/QD (2880 sektorer).

TOM PC XT-BOX

Jag har själv en extra PC XT-Box med tre flexskiveenheter och en hårddisk (plats för 4 halvhöjdare). En tom sådan PC XT-Låda kostar idag 390 kr + moms. Nätaggregat för 150 W kostar 490 kr + moms. Dessa ger mer än 15 A från +5V och mer än 4,2 A från +12 V (värdena gäller 135 W-modellen). Detta är troligen tillräckligt för 4 flexskiveenheter och 1 hårddisk.

REFERENSER

PB 85-3: Corcomps diskkontrollkort
PB 88-1: Myarcs diskkontrollkort
PB 88-2: Att minnas med magnetism
PB 88-2: Kraft till separat drive
Micropendium dec 1988: Disk drivin'
Micropendium jan 1990: Expanding TI

DATAFILER UTAN TÅRAR - 1

av Peter Walker, TI*MES, England

Denna artikel syftar till att förklara mysterierna med datafiler, som inte är bra täckta i bruksanvisningen (User's Reference Guide) och som kan verka mycket förvirrande för nybörjaren. Även om du bara har grund-datorn och kassettspelare, kommer jag att förklara hur du kan använda datafiler.

Du kanske har läst bruksanvisningen och vadat genom information om filprocessering, men konfrontationen med valet mellan INTERNAL/DISPLAY, FIXED/VARIABLE, RELATIVE/SEQUENTIAL kanske har gjort dig undrande:
- När ska jag använda de olika?
- Vilka är dess fördelar?

Om detta verkar intressant så är denna artikel för dig.

VAD ÄR EN DATAFIL?

En datafil är en följd av information som lagras på något media som t.ex. kassettband eller flexskiva. En sådan fil är oberoende av det BASIC-program som läser och skriver till filen, och är helt annorlunda än DATA-satserna som används inom ett program, vilka inte kan ändras av programmet. Alla externa enheter till datorn används inte för datalagring, men det sätt som datorn använder för att kommunicera med dessa är lika både för enheter med och utan lagring. Filer används således både för datalagring och för att sända information till skrivare och modemer m.m. De yttre enheter (device) som fungerar med TI-99/4A är:

CS1 och CS2	Kassettspelare (2)
DSKn (n=1-3)	Flexskiveenheter (3)
RS232, RS232/2	Serieportar (2)
PIO	Parallellport
SPEECH	Terminal Emulator II
ALPHON	Terminal Emulator II
MINIMEM (4kB)	Mini Memory
EXPMEM2 (24kB)	Mini Memory +32kB EM

Uppenbarligen är endast ett fåtal av dessa enheter användbara för datalagring. Med hänsyn till det stora antalet enheter som datorn kan komma att behöva kommunicera med och att 99/4A behöver tilldela en minnesbuffert för varje enhet, så öppnas filerna med OPEN vid kommunikation och stänges med CLOSE när detta är avslutat. Detta begränsar det buffertutrymme som användes. När filen öppnas så tilldelas den ett nummer mellan 1 och 255 som sedan används när datorn ska skriva och läsa filen. Emellertid kan endast tre filer öppnas samtidigt om inte CALL FILES(N) användes innan programmet startas. N är det maximala antalet filer som samtidigt kan öppnas, upp till max 9 (CS1 och CS2 kan anropas utan hänsyn till CALL FILES ö.a.)

HUR ÄR EN DATAFIL ORGANISERAD?

Som när det gäller all datainformation, är datafiler en ström av binära ettor och nollor. Dessa grupperas i 8-bit "bytes" så att varje byte kan anta 256 olika värden. Dessa bytes kan användas för att representera tecken med användning av ASCII-kod, eller kombineras på andra sätt för att representera flyttal. En datafil kan bestå av inget annat än en ostrukturerad ström av bytes. Sådana filer kallas PROGRAM-filer eftersom detta är det sätt på vilket BASIC-programmen själva lagras med SAVE. PROGRAM-filer kan ej kommas åt från BASIC för lagringsändamål, även om moduler som Personal Record Keeping (PRK) använder denna form av datafil. Alla filer som kan kommas åt av BASIC har varierande struktur så att filen delas i ett antal poster (records), vilka var och en innehåller ett eller flera element (items). Det bör påpekas att bruksanvisningen inte förklarar skillnaden mellan poster och element på ett bra sätt. Jag hoppas att jag kan göra detta på ett klarare sätt. Om vi antar att datafilen består av en lista av namn och adresser, som vi vill dela så att varje namngiven person får en post och inom posten finns det flera element som förnamn, efternamn, gatuadress, postadress o.s.v. Om filen å andra sidan innehåller skriven text,

så är det praktiskt att välja en textrad i varje post. Det är normalt inte meningsfullt att ytterligare dela raden (t.ex. ett ord per element) så text har vanligen endast ett element per post d.v.s. den kompletta raden. När du vill lagra information i datafilen, måste du först strukturera din fil: vad ska varje post bestå av, hur många element, vilken typ av element (numerisk/strängtyp), och hur långt ska varje element vara. När du öppnar en fil, kräver BASIC att du specificerar några av dessa attribut i ett OPEN-kommando. Om du inte gör det så kommer vissa förutbestämda (default) attribut att användas som kan vara helt tillräckliga i många fall. Låt oss nu titta på OPEN-kommandot och vilka filattribut som används.

ATT ÖPPNA EN FIL MED OPEN

Formatet för OPEN-kommandot är:

```
OPEN #filnummer:device
[,posttyp][,filtyp]
[,filorganisation]
[,öppningssätt]
```

Attributen inom hakparentes är frivilliga (default-värden om de utelämnas) och kan skrivas i godtycklig ordning. Om du t.ex. öppnar en fil mot en parallell printer-port för 80 kolumnsutskrift, är det endast nödvändigt att använda OPEN #1:"PIO". Default-värdena DISPLAY och VARIABLE 80 passar här.

ATT LÄSA OCH SKRIVA TILL EN FIL

Enbart för att förvilla användes BASIC kommandot INPUT för att läsa från en fil och PRINT för att skriva till en fil. Dessa är samma kommandon som används för inmatning från tangentbordet och för att skriva på skärmen. Skillnaden är att filnumret anges när det gäller filanvändning:

```
PRINT #1:"This goes onto a file"
```

En intressant användning är att om fil #0 anropas gäller tangentbord/skärm på samma sätt som om inget filnummer skrivits ut. Detta är användbart när programmet ska kunna välja utskrift mellan skärm och printer. Fil #0 är alltid öppen och kan inte stängas.

Låt oss nu titta närmare på attributen i OPEN-kommandot.

DEVICE (även kallat filnamn)

Detta motsvarar de externa enheter som finns i listan ovan, som CS1, MINIMEM o.s.v. På grund av den speciella flexibiliteten som finns för disk-filer så måste varje fil ha ett namn, vilket gör att device-namnet har filnamnet som ett tillägg d.v.s:

DSK2.ADDRESSES för en fil med namnet "ADDRESSES" på flexskiveenhet 2.

För alla andra enheter används endast device-namnet även om vissa andra inställningar för speciella enheter vid behov också skrives efter device-namnet. PIO.LF är t.ex. namnet för parallellutgången mot printer när den automatiska radmatningen skall undertryckas.

PERMANENTA ATTRIBUT

Posttyp och filtyp är de enda permanenta attribut hos en fil och dessa är bestämda när filen en gång skapades. I motsats till detta är filorganisation och öppningssätt endast giltiga för det sätt på vilket man kommer åt filen varje gång den öppnas med OPEN, och kan vara olika vid varje tillfälle om så önskas.

POSTTYP (record type)

Denna kan vara FIXED eller VARIABLE. FIXED-poster har alla samma längd, medan VARIABLE-poster kan variera i längd upp till ett angivet maximum (t.ex. VARIABLE 96) eller default-maximum som är 80 för flexskiva, RS232 och PIO medan det är 64 för kassettfiler. Om vi tar vårt ovanstående exempel med en namn/adress-fil och en textfil, är det klart att textraderna varierar i längd och detta gäller även summan av element i namn/adress-filen. Genom att använda poster med VARIABLE längd är det möjligt att spara utrymme på lagringsmediat eftersom det inte blir några mellanrum mellan efter-

dast skriva. Genom att använda UPDATE kan du både läsa och skriva (en skrivskyddad fil kan ej öppnas med UPDATE ö.a.). Med APPEND kan du endast skriva till slutet av filen. P.g.a. skäl som är något oklara, kan APPEND endast användas med VARIABLE-filer vilket något begränsar dess användbarhet. Jag gissar att skälet till detta är att FIXED-filer inte använder "End of file offset" i katalogen för att peka ut slutet av den lagrade informationen. De som inte har ett flexskivesystem kommer också att bli besvikna över att kassettspelare endast kan använda INPUT och OUTPUT. En kassett kan endast spela av eller spela in - inte båda samtidigt.

FILORGANISATION

Denna kan vara RELATIVE eller SEQUENTIAL. Eftersom SEQUENTIAL är "default"-inställningen behöver du aldrig skriva ut detta. Med SEQUENTIAL användning menas att efterföljande INPUT eller PRINT bearbetar filens poster i nummerordning. I motsats till detta, kan du med RELATIVE organisation (som kräver FIXED-poster) läsa och skriva från angiven post. T.ex.kommer:

```
PRINT #1,REC 5:A
```

att skriva värdet av A till 5:e posten i filen eller snarare den 6:e eftersom den första posten är REC 0. När är dessa två former av filanrop nyttiga?

När man uppdaterar eller bearbetar en fil, finns det två sätt på vilket informationen kan användas. Det första sättet är att öppna filen och läsa hela innehållet till en fältvariabel (vektor eller matris). Ett antal vektorer eller en tvådimensionell matris kan användas om det finns flera element per post. När all nödvändig bearbetning är klar, skrives hela fältvariabeln tillbaka till filen. Genom att på detta sätt ladda in alla data till RAM-minnet kan bearbetningen utföras ganska snabbt. Men om ditt program är stort och datafilen är skrymmande, kan du enkelt överskrida tillgängligt internt minne. I detta fall används det andra sättet att bearbeta en fil med hjälp av RELATIVE filorganisation. Istället för att ladda in hela filen till minnet, läses varje enskild post med REC när den ska uppdateras eller bearbetas och skrives tillbaka när detta är klart. Detta kräver uppenbarligen mindre minne men har nackdelen med den fördröjning som blir varje gång en post läses med INPUT eller skrives med PRINT. Att använda REC-nummer liknar indexerade element i en vektor. För att visa detta låt oss titta på de två metoderna att leta genom en fil för att hitta en post med strängen "Kermit" och lägga till så att den blir "Kermit the Frog".

```
 10 REM METOD 1
 50 DIM A$(50)
100 OPEN #1:"DSK1.FILE",INTERNAL,
    FIXED 40
110 FOR I=1 TO 50
120 INPUT #1:A$(I)
130 NEXT I
140 FOR I=1 TO 50
150 IF A$(I)="Kermit" THEN 1000
160 NEXT I
---
```

I detta enkla exempel skulle de två FOR-TO-NEXT slingorna kunna ha kombinerats, men jag ville visa den allmänna principen att ladda hela filen till en vektor.

```
---
1000 A$(I)=A$(I)&" the Frog"
1010 RESTORE #1 ! Detta återställer
     filpekaren till början av filen
1020 FOR I=1 TO 50
1030 PRINT #1:A$(I)
1040 NEXT I
1050 CLOSE #1
```

```
 10 REM METOD 2
100 OPEN #1:"DSK1.FILE",RELATIVE,
    INTERNAL, FIXED 40
110 FOR I=1 TO 50
120 INPUT #1, REC I:A$
130 IF A$="Kermit" THEN 1000
140 NEXT I
---
1000 A$=A$&" the Frog"
1010 PRINT #1, REC I:A$
1020 CLOSE #1
```

Eftersom kassettspelare bara kan gå framåt, kan du inte använda RELATIVE-filer med kassett. Flexskivor kan använda detta sätt att komma åt filen eftersom positionen och sektorn för varje post (med FIXED) kan lokaliseras och läsas individuellt.

MCGOVERNS XB-SKOLA, DEL 3

av Tony McGovern, Australien

Vårt nästa exempel kommer att vara en bra början på ett icke trivialt nyttoprogram för att skriva ut programlistning för TI BASIC eller XB på en 80-kolumnsskrivare med två spalter bredvid varandra med bibehållande av det vanliga formatet från skärmlistning. Om du bara gör LIST "RS232.BA=...." kommer datorn att skriva ut det i DIS/VAR 80 format och det är upp till dig att bestämma hur skrivaren ska behandla detta. Något som liknar skärmformatet kan endast fås (med extra förbrukning av papper) med indragna skrivarmarginaler. 80-kolumnsutskrift imponerar ej så låt oss göra det bättre. Om du saknar skrivare eller flexskiveenhet så kommer denna lektion inte att vara omedelbart användbar, men kommer ändå att vara ett bra exempel att gå igenom som en programmeringsövning. Vi kan lika gärna göra något användbart.

Först tar vi reda på vad som behöver göras, och arbetar fram ett antal procedurer som vid behov kan anropas med CALL. Själva programmet kommer endast att göra minsta möjliga för att utföra arbetet ordentligt. Pukor och trumpeter kan läggas till senare. På ett par ställen ska vi förbereda för att lägga till extrafinesser (pukor och trumpeter har inget med tal att göra) med tomma underprogram som kan fyllas i senare. För en bra beskrivning av användningen av detta se den utmärkta boken "Inside Basic Games" av R.Mateosian. De detaljerade exemplen på programkodning i denna bok är för Apple eller Trash-80 Basic (kommer någon fortfarande ihåg vad en TRS-80 var ?), men Mateosian utvecklar idéer på ett sätt som ännu mer stämmer med ett TI XB:s SUB-programs realisering än med dessa mindre kapabla Basic-dialekter.

Så låt oss börja konstruera vårt program genom att bestämma vad vi vill att det ska göra. Vi vill att utskriften ska formateras snyggt på sidan med övre och undre marginaler, i två spalter med skärmformat (28 tecken/rad). Flera kolumner (om skrivaren kan hantera detta) är inget problem -- när du väl kan räkna till 2 är 3 lätt. Basic-rader får inte delas mellan två spalter eller mellan två sidor. Några saker som ofta uppträder i utskrifter, som indragning av FOR-NEXT-loopar passar inte alls med flera instruktioner per rad i XB (men kanske med TI Basic utskrifter) så vi kommer inte att beakta detta här. Å andra sidan förbättras läsbarheten av XB-listningen avsevärt om en blankrad stoppas in före REM eller SUB, utan att man gör våld på idéen att det motsvarar skärmlistningen. Att lägga till sidnumrering är ingen större vinst (ett XB-program från grunddatorn kan bara fylla 6 sidor).

Vi antar att program-listningen som ska skrivas ut kommer från en diskfil som DSK1.LIST som tidigare skrivits dit med LIST "DSK1.LIST". En enkel svårighet som tas omhand är den första tomma raden som skrivs av LIST. Det verkliga problemet är att LIST inte bryr sig om att bevara XB-raderna som en bestämd enhet. Varje XB-rad startas som en separat post (record) och om den är mindre än 80 tecken lång så bevaras den odelad. XB-rader kan utan vidare utsträckas till två utskriftsrader eller mer (mycket mera sällan för Basic-rader), men LIST placerar inga markeringar för att visa vilken utskriftsrad som innehåller starten på XB-raden. Så om vi ska uppfylla våra krav på att XB-rader ska behandlas precis som på skärmen, krävs det något mera avancerat än ett enkelt LINPUT. Då är ett av våra viktigaste byggblock identifierat --- SUB BASICLINE(...).

Varje nyttoprogram behöver titel- och hjälpskärmar så det finns SUB TITLES för att hålla alla dessa detaljer borta från att skräpa ned huvudprogrammet. Programmet behöver även SUB OPTIONS(...) för att behandla inmatning av fil- och printer-namn och val av de erbjudna utskriftsalternativen.

Nu är den verkliga kärnan i program-

met det sätt på vilket det måste sammanställas till en hel sida innan något skrivs ut eftersom radmatning bara går framåt (åtminstone gör det på vår TI-99 skrivare). Så vi behöver SUB PAGEBUFFER(...) för att ta utmatningen från BASICLINE, hugga sönder det i skärmformatsblock och bestämma var dessa ska placeras på sidan. Sedan behöver vi SUB PRINTPAGE(...) för att forma dessa och sända iväg dem till skrivaren. Detta är alla SUB-program som anropas direkt från huvudprogrammet, och det som återstår är att bestämma initialiseringen -- DIM, förutbestämda filnamn o.s.v. och att skriva logiken för programflödet.

Innan vi börjar skriva någon kod måste vi bestämma vilka SUB-program som ska anropas av de som redan tidigare har definierats. Eftersom listningen skrivs i spalter så är SUB WRITECOL(...) en bra kandidat för upprepad användning, och SUB WRITEPAR(...) för att ta en BASIC-rad och returnera den upphuggen i 28-teckensrader till WRITECOL. Eftersom BASICLINE hämtar inputposten är detta det lämpliga stället att känna av slutet av filen (EOF). Vi kan även använda PRINTPAGE för att sudda ut bufferten innan en ny sida skrivs ut.

Låt oss skapa inmatning av filnamn och Ja/Nej-svar som SUB FILENAME (...) och SUB YN(...), med SUB MORE(...) för att avsluta alltihop. Andra nyttiga SUB-program som kommer att tas med är SUB TXTCOL(...) för att ändra skärmfärgerna i ett anrop, SUB KEYCON för att bära bördan av "press any key to continue", och SUB DELAY(...) är alltid bra att ha tillhands.

Detta avslutar uppsättningen av de nödvändiga procedurerna för att göra programlistningen, och nu startar detaljkodningen efter någon eftertanke när det gäller kedjan av parameteröverföringar. Principen att du ska planera uppifrån (top down) och skriva koden nedifrån (bottom up) gäller lika väl för Extended Basic som när det gäller TI-LOGO och TI-FORTH där språkets utformning gör det svårt att göra annorlunda. SUB-program gör det möjligt att enkelt gå samma väg i XB. Mindre kapabla Basic-dialekter gör det mycket svårare att hålla ihop dina tankar och hålla koden på det raka spåret.

Själva programmet kommer nu att listas bit för bit med noggranna kommentarer. Listningen har överförts till denna TI-Writer fil från ett fungerande program som använde en mera utarbetad version. Det följande programmet är faktiskt en förenklad version av det ursprungliga programmet, men är tillräckligt kraftfullt för att göra ett nyttigt arbete.

```
100 REM ** SIMPLIST **
110 REM * PRINTER LIST *
120 REM ** FROM DISK **
130 REM -FUNNELWEB FARM-
140 OPTION BASE 1 :: DIM PRL
N$(66,2)

150 REM * DEFAULT VALUES *
160 CALL TITLES :: SFIL$="DS
K1.LIST" :: PDEV$="RS232.BA=
4800"
170 CALL KEYCON
```

Den första delen av huvudprogrammet som visas ovan, ställer in "default"-värdena och DIMensionerar fältvariabeln PRLN$ för två spalter på vardera 66 rader. Några rader upptill och nedtill lämnas blanka så att sidformatet erhålles utan sändning av kontrollkoder till skrivaren. Vi utgår från att vi har en skrivare med 66 rader/sida (i Sverige används normalt 72 rader/sida, vilket man enklast kompenserar för, genom att stega fram tre rader på skrivaren både före och efter det att 66-raderssidan skrivs ut ö.a.).

```
180 REM * New File Entry *
190 CALL OPTIONS(SFIL$,PDEV$
):: ENDFILE=0 :: LINPUT #1:N
EW$

200 REM * New Page Entry *
210 CALL PAGEBUFFER(PRLN$(,)
,ENDFILE)
220 CALL PRINTPAGE(PRLN$(,),
PDEV$):: IF ENDFILE=0 THEN 2
10

230 REM * End OR Next *
240 CLOSE #1 :: CLOSE #2 ::
CALL MORE(NM):: IF NM THEN C
ALL SPEAK("GOODBYE"):: GOTO
250 ELSE 190
```

```
250 STOP
```

OPTIONS returnerar fil- och devicenamn som har matats in där, och resten av rad 190 återställer markeringen av slutet på filen, och slänger första raden i listfilen. Vid ny sida fylls sidbufferten och skrivs sedan ut upprepade gånger tills den har slut på listning, och frågar sedan om du vill avsluta. Detta är allt som finns i huvudprogrammet. Och nu till SUB-programmen som gör allt arbete.

```
260 SUB TITLES
270 CALL CLEAR :: CALL SCREE
N(11):: DISPLAY AT(12,6)BEEP
:"PRINTER LISTING"
280 SUBEND

290 SUB OPTIONS(S$,P$):: DIS
PLAY ERASE ALL :: CALL TXTCO
L(16,5)
300 CALL FILENAME(1,2,"Edit
as needed and ENTER","N?")
310 CALL FILENAME(4,4,"Sourc
e file for listing",S$)
320 CALL FILENAME(8,4,"Print
er devicename",P$)
330 CALL YN(" Change mind ?"
,"N",22,5,I):: IF NOT(I)THEN
CALL HCHAR(22,1,32,64):: GOT
O 300
340 DISPLAY ERASE ALL :: IF
S$="" OR P$="" THEN DISPLAY
AT(1,2)BEEP:"NO INPUT/OUTPUT
POSSIBLE" :: CALL DELAY(500)
:: GOTO 300
350 OPEN #1:S$,DISPLAY ,INPU
T ,VARIABLE 80 :: OPEN #2:P$
,DISPLAY ,OUTPUT,VARIABLE 80
360 SUBEND
```

TITLES är här litet mer än en tom snutt, men du kan fylla ut den efter eget tycke. OPTIONS hämtar filnamnen, gör några kontroller, och öppnar filerna.

```
370 SUB PAGEBUFFER(PRLN$(,),
EFL)

380 REM * New Col Entry *
390 PLN=6 :: COL=COL+1 :: IF
 COL>2 THEN COL=0 :: SUBEXIT
E LSE PRINT "":"** Reading c
olumn #";COL:"":""

400 REM * New Para Input *
410 IF EFL THEN PRINT "":" *
":"*** END of FILE ***":" *"
:"" :: SUBEXIT ELSE CALL BAS
ICLINE(NEW$,EFL):: PRINT NEW
$:""
420 CALL WRITECOL(PLN,COL,PR
LN$(,),NEW$)
430 IF NEW$="END of COL" THE
N 390 ELSE 410
440 SUBEND
```

Starten på ny spalt i PAGEBUFFER återställer radräknaren PLN till toppen av sidan med en marginal, stegar fram spalträknaren, och återvänder till huvudprogrammet när sidan är full. Om detta inte är fallet begär den att BASICLINE ska hämta en ny programrad och WRITECOL matar in den i sidbufferten. Om BASICLINE meddelar att den har läst sista raden hoppar den tillbaka till huvudprogrammet som får bekymra sig om detta, i annat fall tar den ytterligare en Basic-rad eller börjar en ny spalt. En snutt här CALL SKIPLINE(NEW$,SK) kunde ha använts.

```
450 SUB BASICLINE(N$,E)
460 N$="" :: IF NX$="" THEN
LINPUT #1:NX$
470 N$=N$&NX$ :: IF LEN(NX$)
<80 OR EOF(1)THEN NX$="" ::
E=EOF(1):: SUBEXIT ELSE LINP
UT #1:NX$
480 PX=POS(NX$," ",1):: IF P
X<2 OR PX>6 THEN 470
490 P=POS(N$," ",1):: IF PX<
P THEN 470
500 NR=-1 :: FOR I=1 TO PX-1
:: C=ASC(SEG$(NX$,I,1)):: N
R=NR AND C>47 AND C<58 :: NE
XT I :: IF NOT(NR)THEN 470
510 IF SEG$(N$,LEN(N$),1)="
" THEN 470
520 IF VAL(SEG$(NX$,1,PX-1))
<VAL (SEG$(N$,1,P-1))THEN 47
0

530 REM ** Check Quotes
540 N Q,I=0
550 I=POS(N$,CHR$(34),I+1)::
 IF I THEN NQ=NQ+1 :: GOTO 5
50 ELSE IF NQ<>2*INT(NQ/2)TH
EN 470
560 SUBEND
```

Proceduren BASICLINE som återvinner kompletta Basic-rader från LIST-filen är den enda delen av programmet som har ett tillräckligt komplext beslutsflöde så att det i förväg är nödvändigt med fristående

planering med papper och penna. Jag tänker inte upprepa detta här, men du kan arbeta fram din egen och se om det leder till liknande kod. Problemet uppstår när proceduren har läst in en rad som är exakt 80 tecken lång. Kommer nästa list-post då att vara en fortsättning av samma Basic-rad eller är det början på en ny Basic-rad? Denna svårighet kan inte försummas om skärmlistningens format ska bibehållas eftersom 80 inte är jämt delbart med 28. Proceduren har en följd av tester som var och en undersöker om posten ska läggas till som en fortsättning av den tidigare Basic-raden. Ytterligare några sällsynta fall kan testas enligt samma principer. Det finns åtminstone ett osannolikt fall (som jag känner till) vilket BASICLINE inte ens teoretiskt kan klara av. Kan du upptäcka det? Det tycks fungera ganska bra trots detta. Den invecklade inmatningskoden behövs eftersom en VARIABLE-fil endast kan läsas i stigande postnummerordning, och om uppsättningen av tester säger att den senaste posten som lästs med LINPUT verkligen börjar en ny Basic-rad, så måste detta sparas tills BASICLINE anropas nästa gång.

Var bara tacksam för statiska variabler i XB-underprogram! Du måste även vara försiktig så att filslutsmarkeringen ej utlöses för tidigt.

```
570 SUB WRITECOL(P,C,P$(,),N
$):: IF NC THEN P=6 :: NC=0
580 IF P>=57 THEN N$="END of
 COL" :: NC=-1 :: SUBEXIT
590 CALL WRITEPAR(P,C,P$(,),
N$)
600 SUBEND
```

När WRITECOL nu har Basic-raden så sändes den iväg för att formas till spalter. Detta förenklade program hanterar slutet av en spalt på ett något slösaktigt sätt som är mycket enkelt att programmera. En normal XB-programrad listar som mest 5 skärmrader, och oberoende av hur smart du är på att mata in längre programrader så har programmet redan begränsat den till en strängvariabel (max längd 255 eller 10 skärmrader) eller kraschar med ett felmeddelande (ERROR). Den närliggande lösningen är att hoppa tillbaka med meddelande om slut på spalten när den föreslagna första raden på den nya spalten är förbi ett bestämt ställe något före slutet av spalten. Det värde som valts, rad #57, är en kompromiss mellan att göra programmet idiotsäkert eller att slösa utrymme. En bättre ansats är att skriva så långt som möjligt, och testa varje nytt stycke för att se om det får plats, och om så ej är fallet, spara det till nästa spalt. Om du undrar varför strängen kallas NEW$, så lämna en tanke åt OLD$ som försvann spårlöst när programmet förenklades för undervisningsändamål.

```
610 SUB WRITEPAR(P,C,P$(,),N
$)
620 P=P+1 :: IF LEN(N$)>28 T
HEN P$(P,C)=SEG$(N$,1,28)::
N$=SEG$(N$,29,LEN(N$)-28)::
GOTO 620 ELSE P$(P,C)=N$ ::
N$=""
630 SUBEND
```

Underprogrammet WRITEPAR var nära att bli kallat SALAMI eftersom det skivar upp NEW$ och fördelar skivorna till successiva utskriftsrader. När rad 620 väl har påbörjats så kommer den att hoppa tillbaka till sig självt tills den återstående delen passar på en skärmrad. Det förutsättes att invervallets storlek har undersökts innan raden påbörjas. Vid en tillbakablick, 5 år senare, ser detta fortfarande ut att vara en fin programrad.

```
640 SUB PRINTPAGE(P$(,),D$):
:PRINT "":"** Page print sta
rted"
650 PRINT "":"** Assembling
printlines":" and printing t
o" :: PRINT "":" ";D$
660 FOR I=1 TO 66 :: PRINT #
2:TAB(9);P$(I,1);TAB(45);P$(
I,2):: P$(I,1),P$(I,2)="" ::
 NEXT I
670 SUBEND
```

Inte mycket behöver sägas om PRINTPAGE utöver att notera att rad 660 formar en enstaka utskriftspost från inmatning av två spalter och suddar radbufferten.

```
680 SUB YN(A$,B$,R,C,X)
690 DISPLAY AT(R,C)BEEP:A$&"
(Y/N) "&B$ :: ACCEPT AT(R,C
```

```
+LEN(A$)+7)VALIDATE("YN")SIZ
E(-1)BEEP:A$ :: X=A$=B$ :: R
=R+2 :: SUBEND

700 SUB KEYCON :: DISPLAY AT
(24,6)BEEP:"ANY KEY TO PROCE
ED"
710 CALL KEY(3,I,ST):: IF ST
=0 THEN 710 ELSE DISPLAY ERA
SE ALL
720 SUBEND

730 SUB FILENAME(R,C,M$,D$)
740 DISPLAY AT(R+1,C):RPT$("
-",LEN(M$)):: DISPLAY AT(R,C
):M$ :: IF D$<>"N?" THEN DIS
PLAY AT(R+2,C):D$ ELSE SUBEX
IT
750 ACCEPT AT(R+2,C)SIZE(-15
)BEEP:D$ :: SUBEND

760 SUB MORE(NM):: DISPLAY E
RASE ALL :: CALL TXTCOL(3,12
):: CALL YN("More listings",
"N",16,2,NM):: SUBEND

770 SUB DELAY(A):: FOR A=1 T
O A :: NEXT A :: SUBEND

780 SUB TXTCOL(A,B):: CALL S
CREEN(B):: FOR I=0 TO 12 ::
CALL COLOR(I,A,B):: NEXT I :
: SUBEND
```

FILENAME-rutinen skriver en understruken rubrik, visar default-svaret med DISPLAY och läser av svaret med ACCEPT. Om det inte ställs någon fråga, "N?", väntar den inget svar. De andra SUB-programmen gör bara sitt jobb när de anropas. YN fungerar som inmatningsrutiner kända från andra TI-moduler.

```
790 SUB SPEAK(A$):: CALL PEE
EK(-28672,SP):: IF SP=96 TH
EN THEN CALL SAY(A$) ELSE CA
LL DELAY(5*LEN(A$))
800 SUBEND
```

Detta är den sista lilla godbiten som har lagts till så att du kan lägga till tal vid inmatningsfrågorna i ditt program där så önskas. Ett CALL SAY har det irriterande uppförandet att det tycks hålla på i evighet innan det ger upp försöket om "speech synthesizer" inte är ansluten. Rad 800 kontrollerar att tal är anslutet och rad 820 ersätter detta med en avsiktlig fördröjning om så ej är fallet. CALL SPEAK("...") kan sedan stoppas in varhelts den behövs i programmet.

Så nu var det klart, ett genomarbetat exempel på ett icke trivialt nyttigt program som på ett betydande sätt använder SUB-program i XB. Det visar att XB-programmeraren kan, med något omdöme som finner naturliga uttryck i språket utan att tumma på funktionen, följa de allmänna principerna för strukturerad programmering. Det beskrivna programmet är en nedbantad version av den sjungande och dansande modellen, COLIST (finns i programbanken ö.a.), som nu har vuxit till >22kB och använder 48 SUB-program. I alla versioner, har underprogram varit ett väsentligt verktyg för programutveckling. Nu är det tid att se tillbaka på vad vi har gjort och jaga ytterligare några finesser.

(Med Super-XB kan du lista direkt med 28 kolumner med t.ex. LIST "PIO":28: ö.a.)

HORIZON 3000 RAM-DISK

En ny variant av Horizon-kort kan använda statiska 32 kbytes (256 kbit) eller 128 kbytes (1 Mbit) chips. Med 32 kB kan upp till 384 KB monteras utan att kretsarna behöver staplas ovanpå varandra, medan 128 kB klarar 1536 kB utan stapling. För 32 kB används HM62256-LP12 och för 128 kB används HM628128-LP12. En annan nyhet är en switch som gör "disable" av kortet så att det blir osynligt för datorn vilket kan vara bra om något skulle krascha RAM-disken. Kortet levereras med MENU version 7.38 och detta operativsystem medger 2x360 kB på samma kort genom att två DSK-nummer samtidigt används. Kortet behåller minnet även när expansionsboxen är avstängd genom att man använder laddningsbara NiCd-, litium-, eller alkaliska batterier. De två senare är inte laddningsbara så en extra diod måste monteras som förhindrar laddning. RAM-disken säljs av Bud Mills Services. Pris 192kB USD 190, 384kB USD 280.

NÄR DATORN LÅSER SIG

av John Guion, Dallas TI user group, USA
(död 8 sept 1989 i en bilolycka)

Ibland kan TI-99/4A råka ut för låsning (lock-up) under ett programs normala körning. Detta kan orsakas antingen av mjukvarufel, hårdvarufel eller yttre störningar av systemet. Antag att mjukvarufel och yttre störningar (som t.ex. att stöta till flexkabelkontakten till expansionsporten) har kunnat uteslutas som möjliga orsaker, finns det flera saker som kan orsaka slumpmässig låsning av datorn. Låsningen kan bero på tre olika saker: kraftaggregat, värme och dåliga kontakter.

KRAFTAGGREGAT

Kontrollera att datorn matas från en stabil nätspänning (c:a 220 V) som omvandlas till en lägre växelspänning i det yttre nätaggregatet. Datorns interna (OBS!!) spänningsomvandlare ger sedan +5V, +12V och -5V. Toleransen för dessa är 5 %. Om datorn alltid fungerar på annan plats eller med andra tillsatsutrustningar, kan man misstänka störningar från någon tillsatsutrustning eller från nätspänningen.

Kontrollera även att kraftkontakten på datorns baksida är ordentligt inskjuten.

VÄRMEPROBLEM

Om låsningen först inträffar efter en viss tids användning, kan man misstänka att det beror på värme. Förvissa dig om att datorns ventilationsgaller ej är blockerad. Kontrollera att kylflänsen till videoprocessorn TMS9929A (PAL i Europa) har tillräckligt med värmeavledande kräm samt även TMS9904 om den har kylfläns. Om du installerar en modernare switchad intern omvandlare, kommer temperaturen att sänkas.

MODULKONTAKTEN (GROM-PORT)

Om datorn har svårigheter med att köra vissa moduler tillfredsställande (som t.ex. Extended Basic) och dessa moduler ofta behöver reset, bör modulporten bytas. Detta gäller i första hand moduler som använder dubbelsidiga kontakter, som t.ex. Extended Basic. Att göra rent GROM-porten kan hjälpa men problemet kommer troligen snart tillbaka igen om inte en ny kontakt installeras (Kan beställas hos Jim Lesher, 722 Huntley, Dallas, TX 75214, USA. Pris USD 10 + porto. En extra GROM-kontakt finns hos Jan Alexandersson, 08-771 05 69).

När du byter GROM-porten bör du avlägsna den rengörare av filt eller skumplast som du tar bort med en skruvmejsel och lämpligt lösningsmedel. Använd inte något som kan lämna kvar lösa fibrer. Sätt sedan på plastskyddet som snäppes fast på den nya GROM-porten.

EXPANSIONSKONTAKTEN

Dålig kontakt vid datorns I/O-port på höger sida kan ibland ge upphov till problem. Om problemet kvarstår efter kontroll av att hopkopplingen är stabil, använd sprit och en styv pappersbit för att göra ren den inre kontakten hos den yttre enhet du anslutet (fristående enhet eller flexkabeln).

Öppna datorn och plocka fram moderkortet och gör rent kontakten med suddgummi följt av rengöring med sprit. Det behövs endast en försiktig rengöring för att avlägsna oxidationen på ytan.

REFERENSER

Micropendium:

aug 1989 p.46: Things to do when your console locks up
sep 1989 p.45: Tips on debugging module problems
dec 1989 p. 8: Console lockups
jan 1990 p.45: Complete cleaning cures console lockup

ERFARENHETER AV HÅRDDISK MED TI-99/4A

av Bertil Stenfeldt

Jag har nu hunnit använda min hårddisk tillsammans med Myarcs HFDC i några månader. Det har verkligen varit angenämt att nyttja ett så snabbt och stort yttre minne! Utan att jag mätt med klocka vågar jag påstå att snabbheten inte understiger RAM-diskens nämnvärt.

Det har emellertid inte varit helt problemfritt. Bl a hade jag otur och fick byta hårddisk ett antal gånger innan jag fick en som fungerade. Leverantören verkade vid åtminstone ett tillfälle knappast veta att en hårddisk måste behandlas varsamt. Eller vad sägs om att skicka en hårddisk på posten endast emballerad i ett vanligt vadderat kuvert utan någon som helst extra stötdämpare. Helt riktigt! Flera hundra sektorer var skadade!

Det finns också en del andra problem i samband med användningen. Det gäller främst 2 ting. Det ena är att DSK.-emuleringen inte tycks fungera helt klanderfritt. Om jag vill spara en fil med t ex SAVE DSK.TW.FILNAMN och filen inte finns tidigare på hårddisken går inte detta utan jag måste skriva SAVE WDS1.DSK.TW.FILNAMN. Om jag däremot skriver på en befintlig fil fungerar det första alternativet utmärkt.

Det andra är att jag använder P-GRAM-kortet vilket förorsakar en del problem vid inladdningen av MDM5 (diskmanagern). När P-GRAM-kortet är aktivt låser sig datorn vid inladdningen av MDM5 och jag måste stänga av och slå på den igen för att komma ur detta läge. Har någon annan medlem råkat ut för detta och kan tipsa mig om en lösning?

Tills vidare har jag löst det på följande sätt: Jag har satt en strömbrytare på fronten på expansionsboxen. Från denna går en sladd till strömbrytaren som sitter i bakkanten på P-GRAM-kortet. Med denna kan jag slå på och av kortet. Om jag slår av kortet innan jag ska ladda in MDM5 går det bra att ladda.

Denna lösning fungerar emellertid inte när jag är inne i XB och denna ligger i P-GRAMKORTET eftersom jag då också stänger av XB när jag stänger av kortet. Det fungerar dock utmärkt om jag laddar MDM5 från vanlig BASIC eller från en assemblerrutin som Funnelweb.

Förutom dessa båda problem har jag ytterligare ett som gäller MDM5 tillsammans med min CorCoMp RAM-disk. MDM5 förstör RAM-diskens filkatalog. MDM5 är inte ensam om att bära sig åt på detta sätt utan är i gott sällskap med DM1000.

Något som jag tycker är mycket viktigt att tänka på är backup. Detta fungerar bra med HFDC även om det tar en förfärlig tid beroende på långsamheten i diskettdrivarna. Det finns flera metoder att ta backup på.

- "vanlig" filkopiering
- backup på endast vissa underkataloger som helheter
- d:o fast endast av uppdaterade filer
- backup av hela hårddisken, vilket även inkluderar katalogstrukturen
- d:o fast endast av uppdateringar.

För att det ska fungera smidigt bör man inte ta hela hårddisken alltför ofta. I stället tar man en backup av hela disken kanske en gång varannan månad beroende på hur mycket man använder den och dessemellan nöjer man sig med att endast backuppa uppdaterade filer. Därutöver kan man ta backup på de viktigaste underkatalogerna för sig för att vara säker på att ha dessa säkrade. Det finns risk att en backupdiskett inte fungerar. Det hände mig med en av de disketter på vilka jag tagit totalbackup på. Jag hade formaterat om hårddisken vilket jag gjorde några gånger i början för att prova olika formateringsalternativ och skulle lägga tillbaka innehållet från min

TI-PD CATALOG #2

from Tigercub Software, USA

During the past 7 years, a great many programmers have contributed a wealth of material to the public domain. Unfortunately, most of these programs have not been readily available to most of the TI users. Only a few of the user groups have really large public domain libraries, and even these are usually cataloged only by alphabetized abbreviated filenames.

I have therefore decided to make the contents of my public domain library available to the TI world, at a copying fee so low that I hope no one will think I am unfairly profiting from the work of others (and I think you will note, in the following listings, that I have probably contributed more to the public domain than anyone else!), but if any author objects to my distributing his work I will certainly stop. My catalog contains the author's name for each program, when available, both in order to give due credit and to aid in distinguishing between programs of the same name. Regrettably, many of the IUG programs distributed by Amnion have had the author's name deleted.

I have gone through my library of over 4000 public domain programs and selected enough of the better ones to fill over 300 disks, arranged by category. Each SS/SD disk contains as many programs as I could fit onto it, if I had enough programs of that category - the number of filled sectors on each disk is indicated in parentheses. All Basic-only programs have been converted to run in Extended Basic (except those which use the TEII speech), and an XBasic loader has been provided for assembly programs whenever possible. Each disk has been provided with an autoloader by full program name, not filename.

This public domain is offered only as a copying service, not as a sale of computer software, and I take no responsibility other than providing a copy equal to the original. Contents of all disks are subject to change without notice.

The following TI-PD disks are offered at $1.50 per disk, POSTPAID in the U.S. and Canada on orders of at least eight disks. Overseas customers add .50 per disk. Make checks payable to Tigercub Software.

Du kan få en kopia av denna katalog genom att sända 2 SS/SD med frankerat svarskuvert till Jan Alexandersson. Du som endast har kassettspelare kan få en utskriven kopia genom att sända 5 rabattmärken eller 15 kr i frimärken. Om det finns intresse kan ett gemensamt inköp göras och program vid behov överföras till kassettband.

---

backup. Den första disketten krånglade varför hela backupen var ogjord. Som tur var hade jag inte hunnit göra så mycket och dessutom hade jag andra backuper av de viktigaste filerna så jag kunde rekonstruera disken.

En förutsättning för att backuppandet ska fungera är att man ALLTID ställer klockan i HFDC. Denna drivs inte med batteri utan måste ställas varje gång man har slagit av expansionsboxen. Alla filer tidsstämplas när de skapas eller uppdateras. När man tar en backup markeras de filer som kommit med i backupen så att man senare inte behöver backuppa dem igen om de inte blivit uppdaterade. För detta ändamål behövs en riktig tidsangivelse på alla filer. Om man har en annan klocka i sitt system t ex CorComps Triple Tech eller P-GRAM-Kortet och denna går på batteri kan man enkelt lägga in rutiner i sitt XB-LOAD program som läser av denna klocka och sedan lägger in dessa uppgifter i HFDC-klockan.

Sammantaget tycker jag att det är synnerligen behagligt att till största delen kommit ifrån "skyfflandet" av disketter och är mycket nöjd med HFDC och hårddisken trots de problem som finns.

TIPS FROM THE TIGERCUB

#52

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213
USA

This one should come in handy for bowling league captains and Little League coaches.

```
100 DIM M(29,29),T$(30)
110 GOTO 130
120 N;Q$;J;I;X;P$;S$;K
130 !@P-
140 DISPLAY AT(3,7)ERASE ALL
:"LEAGUE SCHEDULER":;:"by th
e Burwells              adapt
ed by Tigercub"
150 DISPLAY AT(8,1):" This p
rogram sets up a":"schedule
for up to 30 teams":"so that
 each plays each":"other onc
e and only once."
160 DISPLAY AT(12,1):" If an
 odd number of teams":"are s
cheduled, each gets one":"by
e."
170 DISPLAY AT(16,1):"Number
 of teams?" :: ACCEPT AT(16,
18)VALIDATE(DIGIT):N :: IF N
>30 THEN DISPLAY AT(18,1):"L
IMIT OF 30!" :: GOTO 170
180 DISPLAY AT(18,1)ERASE AL
L:"Schedule teams by name? Y
" :: ACCEPT AT(18,25)SIZE(-1
)VALIDATE("YN"):Q$ :: IF Q$=
"N" THEN 200
190 FOR J=1 TO N :: DISPLAY
AT(20,1):"Team no.";J;"name?
" :: ACCEPT AT(22,1):T$(J)::
 NEXT J :: GOTO 210
200 FOR J=1 TO N :: T$(J)="T
eam No. "&STR$(J):: NEXT J
210 IF N/2<>INT(N/2)THEN N=N
+1 :: T$(N)="bye"
220 DISPLAY AT(23,1):"Schedu
le by day, week, month":"or
what?" :: ACCEPT AT(24,10):S
$ :: FOR J=1 TO N-1 :: M(1,J
)=J+1
230 NEXT J :: FOR J=1 TO N-1
 STEP 2 :: GOSUB 260
240 NEXT J :: FOR J=2 TO N-2
 STEP 2 :: GOSUB 330
250 NEXT J :: GOSUB 390 :: S
TOP
260 FOR I=1 TO N-2 :: IF M(I
,J)=N THEN 280
270 M(I+1,J)=M(I,J)+1 :: GOT
O 290
280 M(I+1,J)=M(I,J):: GOTO 3
00
290 NEXT I
300 X=I+1 :: FOR I=X TO N-2
:: M(I+1,J)=M(I,J)-1
310 NEXT I
320 RETURN
330 FOR I=1 TO N-2 :: IF M(I
,J)=2 THEN 350
340 M(I+1,J)=M(I,J)-1 :: GOT
O 360
350 M(I+1,J)=M(I,J):: GOTO 3
70
360 NEXT I
370 X=I+1 :: FOR I=X TO N-2
:: M(I+1,J)=M(I,J)+1
380 NEXT I :: RETURN
390 DISPLAY AT(12,1)ERASE AL
L:"Output to - 2":;:" (1) Sc
reen":" (2) Printer" :: ACCE
PT AT(12,13)SIZE(-1)VALIDATE
("12"):K :: IF K=1 THEN 440
400 DISPLAY AT(18,1):"Printe
r? PIO" :: ACCEPT AT(18,10)S
IZE(-18):P$ :: OPEN #1:P$ ::
 PRINT #1:"LEAGUE SCHEDULE":
 : :: FOR I=1 TO N-1 :: PRIN
T #1:S$;" #";I :: PRINT #1:T
$(1);" vs ";T$(M(I,1))
410 FOR J=2 TO N-2 STEP 2 ::
 PRINT #1:T$(M(I,J));" vs ";
T$(M(I,J+1))
420 NEXT J :: PRINT #1:"": :
430 NEXT I :: RETURN
440 FOR I=1 TO N-1 :: PRINT
TAB(7);"LEAGUE SCHEDULE": :
:: PRINT "WEEK #";I: : :: PR
INT T$(1);" vs ";T$(M(I,1)):
: FOR J=2 TO N-2 STEP 2 :: P
RINT T$(M(I,J));" vs ";T$(M(
I,J+1))
450 NEXT J :: PRINT "": : ::
 PRINT "PRESS ANY KEY FOR NE
XT WEEK"
460 CALL KEY(0,K,S):: IF S=0
 THEN 460
470 CALL CLEAR
480 NEXT I :: RETURN :: END
```

Some folks seem to think that the subprograms on my Nuts & Bolts disks are just flashy screen displays. Not so! This one will be on the next diskfull, if I ever get it full, which is most unlikely.
ACCEPT AT with a negative size is useful to accept a default string from the screen, but the length of the string is limited to 28 characters; and if you want something other than the default, you must be sure to delete any extra characters.
CALL DEFAULT(R,C,M$,R$),
where R and C are the row and column to accept at, M$ is the default string which can be up to 254 characters long, and R$ is the string accepted, will display the default string, accept it if Enter is pressed, or accept any other string without having to blank out the extra characters. Just don't type too fast!

```
100 M$="TESTING" :: CALL CLE
AR
110 CALL DEFAULT(12,1,M$,R$)
:: DISPLAY AT(24,1):R$ :: GO
TO 110
10000 SUB DEFAULT(R,C,M$,R$)
:: R$="" :: X=ASC(M$)
10001 DISPLAY AT(R,C):M$
10002 CALL HCHAR(R,C+2,ASC(S
EG$(M$,1,1))):: CALL HCHAR(R
,C+2,30)
10003 CALL KEY(0,K,S):: IF S
=0 THEN 10002 ELSE IF K=13 T
HEN R$=M$ :: SUBEXIT ELSE DI
SPLAY AT(R,C):CHR$(K):: ACCE
PT AT(R,C+1):R$ :: R$=CHR$(K
)&R$
10004 SUBEND
```

CALL DEFAULT(R,C,N,RN), with N as the default value and RN as the value accepted, will do the same for numeric input, and will reject any non-numeric input. Errors due to fast typing can be prevented by omitting the DISPLAY AT(R,C):CHR$(K) in line 1002.

```
100 N=176453.897 :: CALL CLE
AR
110 CALL DEFAULTN(12,1,N,RN)
:: DISPLAY AT(24,1):RN :: GO
TO 9999
10000 SUB DEFAULTN(R,C,N,RN)
:: DISPLAY AT(R,C):N :: N$=S
EG$(STR$(N),1,1)
10001 CALL HCHAR(R,C+2,ASC(N
$)):: CALL HCHAR(R,C+2,30)
10002 CALL KEY(0,K,S):: IF S
=0 THEN 10001 ELSE IF K=13 T
HEN RN=N :: SUBEXIT ELSE DIS
PLAY AT(R,C):CHR$(K):: ACCEP
T AT(R,C+1):R$ :: R$=CHR$(K)
&R$
10003 ON ERROR 10004 :: RN=V
AL(R$):: GOTO 10005
10004 CALL SOUND(200,110,5,-
4,5):: DISPLAY AT(R,C):N ::
ON ERROR STOP :: RETURN 1000
2
10005 SUBEND
```

Ed Machonis discovered an easy way to count the words in a TI-Writer file, using TI-Writer itself. Just put in a line before line 0001, with .LM 0;RM 1;FI;PL nnn with nnn being the sector length of the file multiplied by 40. Save it, go into the Formatter and print it to disk under a different filename. Return to Editor, load the resulting file, page through it with FCTN 4 counting any blank lines, subtract the number of blanks from the last line number, and that's it! The Formatter takes about one minute to count 1000 words. If the resulting file is very large, you may have to load it in two sections.

```
100 M$="POS WILL FIND THE FI
RST OCCURRENCE OF A SUBSTRIN
G WITHIN A STRING BUT I OFTE
N NEED TO FIND THE LAST OCCU
RRENCE SO I WROTE THIS SUBPR
OGRAM"
105 INPUT "SUBSTRING?":L$
110 CALL LAST(M$,L$,P):: IF
P=0 THEN PRINT "NOT FOUND" :
: GOTO 105 ELSE PRINT SEG$(M
$,P,255):: GOTO 105
120 SUB LAST(M$,L$,P):: X=1
130 Y=POS(M$,L$,X):: IF Y=0
```

```
THEN P=0 :: SUBEXIT ELSE Z=Y
140 X=Y+1 :: Y=POS(M$,L$,X):
: IF Y=0 THEN P=Z :: SUBEXIT
 ELSE Z=Y :: GOTO 140
150 SUBEND
```

Here's a new way to make music. The algorithm in 110 sets up a 3-octave chromatic scale - note the N(1)=F, I have erroneously omitted it when I previously published that algorithm.
To change the key of the music you have programmed, just change the value of F. Lines 190-220 contain the part of the music that is repeated within the melody. A is the subscript of the melody note, B is the subscript number of the chord. These must be above 13, as the frequency is divided by 2 in the subroutine.
Each beat of the music has a GOSUB, to 230 to play a bass accompaniment with the first note of each bar, to 260 for the other notes of the bar. The chord note is divided by different values to play the three notes of the chord in succession, and multiplied by 3.75 in the 3rd voice to produce a bass note two octaves lower in the -4 noise. The melody note is multiplied by 1.01 in the second voice to give a richer tone.

```
100 DISPLAY AT(12,3)ERASE AL
L:"THE MAORI FAREWELL SONG"
! programmed by
  Jim Peterson
110 F=110 :: DIM N(36):: FOR
 J=1 TO 36 :: N(J)=INT(F*1.0
59463094^(J-1)):: NEXT J ::
N(1)=F :: T=-999
120 GOSUB 190 :: A=30 :: B=2
3 :: GOSUB 230 :: GOSUB 260
:: GOSUB 260 :: A=32 :: B=28
 :: GOSUB 230 :: GOSUB 260 :
: GOSUB 260 :: A=28
130 GOSUB 230 :: GOSUB 260 :
: GOSUB 260 :: A=30 :: B=23
:: GOSUB 230 :: GOSUB 260 ::
 A=28 :: GOSUB 260 :: A=27 :
: GOSUB 230 :: GOSUB 260
140 A=28 :: GOSUB 260 :: A=3
0 :: GOSUB 230 :: GOSUB 260
:: GOSUB 260 :: GOSUB 230 ::
 GOSUB 260 :: GOSUB 260 :: G
OSUB 190
150 A=30 :: B=23 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: A=32 :: B=16 :: GOSUB 230
 :: GOSUB 260 :: A=28 :: GOS
UB 260
160 A=33 :: B=23 :: GOSUB 23
0 :: GOSUB 260 :: A=32 :: GO
SUB 260 :: A=25 :: B=13 :: G
OSUB 230 :: GOSUB 260 :: GOS
UB 260
170 A=27 :: B=23 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: A=28 :: B=16 :: GOSUB 230
 :: GOSUB 260 :: GOSUB 260
180 B=28 :: GOSUB 230 :: GOS
UB 260 :: GOSUB 260 :: B=16
:: GOSUB 230 :: GOSUB 260 ::
 GOSUB 260 :: GOTO 120
190 A=32 :: B=28 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: A=28 :: B=16 :: GOSUB 230
 :: GOSUB 260 :: A=30 :: GOS
UB 260
200 A=32 :: B=28 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
:: B=16 :: GOSUB 230 :: GOSU
B 260 :: GOSUB 260 :: B=28 :
: GOSUB 230 :: GOSUB 260
210 A=30 :: GOSUB 260 :: A=3
3 :: B=23 :: GOSUB 230 :: GO
SUB 260 :: A=27 :: GOSUB 260
 :: A=28 :: B=16 :: GOSUB 23
0 :: GOSUB 260 :: GOSUB 260
220 B=28 :: GOSUB 230 :: GOS
UB 260 :: GOSUB 260 :: B=16
:: GOSUB 230 :: GOSUB 260 ::
 GOSUB 260 :: RETURN
230 CALL SOUND(T,N(A),5,N(B)
/1.585,9,N(B)*3.75,30,-4,9):
: GOSUB 290
240 CALL SOUND(T,N(A),5,N(B)
/1.334,9,N(B)*3.75,30,-4,9):
: GOSUB 290
250 CALL SOUND(T,N(A),5,N(B)
/2,9,N(B)*3.75,30,-4,9):: GO
SUB 290 :: RETURN
260 CALL SOUND(T,N(A),5,N(A)
*1.01,5,N(B)/1.585,9):: GOSU
B 290
270 CALL SOUND(T,N(A),5,N(A)
*1.01,5,N(B)/1.334,9):: GOSU
B 290
280 CALL SOUND(T,N(A),5,N(A)
*1.01,5,N(B)/2,9)
290 FOR D=1 TO 20 :: NEXT D
:: RETURN
```

MEMORY FULL.....

Jim Peterson

SWEDLOW EXTENDED BASIC

```
X   X   BBBB      # 7 to 8
 X X    B   B
  X     BBBB      By
 X X    B   B     Jim
X   X   BBBB      Swedlow
```

(This article originally appeared in the User Group of Orange County, California ROM)

PRODUCT REVIEW: ADVANCED DIAGNOSTICS
By Millers Graphics

I have only had this a few days so this is a preliminary reading. This product is so impressive, however, that it is worth your serious consideration.

Advanced Diagnostics does quite a few things:

- Checks your disk drive's motor speed and operation.

- Checks all of your computer's RAM.

- Checks, displays, edits and copies disks, sector by sector.

- Dumps your screen (including a disk sector's contents) to any device -- printer, etc.

- Formats and checks disks.

- Works with both the TI and CorComp disk controllers.

What's more, you can create command files that allow you to program the 45 commands in Advanced Diagnostics in any sequence you may need. For example, one of the seven Command Files that comes with this will format a box of disks.

This is a powerfull disk diagnostic and control LANGUAGE. It requires XB, Editor/Assembler or Mini Memory.

On a scale of 1 to 10 this is an easy 9.5. Well worth the cost and a valuable tool for the experienced programmer.

WHEN ALL ELSE FAILS . . .

TRACE and UNTRACE are powerful debugging tools. The other day I was looking something up in the XB book and discovered that these can be used as statements in a program line. This makes them all the more helpfull in catching that bug!

IMAGE and USING

XB left justifies all printed and displayed strings and numbers. While this is correct for strings, numbers should be lined up by the decimal point. One hard way to fix this is to turn your numbers into strings and add leading spaces. PRINT USING is much easier.

Enter and run this program:

```
100 FOR I=9 TO 10 STEP .1
110 PRINT USING "##  ##.#  #.##"
    :I,I,I :: NEXT I
```

Note how the first column prints the number rounded to the nearest whole number while the other two display decimals. The string of asterisks shows you that the number was too big for the space alloted.

Only the number on the screen is rounded -- the number in RAM is NOT rounded.

The statement after USING can be a string, a string variable or a line number that refers to an IMAGE statement. Examples:

```
10 A$="The answer is ###.##"
   :: B$="John" :: C$="Dear"
20 IMAGE ## + ## = ###
30 PRINT USING A$:2.45678
40 DISPLAY AT(1,1):USING 20:
   14,19,14+19
50 PRINT #1, USING "$###.##":23.1
60 PRINT USING C$&"#####":B$
```

For more details, look in the XB book. Remember, when all else fails . . . .

You can make your output look more professional with these tools.

PAYMENT

I wrote this program to find out how long it would take to pay off a charge account if I paid the minimum. It has been modified to show a number of ways to use USING.

A note on the program: the answers are approximations. Most lenders these days have complicated formulas to determine interest while this program uses the old simple formula.

REDO

You are editing a line. As you add some code, the end of the line, including the closing quote mark, disappears. You press ENTER and get an error message. All of your work is lost? No. Press FCTN REDO and your line, complete with errors, will reappear. Now you can correct it and finish editing your program.

```
*********************************
* Programlistning för PAYMENT   *
*      se PB 89-1 sid 17        *
*********************************
```

ASC CODES

At a Users Group meetings a list of ASCII codes in both base 10 and hex was passed out. There were some questions about the BASIC commads listed on the chart.

When your computer saves a command on disk or cassette it changes the command statements to tokenized statements. CALL, for example, is saved as HEX 9D or CHR$(157). The chart lists all of these tokenized ASC codes.

SUB PROGRAMS

XB has four (or more) ways of writing utility routines: sub routines, sub programs, assembly language routines and DEFinations. Each of these has pros and cons. This month, we'll look at sub programs.

The best way to describe a sub program is to explain where it differs from a subroutine:

-- Sub routines are called by line number while sub programs are CALLed by name.

-- Sub routines use the same variables as the main program. In sub routines, variables are entirley different unless noted in the CALL statement.

-- Sub rountnes can be anywhere in the program while sub routins must be at the end.

Since you call sub programs by name, you do not loose their identity when you RESquence. You loose the power of ON GOSUB and passing varialbes can be tricky if quite a few are involved.

Enter and run this sample program:

```
10 A,B=10 :: PRINT A;B;C ::CALL TEST
20 CALL NEWS(B) :: PRINT A;B;C
30 SUB TEST :: A=5 :: PRINT A;B;C
   :: SUBEND
40 SUB NEWS(C) :: PRINT A;B;C :: C=4
   :: SUBEND
```

You should get this output:

```
 10 10  0
  5  0  0
  0  0 10
 10  4  0
```

If you think this thru, you will see that there are NINE variables in this program: A, B and C in the program; A, B and C in TEST; and, A, B and C in NEWS.

When you called NEWS(B) the value of B was passed to NEWS and processed as C in that sub program. Play with this program (including inserting BREAK statemnts) to test this out.

ORACLE

This program is loosely based on one that appeared in 99'er years ago. Both programs were designed to demonstrate sub programs. Speech has been added to this version.

The CALL PEEK in line 160 works on my computer but may not work on yours -- if you have a problem, just

delete it. If you don't have speech, change the CALL SAY's to PRINT's.

Various formats for sub progams are shown. The rules are in the XB book, especilly for passing variables between the main program and the sub program. Note that SUB name must start a line and SUBEND must end a line.

A disclaimer: all answers are based on random numbers. If you think the answers fit, well, that's just random chance, right? A computer can't give sood answers, can it?

```
100 ! ORACLE
110 ! VERSION XB.2.1
120 ! 08 MAR 85
130 ! BY JIM SWEDLOW
140 !
150 DISPLAY AT(10,4)ERASE AL
L BEEP:"** I AM  THE ORACLE
**" :: CALL DELAY :: RANDOMI
ZE
160 CALL INIT :: CALL PEEK(-
28672,I):: IF I=0 THEN DISPL
AY AT(20,1):"I cannot operat
e without the    Speech Synt
hesizer!" :: STOP
170 DISPLAY AT(15,1):"   I a
nswer all questions": : :"As
k qustions with  YES or NOan
swers -- When you are donepr
ess ENTER."
180 CALL DELAY :: DISPLAY AT
(24,1):"   PRESS ANY KEY TO
BEGIN"
190 CALL KEY(0,I,S):: IF S=0
 THEN 190 ELSE CALL CLEAR
200 PRINT : :"WHAT IS YOUR Q
UESTION?" :: LINPUT Q$ :: IF
 Q$="" THEN 220
210 CALL DELAY :: CALL REPLY
(Q$):: CALL DELAY :: GOTO 20
0
220 DISPLAY AT(10,1)ERASE AL
L:"THANK YOU FOR CONSULTING"
: : : : : : :"    ** THE ORA
CLE **" :: CALL DELAY :: STO
P
230 !
240 SUB DELAY :: FOR I=1 TO
200 :: NEXT I :: SUBEND
250 !
260 SUB REPLY(A$)
270 A$=SEG$(A$,1,2):: IF A$=
"WH" OR A$="HO" THEN CALL OT
HER ELSE CALL YESNO
280 SUBEND
290 !
300 SUB YESNO
310 ON INT(10*RND)+1 GOTO 32
0,330,320,340,350,350,360,37
0,380,390
320 CALL SAY("YES"):: SUBEXI
T
330 CALL SAY("I THINK SO")::
 SUBEXIT
340 CALL SAY("LOOKS POSITIVE
"):: SUBEXIT
350 CALL OTHER :: SUBEXIT
360 CALL SAY("LOOKS NEGATIVE
"):: SUBEXIT
370 CALL SAY("I DO NOT THINK
 SO"):: SUBEXIT
380 CALL SAY("NO WAY"):: SUB
EXIT
390 CALL SAY("NO"):: SUBEND
400 !
410 SUB OTHER
420 ON INT(10*RND)+1 GOTO 43
0,440,450,460,470,480,490,50
0,510,520
430 CALL SAY("I CAN NOT TELL
 YOU THAT"):: SUBEXIT
440 CALL SAY("SAY THAT A DIF
FERENT WAY"):: SUBEXIT
450 CALL SAY("YOU DO NOT WAN
T TO KNOW"):: SUBEXIT
460 CALL SAY("I DO NOT KNOW"
):: SUBEXIT
470 CALL SAY("I AM NOT SUPPO
SED TO SAY"):: SUBEXIT
480 CALL SAY("I WILL NOT TEL
L YOU"):: SUBEXIT
490 CALL SAY("I CAN ONLY GUE
SS"):: SUBEXIT
500 CALL SAY("I CAN NOT ANSW
ER THAT"):: SUBEXIT
510 CALL SAY("I DO NOT REMEM
BER"):: SUBEXIT
520 CALL SAY("TRY SOME THING
 ELSE"):: SUBEND
```

THE MISSING LINK FÖR XB+32KB EM

är ett program från TEXAMENTS (pris USD 25 + porto 8) som utökar Extended Basic med 30 assembler-rutiner som anropas med CALL LINK. En demonstrationsskiva till detta program kan fås genom att sända en skiva med frankerat svarskuvert till Jan Alexandersson. Programmet fungerar f.n. ej med DIJIT AVPC eftersom interrupt suddar programrader.

FÖRENINGEN PROGRAMBITEN
ÅRSBERÄTTELSE 1990-03-05

STYRELSENS ÅRSREDOVISNING

Verksamheten 1989 har bedrivits i form av tidningsutgivning och programförmedling. Antalet medlemmar har minskat från 130 till 80.

STYRELSE OCH FUNKTIONÄRER

Föreningens styrelse:
Jan Alexandersson, ordf. och red.
John Hanssen, kassör
Åke Olsson, sekreterare
Claes Schibler, registerförare
Börje Häll, programbankir
Peter Odelryd
Peter Olsson
Mikael Nordlin

Under året har 8 protokollförda styrelsemöten hållits.

FÖRENINGENS EKONOMI

Årets förlust uppgår till 2123.35 kr.

FÖRENINGENS VERKSAMHET

Fyra nummer av tidningen Programbiten har framställts. Från PB 89-1 har tryckningen gjorts på ett enklare sätt. Föreningens telefon har sagts upp.

Fairware på flexskiva har skaffats från Charles Earl (TELCO), Miami UG (Boot och Menu), Barry Boone (Archiver), R.A.Green (RAGMAC, RAG Linker, RAG Utilities), Pargon Computing (Text Loader och Enhanced Display Package), Tony McGovern (Funnelweb, Spider, XB-Tutorial m.m.), Alexander Hulpke (Tetris och Etiketter).

Arvode till funktionärer har ej utgått.

BALANSRÄKNING

Ingående Balans 1989

Tillgångar		Skulder	
Postgiro	14970.68	Leverantörsskulder	6012.30
Varulager	855.87	Förutbetalda medlemsavg.	220.00
Skattefordringar	790.00	Eget kapital	10384.25
Summma	16616.55		16616.55

Utgående Balans 1989

Tillgångar		Skulder	
Postgiro	14923.80	Leverantörsskulder	0.00
Varulager	0.00	Förutbetalda medlemsavg.	600.00
Skattefordringar	0.00	Eget kapital	14323.00
Summma	14923.00		14923.00

RESULTATRÄKNING

Intäkter		Kostnader	
Medlemsavgifter	8960.00	Tryckkostnader	7495.00
Varuförsäljning	3035.00	Varuinköp	972.68
Räntor	612.60	Varulagersförändring	855.87
Överskjutande skatt	30.00	Förbrukningmateriel	1958.30
Årets förlust	2123.35	Porto	1641.50
		Telefon	697.00
		Skatt	1140.00
	14760.35		14760.35

Stockholm som ovan

Jan Alexandersson, ordf

John Hanssen, kassör

PRINTER-PROBLEM MED PIO

av Arne Wennberg

Min son köpte för en tid sedan en ny printer, en STAR LC-10. Det fanns en liten baktanke att jag också skulle kunna använda den till min TI-99/4A. Men se det gick inte. Efter flera tappra försök blev vi tvungna att ge upp. Säljaren kunde inte heller hjälpa oss.

När jag läste Jan Alexanderssons artikel om MYARC och TI RS232/PIO i PROGRAMBITEN 90-1 fick jag en misstanke att här kanske lösningen till problemet fanns. Olika datorer ger olika lång s k STROBE på pinne nr 1 i PIO-porten och den passar inte alla printrar.

Efter telefonkontakt med Jan fick jag mig tillsänt ett kopplingsschema som varit publicerat i tidskriften MICROpendium/September 1989. Där visas hur man med hjälp av en IC-krets och en kondensator kan förändra STROBENS längd.

IC-kretsen behöver +5V för att fungera och det föreslogs i schemat att denna spänning skulle tas ut på pinne nr 15 i PIO-porten. I RS232-kortets manual anges vid denna pinne "1-kilohm pull-up resistor to +5V". Spänningen blev tydligen för låg för det fungerade inte. Om jag däremot tog ut spänningen på pinne nr 12, märkt "10-ohm pull-up resistor to +5", så fungerade det.

Jag försökte också att ta ut spänningen från printern på parallellinterfacets pinne nr 18 (checka med manualen till printern) och det gick också bra.

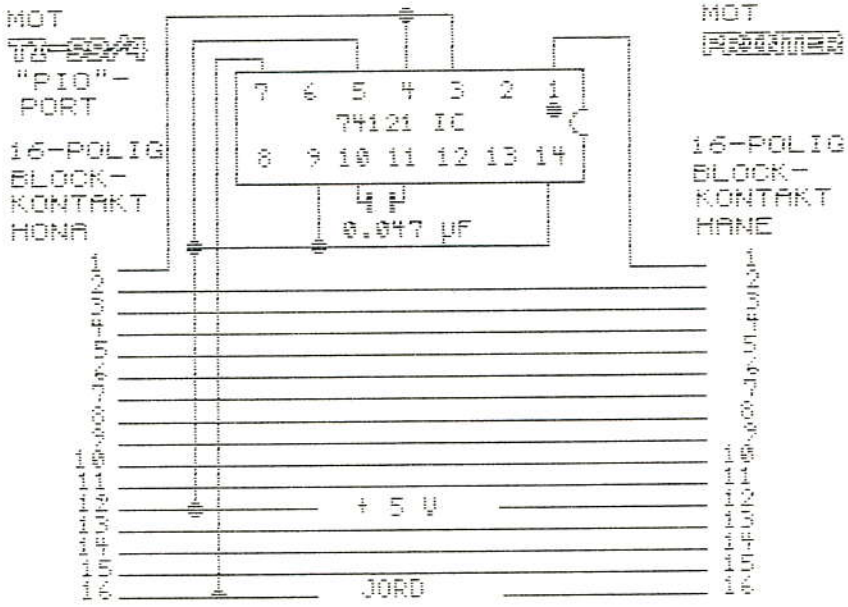
För att tillverka en adapter att koppla in mellan PIO-porten och printerkabeln behövs följande komponenter:

1 st IC-krets 74121
1 st kondensator 0,047 uF
1 st 16-polig blockkontakt, hane
1 st 16-polig blockkontakt, hona
2 dm flatkabel, 16-ledare
en bit enkelledare för inkoppling

Koppla ihop enligt bifogat schema.

STAR LC-10 fungerar utmärkt med denna adapter inkopplad och det är inte osannolikt att den kan få även andra printrar att fungera tillsammans med TI-99:an. Jag kan dock inte lämna några garantier.

Lycka till!



PROGRAM FÖR 9938

(*4A = även 99/4A MP = Micropendium)

PROGRAM	VERS	FÖRFATTARE	SÄLJARE	---GRAFIKMODE----							BILD	NÖDVÄND
				T1	T2	G1	G2	G6	G7	MUS	FIL	TILLBEH
Set 99/4A		David Allen	Dijit	-	-	-	-	-	-	-	-	
Monitor (Edit VRAM)		Elekronik-Serv	?	-	-	JA	-	-	-	-	-	
Horizon ROS(CC/TI)		Barry Boone	Fairware	-	-	-	-	-	-	-	-	Horizon
Horizon ROS(MYARC)		Barry Boone	Fairware	-	-	-	-	-	-	-	-	Horizon
SC-DOS 80	7/4/88	Barry Boone	Fairware	JA	JA	JA	-	-	-	-	-	RAM>6000
Funnelweb Editor	4.21	Tony McGovern	Fairware	JA	JA	-	-	-	-	-	-	*4A
FW Disk Review	4.21	Tony McGovern	Fairware	JA	JA	-	-	-	-	-	-	*4A
Disk Manager 80	4.0	okänd tysk	?	-	JA	-	-	-	-	-	-	
Hard Master	2.18	C.Christensen	Asgard	JA	JA	-	-	-	-	-	-	*4A
Sector One	1.2	Randall Moore	Fairware	JA	JA	-	-	-	-	-	-	*4A
Infocom 80 Loader		TM	?	-	JA	-	-	-	-	-	-	
Multiplan 80		?	?	-	JA	-	-	-	-	-	-	MP-Modul
Telco	2.1	Charles Earl	Fairware	JA	JA	-	-	-	-	-	-	*4A
Hot Bug	1.0	Charles Earl	Fairware	-	-	-	-	-	-	-	-	*4A
AVPC FORTH	1.2	Lutz Winkler	Fairware	JA	JA	JA	JA	-	-	-	-	
FORTH 80 Editor	2färg	Lutz Winkler	MP jun88	-	JA	-	-	-	-	-	-	
FORTH 80 Editor	4färg	Lutz Winkler	MP aug89	-	JA	-	-	-	-	-	-	
FORTH High Res,	part1	Lutz Winkler	MP nov89	-	-	-	-	JA	JA	-	-	
FORTH High Res,	part2	Lutz Winkler	MP feb90	-	-	-	-	JA	JA	-	-	
Tile Breaker	2.0	Michael Ricco	?	-	-	JA	-	-	-	JA	-	
Fractal Generator		Luigi Grilli	Fairware	-	-	-	-	-	JA	-	-	
GIF2 (grafik)	0.41	Paul Charlton	Fairware	-	-	-	-	JA	JA	-	GIF	Geneve
GIF99 (grafik)	1.05	Achim Liese	Fairware	-	-	-	-	JA	JA	-	GIF	TI/CCdsk
Myart Picture Decoder		Barry Boone	Fairware	-	-	-	-	JA	JA	-	MYA	
Art to Art		David Allen	Fairware	-	-	-	-	-	JA	-	MYA	
TI-Artist Plus!		Chris Faherty	Texaments-	-	-	JA	-	-	-	JA	TIA	*4A
Mouse TI-Artist		Mike Dodd	MP nov87	-	-	-	-	-	-	JA	-	TI-Artis
Mouse CALL LINK		Mike Dodd	MP okt87	-	-	JA	-	-	-	JA	-	XB
Mouse Interrupt		Peter Hoddie	Fairware	-	-	JA	-	-	-	JA	-	XB
Palette Master		Jeff Kittka	MP feb89	-	-	JA	-	-	-	-	-	XB
X80 (CALL LINK)	0.91	A.Hulpke	Fairware	-	JA	JA	-	-	-	-	-	XB
XHI (CALL LINK)	3.6	A.Hulpke	Fairware	-	-	JA	-	JA	JA	-	MYA	XB
COLDEF		A.Hulpke	Fairware	-	-	-	-	JA	-	-	-	XB + XHI
Fixsterne		A.Hulpke	Fairware	-	-	-	-	JA	-	-	-	XB + XHI
Kugel		A.Hulpke	Fairware	-	-	-	-	JA	-	-	MYA	XB + XHI
Hard Copy	1.1	A.Hulpke	Fairware	-	-	-	-	-	-	-	MYA	*4A
YAPP (grafik)	0.29	A.Hulpke	Ej klart	-	-	-	-	JA	JA	JA	MYA	

Program för 9938 kan använda den nya videoprocessorn på följande sätt:

- Korrigera fel i äldre program som hindrar användning av 9938.

- Använda VDP-register 8 och högre för t.ex. 512 färger för de äldre 4 grafikmoderna: T1, G1-G2 och MC.

- Använda VDP-RAM för att lagra programsegment som t.ex. TELCO.

- 6 nya grafikmoder: T2 och G3-G7.

- Styrning med mus.

BILD: TIA TI-Artist
MYA MyArt
GIF Graphics Interchange Format

Tony McGovern, 215 Grinsell Street, Kotara, NSW 2289, Australien

Charles Earl, 34 McLeod Street, Ottawa, Ontario, Canada K2P 0Z5

Barry Boone, Box 1233, Sand Springs, OK 74063, USA

Lutz Winkler, 1540 Corsica Street, San Diego, CA 92111, USA

David Allen, 4418 Kansas Street, San Diego, CA 92116, USA

Alexander Hulpke, Sadowastrasse 68, D-5600 WUPPERTAL 1, Västtyskland

Achim Liese, Chattenstrasse 72, D-6500 MAINZ, Västtyskland

The MISSING LINK

The Ultimate Extended Basic Upgrade

The ultimate upgrade. The Missing Link is a powerful extension of the Extended Basic language that allows programmers to access all of the high resolution bit-mapped graphics and advanced text modes of the TI-99/4a. Before The Missing Link was developed these advanced display modes could only be accessed through assembly language programs, or by using optional and often expensive hardware. Now, using The Missing Link, ordinary Extended Basic programs, without the aid of any additional hardware, can be written to take full advantage of these advanced display modes.

High-speed subroutines. The Missing Link consists of over 30 assembly language subroutines that replace the usual methods of accessing the computer display through Extended Basic. With these high-speed subroutines many text, cartesian graphic, turtle graphic, sprite graphic, windowing and miscellaneous peripheral operations can now be incorporated into any Extended Basic program. Novice and expert users alike will find these subroutines easy-to-use, and also fully explained in the 32 page manual included with The Missing Link.

Incredible text functions. Using the text functions found in The Missing Link information can be displayed and input to and from the screen. Text can be displayed both horizontally and vertically with automatic word wrap in a window of any size. The character text size can be changed permitting up to 32 rows by 60 columns to be displayed on the screen. Different sized text can also be displayed simultaneously on the same screen.

Awesome graphics power. A tremendous amount of bit-mapped graphics functions are also available in The Missing Link. With cartesian graphics, points, lines, circles and boxes can be plotted on the screen. Turtle graphics can be used without the ink and color restrictions typically found in Logo. Using the advanced sprite routines up to 32 moving sprites can be defined and controlled simultaneously. Best of all, there are no limits when combining the advanced text and graphics capabilities on the screen.

"Through Extended Basic, The Missing Link allows anyone to access all of the incredible graphics and text capabilities found in the TI-99/4a. This was something people said could never be done... we did it."

Steve Lamberti
President of Texaments

It does windows. With The Missing Link you can display an unlimited amount of windows without any size or color restrictions. Text may be displayed in or input from a window; graphics may be generated inside and outside the boundries of a window. Multiple windows can even be overlapped and text or graphics output controlled within window boundries.

TI Artist compatibility. In addition to its remarkable text and graphics capabilities, The Missing Link can also display and save full color TI Artist pictures. Furthermore, The Missing Link can perform full bit-mapped screen dumps of any current display.

The first one is on us. Included free with The Missing Link is PaperSaver, the first program ever written with and for The Missing Link. PaperSaver is an impressive utility program that, for the first time, lets you see precisely how text prepared with TI Writer is going to look before it is printed.

Go ahead, try it! For only $3.00 (shipping included) we will send you a Live Demostration of The Missing Link that demonstrates almost every function of The Missing Link and PaperSaver. The Live Demostration is written entirely in Extended Basic and is a true representation of what can actually be done with The Missing Link. There is no better way to see what The Missing Link can do (unless you buy it, of course).

Order today. Not only is The Missing Link powerful, but it is affordable as well. For only $24.95 (plus shipping) you get The Missing Link, the PaperSaver utility, a comprehensive 32 page manual, and The Missing Link Live Demonstration.

Requirements. A TI-99/4a system with 32K memory expansion, disk drive system and an Extended Basic cartridge is all that is required to operate The Missing Link. An Epson compatible printer is needed to use the screen dump features. The Missing Link has been tested (but is not guaranteed) to be compatible with the Geneve 9640 (in TI mode), all Myarc and CorComp peripheral expansion cards, HRD, and the Triton/MG Super Extended Basic.

TEXAMENTS

Office: (516)475-3480 53 Center Street, Patchogue, New York 11772 BBS: (516)475-6463

Please add $2.50 for domestic first class (and Canadian) delivery, $8.00 for foreign insured air mail delivery. Orders are usually shipped within a 48 hour period. C.O.D. orders are accepted and must be placed by phone. Sorry, no credit card orders accepted.

The Organizer! For TI Base

The filing cabinet is one of the most powerful tools used today to store and organize information. Although computers are also used for this purpose, they are generally much more difficult to use than the good old filing cabinet. Not anymore.

The Organizer! is a complete information management system that stores and organizes information electronically... and it works just like a standard filing cabinet does. Each electronic cabinet consists of four drawers that can store up to 16,129 folders each (limited by actual disk size). Each folder contains a subject line to identify the folder by and six text lines for storage of virtually any kind of data.

The folders stored in each drawer may be be searched for, displayed, changed, printed, deleted and sorted by subject. The drawers can even be labeled anything you wish. All operations are fully menu driven, which makes The Organizer! as easy ot use as an ordinary filing cabinet!

Only $14.95

TEXAMENTS
53 Center Street, Patchogue, New York 11772

Please add $2.50 for domestic first class (and Canadian) delivery, $8.00 for foreign air mail delivery. Sorry, no credit card orders accepted.

TI Base is required to operate The Organizer!. TI Base may be purchased from Texaments for only $24.95 plus shipping.

Publication Index For TI Base

Did you forget where you read something? Can't remember what magazine issue contained the article you're looking for? Not sure who wrote a particular book? If so, Publications Index is for you!

With the Publications Index system, anyone, a novice or expert, can develop, organize, and maintain their own personal index of publication references in a single centralized database. Indexes can be created for any type of publication, including magazines, newspapers, books, and newsletters. Publication references can be entered into the database and later searched for, displayed, changed and printed... all with very little effort.

Publications Index can even help you better understand the powers of TI Base; all of the command files included with Publications Index can be veiwed using the TI Base editor and altered to suit your own needs.

Only $14.95

TEXAMENTS
53 Center Street, Patchogue, New York 11772

Please add $2.50 for domestic first class (and Canadian) delivery, $8.00 for foreign air mail delivery. Sorry, no credit card orders accepted.

TI Base is required to operate Publication Index. TI Base may be purchased from Texaments for only $24.95 plus shipping.

THE MISSING LINK
The Ultimate Extended Basic Upgrade
A Live Demonstration

They said it couldn't be done. We did it. For the first time ever users can now exploit the full power of bit-mapped graphics from TI Extended Basic without any additional hardware. Impossible? Not anymore... Introducing The Missing Link.

There is no better way to explain the incredible capabilities of The Missing Link than to show you exactly what it can do. So lets do it!

To run The Missing Link live demonstration you will need an ordinary TI-99/4a computer with 32K, one floppy disk drive, and TI Extended Basic. That's all. No special memory card or module is required to run the demonstration or The Missing Link itself. The demonstration will automatically load and run from TI Extended Basic.

You can order The Missing Link directly from Texaments for only $24.95 plus $2.50 shipping. It comes complete with the PaperSaver application example and an extensive 32 page manual.

Texaments
53 Center Street
Patchogue, New York 11772

Office: (516)475-3480 BBS: (516)475-6463

This live demonstration disk may be freely distributed provided no modification is made to the program and data files stored on it, and that the disk is distributed in its entirety. This includes, but is not limited to, distribution through user groups and online services (such as CompuServe, Genie, and local bulletin board systems). Under no circumstances is this disk and the files on it to be distributed to others for any charge or fee whatsoever.

Printed in U.S.A.

A utility every Geneve owner should have...

A>EXEC

How do you run TI-99/4a Option 5 Editor/Assembler programs? Most people load up GPL, then some option 5 loader, and then finally their program. With Exec, you can now run most E/A Option 5 programs directly from MDOS using one simple command! A dream come true.

Included with Exec are GETKEY and GETSTR, two short external MDOS batch file commands. With these two powerful commands you can create interactive batch files. GETKEY captures single key input, while GETSTR handles string input.

But that's not all... also included is Archiver III, the standard for file packing and compression. Archiver III saves disk space, and modem transfer time -- and its used by all the major telecommunications networks!

All Four Programs - Only $17.95

TEXAMENTS
53 Center Street, Patchogue, New York 11772

Please add $2.50 for domestic first class (and Canadian) delivery, $8.00 for foreign air mail delivery. Sorry, no credit card orders accepted.

Texaments has been exclusively authorized by Barry Boone to offer and distribute these REGISTERED programs internationally to end users.

"FirstBase has an excellent Query Editor akin to the best I have seen in MS-DOS database managers."

"Error trapping is superior and program flexibility is in the advanced range."

MICROpendium Review, January 1989

FirstBase

Database Manager
Version 1.1

Over two years in the making, FirstBase from JP Software has become an unqualified hit with users and major publications throughout the TI Community. The reason is simple - FirstBase heralds the arrival of a new breed of database management software: easy to use yet extremely flexible. Enormous file capacities, clean design, powerful IBM-style queries, global updates, and thorough documentation make FirstBase the one to beat.

Enormous Capacity

FirstBase's file capacities easily exceeds the standards set by the competition. You're no longer limited to 255-character fields - FirstBase allows 720-character fields, and up to 75 fields per record! You can have up to 32,767 records per database, making FirstBase perfect for the most storage hungry database needs.

Clean Design

FirstBase is built around a well-designed interlocking system of menus which carry you from task to task. A single keypress is all that is needed to accomplish many jobs.

Queries

Searching for information in a large database has never been easier. A query allows you almost total flexibility in creating a set of search parameters to narrow or widen your search. You can use a full range of relational operators (=, >, >=, <>, etc.) as well as the logical operators AND and OR. The advanced feature of single and multiple wildcards in search strings are also supported. Once the desired information is found, you can send the records to the screen, as existing database, or the report generator.

Global Updates

Have you ever wanted to make the same change on every record in a database, or to a certain group of records? Perhaps you want to perform some math or change the contents of a particular field from "YES" to "NO". Such operations, including four function floating-point math, are easy to do with FirstBase. Change all records or only those located with a query command.

Documentation

FirstBase comes with a 120+ page manual that extensively and clearly explains every feature the package has to offer. Special sections include a tutorial to get you up and running quickly, and an exhaustive glossary of all error messages.

Come and see what the fuss is all about. Order your copy of FirstBase today and enter a new era of information retrieval for the 99/4A and Geneve. FirstBase requires a TI-99/4A or Geneve, and either Editor/Assembler, Extended BASIC, or TI-Writer.

FirstBase is now available for only $39.95. Please send check or money order plus $2 for shipping and handling to:

JP Software
2390 El Camino Real, #107
Palo Alto, CA 94306
(415) 328-0885

- Please write for our catalog describing our complete software line for the TI-99/4A and Geneve 9640
- If you have any questions please call or write. Best time to phone is evenings (P.S.T) and weekends
- FirstBase was created by the amazing Warren Agee.

Version 1.1 - New Features!!

Version 1.1 of FirstBase has just been completed. The update includes numerous new features and enhancements including: "similarity searches" so you can find information even if you can't remember the spelling, a fast binary search, numeric sorting, full hard drive support, auto-repeat on all keys, true lower case letters, and more. The new version of FirstBase makes this powerful database manager even stronger!

Special Offer to TI-Base Owners!!

Until February 28 1990, TI-Base owners may purchase FirstBase for only $25. That's right the power and ease of use of FirstBase for only $25!

So step up to FirstBase, the powerful database that you don't have to be a programmer to use. To qualify for the discount, simply send us a photo-copy of your TI-Base manual cover.

Chainlink

JP Software™
Formerly Genial Computerware

Chainlink is one of the most challenging and highly addictive of all solitaire card games. While a high solution rate is possible if played very well, Chainlink never ceases to present a challenge to the experienced player. About half of all games can be won, but for the beginner it may take many attempts before the first solution.

Once Chainlink deals the cards out, the entire deck is visible. There are no face down or hidden cards. Because all cards are visible, no luck is involved once the cards are dealt. The outcome depends entirely on your playing skill.

As with most solitaire games, the object of the game is to build all the cards in order from ace to king in each suit. Cards are played one at a time beginning with the aces in four piles at the top of the screen, one for each suit. When all 13 cards in each suit are played to the four top piles, you've won the game!

In the August 1989 MICROpendium Chainlink received an "A." Reviewer Ruth O'Neil called Chainlink "immensely satisfying" and wrote "Obviously, it would be possible to play... with an ordinary deck of cards, but it is much more fun to use this program.

Written in 100% assembly language Chainlink features animated moving cards, sound effects, and blinding speed. According to MICROpendium "The excellent graphics and pleasant sounds... add to the game's enjoyment."

So you can hone your Chainlink skills, 50 saved games that are guaranteed possible to win are included. The manual not only describes how to play Chainlink, but also offers many hints on how to win from the game's creator, Walt Howe.

Chainlink was written by Wayne Stith based on a game by Walt Howe. Chainlink runs on a TI-99/4A or 9640 and requires either Editor/Assembler, TI-Writer, or Extended BASIC. Chainlink sells for $12.

AV-Index

AV-Index allows you to easily create high quality video cassettes labels, audio cassette labels, and audio cassette box inserts with space for listing song titles and other notes. Labels created with AV-Index can be edited, printed, or saved for later use.

A carefully designed user interface makes entering labels as natural as possible. The label is displayed on screen exactly as it will print out on paper - there's no guess work involved! AV-Index is menu driven so there are no commands to memorize. A disk catalog is always available so you don't have to guess at file names.

AV-Index features a smooth scrolling 80 column editor on the 99/4A providing an environment of unparalleled convenience for creating your labels. On a 9640, AV-Index operates in 80 column mode.

While the primary function of AV-Index is to assist you in the creation and maintenance of audio and video cassette labels, AV-Index also includes a library feature which automatically builds an index of audio or video cassettes. The index can be alphabetized, printed, or viewed on screen. There is even room for comments about each audio or video cassette.

Files are stored in a convenient Display/Variable 80 format so you easily access your files with TI-Writer or MY-Word. For those now using Asgard's Cassette Labeler, AV-Index automatically loads and converts files existing files.

AV-Index comes with extensive documentation and several sample labels on disk. As a bonus, portions of the program's assembly code are supplied on disk including QuickSort and 9640 color palette management code.

AV-Index was created by Don and Aaron West. It requires a TI-99/4A or 9640 with at least one floppy drive, Epson compatible dot-matrix printer, and Extended BASIC. AV-Index sells for $15.

JP Software
2390 El Camino Real, #107
Palo Alto, CA 94306

- Write for a complete catalog listing our products for TI-99/4A and 9640
- Send check or money order . Include $1 for shipping and handling.
- Credit Card orders (Visa, MC, AmEx) may be placed through Disk Only Software at 1-800-456-9272.

följande poster. Om vi istället använt poster med FIXED längd, hade varje post bestått av element plus utfyllnads-bytes för att fylla ut till FIXED längd. Det kan således synas som om VARIABLE-filer alltid var att föredra framför FIXED. Varför kommer man då överhuvudtaget att använda poster med FIXED längd? Svaret är att man bör använd FIXED postlängd i följande fall:

- När kassett CS1 och CS2 används. Kassett kan beklagligtvis ej använda VARIABLE-filer.

- När filorganisationen är RELATIVE (se nedan). Varje fil som öppnas från början eller vid något annat tillfälle som RELATIVE måste använda poster med FIXED längd.

- När informationen som du ska spara i sig självt har konstant längd, t.ex. om varje post är ett enstaka tal, kommer det alltid att uppta 9 bytes i INTERNAL format (oberoende av antalet decimaler).

- När du behöver läsa filer som skapats av andra program som använder FIXED format. Både Multiplan och Editor/Assembler objektfiler använder FIXED format.

Filer med VARIABLE längd bör användas i övriga fall för att spara lagringsutrymme. VARIABLE-poster måste användas om öppningssättet APPEND kommer att användas. För att datorn ska veta var en VARIABLE-post slutar och nästa börjar, så börjar varje post med en enstaka byte som anger längden. Du behöver inte bekymra dig om detta eftersom detta stoppas in automatiskt.

FILTYP

Denna kan vara DISPLAY eller INTERNAL. Strunta i vad bruksanvisningen säger om filtyper eftersom det mesta är missvisande. DISPLAY-format används alltid när data ska lagras som ASCII-tecken. Ordet "BOX" kommer t.ex. att lagras som 3 bytes: 66,79,88 som motsvarar ASCII-koden för B,O,X. På motsvarande sätt kommer 23 att lagras som 50,51. DISPLAY måste användas om respektive device kräver detta. Lagringsmedia kan använda antingen DISPLAY eller INTERNAL, men printrar som skriver tecken som ASCII-kod måste använda DISPLAY. Om du valt DISPLAY-format för en datafil, bör du endast använda ett element per post. Skälet till detta är att om du skriver två eller flera element till en DISPLAY-post t.ex.:

```
PRINT #1:"BOX";"TOP"
```

så får du endast ett element "BOXTOP" tillbaka när du läser ett element från filen. DISPLAY-filer kräver kommatecken mellan element och situationstecken omkring varje strängdata som du själv måste skriva in. Detta är mycket obekvämt vilket gör att jag inte rekommenderar flera element per post för DISPLAY-filer. Om du insisterar bör du läsa bruksanvisningen, men mitt råd är strunta i detta. Om varje post består av flera dataelement ska du använda INTERNAL-format. Jag misstänker att svårigheten med att använda flera element per post med DISPLAY-poster är en typisk orsak som gör att folk slutar att använda datafiler överhuvudtaget. Jag ska längre fram visa hur man kan lagra flera element i en DISPLAY-fil utan denna komplikation.

Vad är nu INTERNAL-format? Strängelement kodas fortfarande som ASCII-strängar men det finns en enstaka byte som längdindikator i början av varje element, vilket avsevärt underlättar när det gäller att hitta element inom en post. Så t.ex. kommer "BOX" att lagras som 3,66,79,88 och "23" (sträng) som 2,50,51. Numeriska element lagras som 8 bytes plus en längdbyte liksom när det gäller strängelement. Med 8 bytes kodning kan flyttal lagras på samma sätt som sker internt i datorn. Detta kallas Radix 100 och bruksanvisningen beskriver detta närmare. Som exempel kan nämnas att talet 23 lagras numeriskt (9 bytes) på följande sätt: 8,64,23,0,0,0,0,0,0.

ÖPPNINGSSÄTT

Detta kan vara INPUT, OUTPUT, UPDATE eller APPEND. Detta beskriver hur du vill komma åt filen vid just detta tillfälle. Med INPUT kan du endast läsa data och med OUTPUT kan du en-