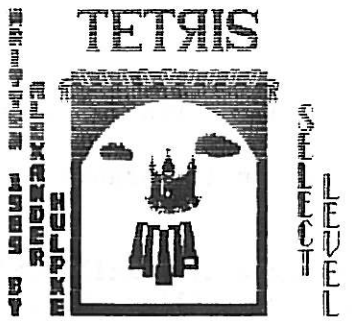
PROGRAM BITEN 89-2



FW*4*13
FIL=37 LED=15 ANV=705

filnamn	sekt	typ	längd
-READ-ME	22	DIS/VAR	80
AS	33	PROGRAM	8192
AT	22	PROGRAM	5304
C99PFI;O	2	DIS/FIX	80
CF	32	PROGRAM	7850
CG	25	PROGRAM	6060
CHARA1	5	PROGRAM	1024
CHARA2	5	PROGRAM	1024
CP	4	PROGRAM	545
CT8RAM	9	DIS/FIX	80
DP	18	PROGRAM	4138
DS	4	PROGRAM	542
EA	9	PROGRAM	1860
ED	33	PROGRAM	8192
EE	19	PROGRAM	4442
FMSAVE	6	DIS/FIX	80
FO	33	PROGRAM	8192
FP	15	PROGRAM	3566
FWDOC/EASM	33	DIS/VAR	80
FWDOC/LOAD	38	DIS/VAR	80
FWDOC/REPT	55	DIS/VAR	80
FWDOC/TIWR	25	DIS/VAR	80
FWDOC/UTIL	63	DIS/VAR	80
FWSAVE	6	DIS/FIX	80
LDFW	9	DIS/FIX	80
LH	16	PROGRAM	3836
LL	10	PROGRAM	2064
LOAD	31	PROGRAM	7661
MG	33	PROGRAM	8192
MH	20	PROGRAM	4856
QD	11	PROGRAM	2512
SAVIT	2	DIS/VAR	80
SL	10	PROGRAM	2240
SYSCON	6	PROGRAM	1198
UL	4	PROGRAM	542
UTIL1	33	PROGRAM	8152
XB4THLD	2	PROGRAM	203

(delad på 2 skivor i progr-bank)

INNEHALL

ISSN 0281-1146

UTMANING FÖR MEDLEMMAR

Det saknas medlemmar som skriver i tidning. Hitta på något eget eller välj ur följande lista. Om du behöver underlag, tag kontakt med Jan Alexandersson.

A. ÖVERSIKTSARTIKLAR

1. Databas
2. Grafik
3. Musik
4. Ordbehandling
5. Spel
6. Modem och Terminalprogram
7. Programmeringsspråk
8. Assembler för nybörjare
9. FORTH för nybörjare

B. SAKNADE TESTER I PB (FAIRWARE)

1. XB*Tools
2. Linker + Linker Librarian
3. Macro Assembler
4. Fast Term, Telco, Omega (ev även Terminal Emulator II)
5. TEST från TI av dator och box
6. Colist
7. Creative Filing System
8. Basic Sort
9. Emusic Preprocessor
10. Disk Hacker
11. Fractal Explorer
12. Pulsar

C. ÖVERSÄTTA EAST ANGLIA REGION

1-7 Spad XIII Mk2
1-9 Picasso, grafik
1-9 Legends, adventure
1-11 Super Space, review

D. MINI MEMORY TUTORIAL FRÅN EAR

Det finns 11 olika TUTORIAL om Mini Memory. Den som översätter bör skriva om det något så att det går att knappa in både med Editor/Assembler (som finns i Funnelweb) och Mini Memory. Eventuellt kanske några av de inledande avsnitten kan kortas ned.

E. ÖVERSÄTTA OCH REDIGERA SWEDLOW

Ett stort antal artiklar om Basic och XB, men de måste redigeras så att allt omodernt tas bort och allt lokalanknytet minskas eller tas bort. Bra skrivet men EJ tidlöst.

Redaktör:
Jan Alexandersson

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

Postgiro 19 83 00-6
Medlemsavgiften för 1989 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 m.m.

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 magazine nr 12/82, 1-5, 7-9/83(st)	40:-
Gamla årgångar av Programbiten	50:-
TI-Forth manual	100:-
Hel diskett ur programbanken(st)	30:-

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

NYA PROM FÖR MODUL & KORT

av Jan Alexandersson

Det finns en modifierad PROM att montera i Super Extended Basic-modulen, TI Diskkontrollkort och TI RS232 om du har en TI-99/4A.

Inga av dessa modifieringar fungerar med Myarc Geneve beroende på den universal-DSR som finns i MDOS (Myarc DOS) som gör att MDOS från början måste känna till DSR-namn m.m. Myarc har dock vunnit snabbhet men förlorat flexibilitet. Läs mer om Myarcs förhistoriska monster i McGoverns artikel om DSR-LINK. Du som har en Geneve kan ej räkna med att kort med helt nya funktioner kan användas. Jag tror dock att allt fungerar så länge som det nya kortet har ett välkänt DSR-namn men jag vet inte om det även finns restriktioner när det gäller CRU-adresser eller texten efter punkten i DSR-namnet.
Följande priser gäller:

MULTI-MOD för Super-XB	USD 22.95
Disk Controller upgrade	USD 19.95
TI RS232 Upgrade kit	USD 14.95
Porto till Europa	USD 2.00

Dessa säljs av:
John Guion
P.O. Box 4628
Lubbock, TX 79409
USA

MULTI-MOD för Triton Super Extended Basic modul ger dig Editor/Assembler, Disk Manager III och TI-Writer samtidigt som Super-XB finns kvar. Programfilerna till EA och TIW finns fortfarande på fristående flexskiva. Alla dessa fyra moduler kan sedan väljas från TI:s startmeny. Disk Manager III kan komma åt DSK1-DSK9. Du monterar den nya PROM i en hållare i modulen så du behöver inte löda. Innan du skaffar detta bör du ta dig en ordentlig titt på Funnelweb (finns i Programbanken på två skivor) som jag tror ofta är ett bättre alternativ än denna PROM.

Upgrade kit för TI Diskkontrollkort ger dig möjlighet att ha fyra flexskiveenheter samtidigt. Du kan även adressera med "dsk1" som ett alternativ till "DSK1" vilket kan vara bra om en RAM-disk har samma nummer. Stegtiden kan minskas från 20 till 12 ms. Du måste välja detta före köpet eftersom olika PROM levereras för de båda stegtiderna. Alla dina flexskiveenheter måste klara den stegtid du väljer. Observera att modifieringen INTE ger Double Density. Det är nödvändigt att LÖDA genom att byta två ROM och placera ytterligare några chips ovanpå de befintliga.

Upgrade kit för TI RS232 ger några nya DSR-namn. TP emulerar Termal Printer till en Epson-kompatibel skrivare som även ger skärmdumpning av ett XB-program. SIO är en förprogrammerad inställning av RS232 med valfria mjukvaruswitchar t.ex. kan RS232/2.BA=4000.DA=8.PA=0 anropas med enbart SIO. Observera att PROM måste skräddarsys till din egen printer för både TP och SIO i samband med beställningen. Denna PROM saknar de modifieringar som DIJIT har gjort för RS232. Du måste löda för att byta PROM på kortet.

OVERLAY FÖR TANGENTBORD

av Jan Alexandersson

Det finns nya mallar (overlay) till tangentbordet utgivna av SouthWest Ninety-Niners i USA. Exempel på liknande mallar kan du se i PB 88-4 sid 16. Den nya mallen består av mallar för fem olika moduler som levereras hopsatta, men det går även att klippa isär dessa. De är tryckta med mycket svarta bokstäver på styvt papper som har plastbehandlats på båda sidorna. Mallar finns för följande fem moduler: Konsolen, TI-Writer, Multiplan, Editor/Assembler och Terminal Emulator II. Jag har skaffat 10 exemplar från USA för att minska portot per styck. Om du vill ha en mallsats kan du sätta in 22 kr på mitt personliga postgiro 61 68 86-8. Om mina exemplar tar slut så kan du räkna med längre leveranstid men jag beställer då nya från USA.

ANNONS: Köpes: Expansionsbox med TI diskdrive, kontroller & 32kb minne. Telefon 0451-16468 Jesper Nordström

---

NYHETER TILL FUNNELWEB

av Tony McGovern, Australien

Originalversionen av Ottawa DM 1000 version 3.5 har en bug som gör att en tillfällig DATA-lagring utförs som en instruktion. DM 1000 tillhörande Funnelweb är vad vi vet felfri förutom en bug vid ändring av skivnamn (ej blanktecken efter namnet).

Mjukvaruutvecklingen här har mest ägnats åt en betydande utökning av QDAV-katalogprogrammet för AVPC-systemet. Detta inkluderar alla funktioner som SD (Show Directory) i Funnelwebs editor till AVPC har, tillsammans med en rad nya funktioner. Programkodens storlek har ökat till 32 sektorer från tidigare 14 i SD och 11 i QD (Quick Directory). För närvarande finns programmet som en fristående programfil för utvecklingsändamål, och en integration med AVPC-editorn till Funnelweb har skjutits på framtiden. Det har givit mig några idéer till ett förbättrat QD-liknande program för den vanliga 40-kolumnsversionen vilken måste finnas kvar som en egen fil. Nya funktioner är bl a komplett tillgång till Funnelwebs programladdare, och direkt laddning av XB-program om XB-modulen används, både som programfiler och längre INT/VAR 254-filer. Även "View file" kommer att tillåta skrollning i båda riktningar, enkel eller självrepeterande, per rad eller sida, till "Display file" av godtycklig längd, och om fullt VDP-RAM (192 kbytes) finns, samtidig "View file" av två filer (från oberoende 64K-buffertar tillräckligt stora att hålla så gott som alla filer som kan dyka upp) på skärmen med av varandra oberoende skrollning. Hela eller markerad del av filen kan skrivas ut från bufferten.

Ett beslut som jag nyligen har kommit fram till är att INTE KÖPA en Geneve. Detta avviker från den tidigare attityden där jag tänkte att det kunde vara en bra sak men bara inte kommit mig för att skaffa den. Skälet till detta är i första hand mjukvaran och informationsförhållandena. Jag har upptäckt att jag inte är ensam om min slutsats att en liten inre cirkel av mjukvaruutvecklare försöker att hålla allt för sig själva, och utestänger alla andra från seriös mjukvaruutveckling. Lärde de sig inget från TI:s misslyckande? För mig får de gärna leka med detta själva. Kombinera detta med opålitlig och icke komplett systemmjukvara, och en liten användarbas så blir det mycket oatraktivt att ge sig in i detta som en utomstående eller ny användare. Detta är kanske synd, eftersom en 9995-baserad maskin skulle vara intressant att ge sig på med all vår erfarenhet av 9900. Det enda sättet för maskinen att lyckas är att Myarc gör ett fullständigt öppet system, och när de ännu inte har gjort det, så tror jag aldrig att detta kommer att ske. Som det nu är kommer AVPC att öka livslängden för 99/4A så bra att jag känner att jag kan kosta på mig att vänta ännu längre tills en framtidslösning dyker upp. I Australien är den tydliga vägen att gå för närvarande när der gäller hemdatorer (till skillnad från kontorsdatorer) Amiga A500. William har gått den vägen men jag vill vänta och se vad som kommer efter detta.

Jag har en annan liten sak som borde vara känd tidigare. Jag vill använda <fctn-V> som funktionstangent i QDAV och jag har aldrig tidigare tänkt på vilken ASCII-kod den gav. Ingen manual, inte ens tangentborden i Basic-manualen som är min vanliga referens när det gäller detta, beskriver detta. Koden som ges från två olika konsoler hos oss är >7F (decimalt 127), ASCII-koden för "Delete". I alla teckenset hos oss ekar den tillbaka på skärmen som blanktecken, men det finns ingen "Delete" i vare sig TI-Writer eller Basic. Manualen listar Delete (127) som ett tecken som ges av en tangent utan att tala om vilken. Finns det någon som vet om detta är <fctn-V> på alla konsoler?

QDAV 9938 MENY/KATALOG

av Jan Alexandersson

DIRECTORY UTILITY FÖR 9938

Det har kommit ett helt nytt program från Tony McGovern, Funnelweb Farm, Australien. Tony kallar QDAV "Directory Utility" för att markera att detta är mycket mer än ett katalogprogram. QDAV måste alltid laddas via Funnelweb och kräver videoprocessorn 9938 som finns i DIJIT AVPC, Myarc Geneve eller Mechatronic 80-kolumnskort. Programmet finns på en skiva i programbanken som även har 80-kolumnseditorn (ED/EE) till Funnelweb. Skivan AVPC/QDAV har datum July/6/89 och passar som tilllägg till alla 9938-system men kan inte fungera utan den vanliga Funnelweb som finns på två skivor i programbanken nu med datum July/4/89.

När du använder 60 Hz bildfrekvens bör du ej placera monitorn ovanpå expansionsboxen eftersom monitorn störs av nätdelens 50 Hz.

KATALOG AV FLEXSKIVA

Till det yttre liknar QDAV Show Directory till 80-kolumnseditorn men innehåller ett stort antal nyheter. Alla enheter som kan anropas med "DSKx." kan användas där x kan bytas mot godtycklig siffra, bokstav eller tecken. Den mest närliggande användningen är att skriva ut flexskivors kataloger. Utskriften görs med två kolumner vilket gör att det ryms mycket på ett papper. PROGRAM-filer markeras med BX om de kan laddas av Basic/XB eller EA om de kan laddas av Editor/Assembler. Med QDAV kan du även skrivskydda filer, ta bort skrivskydd, radera filer, skriva ut katalog, titta på filer och ladda program.

PROGRAMLADDARE

Med QDAV kan du ladda alla sorters program direkt från katalogmenyn. Alla programtyper som Funnelweb kan ladda kan startas men även Extended Basic-program från en Assemblermiljö, vilket jag aldrig tidigare har sett till 99/4A. XB-laddaren använder direktadressering av GROM så du måste ha en TI-kompatibel modul (Triton Super-XB fungerar bra). De laddare som funnits tidigare har alltid laddat XB-program från en XB-miljö vilket ju inte är så svårt. Följande program kan således startas med QDAV:

- DIS/FIX 80 objektfiler
- DIS/VAR 80 FW script-filer
- XB PROGRAM-filer (Basic)
- XB INT/VAR 254 (Basic)
- EA PROGRAM-filer (Assembler)

TITTA PÅ FILER

"View file" är avsevärt utökad så att den klarar DIS/VAR, DIS/FIX, INT/VAR och INT/FIX oberoende av bredd. Du kan även titta på PROGRAM-filer, filhuvud och diskhuvud, men då kommer data upp samtidigt i två former, dels som hexadecimal kod och dels som ASCII-kod ungefär som du kanske har sett i många disassemblerare. Även med INT/VAR och INT/FIX kan du ändra presentationen till HEX och ASCII efter att den har laddats in i bufferten.

Du kan läsa in en 64 kbytes lång fil i tittbufferten som finns i det stora VDP-minnet som 9938 alltid har. När filen finns i bufferten kan du skrolla framåt och bakåt samt hoppa till början eller slutet m.m. Det är även möjligt att skriva ut hela filen eller markerad del av filen.

Om du har 192 kbytes VDP-RAM som jag har till min DIJIT AVPC så kan även en andra fil läsas in till buffert nummer två. Detta kan ej användas till Geneve eftersom denna endast har 128 kbytes VPD-RAM. När två filer finns i buffertarna kan du se båda samtidigt på en delad skärm men även skrolla båda varför sig.

DSR-LINK OCH RS232/PIO

av Tony McGovern, Australien

Den funktion hos TI-99/4A som kanske mest har bidragit till dess långa liv som "orphan" är dess mycket generella sätt att lägga till yttre enheter till systemet, på ett sätt som är oberoende av interna detaljer i varje enhet (device). Alla enheter adresseras på ett enhetligt sätt som en del av ett filhanteringssystem, där var och en är ansvarig för sina egna åtgärder med sin egen Device Service Routine (DSR), vanligen i ROM, som switchas in för varje särskilt kort. De yttre enheterna anropas med namn och och den enda fysiska förinställning som är nödvändig är att ställa in CRU-basadress på kortet för att undvika konflikt när olika DSR kopplas in till en adress som TI ej har använt. All adressering görs med enhetens namn så den fysiska placeringen i expansionsboxen eller CRU-adressen saknar betydelse i anropande program. Detta system står sig mycket bra även vid en jämförelse med de flesta hem- och persondatorer som säljs idag.

De flesta program aktiverar en speciell enhet genom att anropa en standardtyp av gränssnittsprogramdel kallad Länkare för DSR (DSRLNK) som accepterar anropande data i en generellt angiven form som sköter om detaljerna för anslutning till enheten. Data anordnas i en Peripheral Access Block (PAB) vilken för 99/4A finns i VDP-RAM med pekare i en standardadress i CPU PAD RAM. Detta liknar "File Control Block" hos andra system, men TI:s namn indikerar fullständigheten i dess tillämpning för 99/4A. Inget krångel med att ladda "device driver" eftersom när enheten är aktiverad till 99/4A så har den med sig sina egna drivrutiner. MDOS till Geneve ser ut att vara ett ordentligt steg tillbaka till ett levande fossil när det gäller detta. Några funktioner adresseras direkt genom hårdvara i konsolen, video, ljud och tal, de två första pga snabbhet och den sista pga kostnader. Det finns ett pris att betala för detta enkla anrop med namn av yttre enheter, nämligen snabbhet i anropet eftersom kortet ej är förinställt på en standardväg, utan söks för varje enstaka anrop med en DSRLNK. Detta kan snabbas upp vid upprepade anrop av samma enhet, det är möjligt att gå ännu längre genom att adressera enhetens hårdvara direkt om det ej går att få tillräckliga prestanda på annat sätt för speciella uppgifter. Detta är liktydigt med att använda dina egna drivrutiner och skriva direkt till hårdvaran. Med tanke på det begränsade RAM-minnet i 99/4A är detta något som en programmerare endast gör när det finns ett tvingande behov av detta. När 99/4A blev övergiven av TI var detta inget stort problem eftersom det fanns mycket få alternativa kort, men det har blivit mer och mer besvärligt för programskrivare att hålla reda på alla olika sorters kort till expansionsboxen för varje funktion.

Avsikten med denna artikel är inte att gå in på detaljer vad gäller PAB och filhantering vilket är utförligt behandlat på andra ställen, inte minst i Editor/Assembler-manualen. Det jag kommer att behandla är själva DSRLNK, vilket är ett ämne som TI aldrig har behandlat detaljerat. Så vitt jag vet finns den enda källan från TI i den kommenterade källkoden till FORTH. Inte ens där är kommentarerna helt tillförlitliga. DSRLNK kommer i många varierande former i tillämpade program, där de huvudsakliga variationerna är graden av felhantering i själva DSRLNK, och hur mycket intern information som sparas utanför rutinen. Konsolens ROM innehåller en DSRLNK (och gör även en GSRLNK för att söka GROM) men denna är inte lättillgänglig även om den är möjlig att använda. TI:s egna välkända nyttoprogram TI-Writer och E/A som körs i expansionsminnet har båda sina egna DSRLNK, liksom E/A-utilities och Mini Memory, och ingen av dessa använder konsolens programkod.

Jag har faktiskt inte ens tänkt gå in på några detaljer när det gäller programkoden särskilt mycket om ens

något, men låt oss först se vad som händer. För det första kommer varje DSRLNK att rensa Status <=> bit vilken är den huvudsakliga felflaggan vid återhopp. Eftersom DSRLNK vanligen är BLWP-rutiner så görs detta genom att trixa med lämplig bit i R15 vilket sedan laddas in i statusregistret när återhopp görs med RTWP. PAB i VDP-RAM hittas sedan genom den pekare som lagras i PAD-adress >8356. (PAD finns på >8300 i alla sålda 99/4A trots att TI:s DSR-specifikation tillåter att den finns varsomhelst, och väljes med en STWP-instruktion från GPL-workspace pekare på PAD+>E0). Detta pekar på längdbyten för device-namnet i PAB som är allt det anropande programmet, vare sig det är Basic eller nyttoprogram, redan har satt upp. Namnet hämtas till PAD, typiskt vid FAC (PAD+>4A) och dess längd kontrolleras samtidigt. Device-namnet före första punkten kan vara högst 7 tecken långt om inte DSRLNK skall ge ett felmeddelande, även om några DSRLNK skrivna av TI tillåter 8. Tilläggsnamn efter punkten i device-namnet hanteras av DSR självt, som filnamn eller mjukvaru-switchar eller vad som passar.

Nästa steg blir att söka efter device-namn i alla DSR. Detta görs normalt med GPL-workspace satt för snabbheten av att vara på 16-bitars buss. DSR-ROM aktiveras en i taget med början på >1000 i en normal DSRLNK och i var och en av dessa söks den länkade listan av namn igenom. Se den tekniska manualen eller andra källor om du vill följa upp detaljerna. Den första pekaren till denna lista är på >4008, en ursprunglig offset från >4000 ställs in med DATA 8 specificerad i E/A-manualen att följa efter anrop av BLWP @DSRLNK. Den andra nyttiga data som kan följa detta anrop är DATA 10 vilket startar sökning i listan av underprogram (CALL), t.ex. att hitta sektor skriv/läs-rutinen i flexskive-DSR. Inte alla DSR-anrop i program följer exakt detta format. TI-Writer Editors DSRLNK utgår endast från anrop av filtyp och anropet följs inte av DATA. Funnelweb har följt detta mönster på gott och ont, och det sparar några ord i huvudprogrammet för att slippa upprepa samma DATA många gånger. Dessa DSRLNK bibehåller tillräcklig allmängiltighet och söker genom alla DSR-adresser. Förenklade versioner har publicerats vilka förutsätter en speciell CRU-bas och låter bli att söka alla DSR. Dessa är inte lämpliga för allmänt bruk. Utökningar som kan handha flera RAM-diskar har tidigare behandlats i HV99 News och i andra publikationer.

Om device-namn inte hittas så kommer nästa DSR att sökas igenom till slutet eller tills namnet hittas. När namnet hittas kommer GPL-workspace att sättas om det inte redan används och startadress att sättas i GPL-workspace R9 och DSR att startas med BL *R9. TI:s specifikation för återhopp från en DSR är att hoppa över nästa ord i den anropande koden med ett INCT R11 eller motsvarande före återgången, trots att ingen förklaring ges till varför detta skall göras annorlunda än power-up- och interruptrutinerna. Detta ord i DSRLNK är vanligen JMP för att återuppta sökningen i nästa ROM. Mer om detta senare.

Normala E/A-liknande DSR sparar viss information som fåtts under vägen. I E/A-utilities görs detta i UTLTAB med flera standard namn av vilka de följande är viktigast:

SAVCRU - CRU-basadressen för den DSR-ROM där device-namnet fanns.

SAVENT - aktuell startadress för DSR

Det finns även SAVVER avsedd för versionsnummer av DSR. Om du vet att du kommer att anropa samma DSR upprepade gånger så är det möjligt att snabba upp saker genom att hoppa in direkt och hoppa över sökningen av DSR. E/A 3 Load/Run gör detta för att snabba upp läsningen av DIS/FIX 80-objektfiler. Detta är ett trick som inte tycks användas så ofta i andra program men ett exempel är LINEHUNTER-programmet i Funnelweb som snabbas upp genom att använda denna teknik för både master- och kopiefilen.

SAVVER tycks inte användas av TI på det sätt som antyds av E/A eller andra TI dokument. Detta kom fram helt nyligen i form av en fråga från

Geoff Trott i Wollongong om varför det ibland är problem med att anropa ett andra RS232-kort i hans system. Detta var något jag aldrig tidigare gått in på eftersom vi inte haft något speciellt intresse av kommunikation och således ej heller av detaljer i RS232. Så jag letade fram min kopia av Colin Hinsons utmärkta detaljerade beskrivning av RS232 DSR-ROM och där stirrade det rakt i mina ögonen. I sammanfattning var den fråga som behövde besvaras hur två RS232-kort med exakt samma DSR-ROM är användbara på olika CRU-adresser, vilket vad jag förstår är den enda ändringen på det andra kortet. Varför kommer inte device-namn som RS232/3 och PIO/2 som är avsedda att adressera det andra kortet att få det första kortet att svara? Det hemska otrevliga faktum är att RS232-kortet använder ett odokumenterat tillägg till DSRLNK. R1 i GPL-workspace används som en räknare av det antal gånger en DSR med riktigt namn har hittats. Anrop av device-namn tillhörande det andra RS232-kortet undersöker om R1 innehåller 2 och om inte hoppar vidare utan INCT R11 så att sökningen fortsättes. Detta förklarar CLR R1 före sökningen av ROM och INC R1 före hoppet till DSR som du hittar i en vanlig DSRLNK. Jag hade lagt märke till dessa som till synes meningslösa instruktioner tidigt i samband med utvecklingen av Funnelweb. När de togs bort så fungerade flexskiveenheterna bra men det upptäcktes något senare att RS232 inte längre fungerade. Eftersom detta inte är ett kort som uppför sig bra överhuvudtaget, så antog jag bara att det tillhörde underligheterna i kortet och återställde koden i DSRLNK. Nu vet jag varför det finns där!

Detta är kanske en ny upptäckt på Funnelweb Farm men det borde ha varit känt för ett antal andra människor, inte minst de som skriver nya DSR. Det kan ha diskuterats lång tid tillbaka, men eftersom detta är det första vi sett av detta här så är det troligen nytt för de flesta läsare. Nu när jag skrivit detta, kan någon komma och påpeka var det hela tiden har stått och stirrat på mig! Jag antar att någon som bestämmde på TI insisterade, när möjligheten av två RS232-kort i expansionsboxen aktualiserades, att det gjordes utan att byta ROM i det andra kortet. Små ändringar i ROM mellan de två korten är uppenbart det bästa sättet att göra det, men dyrare för tillverkaren. Så detta extra lades till DSRLNK och har alltid kopierats sedan dess. I källkoden till TI-FORTH är det kommenterat som samhörande med funnen version, och sparat på SAVVER. Ingenstans har det uppstått någon tvetydighet mellan detta och DSR-ROM version som lagras som byte två i respektive huvud, och ingen TI dokumentation av något slag som jag har sett behandlar frågan och hur det används i RS232. Kanske var alla för generade av detta brott mot den väsentliga elegansen hos 99/4A-systemkonstruktionen för att fästa det på papper.

Allt detta förklarade inte varför vissa DSRLNK syntes fungera med RS232/3 och andra inte trots att de hade mekanismen för att räkna i R1. Så uppmanad av ett påpekande från Geoff Trott rörande detta, tittade jag ännu en gång på TI-Writer DSRLNK, och uppenbart fanns det där. Det som hände var att denna speciella DSRLNK använde R6 för att lagra ett värde mellan hopp till kortrutiner, och detta var samma register som användes av RS232-DSR för att undersöka R1 genom att göra LI R6,1 eller 2 beroende på resultatet av C R1,R6. Så det första kortet förstörde DSRLNK-funktionen, men bara när det hoppade ut via denna speciella utgång för att leta efter namnet på nytt. Jag gissar att en del av TI Lubbock aldrig klart berättade för andra berörda exakt vad de hade gjort med RS232. Kom ihåg att RS232-kort fanns långt innan TI-Writer kom fram. Samma kod som orsakar detta problem finns även i Assembler men TI-Writer Formatter är annorlunda, och liknar mer E/A-DSRLNK och fungerar utan problem. Denna bekymmersamma kod blev också inbyggd i Funnelweb LOAD/UTIL1. Detta problem kan tas bort med patch av koden vilket har gjorts för både 40 och 80 kolumnseditorn till Funnelweb 4.13 från Mar 03/89 eller senare. Detta visar bara att även den mest genomarbetade kod fortfarande kan innehålla överraskningar.

ASSEMBLER 2

av Tony McGovern, Australien

Låt oss fortsätta att titta på hur man krymper bort bytes i assemblerkod genom att titta på flyttning av block av dataord i CPU-RAM. Antag att detta block är av känd längd och att det skall kopieras från en bestämd adress till en annan. Med andra ord kan allt detta representeras med symboler eller uttryck i din källkod till assembler. Det närliggande sättet att koda detta är

```
      LI   R0,BLOK1
      LI   R1,BLOK2
      LI   R2,BLLEN
LOOP  MOV  *R0+,*R1+
      DECT R2
      JGT  LOOP
```

Register 0-2 har laddats med passande värden, och sedan räknas ordskiftningen ned. Att använda JGT istället för JNE är försiktig kod ning på samma sätt som försiktig bilkörning. Detta tar 9 ord (18 bytes) och använder 3 register. Låt oss prova en första krympning

```
      LI   R0,BLOK1
      LI   R1,BLOK2
LOOP  MOV  *R0+,*R1+
      CI   R0,BLOK1+BLLEN
      JL   LOOP
```

Nu tar det ett ord (2 bytes) mindre och behöver endast 2 register. I en komplex kod kan registerekonomi vara viktigare än 2 sparade bytes. Notera att en logisk jämförelse används eftersom adresser jämförs i stället för test av nedräknare. Låt oss göra det ännu bättre!

```
      LI   R1,BLLEN
LOOP  MOV  @BLOK1-2(R1),@BLOK2-2(R1)
      DECT R1
      JGT  LOOP
```

Detta tar nu bara 7 ord av koden och använder endast ett register. Observera att R0 inte kan användas här. Detta skiljer sig från den tidigare koden genom att det startar från högsta ord och arbetar sig nedåt. Om inte blocken överlappar varandra så spelar det ingen roll. Om du har läst E/A-manualen flitigt så har du redan känt till detta sätt att skriva kod genom att läsa sidan 101. Om blocken överlappar så blir du tvingad att välja.

PUSH, TETRIS

av Jan Alexandersson

Det finns nu två mycket bra spel på flexskiva som du kan beställa från programbanken för 30 kr styck.

Perfect Push av Martin Flodén (medlem i föreningen) är mycket professionellt gjort med fin grafik, musik och tal. Talet fungerar utan Speech Synthesizer. Detta är det enda program jag har sett som har talet ordnat på detta sätt. Du måste ha 32 kbytes expansionsminne och Editor/Assembler, Mini Memory eller Funnelweb med godtycklig modul. Spelet går ut på att plocka ihop en raket av de delar som är utspridda över spelplanen (planeten). Du kan endast flytta delarna genom att knuffa dem (push) så det gäller att logiskt tänka igenom hur delarna ska flyttas så att raketdelarna kommer på rätt plats. Det finns risk att du knuffar delarna till kanten av spelplanen och då kan du inte få dem på rätt plats. Du hindras hela tiden av ett monster som du måste hålla dig ifrån. När du har alla delar på plats så startar du raketen som fortsätter till en ny planet med ökade svårigheter.

TETRIS av Alexander Hulpke i Tyskland går ut på att stapla fallande stenar på ett golv så att det blir så litet mellanrum som möjligt mellan stenarna. Du kan flytta i sidled och rotera stenen runt dess axel så att den passar så bra som möjligt på golvet. Varje sten är en slumpartad hopsättning av fyra kvadrater ("Tetromino") som då bildar en avlång sten, vinklad sten eller en större kvadrat. Denna typ av spel finns även till Atari, Amiga och PC. TETRIS till TI-99/4A har mycket fin grafik och musik och en från spelplanen fristående startskärm. Tetris kräver 32 kbytes EM och Extended Basic, Editor Assembler, TI-Writer eller Funnelweb.

XHI - GRAFIK FÖR 9938

av Jan Alexandersson

Ett helt nytt fantastiskt program för DIJIT AVPC, Geneve eller Mechatronic med videoprocessorn 9938 har kommit från:

Alexander Hulpke
Sadowastrasse 68
D-5600 WUPPERTAL 1
Västtyskland

Du kan beställa det från adressen ovan genom att sända lämplig donation (jag rekommenderar minst 20 DM) samt 7 DM för disk och porto. Skaffa utlandsblanketter till postgirot och sänd 27 DM. Programmet ger dig möjlighet att använda ett stort antal CALL LINK för att komma åt högupplösande grafik från ett Extended Basic-program. En manual på 20 sidor finns på skivan till XHI version 3.4 från 9 aug 1989.

Det finns en ny PROM till Mechatronic 80-kolumnskort. Tag kontakt med Barry Boone (Archiver-författaren).

1. GRAFIKMODER FÖR 9938

Av de tio olika grafikmoder som finns till videoprocessor 9938 är sex helt nya medan de övriga fyra även finns på den vanliga 99/4A.

Text1-2 eller Graphics1 väljer du om du i första hand vill åstadkomma text. Dessa tre moder är de enda som har gemensamt teckenset för hela skärmen. Text2 bör användas för alla nya program där mycket text skall visas för t.ex. ordbehandling, kalkylering och databas. Graphics1 kan trots detta vara det rätta för startskärmar och menyer där man enkelt får stora och feta bokstäver om nu 32 tecken per rad räcker. Du kan även ordna grafik med ett visst krångel till Graphics1 med CALL CHAR eller motsvarande.

Graphics6-7 väljer du om du i första hand vill använda grafik. Graphics6 ger flest punkter (512x212) på skärmen om 16 (av 512) färger räcker. Om du vill prioritera antalet färger så tar du Graphics7 som har 256 färger samtidigt men färre punkter (256x212). Graphics6-7 kan med ett visst krångel användas för text. Tänk bara på att resultatet blir pixels så det går inte att läsa text med CALL GCHAR eller motsvarande i assembler.

Graphics2 har skärmen delad i tre delar som var och en har sitt eget teckenset. Varje tecken kan ha många färger samtidigt eftersom varje pixelrad har sin egen motsvarighet till CALL COLOR. Graphics2 är främst intressant om du även vill kunna köra på 99/4A. Graphics3 är endast bra för gamla program skrivna för Graphics2 där man vill gå över till flerfärgade sprites (mode2) utan att behöva ändra koden i övrigt. Kanske något för Parsec. Graphics4 är endast bra för gamla 99/4A-program skrivna för Graphics2 där man vill gå över till äkta bitmap med egen färg för varje pixel utan att ändra koden särskilt mycket. Kanske något för TI-Artist.

Graphics5 klarar endast fyra färger men behöver bara 32 kbytes vilket kan passa bra om man vill växla mellan ett större antal bilder som samtidigt finns i VDP-RAM.

2. SPRITE-MODER FÖR 9938

Spritemode2 som används för Graphics 3-7 ger dig flerfärgade sprites (max 32 stycken) där varje pixelrad har sina egna attribut som färg, kollision, prioritet och "early clock". Du kan se 8 sprites på samma pixelrad innan någon skyms. Med Magnify 3 eller 4 har du möjlighet till 16 olika attribut eftersom en sprite med fyra tecken har 16 rader. På samma sätt som för Graphics2 kan du inte använda auto-motion av sprites vilket betyder att programmet självt måste sköta flyttningen.

3. FÄRGER MED 9938

De fyra vanliga grafikmoderna som även finns för 99/4A får med 9938 en

möjlighet att definiera om färgerna så länge man har max 16 olika av 512 möjliga för Graphics1-2 och 2 av 512 möjliga för Text1. Även Graphics6 har 16 av 512 möjliga färger för både pixels och sprites medan Graphics7 har 256 av 256 färger för pixels och 16 av 16 färger för sprites. Dessa 16 sprite-färger skiljer sig från 99/4A-färgerna i tabellen nedan men är samma normalfärger som används i Myart för Geneve. De 512 färgerna ställs in genom att de tre grundfärgerna röd, grön och blå sätts till ett värde mellan 0-7. För Graphics7 får man 256 färger genom att blått endast kan ställas in i 4 nivåer.

Den normala inställningen av de 16 färger som motsvarar 99/4A är:

Nr	Färg	Röd	Grön	Blå
0	Transparent	0	0	0
1	Svart	0	0	0
2	Grön	1	6	1
3	Ljusgrön	3	7	3
4	Mörkblå	1	1	7
5	Blå	2	3	7
6	Mörkröd	5	1	1
7	Cyan	2	6	7
8	Röd	7	1	1
9	Ljusröd	7	3	3
10	Mörkgul	6	6	1
11	Ljusgul	6	6	4
12	Mörkgrön	1	4	1
13	Magenta	6	2	5
14	Grå	5	5	5
15	Vit	7	7	7

Färg 0 ändras endast om skärmfärgen sätts till 0 eller om VDP-register R8 bit 5 sätts till 1. I Basic och XHI numreras de 16 färgerna 1-16 (motsvarar 0-15 i tabellen).

4. ATT LADDA IN XHI

Programmet XHI laddas från Extended Basic med CALL INIT :: CALL LOAD ("DSK1.XHI"). Du får tillgång till ett stort antal CALL LINK för Graphics6 och Graphics7. Du kan hela tiden i ett program växla mellan den vanliga XB-skärmen (Graphics1) och högupplösande skärm (Graphics6 eller 7). Du kan både skriva till XB-skärmen och XHI-skärmen och dessa finns kvar oförändrade i minnet även när den andra skärmen visas. Med aktiverad XHI-skärm kan du fortfarande använda alla XB-kommandon som läser och skriver på XB-skärmen trots att den inte syns på monitorn. XHI-skärmen visas utan störningar även när olika programrader utföres. Det är inget problem med t.ex. SINUS som är svårt med 99/4A och Graphics2. XHI-LINK kan dock endast användas när XHI-skärmen är aktiverad. CALL SCREEN och CALL MAGNIFY påverkar XB- och XHI-skärmen samtidigt eftersom de endast skriver till VDP-register.

5. CALL LINK MED XHI

Alla XHI-kommandon kan även förses med logiska operationer IMP, AND, OR, XOR och NOT. Om du laddar in en ny Myart-bild så kan den adderas till tidigare bild med OR. Högupplösande grafik anropas med t.ex. CALL LINK ("PLOT",VCOOR,HCOOR,COLOR) enligt följande:

a.Grafikmoder

```
HICLR  Graphics6 + rensa skärm
HIRES  Graphics6
CLR256 Graphics7 + rensa skärm
MOD256 Graphics7
NORMAL Graphics1
COLMIX definiera om valfri färg
COLRES återställer färger
```

b.Grafik

```
BACK   skärmfärg (max 256 olika)
även CALL SCREEN (endast 16 färger)
DCOL   färg att rita med
PLOT   rita punkt
CLS    rensa/färga pixel-ytan
GPIX   läsa punkt
CIRCLE rita cirkel
LINE   rita linje
DRAWTO fortsätta linje (turtle)
SAVE   spara Myart-bild
LOAD   ladda Myart-bild
ARTLES TI-Artist till Myart-bild
```

Ett fristående program till XHI kan skriva ut Myart-bilder på skrivare. Denna HARDCOPY version 0.95 är ej helt färdigt men fungerar även utan 9938-videoprocessor.

c.Fönster

```
VIPORT skapa fönster för grafik
FILSCR rensa/färga fönster
```

```
COPY   kopiera fönster
XPAND  expandera fönster horisontellt
YPAND  expandera fönster vertikalt
REDUCX krympa fönster horisontellt
REDUCY krympa fönster vertikalt
```

d.Text

```
PRINT  skriver textsträng
CWIDTH väljer pixelbredd för bokstav
```

Med CWIDTH 6 får du 80 tecken/rad och med CWIDTH 5 får du 102 tecken/rad. I vanlig BASIC är bredden 5 genom att pixel 2-6 används. Om du skapar egna tecken med bredden 4 (+ 1 pixels mellanrum) kan du enkelt skriva 102 tecken/rad.

e.Sprites

```
SPRITE aktiverar sprite
LOCATE flyttar sprite
SCOL   ändrar flerfärgad sprite
PATTER sprite-tecken
DELSPR ta bort sprite
```

även CALL MAGNIFY fungerar
MOTION interrupt-rörelse av sprite
STOP stannar sprite

Utöver dessa har författaren planer på att i framtiden utöka XHI med FILL, PEEKV, POKEV, cirkel med rätt proportioner i G6 och COLOR HARDCOPY.

Ett program som liknar XHI men arbetar med CALL LINK för Text2 kallat X80 finns nu i en testversion 0.9 utan komplett manual.

6. PROBLEM

G6 CIRCLE har fel proportioner eftersom punkterna ligger tätare i X-led än Y-led.

G7 CLR256 gör alla pixels svarta så du måste även använda CLS. Transparent ser ut att saknas bland de 256 färgerna så du måste välja samma färg för CLS och BACK.

MOTION av sprite med interrupt fungerar inte med mitt DIJIT AVPC-kort vilket kanske inte är så överraskande eftersom XHI har skrivits på en Myarc 9640 Geneve. Andra interruptprogram fungerar dock bra med AVPC t.ex. musrutinen som styr sprite 1 i Extended Basic.

```
100 ! LOAD MYART GRAPHICS6
110 ! JAN ALEXANDERSSON
130 ! 1989-08-19
140 ! XB + 9938 + XHI
150 CALL CLEAR
160 INPUT "FILE ":FILE$
170 IF FILE$="" THEN END
180 CALL LINK("HICLR")
190 ON ERROR 220
200 CALL LINK("LOAD",FILE$)
210 CALL KEY(5,K,S):: IF S<1
 THEN 210
220 CALL LINK("NORMAL")
230 CALL LINK("COLRES")
240 GOTO 160
```

```
100 ! LOAD MYART GRAPHICS7
110 ! JAN ALEXANDERSSON
120 ! SWEDEN
130 ! 1989-07-28
140 ! XB + 9938 + XHI
150 CALL CLEAR
160 INPUT "FILE ":FILE$
170 IF FILE$="" THEN END
180 CALL LINK("CLR256")
190 ON ERROR 260
200 OPEN #1:FILE$,FIXED 128,
INPUT :: LINPUT #1:COLOR$ ::
 CLOSE #1
210 COLOR=ASC(COLOR$)
220 CALL LINK("BACK",COLOR)
230 CALL LINK("CLS",COLOR)
240 CALL LINK("LOAD",FILE$)
250 CALL KEY(5,K,S):: IF S<1
 THEN 250
260 CALL LINK("NORMAL")
270 GOTO 160
```

```
100 REM XHI ARTLES G6
110 INPUT "TIARTIST FIL ":F$
120 INPUT "MYART G6-FIL ":M$
130 CALL LINK("HICLR")
140 ON ERROR 180
150 CALL LINK("CLS",16)
160 CALL LINK("ARTLES",F$)
170 CALL LINK("SAVE",M$)
180 CALL LINK("NORMAL")
```

```
100 REM XHI FILSCR G7
110 CALL LINK("CLR256")
120 ON ERROR 230
130 CALL LINK("BACK",255)
135 CALL LINK("CLS",255)
160 CALL LINK("VIPORT",10,50
,100,100)
170 CALL LINK("FILSCR",7)
180 CALL LINK("VIPORT",10,15
0,100,200)
181 CALL LINK("FILSCR",14)
220 CALL KEY(5,K,S):: IF S<1
 THEN 220
230 CALL LINK("NORMAL")
```

TIPSPROGRAM FÖR XB

av Arne Dahlander

I tidigare nummer efterlystes bidrag från medlemmar med enklare utrustning och hur de har användning av denna. För min del har jag TI-99/4A grundenhet, XB, PRK och bandspelare. Mitt programförråd består av 220 st nummer, och de flesta programmen har jag tagit från olika datatidningar. Ämnesområdena är spel, geografi, matte, språk, astronomi, bokföring m.fl. Men jag måste tillstå att efter det jag knappat in programmen, provat att de fungerar ordentligt, och använt dem några gånger blir de liggande ospelade.

Men jag har en hobby sen många år tillbaka, nämligen tippning. Haha, tänker kanske många och drar sina slutsatser men låt mig förklara. Jag har alltid varit road av att konstruera egna små reducerade tipssystem, vilka jag sedan provar och kontrollerar med datorns hjälp. Sedan lämnar jag in mina små system. Varje vecka använder jag regelbundet datorn och kör ett eget program, där de för veckan aktuella lagens chanser värderas. Sedan använder jag dessa uppgifter i ett annat eget konstruerat program, baserat på eget system som på skärmen visar rad för rad, som skall skrivas på kupongen. I början visades alla raderna på en kupong samtidigt, men det ändrade jag ganska snart, ty det blev för gyttrigt. När tipsresultaten kommer på lördagen använder jag samma program för rättning av tipskupongen.

Som ett prov medsänder jag rättningsprogram för Tipstjänst reducerade system R7, R8, R9, R10 och R11. I början fick jag lära mig att man måste DIMENSIONERA det antal rader, som skulle rättas. Tyckte rätt snart att det tog för lång tid för PRE-SCAN-proceduren och slukade för mycket av dator-minnet. Gjorde om programmen med ett minimum av dimensioneringar och R7-R8-R9 kom till. Nu var 99:ans minnesutrymme nästan helt utnyttjad. För att klara R10 och R11 fordrades ju ännu mera minnesutrymme. Tyckte att programmet R7-R8-R9 hade alldeles för många kommatecken i data-satserna. Ändrade programmen ytterligare med användande av SEG$ och ett data-ord per rad. På det sättet fick jag plats med R10 och R11. Hoppas att bifogade program är av intresse för någon föreningsmedlem. I så fall håll till godo!

De tre programmen R7/R8/R9, R10 och R11 kan beställas från programbanken på kassett eller flexskiva.

```
100 REM KONTROLL-R10, ARNE DAHLA
NDER, GULLMARSVÄGEN 110, 121 41
JOHANNESHOV
110 CALL CHAR(91,"00440038447C44
440044007C4444447C00100038447C44
44")
120 CALL CLEAR
130 DISPLAY AT(8,7):"***TIPSKONT
ROLL***"
140 DISPLAY AT(10,7):"***     AV
     ***"
150 DISPLAY AT(12,7):"***    R-1
0    ***"
160 DISPLAY AT(14,7):"***   10-5
67   ***"
170 DISPLAY AT(16,7):"***  GJORT
 AV  ***" :: DISPLAY AT(18,7):"*
**A. DAHLANDER***"
180 FOR DY=1 TO 500 :: NEXT DY
190 DIM A$(10)
200 DIM B$(10)
210 PRINT : : : : :
220 INPUT "ANT. RÄTTA SÄKRA MATC
HER? ":N
230 IF N>3 THEN 220
240 PRINT : : :
250 PRINT : :"SKRIV ÖVR. 10 RÄTT
A":" TECKEN,"
260 PRINT : :"ETT TECKEN I TAGET
": :
270 FOR K=1 TO 10
280 PRINT USING "   ## ":K;:: IN
PUT B$(K)
290 NEXT K
300 A,B,C,D,E=0
310 FOR J=1 TO 567
320 READ C$
330 FOR I=1 TO 10
340 A$(I)=SEG$(C$,I,1)
350 IF A$(I)<>B$(I)THEN 370
360 A=A+1
370 A$(I)="" :: NEXT I
380 AX=A+N
```

```
390 IF AX<10 THEN 440
400 PRINT : :"RAD";J;"=";AX;"MAT
CH. RÄTT"
410 PRINT : :"GRATULERAR"
420 IF AX=13 THEN B=B+1 ELSE IF
AX=12 THEN C=C+1 ELSE IF AX=11 T
HEN D=D+1 ELSE IF AX=10 THEN E=E
+1
430 PRINT : :B;"/";C;"/";D;"/";E
 :: PRINT
440 A,AX=0
450 DISPLAY AT(1,26):USING "###"
:J+1
460 NEXT J
470 PRINT : : :"13:A";TAB(9);"12
:A";TAB(17);"11:A";TAB(25);"10:A
"
480 PRINT : :B;TAB(9);C;TAB(17);
D;TAB(25);E
490 PRINT
500 END
510 DATA 1111111111,1111111XXX,1
111111222,11111X211X,11111X2XX2
520 DATA 11111X2221,111112X112,1
11112XXX1,111112X22X,1111X1X11X
530 DATA 1111X1XXX2,1111X1X221,1
111XX21XX,1111XX2X22,1111XX2211
540 DATA 1111212112,1111212XX1,1
11121222X,111122X1X1,111122XX2X
550 DATA 111122X212,111X112122,1
11X112X11,111X1122XX,111X1X1121
560 DATA 111X1X1X1X,111X1X12X2,1
11XX12121,111XX12X1X,111XX122X2
570 DATA 111XXXX111,111XXXXXXX,1
11XXXX222,111XX211X1,111XX21X2X
580 DATA 111XX21212,111X2X21X1,1
11X2X2X2X,111X2X2212,111X221112
590 DATA 111X221XX1,111X22122X,1
11211X121,111211XX1X,111211X2X2
600 DATA 1112121122,1112121X11,1
1121212XX,1112XX111X,1112XX1XX2
610 DATA 1112XX1221,1112X2X1XX,1
112X2XX22,1112X2X211,111221X122
620 DATA 111221XX11,111221X2XX,1
1122X11XX,11122X1X22,11122X1211
630 DATA 1112222111,1112222XXX,1
112222222,1X211111X2,1X21111X21
640 DATA 1X2111121X,1X211XX1X1,1
X211XXX2X,1X211XX212,1X211221XX
650 DATA 1X21122X22,1X21122211,1
X21X211X1,1X21X21X2X,1X21X21212
660 DATA 1X21X2X121,1X21X2XX1X,1
X21X2X2X2,1X212X11XX,1X212X1X22
670 DATA 1X212X1211,1X212X2122,1
X212X2X11,1X212X22XX,1X2X12111X
680 DATA 1X2X121XX2,1X2X121221,1
X2X12X112,1X2X12XXX1,1X2X12X22X
690 DATA 1X2XX11112,1X2XX11XX1,1
X2XX1122X,1X2XXXX1X2,1X2XXXXX21
700 DATA 1X2XXXX21X,1X2XX22122,1
X2XX22X11,1X2XX222XX,1X2X21X122
710 DATA 1X2X21XX11,1X2X21X2XX,1
X2X2121XX,1X2X212X22,1X2X212211
720 DATA 1X221X1112,1X221X1XX1,1
X221X122X,1X221X211X,1X221X2XX2
730 DATA 1X221X2221,1X22X1X1X1,1
X22X1XX2X,1X22X1X212,1X22X12121
740 DATA 1X22X12X1X,1X22X122X2,1
X2221111X,1X22211XX2,1X22211221
750 DATA 1X222XX121,1X222XXX1X,1
X222XX2X2,1X222221X2,1X22222X21
760 DATA 1X2222221X,12X111112X,1
2X1111X12,12X11112X1,12X1X12121
770 DATA 12X1X12X1X,12X1X122X2,1
2X1XX1122,12X1XX1X11,12X1XX12XX
780 DATA 12X1X2211X,12X1X22XX2,1
2X1X22221,12X121X122,12X121XX11
790 DATA 12X121X2XX,12X12XX112,1
2X12XXXX1,12X12XX22X,12X1221121
800 DATA 12X1221X1X,12X12212X2,1
2XX11X1XX,12XX11XX22,12XX11X211
810 DATA 12XX1X211X,12XX1X2XX2,1
2XX1X2221,12XX122121,12XX122X1X
820 DATA 12XX1222X2,12XXXXX12X,1
2XXXXXX12,12XXXXX2X1,12XX2111X1
830 DATA 12XX211X2X,12XX211212,1
2XX2X11XX,12XX2X1X22,12XX2X1211
840 DATA 12XX22X11X,12XX22XXX2,1
2XX22X221,12X21121X1,12X2112X2X
850 DATA 12X2112212,12X21XX122,1
2X21XXX11,12X21XX2XX,12X212X112
860 DATA 12X212XXX1,12X212X22X,1
2X2X111XX,12X2X11X22,12X2X11211
870 DATA 12X2XX2112,12X2XX2XX1,1
2X2XX222X,12X2X211X1,12X2X21X2X
880 DATA 12X2X21212,12X222212X,1
2X2222X12,12X22222X1,X12111112X
890 DATA X121111X12,X1211112X1,X
121X12121,X121X12X1X,X121X122X2
900 DATA X121XX1122,X121XX1X11,X
121XX12XX,X121X2211X,X121X22XX2
910 DATA X121X22221,X12121X122,X
12121XX11,X12121X2XX,X1212XX112
920 DATA X1212XXXX1,X1212XX22X,X
121221121,X121221X1X,X1212212X2
930 DATA X12X11X1XX,X12X11XX22,X
12X11X211,X12X1X211X,X12X1X2XX2
940 DATA X12X1X2221,X12X122121,X
12X122X1X,X12X1222X2,X12XXXX12X
950 DATA X12XXXXX12,X12XXXX2X1,X
12X2111X1,X12X211X2X,X12X211212
960 DATA X12X2X11XX,X12X2X1X22,X
12X2X1211,X12X22X11X,X12X22XXX2
970 DATA X12X22X221,X1221121X1,X
122112X2X,X122112212,X1221XX122
980 DATA X1221XXX11,X1221XX2XX,X
12212X112,X12212XXX1,X12212X22X
990 DATA X122X111XX,X122X11X22,X
122X11211,X122XX2112,X122XX2XX1
1000 DATA X122XX222X,X122X211X1,
X122X21X2X,X122X21212,X12222212X
1010 DATA X122222X12,X1222222X1,
XXX1111111,XXX1111XXX,XXX1111222
1020 DATA XXX11X211X,XXX11X2XX2,
```

```
XXX11X2221,XXX112X112,XXX112XXX1
1030 DATA XXX112X22X,XXX1X1X11X,
XXX1X1XXX2,XXX1X1X221,XXX1XX21XX
1040 DATA XXX1XX2X22,XXX1XX2211,
XXX1212112,XXX1212XX1,XXX121222X
1050 DATA XXX122X1X1,XXX122XX2X,
XXX122X212,XXXX112122,XXXX112X11
1060 DATA XXXX1122XX,XXXX1X1121,
XXXX1X1X1X,XXXX1X12X2,XXXXX12121
1070 DATA XXXXX12X1X,XXXXX122X2,
XXXXXXX111,XXXXXXXXXX,XXXXXXX222
1080 DATA XXXXX211X1,XXXXX21X2X,
XXXXX21212,XXXX2X21X1,XXXX2X2X2X
1090 DATA XXXX2X2212,XXXX221112,
XXXX221XX1,XXXX22122X,XXX211X121
1100 DATA XXX211XX1X,XXX211X2X2,
XXX2121122,XXX2121X11,XXX21212XX
1110 DATA XXX2XX111X,XXX2XX1XX2,
XXX2XX1221,XXX2X2X1XX,XXX2X2XX22
1120 DATA XXX2X2X211,XXX221X122,
XXX221XX11,XXX221X2XX,XXX22X11XX
1130 DATA XXX22X1X22,XXX22X1211,
XXX2222111,XXX2222XXX,XXX2222222
1140 DATA X2111111X2,X211111X21,
X21111121X,X2111XX1X1,X2111XXX2X
1150 DATA X2111XX212,X2111221XX,
X211122X22,X211122211,X211X211X1
1160 DATA X211X21X2X,X211X21212,
X211X2X121,X211X2XX1X,X211X2X2X2
1170 DATA X2112X11XX,X2112X1X22,
X2112X1211,X2112X2122,X2112X2X11
1180 DATA X2112X22XX,X21X12111X,
X21X121XX2,X21X121221,X21X12X112
1190 DATA X21X12XXX1,X21X12X22X,
X21XX11112,X21XX11XX1,X21XX1122X
1200 DATA X21XXXX1X2,X21XXXXX21,
X21XXXX21X,X21XX22122,X21XX22X11
1210 DATA X21XX222XX,X21X21X122,
X21X21XX11,X21X21X2XX,X21X2121XX
1220 DATA X21X212X22,X21X212211,
X2121X1112,X2121X1XX1,X2121X122X
1230 DATA X2121X211X,X2121X2XX2,
X2121X2221,X212X1X1X1,X212X1XX2X
1240 DATA X212X1X212,X212X12121,
X212X12X1X,X212X122X2,X21221111X
1250 DATA X212211XX2,X212211221,
X2122XX121,X2122XXX1X,X2122XX2X2
1260 DATA X2122221X2,X212222X21,
X21222221X,21X11111X2,21X1111X21
1270 DATA 21X111121X,21X11XX1X1,
21X11XXX2X,21X11XX212,21X11221XX
1280 DATA 21X1122X22,21X1122211,
21X1X211X1,21X1X21X2X,21X1X21212
1290 DATA 21X1X2X121,21X1X2XX1X,
21X1X2X2X2,21X12X11XX,21X12X1X22
1300 DATA 21X12X1211,21X12X2122,
21X12X2X11,21X12X22XX,21XX12111X
1310 DATA 21XX121XX2,21XX121221,
21XX12X112,21XX12XXX1,21XX12X22X
1320 DATA 21XXX11112,21XXX11XX1,
21XXX1122X,21XXXXX1X2,21XXXXXX21
1330 DATA 21XXXXX21X,21XXX22122,
21XXX22X11,21XXX222XX,21XX21X122
1340 DATA 21XX21XX11,21XX21X2XX,
21XX2121XX,21XX212X22,21XX212211
1350 DATA 21X21X1112,21X21X1XX1,
21X21X122X,21X21X211X,21X21X2XX2
1360 DATA 21X21X2221,21X2X1X1X1,
21X2X1XX2X,21X2X1X212,21X2X12121
1370 DATA 21X2X12X1X,21X2X122X2,
21X221111X,21X2211XX2,21X2211221
1380 DATA 21X22XX121,21X22XXX1X,
21X22XX2X2,21X22221X2,21X2222X21
1390 DATA 21X222221X,2X1111112X,
2X11111X12,2X111112X1,2X11X12121
1400 DATA 2X11X12X1X,2X11X122X2,
2X11XX1122,2X11XX1X11,2X11XX12XX
1410 DATA 2X11X2211X,2X11X22XX2,
2X11X22221,2X1121X122,2X1121XX11
1420 DATA 2X1121X2XX,2X112XX112,
2X112XXXX1,2X112XX22X,2X11221121
1430 DATA 2X11221X1X,2X112212X2,
2X1X11X1XX,2X1X11XX22,2X1X11X211
1440 DATA 2X1X1X211X,2X1X1X2XX2,
2X1X1X2221,2X1X122121,2X1X122X1X
1450 DATA 2X1X1222X2,2X1XXXX12X,
2X1XXXXX12,2X1XXXX2X1,2X1X2111X1
1460 DATA 2X1X211X2X,2X1X211212,
2X1X2X11XX,2X1X2X1X22,2X1X2X1211
1470 DATA 2X1X22X11X,2X1X22XXX2,
2X1X22X221,2X121121X1,2X12112X2X
1480 DATA 2X12112212,2X121XX122,
2X121XXX11,2X121XX2XX,2X1212X112
1490 DATA 2X1212XXX1,2X1212X22X,
2X12X111XX,2X12X11X22,2X12X11211
1500 DATA 2X12XX2112,2X12XX2XX1,
2X12XX222X,2X12X211X1,2X12X21X2X
1510 DATA 2X12X21212,2X1222212X,
2X12222X12,2X122222X1,2221111111
1520 DATA 2221111XXX,2221111222,
22211X211X,22211X2XX2,22211X2221
1530 DATA 222112X112,222112XXX1,
222112X22X,2221X1X11X,2221X1XXX2
1540 DATA 2221X1X221,2221XX21XX,
2221XX2X22,2221XX2211,2221212112
1550 DATA 2221212XX1,222121222X,
222122X1X1,222122XX2X,222122X212
1560 DATA 222X112122,222X112X11,
222X1122XX,222X1X1121,222X1X1X1X
1570 DATA 222X1X12X2,222XX12121,
222XX12X1X,222XX122X2,222XXXX111
1580 DATA 222XXXXXXX,222XXXX222,
222XX211X1,222XX21X2X,222XX21212
1590 DATA 222X2X21X1,222X2X2X2X,
222X2X2212,222X221112,222X221XX1
1600 DATA 222X22122X,222211X121,
222211XX1X,222211X2X2,2222121122
1610 DATA 2222121X11,22221212XX,
2222XX111X,2222XX1XX2,2222XX1221
1620 DATA 2222X2X1XX,2222X2XX22,
2222X2X211,222221X122,222221XX11
1630 DATA 222221X2XX,22222X11XX,
22222X1X22,22222X1211,2222222111
1640 DATA 2222222XXX,2222222222
```

STRUKTURERAD PROGRAMMERING – METODER OCH HJÄLPMEDEL

av Mikael Nordlin

Tyvärr kan man även i Programbiten få ta del av exempel på Basic-program, som lämnar övrigt att önska vad gäller klarhet, överskådlighet och begriplighet. Det kan vara felfria och i övrigt ypperliga program som utgör resultatet av stor möda av programmeraren under många timmars arbete.

Fenomenet som här skall beröras - allt i syfte att ge tips för hur det skall kunna undvikas - kallas ibland "spagetti-kod". Det förekommer inte bara i Basic, även om det till följd av vissa inneboende egenskaper (t ex GOTO <radnr> vid repetition, GOSUB <radnr> vid procedur-anrop) är vanligast i detta programspråk.

I 9900-assembler finns ett prominent exempel på "spagetti-kod" i källkoden till Disk Manager 1000 (av Bruce Caron och Ralph Romans, Ottawa TI-99/4 User's Group; finns i programbanken). Just detta program är ändå ett av de mest framgångsrika i den numera ganska talrika floran av "freeware" till TI-99/4A. "Spagetti-koden" märks ju inte vid körningen av programmet utan vid försök till vidareutveckling, antingen av den ursprunglige programmeraren eller ännu hellre av någon annan som tar vid.

McGovern i Australien har i andra sammanhang redogjort för sina vedermödor beträffande "avlusning" (jmf eng "debugging") av Disk Manager 1000 v3.5 och anpassning av programmet till Funnelweb. Att "spagetti-kod" inverkar menligt även på den ursprunglige programmerarens ambitioner att vidareutveckla sitt program framgår kanske av att senare versioner av Disk Manager 1000 (v3.8 finns i programbanken) rapporterats ha fel, som inte fanns i tidigare versioner. McGoverns vedermödor har föranlett rekommendationen från flera håll att endast använda Disk Manager 1000 i Funnelweb-version. Dess källkod har vi dock inte sett.

Det sägs att just Basic-programmering med nödvändighet innebär "spagetti-kod", och att programmerare som lärt sig konsten i detta programspråk har särskilt lätt för att fortsätta i de invanda fotspåren efter övergång till något annat programspråk. Även om dessa uttalanden kanske är något tillspetsade kan det nog ligga tillräckligt däri för att det skall vara mödan värt att - om det inte är för sent! - se vad som kan göras för att söka skaffa sig goda programmeringsvanor.

Botemedlet kan sammanfattas genom uttrycket strukturerad programmering. Vad som ingår i detta samlingsbegrepp är främst följande (termer används i allmängiltig bemärkelse, utan syftning på något speciellt programspråk). Variabler ges namn som anger innehållet (t ex INDEX). Logiskt sammanhängande instruktioner (tillsammans utgörande ett s k block) markeras visuellt genom t ex omgivande tomrader och indrag i vänster marginal. Segment (d v s ett eller flera block) som används mer än en gång deklareras som procedurer (med eller utan funktionssvar) och anropas med namn som anger vad proceduren utför (t ex Conv_Base ("3ef0", 16, RETN)). Kommentarer som förklarar programflödet används flitigt.

Vanlig Basic (TI BASIC liknar standard Basic) lämnar övrigt att önska i dessa hänseenden. Visserligen kan variabler ges namn av nästan obegränsad signifikant längd, men sedan är det slut på godsaker i den strukturorienterade programmerarens önskelista i föregående stycke. Kommentarer - i form av REM-satser - måste anges i egna rader och fråga uppstår gärna om kommentaren syftar på det närmast tidigare eller kommande. Procedurer måste deklareras någonstans där programexekvering endast sker efter procedur-anrop (GOSUB <radnr>) och anropet ger ingen ledtråd till vad som kommer att utföras. Jag bortser här medvetet från en mycket grov möjlighet att skapa korta funktioner

(DEF). En annan nackdel med standard Basic är att villkorssatser (IF-THEN) endast kan styra program-exekveringen till ett visst rad-nummer. Särskilt vid längre program blir det lätt att "tappa tråden".

Vissa Basic-dialekter (dock ej till 99:an) tillåter ett vulgärt sätt att erhålla begripliga GOSUB-anrop; det anropade radnumret kan först lagras i en variabel som genom sitt namn antyder vad proceduren gör, därefter sker anrop med GOSUB ‹variabelns namn› (t ex GOSUB Conv_Base). Att metoden är "livsfarlig" vid omnumre-ring av radnummer förstår var och en.

TI Extended BASIC är ett bra steg på vägen, och anses som en mycket bra Basic i jämförelse med andra vidare-utvecklingar av Basic till andra da-torer. Således finns villkorssatser med alternativ (IF-THEN-ELSE) och vid uppfyllt villkor behöver inte nödvändigtvis programflödesstyrning till visst radnummer ske; instruk-tioner kan utföras direkt beroende på utfallet av testet, så länge de ryms på samma rad. Förutom REMark finns "Trailing Exclamation Mark" ("!") för kommentarer i direkt an-slutning till det kommenterade. En stor förbättring är möjligheten till äkta procedurer (kallas här sub-program) med lokala, statiska vari-abler; anrop sker med CALL.

Tyvärr saknas vissa hos andra moderna, utökade Basic-dialekter förekommande programflödesstyr-ningar, såsom REPEAT-UNTIL och WHILE-WEND. Möjlighet finns dock att samla flera instruktioner (t ex ett block) i en programrad. Den sistnämnda möjligheten bör emeller-tid - ur struktursynpunkt - användas med viss urskillning för att inte förvärra överskådligheten.

Genom att vid användning av FOR-NEXT indexvariabeln ändras inne i slingan så att 'sant' eller 'falskt' er-hålles och vid FOR kan avgöra om slingan skall genomlöpas ytterligare ett varv, kan REPEAT-UNTIL simu-leras. Detta är enda sättet att styra programexekveringen till något annat programsegment än början av en Basic-rad. En på detta vis simulerad REPEAT-UNTIL struktur kan alltså vara kringgärdad av andra instruk-tioner i en och samma Basic-rad. Detta ger kompakta program, men priset man kan få betala är att läs-barheten försvåras. Vi skall dock visa hur det kan undvikas. Ät kakan och ha den kvar!

IF-THEN ‹radnr› är alltså utökat i TI Extended Basic, så att även IF-THEN /-ELSE/ ‹instruktion› finns, vilket är bra. Dock utgör program-raden blockbegränsare, varför ett IF-THEN-block inte kan åtföljas av andra instruktioner i samma program-rad. Denna onödiga begränsning kan förebyggas genom ett utökat använd-ande av logiska operatorer. Man får ha i åtanke att logiskt 'sant' är -1 och 'falskt' är 0.
Därför kan programraden

```
100 IF A<0 THEN A=A+10
```

ersättas med likvärdiga

```
100 A=A-(A<0)*10 !OBS-,ty -1 är sant
```

Om alternativet används, kan pro-gramraden utökas med ytterligare instruktioner, åtskilda av "::". Tätare programkod kan därigenom erhållas. De logiska operatorerna AND, OR och XOR kan användas, t ex för att maska bort oönskade bitar.

Det finns ett programmeringsverktyg (jmf eng "Programming Tool"), skri-vet i TI Extended BASIC, som så långt möjligt komprimerar ett Basic-program genom att foga samman Basic-rader, korta av variabelnamn och ta bort kommentarer. Programmet heter SMASH och har marknadsförts av OAK TREE SYSTEMS (se även XB*TOOLS).

SMASH bearbetar ett Basic-program lagrat i MERGE-format, som efter komprimering kan lagras på diskett (eller kassett) i PROGRAM IMAGE-format igen. Ett på detta vis kom-primerat program är visserligen snabbare att ladda in och tar mindre plats på diskett (eller kassett) och är i allmänhet snabbare att köra, men blir i gengäld i det närmaste obegripligt. Det är vidare omöjligt att automatiskt dekomprimera ett på detta vis komprimerat program. Ett välskrivet strukturerat Basic-pro-gram bör därför sparas även i sin ursprungliga, längre och kommentera-

de form.

Man har alltså bättre strukturmöjligheter i TI Extended BASIC. Denna modul erfordras också för riktigt långa program, eftersom minnesexpansionen inte utnyttjas i TI BASIC. Men läsbarheten kan ökas högst väsentligt genom ännu bättre struktur, om man använder ett annat programmeringsverktyg som i form av "freeware" har funnits i programbanken ett tag; PARAGON COMPUTING's TEXTLOADER.

Programmet kan även användas till att skapa batch-filer (på känt MS-DOS-maner) av Basic-kommandon som exekveras när datorn slås på med isatt TI Extended Basic. Vidare kan programmet användas för att ladda ned Basic-program i ASCII-format - kanske skrivna för en annan datortyp - från t ex en databas.

Generiskt rör det sig om ett assemblerprogram (egentligen en hybrid mellan Basic och assembler) som används tillsammans med Extended BASIC-modulen och möjliggör exekvering av Basic-satser eller laddning av Basic-program, skrivna i DIS/VAR80-format. Det är detta format, som blir resultatet om ett Basic-program listas till en fil (LIST "DSK1.filnamn"). På detta vis kan man undvika Basic's radorienterade editor och skriva sina Basicprogram med hjälp av någon texteditor, t ex TI Editor/Assembler eller MITEC. För övrigt är man inte tvungen att lista hela programmet till en fil; LIST-kommandot medger att man infogar en s k "slice", d v s anger det intervall av programrader som skall listas. TEXTLOADER förutsätter inte för att laddningen skall fungera att programlistan ser ut så som resultatet blir sedan man listat till en fil. Tomma rader och mellanrum - som alltså kan användas för att ge struktur åt programlistningen - ignoreras av TEXTLOADER. Ett varningens ord är på sin plats - om den programrad som skall läsas in är längre än Basic tillåter, kraschar datorn utan föregående varning. Man får då gå tillbaka till färgbalksskärmen med <FCTN => (QUIT). Följande exempel fungerar. Se listning på sidan 19.
(Ü = tak, ¤ = dollar)

Detta programexempel - ett kort huvudprogram och två procedurer - uppfyller väl grundläggande krav på struktur. Vidare åskådliggör det hur man kan använda logiska operatorer och hur man med hjälp av FOR-NEXT kan simulera REPEAT-UNTIL.

Om det föregående programavsnittet sparas i en textfil i DIS/VAR80-format under namnet DSK1.PROGRAM, kan det läsas in i Basic-tolken om först TEXTLOADER körs med Extended Basic-modulen(RUN "DSK1.TEXTLOADER") och därefter kommandot CALL LINK ("OLD","DSK1.PROGRAM") ges. När TEXTLOADER lagt in hela programmet i Basictolken kan det direkt lagras och köras. De långa variabelnamnen behålls, men kommentarerna i form av rader som inleds med "!" tas bort. Flera kommandon - åtskilda av "::" - återfinns i mån av plats på samma rad i förekommande fall. Om lagring sker i MERGE-format, är programmet klart att bearbetas av SMASH och ett verkligt kompakt - men obegripligt! - resultat erhålles. Skillnaden framgår av följande listning av den SMASHade versionen av ovanstående programexempel.

```
100 CALL CLEAR::DISPLAY AT(12,5):
"Hex:" ::FOR A=0 TO 0::CALL A(B)::
A=B>47 AND B<58 OR B>64 AND B<71 OR
B>96 AND B<103
120 IF A THEN
A$=A$&CHR$(B)::DISPLAY
AT(12,17):A$
130 NEXT A::CALL
B(A$,16,C)::DISPLAY
AT(14,5):"Denary =":::DISPLAY
AT(14,16):C
30000 SUB A(A)::FOR B=0 TO 1::CALL
KEY(5,A,B)::NEXT B::SUBEND
31000 SUB B(A$,A,B)::B=0::FOR
C=1 TO
LEN(A$)::D=ASC(SEG$(A$,C,1))
31010 B=((D OR(D>57)*-96)-
(48-(D>57)*39))*AÜ(LEN(A$)-C)+B
:: NEXT C::SUBEND
```

Programutveckling går fort om man använder ramdisk (Horizon i mitt fall, med laddning av såväl texteditor som TEXTLOADER från denna). Avslutningsvis får jag ändå tillstå, att det kanske inte är helt realistiskt att förvänta sig att Basicprogram kommer att utvecklas efter ovanstående riktlinjer annat än i försökssyfte. Den som eftersträvar

struktur lär nog istället välja något för detta ändamål lämpligare programspråk, särskilt c99 eller någon av Pascal-varianterna (UCSD eller Turbo Pasc'99). Jag hoppas snarare att med dessa rader ha tillgodosett eller väckt ett intresse för allmängiltiga programmeringsmetoder.

```
! ***********************************
! 100 Rensa skärmen, skriv rubrik
! ***********************************
00100 call clear ::
      display at (12, 5):
        "Hex:"

! ***********************************
! 110-130 Läs tangentbordet, avbryt
!         vid ogiltig tangent
! /* do ... while (test == true) */
! ***********************************
00110 for TRUE = 0 to 0 ::
        call get (KEY) ::
        TRUE = KEY > 47 and KEY < 58
            or KEY > 64 and KEY < 71
            or KEY > 96 and KEY < 103

! ***********************************
! 120 Skriv ut ny sträng (om giltig)
! ***********************************
00120     if TRUE then
            HEX¤ = HEX¤ & chr¤ (KEY) ::
            display at (12, 17): HEX¤

00130 next TRUE ::

! ***********************************
! 130 Anropa baskonverteringsrutin,
!     skriv ut resultatet
! ***********************************
      call Conv_Base (HEX¤, 16, RETN)::
      display at (14, 5):
        "Denary =" ::
      display at (14, 16):
        RETN

! ***********************************
! 30000 Rutin för avläsning av
!       tangentbordet. Väntar tills en
!       tangent trycks och ger ASCII-
!       värdet i parametern KEY
! ***********************************
30000 sub get (KEY) ::

! ***********************************
! /* do ... while (index <> 1) */
! ***********************************
      for INDEX = 0 to 1 ::
         call key (5, KEY, INDEX) ::
      next INDEX ::

      subend
```

```
! ***********************************
! 31000 Rutin för baskonvertering.
!       Argumentet NUM¤ är den sträng
!       som skall konverteras till
!       basen 10. Argumentet BAS är
!       basen för NUM¤. Parametern RETN
!       kommer att innehålla svaret.
! ***********************************
31000 sub Conv_Base (NUM¤, BAS, RETN)::

! ***********************************
! Eftersom variabler i subprogram är
! statiska, måste nollställning ske.
! ***********************************
      RETN = 0 ::

! ***********************************
! Konvertering utförs tecken för
! tecken, från vänster till höger.
! ***********************************
      for INDEX = 1 to len (NUM¤) ::

         TEMP = asc
            (seg¤ (NUM¤, INDEX, 1))

31010    RETN =

         (

! ***********************************
! Konvertering från stora bokstäver
! till små sker genom (ASCII OR 96).
! ***********************************
           (
             TEMP or (TEMP > 57) *
             -96
           )

! ***********************************
! Tecknets värde beräknas genom att
! ASCII-koden för 0 resp a dras ifrån.
! ***********************************
           -

           ( 48 - (TEMP > 57) * 39 )

         )

! ***********************************
! Tecknets värde i relation till dess
! placering beräknas genom upphöjning.
! ***********************************
           *

           BAS Ü (len (NUM¤) - INDEX) +
           RETN ::

      next INDEX ::

      subend
```

TIPS FRÅN TIGERCUB

TIPS FROM THE TIGERCUB

#48

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 120 original programs in Basic and Extended Basic, available on cassette or disk, NOW REDUCED TO JUST $1.00 EACH!, plus $1.50 per order for cassette or disk and PP&M. Minimum order of $10.00. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Color Programming Tutor, are no longer available on cassette.
Descriptive catalogs, while they last, $1.00 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 postpaid. Each of these contains either 5 or 6 of my regular catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs - they are a free bonus!
TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION GAMES, REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

NUTS & BOLTS DISKS
These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended Basic. Each is accompanied by printed documentation giving an example of the use of each.
NUTS & BOLTS (No. 1) has 100 subprograms, a tutorial on using them, and 5 pp. documentation. NUTS & BOLTS No. 2 has 108 subprograms, 10 pp. of documentation. NUTS & BOLTS #3 has 140 subprograms and 11 pp. of documentation.
NOW JUST $15 EACH, POSTPAID.

TIPS FROM THE TIGERCUB
These are full disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready-to-run program format, plus text files of tips and instructions.
TIPS (Vol. 1) contains 50 original programs and files from Tips newsletters No. 1 through No. 14. TIPS VOL. 2 contains over 60 programs and files from Nos. 15 thru 24. TIPS VOL. 3 has another 62 from Nos. 25 through 32. TIPS VOL. 4 has 48 more from issues No. 33 through 41.
NOW JUST $10 EACH, POSTPAID.

```
*****************************
*        NOW READY          *
* TIPS FROM TIGERCUB VOL.5  *
* Another 49 programs and   *
* files from issues No. 42  *
* through 50. Also $10 ppd  *
*****************************
```

TIGERCUB CARE DISKS #1,#2,#3 and #4. Full disks of text files (printer required). No. 1 contains the Tips news letters #42 thru #45, etc. Nos. 2 and 3 have articles mostly on Extended Basic programming. No. 4 contains Tips newsletters Nos. 46-52. These were prepared for user group newsletter editors but are available to anyone else for $5 each postpaid.

If you have ever used TRACE to debug a program, you know that it won't dump to a printer, and that it messes up the screen format. The new Super Extended Basic, or the Gram Kracker, will dump to the printer, but you still won't know what is going on line by line or within multiple-statement lines. Now, Supertrace will break the program into single-statement lines and TRACE each statement in the corner of the screen, or dump it to the printer, or both - and you can also pause at any time, or step through line by line.

```
100 GOTO 140
110 SET,C$,END$,Z$,E$,K$,S$,
K,S,IF$,OF$,Q$,FL,TL,M$,LN,L
N2,P,T,LN$,A$,R,P$,QQ,PD$,KC
,KC$
120 CALL CHAR :: CALL CLEAR
:: CALL COLOR :: CALL SCREEN
 :: CALL KEY :: CALL SOUND
130 !@P-
140 CALL CHAR(94,"3C4299A1A1
99423C"):: CALL CLEAR :: FOR
 SET=1 TO 14 :: CALL COLOR(S
ET,13,15):: NEXT SET :: CALL
 SCREEN(13)
150 C$=CHR$(157)&CHR$(200)&C
HR$(1)&"A"&CHR$(183)&CHR$(20
0):: END$=CHR$(255)&CHR$(255
):: Z$=CHR$(131)&CHR$(147)&C
HR$(154)&CHR$(163)
160 E$=CHR$(0):: K$=CHR$(182
):: S$=CHR$(130)
170 DISPLAY AT(2,5)ERASE ALL
:"TIGERCUB SUPERTRACE": :"^
Tigercub Software for free":
"distribution but no price o
rcopying fee may be charged.
" !programmed by Jim Peterso
n 1/88
180 DISPLAY AT(8,1):" Howeve
r, if anyone should feel mo
ved to send me a few bucks f
or the use of this program
, I would not be":"offended!
"
190 DISPLAY AT(15,1):"Jim Pe
terson":"156 Collingwood Ave
.":"Columbus, OH 43213"
200 DISPLAY AT(23,8):"PRESS
ANY KEY" :: DISPLAY AT(23,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 200
210 DISPLAY AT(2,1)ERASE ALL
:" Will break each program":
"line into single statement"
:"lines, unless they contain
"
220 DISPLAY AT(5,1):"an IF,
and add a CALL to a":"subpro
gram which will":"display ea
ch line number in":"the corn
er of the screen as"
230 DISPLAY AT(9,1):"it is b
eing executed, or":"will out
put it to a printer."
240 DISPLAY AT(13,1):" Progr
am must first be -": :"RESeq
uenced to greater in-":"crem
ents than the number"
250 DISPLAY AT(17,1):"of sta
tements in any one":"line. (
recommend RES 100,20)": :"an
d SAVEd by":"  SAVE DSK(file
name),MERGE"
270 DISPLAY AT(23,8):"PRESS
ANY KEY" :: DISPLAY AT(23,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 270
310 DISPLAY AT(23,8):"PRESS
ANY KEY" :: DISPLAY AT(23,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 310 EL
SE CALL CLEAR
320 DISPLAY AT(3,1):"INPUT F
ILENAME?":"DSK" :: ACCEPT AT
(4,4):IF$ :: ON ERROR 330 ::
 OPEN #1:"DSK"&IF$,INPUT ::
GOTO 340
330 CALL SOUND(300,110,0,-4,
0):: DISPLAY AT(6,1):"CANNOT
 OPEN FILE!" :: RETURN 320
340 DISPLAY AT(6,1):"OUTPUT
FILENAME?":"DSK" :: ACCEPT A
T(7,4):OF$ :: ON ERROR 350 :
: OPEN #2:"DSK"&OF$,VARIABLE
 163,OUTPUT :: ON ERROR STOP
 :: GOTO 355
350 CALL SOUND(300,110,0,-4,
```

```
0):: DISPLAY AT(9,1):"CANNOT
 OPEN FILE!" :: RETURN 340
355 DISPLAY AT(9,1):" Progra
ms of more than 50":"sectors
 in length may become":"too
long to run if you break":"a
nd trace all lines."
360 DISPLAY AT(15,1):"Break
all lines? (Y/N)" :: ACCEPT
AT(15,24)SIZE(1)VALIDATE("YN
"):Q$ :: IF Q$="Y" THEN 390
370 DISPLAY AT(17,1):"From l
ine?" :: ACCEPT AT(17,12)VAL
IDATE(DIGIT):FL
380 DISPLAY AT(17,18):"To?"
:: ACCEPT AT(17,22):TL
390 DISPLAY AT(15,1):"TRACE
to 1":"":" (1) Screen":" (2)
 Printer":" (3) Both" :: ACC
EPT AT(15,10)SIZE(-1)VALIDAT
E("123"):QQ :: IF QQ=1 THEN
405
400 DISPLAY AT(21,1):"Printe
r? PIO" :: ACCEPT AT(21,10)S
IZE(-18):PD$
405 DISPLAY AT(3,1)ERASE ALL
:" Key code 1 allows the pro
-":"gram to run until you ho
ld":"down any key. It will b
e"
406 DISPLAY AT(6,1):"difficu
lt to execute CALL":"KEYs in
 the program.":"":" Key code
 2 requires a key":"to be pr
essed to execute"
407 DISPLAY AT(11,1):"each p
rogram line. You can":"step
through the program":"line b
y line, but this may":"be ve
ry slow if all lines"
408 DISPLAY AT(15,1):"are be
ing traced.":"":" Key code 3
 does not allow":"stopping t
he program."
409 DISPLAY AT(20,1):"Key co
de? 1" :: ACCEPT AT(20,11)SI
ZE(-1)VALIDATE("123"):KC410
IF KC=1 THEN KC$=CHR$(191)&C
HR$(192)&CHR$(200)&CHR$(1)&"
0" ELSE KC$=CHR$(191)&CHR$(2
00)&CHR$(1)&"1"
411 DISPLAY AT(12,7)ERASE AL
L:"Working line"
420 LINPUT #1:M$ :: IF M$=EN
D$ THEN 570
430 LN=ASC(SEG$(M$,1,1))*256
+ASC(SEG$(M$,2,1)):: IF Q$="
Y" THEN 440 :: IF LN<FL OR L
N>TL THEN PRINT #2:M$ :: GOT
O 420
440 IF LN>LN2 THEN 460
450 DISPLAY AT(12,1)ERASE AL
L BEEP:"ERROR! RESEQUENCE PR
OGRAM TO":"GREATER INCREMENT
S AND TRY":"AGAIN." :: CLOSE
 #1 :: CLOSE #2 :: STOP
460 LN2=LN :: IF POS(Z$,SEG$
(M$,3,1),1)<>0 THEN PRINT #2
:M$ :: DISPLAY AT(12,19):LN
:: GOTO 420
470 P=POS(M$,S$,3):: T=POS(M
$,CHR$(161),3):: IF T=0 THEN
 500
480 IF P=0 THEN PRINT #2:SEG
$(M$,1,LEN(M$)-1)&S$&C$&CHR$
(LEN(STR$(LN)))&STR$(LN)&K$&
E$ :: DISPLAY AT(12,19):LN :
: GOTO 420
490 PRINT #2:SEG$(M$,1,P)&C$
&CHR$(LEN(STR$(LN)))&STR$(LN
)&K$&E$ :: DISPLAY AT(12,19)
:LN :: LN=LN+1 :: GOSUB 690
:: M$=LN$&SEG$(M$,P+1,255)::
 GOTO 430
500 IF P=0 THEN PRINT #2:SEG
$(M$,1,2)&C$&CHR$(LEN(STR$(L
N)))&STR$(LN)&K$&S$&SEG$(M$,
3,255):: DISPLAY AT(12,19):L
N :: GOTO 420
510 A$=SEG$(M$,1,P-1):: R=PO
S(A$,CHR$(132),3):: S=POS(A$
,CHR$(201),3)
520 IF R=0 THEN GOSUB 750 ::
 GOTO 560
530 IF S=0 AND R<>0 THEN GOS
UB 700 :: GOTO 420
540 IF S<>0 THEN IF S-R<3 TH
EN GOSUB 750 :: GOTO 560
550 GOSUB 700 :: GOTO 420
560 LN=LN+1 :: LN2=LN :: GOS
UB 690 :: M$=LN$&SEG$(M$,P+1
,255):: P=POS(M$,S$,3):: GOT
O 500
570 LN=29999 :: GOSUB 690 ::
 PRINT #2:LN$&CHR$(131)&CHR$
(64)&CHR$(80)&CHR$(43)&CHR$(
0)
580 LN=30000 :: GOSUB 690 ::
 PRINT #2:LN$&CHR$(161)&CHR$
(200)&CHR$(1)&"A"&CHR$(183)&
"X"&K$&E$ :: IF QQ=1 THEN 63
0
590 LN=30001 :: GOSUB 690 ::
 P$=LN$&CHR$(132)&"F"&CHR$(1
90)&CHR$(200)&CHR$(1)&"0"&CH
R$(176)&CHR$(159)&CHR$(253)&
CHR$(200)&CHR$(3)&"250"
600 P$=P$&CHR$(181)&CHR$(199
)&CHR$(LEN(PD$))&PD$&CHR$(13
0)&"F"&CHR$(190)&CHR$(200)&C
HR$(1)&"1"&S$&CHR$(156)&CHR$
(253)&CHR$(200)&CHR$(3)&"250
```

```
"&CHR$(181)&CHR$(214)
610 P$=P$&CHR$(183)&CHR$(200
)&CHR$(2)&"27"&K$&CHR$(184)&
CHR$(199)&CHR$(1)&"N"&CHR$(1
84)&CHR$(214)&CHR$(183)&CHR$
(200)&CHR$(1)&"6"&K$&E$ :: P
RINT #2:P$
620 LN=30002 :: GOSUB 690 ::
 PRINT #2:LN$&CHR$(156)&CHR$
(253)&CHR$(200)&CHR$(3)&"250
"&CHR$(181)&"X"&CHR$(180)&E$
630 IF QQ=2 THEN 650
640 LN=30003 :: GOSUB 690 ::
 PRINT #2:LN$&CHR$(162)&CHR$
(240)&CHR$(183)&CHR$(200)&CH
R$(2)&"24"&CHR$(179)&CHR$(20
0)&CHR$(1)&"1"&K$&CHR$(181)&
"X"&CHR$(180)&E$
645 IF KC=3 THEN 670
650 LN=30004 :: GOSUB 690 ::
 P$=LN$&CHR$(157)&CHR$(200)&
CHR$(3)&"KEY"&CHR$(183)&CHR$
(200)&CHR$(1)&"0"&CHR$(179)&
"K"&CHR$(179)&"S"&K$
660 P$=P$&CHR$(130)&CHR$(132
)&"S"&KC$&CHR$(176)&CHR$(201
)&CHR$(INT(LN/256))&CHR$(LN-
256*INT(LN/256))&E$ :: PRINT
 #2:P$
670 LN=30005 :: GOSUB 690 ::
 PRINT #2:LN$&CHR$(168)&CHR$
(0):: PRINT #2:CHR$(255)&CHR
$(255)
680 CLOSE #1 :: CLOSE #2 ::
DISPLAY AT(12,1)ERASE ALL:"E
nter NEW": :"Then Enter":"
MERGE DSK"&OF$ :: END
690 LN$=CHR$(INT(LN/256))&CH
R$(LN-256*INT(LN/256)):: RET
URN
700 IF LEN(M$)>150 THEN 720
:: PRINT #2:SEG$(M$,1,2)&C$&
CHR$(LEN(STR$(LN)))&STR$(LN)
&K$&S$&SEG$(M$,3,255)
710 DISPLAY AT(12,19):LN ::
RETURN
720 PRINT #2:SEG$(M$,1,2)&C$
&CHR$(LEN(STR$(LN+1)))&STR$(
LN+1)&K$&E$
730 DISPLAY AT(12,19):LN
740 LN=LN+1 :: PRINT #2:CHR$
(INT(LN/256))&CHR$(LN-256*IN
T(LN/256))&SEG$(M$,3,255)::
DISPLAY AT(12,19):LN :: LN2=
LN :: RETURN
750 PRINT #2:SEG$(A$,1,2)&C$
&CHR$(LEN(STR$(LN)))&STR$(LN
)&K$&S$&SEG$(A$,3,255)&E$ ::
DISPLAY AT(12,19):LN::RETURN
```

This "tinygram" might give you a surprise. SAVE it before you run it.

```
100 CALL CLEAR :: CALL KEY(3
,K,S):: ON BREAK NEXT ! by J
im Peterson
110 DIM CH$(26):: FOR J=1 TO
 26 :: CALL CHARPAT(J+64,CH$
(J)):: NEXT J :: FOR J=1 TO
26 :: CALL CHAR(J+64,CH$(27-
J)):: NEXT J
120 DISPLAY AT(3,8):"MZNV ZM
ZOBAVI":"":"GSRH KILTIZN DRO
O ZMZOBAV   BLFI MZNV."
130 INPUT "BLFI MZNV? ":M$ :
: CALL SOUND(200,110,0,-4,0)
:: X=X+1 :: IF X<2 THEN 130
140 DISPLAY AT(12,1):"ZMZOBH
RH - ":"":"VRGSVI BLF XZM'G
HKVOO BLFI LDM MZNV LI MLYLW
B XZM      KILMLFMXV RG."
150 GOTO 150
```

Here's another tinygram that might help you editors who reformat my Tips to wider column widths.

```
100 DISPLAY AT(3,6)ERASE ALL
:"TIGERCUB UNFILLER":"":" To
 remove extra spaces from":"
a TI-Writer text which has":
"been Filled and Adjusted by
"
110 DISPLAY AT(8,1):"the For
matter, prior to":"reformatt
ing.":" It will, however, al
so":"remove paragraph indent
a-":"tions and other intende
d":"spacings."
120 DISPLAY AT(15,1):"Input
file? DSK" :: ACCEPT AT(15,1
6):IF$ :: OPEN #1:"DSK"&IF$,
INPUT
130 DISPLAY AT(17,1):"Output
 file? DSK" :: ACCEPT AT(17,
17):OF$ :: OPEN #2:"DSK"&OF$
140 LINPUT #1:M$
150 X=POS(M$,"  ",1):: IF X=
0 THEN PRINT #2:M$ :: GOTO 1
70
160 M$=SEG$(M$,1,X)&SEG$(M$,
X+2,255):: GOTO 150
170 IF EOF(1)<>1 THEN 140 ::
 CLOSE #1 :: CLOSE #2
```

MEMORY AMOST FULL....
Jim Peterson

Finns det någon kabel som gör att man kan koppla ihop TI-74 med 99:an för dataöverföring? Jesper Nordström 0451-16468

KYLNING AV MYARC-KORT

1. MYARC 512 KBYTES RAM-DISK

av Tony McGovern, Australien

Ett bekymmer som jag har haft är värmeutvecklingen i Myarcs RAM-disk. 5V-stabilisatorn är monterad direkt på kretskortet, inte ens på en större kopparyta. Resultatet blir att kretskortet ser missfärgat ut i närheten. TI-korten har också sina stabilisatorer på kortet men ser ut att ha värmeledning till metallhöljet och tjockare kopparfolie på kortet. Myarc-korten har bara ett plasthölje som inte är till någon hjälp. Jag har funderat på att göra något åt detta ett tag, men det var först när jag såg kylflänsen på Sydney-AT-kortet som böjer sig fint bakåt över stabilisatorn så att dess huvudyta är parallell med kortet, som jag slutligen beslöt att göra något åt detta. En snabb undersökning visade att du inte kan köpa en bra sådan i Newcastle (den fortlöpande historien om livet här) så jag gjorde en egen J-formad och monterade den. Den går bakåt parallellt med kortet för att ge tillräcklig yta och passar fint innanför höljet. Från vad jag kommer ihåg av det Geneve-kort som jag sett så tror jag att ett liknande tillägg inte skulle skada. Myarc tycks strunta i kylflänsar. Jag kanske även gör något liknande för Quest RAM-disk eftersom dess kylfläns inte tillåter ett kort bredvid vilket är besvärande i en full expansionsbox.

2. MYARC 9640 GENEVE

av Sören Bernle

Geneven har vid flera tillfällen i våras kollapsat för mig. Efter att ha ringt runt i nästan hela världen, fick jag förklaring på problemet och anvisning på åtgärd. Förklaringen var värmeböljan i våras och otillräcklig kylning. Åtgärder:

1. Flytta bort datorkortet från den värmeavgivande nätdelen i boxen.

2. För att öka avkylningen, ta bort höljet till datorkortet.

3. Sätt boxens spännings-ställare till 240 V (tidigare 220 V).

Efter att ha åtgärdat punkt 1 och 3, har allt fungerat helt tillfredsställande.

3. INSPÄNNING TILL EXPANSIONSBOXEN

av Stephen Shaw, TI*MES, England

Flera rapporter visar att den oreglerade spänningen som matas till korten ofta kan vara så hög som +11,5, +22,5 och -23,5 V där det nominellt ska vara +8, +16 och -16 V. Detta är inget problem för en glest bestyckad box men för en full box kan det bli problem eftersom när spänningen regleras ned på korten så uppstår det värme. Om spänningen sänks så minskar även värmeutvecklingen. Ett förslag är en fristående transformator som reglerar inspänningen till 200 V innan den går in i expansionsboxen samtidigt som boxen byglas för 240 V (se teknisk manual). Transformatorn behöver klara 30 W.

4. 220 V - 240 V BLIR 230 V I EUROPA

av Jan Alexandersson

Vi har idag 220 V nätspänning i Sverige men denna kan variera mellan 205 och 242 V. I England har man 240 V som kan vara 225-254 V. Det finns nu en ny europeisk standard på 230 V som kommer att införas i Sverige så att 220/380 V blir 230/400 V (svensk standard SS 421 05 01). Den nominella spänningen 230 V får vara 207-244 V. Övergången till den högre spänningen skall vara genomförd före utgången av år 1995. Spänningen är definierad i eldistributörens leveranspunkt vilket betyder att väggkontakten kan ha 198-244 V. Införandet av den nya standarden innebär att det blir ännu varmare i boxen vilket gör det önskvärt att bygla boxen för 240 V istället för 220 V.