Programbiten 87-4

nittinian

Checklist

BONGO SOFT.

PRESENTS

PLUS

Mer om RAM-diskar

Innehåll

ISSN 0281-1146

Redaktören ...

Till och från PC

I förra numret av PB nämndes att CorComp tagit fram en modul som gör det möjligt att läsa disketter från en PC på en 99:a (om man har CorComps diskkontrollkort). Att skriva disketter i PC-format går också. En av våra australiska vänner, Will McGovern, håller på och utvecklar ett liknande program. Det är i första hand gjort för Myarcs diskkontrollkort, men skall enligt uppgift vara skrivet på ett sätt att det lätt kan anpassas för andra kontrollkort som klarar dubbel packningstäthet (dvs TI:s originalkort är uteslutet). Programmet beräknas vara klart vid årsskiftet.

Ytterligare ett program för överföring mellan PC och 99:an/Geneve har tagits fram. Det heter PC-Transfer och säljs kommersiellt.

Omslagspojke

Bilder på styrelsen har vi aldrig publicerat i tidningen. En av våra medlemmar har i alla fall blivit avbildad i en av våra kollegor inom datapressen, *MikroDatorn*. Mikael Nordlin är omslagspojke på oktobernumret (nr 8/87). Såvitt man kan se är det inte en 99:a han knappar på, det ser snarare ut som en PC, men vad gör man inte för att komma med ...

Oförändrad medlemsavgift 1988

Styrelsen beslöt på sitt senaste möte att rekommendera att medlemsavgiften blir oförändrat 120 kr för 1988. Beslut tas på årsmötet.

Inget möte i Stockholm

Det medlemsmöte som vi hade planerat till i november i Stockholm fick vi inställa p g a lokalproblem. Nästa stora möte blir årsmötet den 12 mars. Lokal blir Militärhögskolan på Valhallavägen i Stockholm. Närmare kallelse kommer.

Föreningens databas

Vi hade hoppats ha klart med den databas som vi planerar till detta nummer. Det har dock tagit längre tid än vi trott. Vi har i alla fall fått ett par program från USA som bör kunna användas för ändamålet.

95:an och 74:an

I förra numret utlovades material om dessa till detta nummer. Det kommer i stället att publiceras i nummer 88-1. Vi planerar att ha en temasektion om dem då.

Tryckfelsnisse

brukar härja i tidningens spalter emellanåt. Att han även är aktiv på annat håll syntes i Dagens Nyheter i höstas. Vi vill inte gärna undanhålla läsarna följande annons:

...dsp gott ... 50:- 54 71 ...	Färgmon + modulator, garanti, pass Amiga 1.500:-. 0753-795 64
MSX 728 bandst 2 joy 4 orig.-spel 3 cartr 1.800:- 0752-13087	Skönskrivare Olivetti helt ny nyp 10.000:- NU 3000:- 367735
Spectrum 48 K m tillbh. 1200:- Tel 0511-152 03	TSR-80 fickdator, 1,6 kB + böcker 475:-. Tel 08-767 30 52
128 Comm. m bandsp joyst spel man + litt 3000:- 0762-509 08	Nyb.dator TI99/4A + isterband litt 1400:- Tel 08-767 30 52
Commodore +4 oanv: 1.000:-. Tel 08-742 21 68	Mikrobean 32kb Philips monitor,tillb:1 ...0:-. 305766 ...
C-64 bands 30 kass m 1500 spel o pr,10 böck ...-. 770157?	

I redaktionen:

Redaktör	Peter Odelryd
Utmaningsredaktör	Anders Persson
Forth-redaktör	Lars-Erik Svahn
Programförmedlare	Börje Häll
Allt-i-allo	Claes Schibler

Föreningens och redaktionens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9, nb
12146 Johanneshov

Datainspektionens licensnummer: 82100488

Postgiro: 198300-6
Medlemsavgiften för 1987 är 120:–

Föreningens tillbehörsförsäljning

Följande finns att köpa för medlemmar genom att motsvarande belopp insätts på postgiro 198300-6.

För TI-99/4A:	
Användartips med Mini Memory	60:–
Nittinian T-tröja	40:–
99'er magazine nr 12/82, 1–5, 7–9/83 (per styck)	20:–
Programbiten/Nittinian 1986 (TI 59+99)	80:–
Programbiten/Nittinian 1985 (TI 59+99)	80:–
Programbiten/Nittinian 1984 (TI 59+99)	80:–
Nittinian, årgång 1983 (TI-99/4A)	60:–
För TI-59 mm	
Programbiten, årgång 1983 (kalkylatorer)	40:–
Programbiten, årgång 1982 (kalkylatorer)	40:–
Programbiten, årgång 1981 (kalkylatorer)	40:–
Programbiten, årgång 1980 (kalkylatorer)	40:–
Programbiten, årgång 1978/79 (kalkylatorer)	40:–
Programbiten, fem årgångar 1978–83	160:–
Programbiten, sex årgångar 1978–84	220:–
Programbiten, sju årgångar 1978–85	300:–
Katalog med belgiska och engelska program för räknare TI-57, TI-58, TI-59	20:–
Föreningens programmeringsblanketter (TI-59), olika typer, block om 50 blanketter (se PB 83-1 sid 30) per block	11:–
Patenthandlingar TI-59	25:–
Tom magnetkortsplånbok	10:–

Corcomps RAM-disk

av Lennart Thelander

Jag har sedan i somras en Corcomp 512 KB RAM-disk i min ägo. De erfarenheter jag har av den skulle jag vilja dela med mig. En del uppgifter i PB 87-3 om RAM-diskar får mig att lyfta litet på ögonbrynen. Man får i artikeln uppfattningen att man endast kan ha en Corcomp ramdisk i systemet och att den måste ersätta minnesexpansionen. Det är inte helt rätt. Det finns två sorter av Corcomps ramdisk, nämligen fristående eller inte. Den fristående sätter man i sidan av konsolen, och sedan flatkabeln till boxen i ramdisken. Det finns sedan två storlekar av varje sort (256/512 KB). Av detta kan man utnyttja 224 resp 480 KB som ramdisk om kortet inte är fristående, resten ersätter mycket riktigt minnesexpansionen. Om kortet är fristående kan man utnyttja allt minne som ramdisk. Om man har två Corcomp ramdiskar (maximum) kan man ha antingen en eller två fristående. Man kan alltså inte ha två i expansionsboxen. Den som anser sig behöva mer än 1 MB som ramdisk borde byta dator till åtminstone en persondator. Man måste komma ihåg att 99:an trots allt är en liten hemdator.

Vidare står det att om datorn varit avslagen måste man initiera ramdisken igen. Det behöver man inte på Corcomps; den är visserligen tom, men den är redo att användas. Man behöver inte använda Corcomps diskmanager till disken, det går bra med Disk manager 1000 om ramdisken är formaterad till max 400 KB. Jag ser det inte som någon nackdel att använda Corcomps egen disk manager. Den är nästan lika bra som disk manager 1000 eller den man får med till Corcomps diskcontroller. Framförallt behöver man *inte* ladda den från disk; den finns med i de 16KB EPROM som sitter på ramdisken. I BASIC eller X-BASIC skriver man bara DELETE "RAMGR" så startar disk managern. Man kan också skriva DELETE "SD.x", där x är drivenumret man vill att ramdisken ska få. Man kan ange. Dessutom kommer man alltid åt ramdisken genom "DSKR.filnamn". Det finns en rutin till man kan utnyttja i BASIC eller X-BASIC. Den anropar man med DELETE "LOWER", varpå man har riktiga gemena bokstäver med underslängar. Dessa försvinner som vanligt så fort programmet stannar. De ser inte så snygga ut så jag får väl bränna nytt EPROM med snyggare bokstäver (med å, ä och ö!).

Corcomps ramdisk fungerar alldeles utmärkt, jag har inte hittat några buggar alls. Kortet fungerar i alla lägen där andra ramdiskar fungerar, dvs i 99,9 % av programmen. I den sista tiondelen ingår Multiplan (det enda program jag hittat som inte fungerar) som envist läser disketten i drive 1 i stället för ramdisken. Precis som Horizon fungerar inte Corcomp tillsammans med Pascal och p-systemet till en början. Tack vare mycken hjälp från utmaningredaktören Anders Persson (TACK!) fungerar nu ramdisken med p-systemet. Man måste skapa en PAB i VDP för ramdisken. Den som är intresserad kan höra av sig till mig. Det borde fungera till andra ramdiskar också. Tack vare en del "patchar" i p-systemet kör jag nu systemet med 4×360KB diskdrivar och 480KB ramdisk. Om man lägger COMPILER, EDITOR, ASSEMBLER, FILER och LINKER på ramdisken arbetar systemet riktigt snabbt!

Med ramdisken får man en batterieliminator för 115 V. Det finns så många sådana att köpa för 220 V som lämnar 8—12V 400 mA. De kostar under hundralappen. De ramdiskar som sitter i boxen fungerar ändå (men är tomma när datorn slås på), medan de fristående *måste* ha strömförsörjning. Det finns en liten tryckknapp på kortet man ska trycka på när man slår av strömmen om man vill ha kvar filerna (endast med batterieliminator). Detta är samma sak som att öppna driven och ta ut disketten. Om man inte trycker på knappen *blir* det fel på filerna, medan jag inte fått några fel om jag tryckt på knappen.

På kortet sitter som sig bör en gul lysdiod som visar när kortet används. Den gör ingen skillnad på om man använder minnesexpansionen eller ramdisken. Det tyckte jag inte var bra, så jag satte dit en extra lysdiod som lyser bara när ramdisken används.

Jan Alexandersson frågar i sin artikel efter erfarenheter i att skriva program som använder bankswitchning. Jag undrar till vilken nytta det skulle vara. Finns den verkligen någon som skriver så vansinnigt stora assemblerprogram? Om man har behov av stora datamängder kan man faktiskt använda ramdisken till en helt fantastisk sak — som ramdisk, hör och häpna. Det går faktiskt mycket snabbt att hämta in en speciell sektor från ramdisken, särskilt om man har RELATIVE filer. Jag kör mest med p-systemet numera, och där sköter systemet själv att ha hand om så stora program så de inte rymms i minnet. Man måste visserligen segmentera programmen när man programmerar dem, men det är inte svårt. I p-systemet är alla filer relativa (utom textfiler) och man kan snabbt hitta rätt sektor (speciellt om man har ramdisk).

Den som vill köpa en Corcomp ramdisk kan vända sig till:

Texcomp
P. O. Box 33084
Granada Hills, CA 91344
USA
Tel. 0091-818-366 6631

Deras priser är:

256 KB för boxen 169,95 dollar
512 KB för boxen 229,95 dollar
256 KB fristående 249,95 dollar
512 KB fristående 269,95 dollar

Tyvärr framgår inte i deras katalog hur mycket fraktkostnaden blir. Jag köpte min ramdisk när jag var i USA i somras, så jag vet inte hur mycket det skulle bli (jag gissar på 10 till 20 dollar).

Den som vill fråga mig om något är välkommen att göra det på kvällar och helger:

Lennart Thelander
Dalhemsvägen 103 A
252 65 Helsingborg
Tel. 042/15 18 73

Möte i Göteborg

Dataklubben West 99 höll höstmöte i Göteborg den 10 oktober.

15 personer hade hörsammat inbjudan och fick vara med om ett givande möte där beslut togs på klubbens inriktning, regler m m.

Vi samlades kl 12.00 i Sjöscoutgården Torslanda där 2 kompletta anläggningar hade monterats upp. Demonstrationer av olika program och hårdvara stod på programmet. Bl a demonstrerades *Acorn 99* som visade sig vara en kraftfull databas. Vi fick även en demo av ett RAM/GRAM-kort. Detta hade en hel del möjligheter, dumpning av moduler m m. Ett uppskattat inslag var körning mot klubbens databas YARN.

Basens *sysop* fanns med och visade hur basen såg ut "inuti". West 99 har ju egna areor i basen och flera medlemmar kommer tydligen att skaffa sig modem.

Vid 15-tiden var det så dags för mat. Denna smakade förträffligt och vi fick förnyade krafter att ta itu med Texas igen. Dagen blev lång och kaffekopparna många men vi gick hem nöjda och belåtna med dagen.

Vi hoppas att till våren ha ett nytt stormöte och att modemandet då har ökat. Modemande ja, det kan ju ge oss möjligheter att få kontakt med andra användargrupper både inom och utom landet. Vår bas finns ju med i ett världsomfattande databasnät.

— Sten Gunnarsson

Checklist

I PB 87-2 publicerade vi en artikel om programmet Checklist och hur det används. Den här gången publicerar vi hela källkoden (i assembler) och en beskrivning av koden. Den är skriven av Lars Thomasson och bearbetad av Anders Persson.

SCREEN

Nollställer en flagga för att indikera att utmatningen skall komma på skärmen.

LINEL

Mottager en numerisk parameter, omvandlar den till heltal samt kontrollerar om värdet ligger inom det tillåtna intervallet 10—80 tecken per rad. Om talet visar sig vara godtagbart lagras det i den för detta ändamål reserverade minnespositionen. I de fall då detta kriterium ej uppfylles hoppar exekveringen till rutinen BAERR vilken avbryter körningen med felmeddelandet 'ERROR BAD ARGUMENT'.

DEVICE

Eftersom device ska användas när utmatningen ej önskas dirigerad till skärmen, börjar denna rutin med att ettställa en flagga vilken markerar att skärmen ej skall användas. Därefter hämtas den strängparameter som anger utenhetens namn. Lagringen av namnet, med tillhörande längd-byte, sker direkt i den PAB-buffert som finns uppbyggd i minnet efter programmet.

LISTP

Detta kommando genererar listningen av den aktuella programdelen. Två parametrar utnyttjas för att ange första och sista raden som skall listas. Efter inladdandet av egen workspace börjar rutinen med att hämta de två numeriska parametrarna och omvandla dessa till heltal. Start- och stopp-radnummer jämförs med varandra och om start är mindre än stopp, vilket ju inte förefaller vara någon direkt intelligent utgångspunkt för eventuell listning, hoppar Checklist till en rutin LNFERR som avbryter exekveringen med ett 'ERROR LINE NOT FOUND'. Skulle däremot värdena vara acceptabla hämtas nu en pekare till början, eller snarare slutet, på radnummertabellen.

Då radnumren här är sorterade i sjunkande ordning är det alltså pekaren till slutet på tabellen som pekar på radnumret i början på programmet. Denna pekare måste först justeras då den pekar på sista byten i tabellen. Tabellen är uppbyggd så att varje radnummer i programmet representeras av fyra bytes, där de två första är radnumret och de två sista radens adress i minnet. Checklist är i detta stadium endast intresserad av radnummerna och då pekaren pekar på den sista av de två adressbyten subtraheras den med tre för att komma direkt på radnumret. Efter denna operation hämtas också pekaren till slutet av tabellen (härmed avses den del av tabellen där det högsta radnumret är lagrat). Nu kommer programmet att snurra tills det antingen hittar ett radnummer större än eller lika med startvärdet eller har genomlöpt tabellen utan resultat. I det senare fallet anropas LNFERR enligt ovan. Annars övergår Checklist till att leta upp sista raden att lista. Här ifrån hoppar programmet till INIT.

LIST

Då hela Extended BASIC programmet i minnet skall listas behöver bara de två ovannämnda pekarna kopieras över och korrigeras.

INIT

När Checklist kommit hit har alla inparametrar emottagits och själva listningen kan börja. Först kontrolleras flaggan vilken indikerar huruvida den genererade listningen skall sändas till skärmen eller någon annan yttre enhet. I de fall då skärmen ej är definierad som utfil till Checklist fortsätter exekveringen vid DSROUT. Annars kalkyleras en pekare till position 30 i radbufferten. Den lagras sedan i den minnescell vilken anger maximal radlängd. Detta på grund av att den yttersta kolumnen vid var sida ej användes då dessa kan vara svåra att få synliga i sin helhet på TV-skärmen. Då tangentbordet kommer att kännas av i ett senare skede lagras en nolla i den byte som anger vilken del av bordet som skall avkännas. Detta får till följd att hela tangentbordet undersökes för nedtryckta tangenter.

När dessa åtgärder är vidtagna hoppar programmet till LOOP. Då en yttre enhet har angivits sparas en minnesarea i VDP-RAM, där sedan PABen kommer att residera. Direkt därefter uträknas den aktuella längden på utfilens PAB genom att ta längden på enhetens namn och addera tio för de i PABen ytterligare ingående oktaderna. För att DSRLNK-anrop skall kunna utföras till, för användaren, belåtenhet, måste det lagras en pekare till namnlängdsbyten i minnesutrymmet här kallat PABPTR. Denna cell har adressen >8356. En kopia av denna pekare lagras också undan för framtida användning, då PABPTR måste återställas före varje nytt DSRLNK-anrop. Då allt nu, förhoppningsvis, är färdigt för att öppna filen, göres detta. Om något ändå inte stämde, icke existerande filnamn etc, upptäcks detta av DERROR-rutinen vilken alltid anropas direkt efter DSRLNK. Beskrivning av DERROR kommer senare.

Emedan det i fortsättningen skall skrivas till enheten, ändras PABens I/O-opkod till att innehålla just koden för skrivning, en åtgärd som väl inte förefaller helt ologisk. Pekaren till det sista tecknet i radbufferten, med hänsyn taget till den angivna maximala radlängden, uträknas samt lagras. I detta skede av Checklists exekvering är allt förarbete till listningen slutfört, varför programmet nu fortsätter med listningen i nästa rutin.

LOOP

Först initieras radbuffertpekaren till att peka på den första positionen i bufferten. Sedan kontrolleras om radnummertabellsindexet är lägre än det lägsta tillåtna, eller med andra ord om hela programmet är listat. Skulle denna jämförelse antyda att så är fallet hoppar Checklist till rutinen OUT. Finns det mer att lista tager programmet nu hand om nästa rad i ordningen. Först hämtas radnumret i tabellen och kontrolleras huruvida det kan anses vara ett giltigt sådant, eller om det behagat inkomma någon, mer eller mindre, svårdefinierbar lustighet i därför ej avsedd dataarea. Skulle något oförutsett ha inträffat anropas OUT. Annars fortsätter exekveringen med NUMSTR-rutinen, vilken omvandlar radnumret till en sträng. Efter återkomsten därifrån hämtas adressen till raden och anrop sker av CHECK, där radens kontrollbokstav kommer att genereras och utskrivas först i radbufferten. Då exekveringen återupptages i LOOP lagras ett mellanslag i bufferten efter vilket radnumret kommer att följa. Detta inskrives genom att de sex tecken som utgör radnumret, flyttas, ett och ett, till sin destination.

Nu följer givetvis själva raden. Här hämtas en byte i taget. Först jämföres den med noll eftersom det innebär att raden är slut. Om så är fallet hoppar programmet till NEXT. Därefter följer test av om byten råkar vara ett kontrolltecken, vilket innebär att nästa byte skall hämtas eller om det är ett token-värde. Det sistnämnda innebär ett numeriskt värde som läggs i minnet i stället för det reserverade ord i BASIC som det motsvarar. (Se även artikeln "Merge från kassett" i Utmaningen, PB 85-1.) I så fall anropas TOKEN. Det kvarvarande alternativet är att byten är en skrivbar ASCII-kod, varvid den helt enkelt skrives i bufferten. Därefter testas om bufferten uppnått maximal radlängd. Om så är fallet körs en rutin kallad NEWL, annars tas nästa byte i raden för behandling.

TOKEN

Här omhändertager Checklist de token-värden som representerar reserverade ord. Av dessa värden kräver tre

stycken ytterligare specialbehandling, nämligen 199—201. Sålunda kontrolleras det om det aktuella värdet råkar sammanfalla med något av dessa. Skulle så vara fallet, subtraheras värdet med 199 och multipliceras med två, varefter det användes som index i en tabell. Tabellen ger adressen till rätt rutin, vilken sedan anropas. När den valda rutinen återlämnar programkörningen till TOKEN, hoppar denna tillbaka för att hämta nästa tecken i raden. Om token-värdet ej ingår i ovannämnda intervall provas det först om det är en satsavdelare, dvs ett "::". Är tecknet inte ett sådant, hoppar Checklist till TESTEL. Annars skrivs radbufferten ut via PRINT rutinen, varefter exekveringen återupptages vid TABPR. TESTEL kontrollerar om token-värdet motsvarar ELSE, emedan detta värde skapar ett visst problem. Det är nämligen så att ELSE ibland föregås av ett blanktecken och ibland inte. Detta måste sålunda kontrolleras för att utskriften inte ska få ett närmast tragiskt utseende. Detta test utföres naturligtvis endast om det verkligen är ett ELSE. Annars fortsätter programmet vid TABPR. Följden av testet är att om det inte finns något mellanslag före provningen så skall det i alla fall finnas ett efteråt. På dessa slingrande vägar har programmet nu kanske kommit till TABPR, där alla token-värden utom 199—201 skrivs in i radbufferten. I likhet med TOKEN räknas det också här ut ett tabellindex, denna gång med formeln värdet subtraherat med 129 och multiplicerat med två. Utnyttjande detta index erhålles, med hjälp av en tabell kallad TABLE, adressen till de data som motsvarar respektive token-värde. Första byten i datafältet anger längden på det reserverade ordet. Om denna längd är noll existerar inte detta token-värde, varför nästa tecken hämtas och behandlas. I övriga, normala fall flyttas det antal tecken som längdbyten anger från datafältet till radbufferten. Efter varje bokstavsflyttning utföres det en kontroll av om maximal radlängd uppnåtts. När så sker ska NEWL anropas. När hela det reserverade ordet är skrivet hoppar Checklist tillbaka och hämtar nästa tecken i raden.

NEXT

Denna rutins existensberättigande ligger i att den först anropar PRINT för att skriva ut det aktuella innehållet från radbufferten, Sedan justerar den radnummertabellspekaren till att peka på nästa rad.

OUT

Först måste givetvis undersökas om utmatningen skett på skärmen eller till någon yttre enhet via DSRLNK. Om flaggan visar på att skärmen nyttjats för att ge användaren den önskade informationen, skall programkörningen hoppa till COMEND. Annars utföres följande operationer. Utfilen stängs genom att opkoden för detta laddas in i PABens första byte, varefter namnlängdspekaren återställes och ett DSRLNK-anrop tager plats. Som vanligt följs detta omedelbart av att rutinen DERROR exekveras för att ett eventuellt uppkommet fel skall detekteras och rapporteras. Den del av VDP-RAM som sparades, för att ge plats åt PABen, flyttas nu tillbaka i det ädla syftet att undvika oförutsägbara sensationer när Checklist återlämnar kontrollen till Extended BASIC. COMEND avslutar helt enkelt exekveringen av Checklist.

NUMSTR

Denna rutin omvandlar heltal till ASCII-strängar genom att flytta över resten vid heltalsdivision med tio till talbufferten tills positionerna i denna är fyllda med korrekt värde. Då det i början av strängen kan finnas nollor, indikerande att talets ASCII-koder ännu ej börjat på grund av att heltalet icke består av de maximalt fem siffror som står till förfogande, kontrollerar Checklist om det aktuella talet har några dylika. Om så är fallet utbytes de mot mellanslag. För att möjliggöra en meningsfull utskrift, adderas till de återstående siffrorna ett offset på 48 i syfte att transsubstansiera dessa till motsvarande skrivbara ASCII-koder. Då heltalet på detta sätt blivit en, högerinpassad, sträng i talbufferten, avslutas den med ett nytt blanktecken.

CHECK

Rutinen här är ansvarig för uträknandet av varje rads kontrollbokstav. Bokstaven kalkyleras med hjälp av de koder som finns lagrade i minnet, dvs token-värden, ASCII-tecken och heltal. Först sparas pekaren till raden, vartefter radlängden hämtas. Då denna information finns är det endast att, i en repetition, beräkna checksumman för ett tecken i taget, genom anrop av SUM. Efter detta återstår att tillägga checksumman för radnumret vilken uträknas bytevis. Den på detta sätt uträknade checksumman skalas sedan ner till att bli ett tal motsvarande ASCII-koden för någon versal. Kontrollbokstaven lagras först i radbufferten, radpekaren återställes och rutinen gör ett återhopp.

SUM

Här beräknas checksumman för varje enskild byte. Denna adderas till radens ackumulerade summa. Den använda formeln är en variant av Horners metod, vilken resulterar i ett olinjärt samband mellan ett tecken och dess checksumma. Detta skall, förhoppningsvis, försvåra för ett fel på raden att ta ut ett annat.

NEWL

Om en rad har uppnått sin maximalt tillåtna längd anropas denna rutin för att skapa flera rader i utfilen. Det första som inträffar är att den fulla raden skickas i väg via PRINT-rutinen. Efter detta återställes radbuffertpekaren till början på bufferten, varvid det där inlagras blanktecken för att texten skall bli vänterjusterad vid utskriften.

PRINT

Härifrån sköts all utmatning från Checklist. Om en rad innehåller åtta tecken är den tom, varför det kan betraktas som ganska onödigt att skriva ut den. Därför görs det inte. I det fall en rad består av nio tecken och det nionde tecknet är ett mellanslag är raden inte heller särskilt matnyttig, enär en rad med så få tecken endast kan uppstå vid delning av en rad som ernått maxlängden. En sådan rad inledes alltid med åtta blanktecken. Då raden nu befunnits värdig att utskrivas räknas dess längd ut. Återigen testas det om listningen skall komma på skärmen eller ej. Skall den inte dit, hoppar exekveringen till DSRPR. Annars kalkyleras hur många blanktecken som måste fyllas på i slutet av raden, för att denna garanterat skall överskriva hela den gamla raden på skärmen. Då skärmen i Extended BASIC-läge har ett offset på >60 till ASCII-koderna, får detta adderas till tecknen i radbufferten. Härpå utfylles med blanktecken till det ovan uträknade antalet. Skärmen scrollas genom att alla rader utom den översta läses in och utskrives omedelbart igen, denna gång en rad högre upp. Den på tidigare beskrivna sätt preparerade radbufferten flyttas ut till understa raden.

Då skärmlistningen torde bli totalt oläslig för alla utom extrema snabbläsare av omänsklig art, eller eventuellt ägare till höghastighetskameror, känner Checklist av tangentbordet för kontroll av någon tangentnedtryckning. Om en sådan upptäcks stoppar programmet listningen tills det att tangenten är uppsläppt och en ny tryckning på valfri tangent utförts samt avslutats med densammas avlastning. Då allt detta gjorts, återvänder programmet och ger sig i kast med nästa rad. Vid utmatning via DSRLNK skrivs radlängden ut i den för detta ändamål existerande byten i PABen varefter namnpekaren och PABens flagg/status-byte återställes till ursprungligt skick. Själva skrivningen kan nu utföras genom ett anrop till DSRLNK med vidhängande felsökning i DERROR.

T199

Token-värde 199 får sin specialbehandling här. Detta värde innebär att det följs av en byte som anger en längden på en sträng, samt de ingående ASCII-tecknen. Denna beskrivning gäller också värdet 200 med skillnaden att 199 innebär att strängen skall omslutas av citations-

tecken. Sålunda skrivs ett dylikt ut, varpå T200 anropas följt av ett nytt citationstecken.

T200

Här tas längdbyten fram och det antal tecken denna anger kopieras över till radbufferten, under det att radlängden kontinuerligt jämföres med den maximalt angivna.

T201

Detta token-värde anger att det följs av två bytes vilka äro att betrakta såsom ett 16-bitars heltal. Heltalet omvandlas i den tidigare avhandlade rutinen NUMSTR till en sträng vilken, efter borttagande av inledande blanktecken, inskrives i radbufferten.

DSRERR

Här kontrolleras huruvida en DSRLNK-körning har förorsakat något fel. I tur och ordning kontrolleras equalbiten i statusregistret — detta är ju lagrat i R15 efter en BLWP — condbiten i GPL statusbyte samt flagg/statusbyten i PABen. Om någon av dessa indikerar att fel uppstått, resulterar detta i att ERR-rutinen anropas efter det att felkoden för I/O-fel lagrats i R0 samt att en pekare till PABen lagrats i >8304.

Observera att Extended BASIC förutsätter att varje PAB är fjorton bytes lång, förutom filnamnet. (Se E/A-manualen sidan 302.) Pekaren skall alltså ange PABens adress minus fyra. Om detta står det noll och intet i manualen. Dessutom anger den att pekaren ska finnas i >831C i stället för i >8304. Beskrivningen till Mini Memory säger däremot >8304. Inget där heller om att adressen ska minskas med fyra. Detta lyckades dock hr A Persson fundera ut med hjälp av programmet *Explorer*, när vi funderade över varför Checklist gav ifrån sig mystiska felkoder.

DSRLNK

Denna rutin är skriven av hr Utmaningsredaktören A Persson, varför eventuella kommentarer häröver, med varm hand, överlåtes till nyss nämnda individ.

(Tack så mycket. Minnesgoda läsare drar sig kanske till minnes att den rutinen figurerat i tidningen tidigare, nämligen i LT-Support, PB 85-3. Den utnyttjar det program för DSR-anrop som finns i operativsystemet, genom att göra diverse tricks. För att fullt förstå varför den ser ut som den gör, måste du känna till lite om hur GPL fungerar. En detaljerad förklaring skulle därför föra för långt i det här sammanhanget. Om du vill utnyttja den i något eget program ska du dock tänka på att den endast kan användas ihop med Extended BASIC! Det beror på att den utnyttjar värden som finns i GROM i Extended BASIC. Jag är medveten om att det här går att lösa smartare, men alla är ju barn i början, sägs det.)

```
*********************
*
* CAT-COMPUTING'S CHECKLIST-UTILITY
*
* THIS PROGRAM CALCULATES A CHECKSUM CHARACTER FOR EACH LINE IN A BASIC-PROGRAM,
* THE CHARACTER IS PRINTED IN FRONT OF IT'S LINE.
*
* IN ORDER TO MAKE THE LISTING MORE READABLE THIS UTILITY DEVIDES THE LINES
* SO THAT ONLY ONE STATEMENT IS PRINTED ON EACH ROW.
*
*
* TO USE THIS PROGRAM JUST DO AS THE FOLLOWING.
* FIRST LOAD THE X-BASIC PGM IN MEMORY.
* AFTER THAT CALL UPON THIS UTILITY BY:
*    CALL INIT
*    CALL LOAD("DSKn.CHECKLIST")
*    CALL LINK("LIST")
*
* THE OUTPUT DEVICE IS DEFINED BY:
*    CALL LINK("DEVICE",DEVICE_NAME)   (DEFAULT = PIO)
*
* IF THE OUTPUT LIST SHOULD COME ON SCREEN JUST TYPE IN:
*    CALL LINK("SCREEN")
* IN THIS MODE THE OUTPUT WILL STOP WHEN A KEY IS PRESSED, TO RESUME LISTING
* JUST PRESS A KEY AGAIN.
*
* TO SET A MAXIMUM LENGTH ON EACH ROW OF OUTPUT FILE:
*    CALL LINK("LINEL",MAX_ROW_LENGTH)   (DEFAULT = 80)
* 10 <= MAX_ROW_LENGTH <= 80
*
* TO ONLY PRINT LINES INBETWEEN AND INCLUDING TWO GIVEN LINE-NUMBERS
* USE:
*    CALL LINK("LISTP",START_LINE_#,STOP_LINE_#)
* THE GIVEN LINE-NUMBERS DO NOT HAVE TO EXIST IN THE X-BASIC PGM
*
* OUTPUT FILE ON DISKETT IS CREATED AS DIS/VAR 80 WHICH MEANS IT CAN BE READ
* FROM TI-WRITER AND EDITOR/ASSEMBLER EDITORS.
*
*
* PGM BY CAT-COMPUTING 860918.
*

       DEF  LIST,LISTP,DEVICE,LINEL,SCREEN

* REGISTERS USED

ADDR   EQU  0
ARRAY  EQU  0
PARNUM EQU  1
INDEX  EQU  2
STRBUF EQU  2
MSW    EQU  3
LSW    EQU  4
COUNT  EQU  5
```

```
REP    EQU  6
SEND   EQU  7
CHAR   EQU  8
TABMIN EQU  9
SPTR   EQU  10
RTN    EQU  11
LENGTH EQU  12
MINNUM EQU  13
CHSUM  EQU  13
MAXNUM EQU  14
BUFIND EQU  14
TINDEX EQU  15

* VDP ACCESS

VDPADD EQU  0
CPUADD EQU  1
VALUE  EQU  1
SIZE   EQU  2

PAB    EQU  >1000
VDPBUF EQU  >1028
CFI    EQU  >12B8
ERRLNF EQU  >1600
ERRBA  EQU  >1C00
NUMREF EQU  >200C
STRREF EQU  >2014
XMLLNK EQU  >2018
KSCAN  EQU  >201C
VSBW   EQU  >2020
VMBW   EQU  >2024
VMBR   EQU  >202C
ERR    EQU  >2034
ERRIO  EQU  >2400
STOP   EQU  >8330
START  EQU  >8332
FAC    EQU  >834A
PABPTR EQU  >8356
KDEV   EQU  >8374
KEY    EQU  >8375
GPLST  EQU  >837C
GPLWS  EQU  >83E0

*--------------------------------------

SCREEN
       LWPI LISTWS

       CLR  @SCRFLG          CLEARS FLAG FOR SCREEN OUTPUT

       LWPI GPLWS
       B    *RTN

*--------------------------------------

LINEL
       LWPI LISTWS

       CLR  ARRAY            GET LINE-LENGTH
       LI   PARNUM,1
       BLWP @NUMREF
       BLWP @XMLLNK
       DATA CFI
       C    @FAC,@MINLEN     CHECKS IF LINE LENGTH IN 10..80
       JLT  BAERR
       C    @FAC,@MAXLEN
       JH   BAERR

       MOV  @FAC,@ROWUSR

       LWPI GPLWS
       B    *RTN

*--------------------------------------

BAERR
       LI   R0,ERRBA         RETURNS 'ERROR BAD ARGUMENT'
       BLWP @ERR

*--------------------------------------

DEVICE
       LWPI LISTWS

       SETO @SCRFLG          SETS FLAG FOR NOT SCREEN OUTPUT

       CLR  ARRAY            GET DEVICE NAME
       LI   PARNUM,1
       LI   STRBUF,PABDAT+9
       LI   CHAR,>1A00       26 IN MSBy
       MOVB CHAR,@PABDAT+9
       BLWP @STRREF

       LWPI GPLWS
       B    *RTN

*--------------------------------------

LNFERR
       LI   R0,ERRLNF        RETURNS 'ERROR LINE NOT FOUND'
       BLWP @ERR

*--------------------------------------

LISTP
       LWPI LISTWS
       LI   SPTR,STACK

       CLR  ARRAY            GET START LINE-#
       LI   PARNUM,1
       BLWP @NUMREF
       BLWP @XMLLNK
       DATA CFI
       MOV  @FAC,MINNUM

       INC  PARNUM           GET STOP LINE-#
       BLWP @NUMREF
       BLWP @XMLLNK
       DATA CFI
       MOV  @FAC,MAXNUM

       C    MINNUM,MAXNUM    CHECK IF VALUES ACCEPTABLE
       JH   LNFERR           NO
```

```
        MOV  @START,TINDEX      GET START ADDRESS OF LINE-# TABLE
        AI   TINDEX,-3
        MOV  @STOP,TABMIN       GET STOP ADDRESS
GETMIN
        C    TINDEX,TABMIN      FIND START LINE TO BE PRINTED
        JL   LNFERR             LINES NOT FOUND

        MOVB *TINDEX+,LSW
        SWPB LSW
        MOVB *TINDEX+,LSW
        SWPB LSW

        C    LSW,MINNUM         START LINE FOUND?
        JHE  GOTMIN             YES

        AI   TINDEX,-6
        JMP  GETMIN
GOTMIN
        AI   TINDEX,-2
GETMAX
        MOVB *TABMIN+,LSW       FIND STOP LINE TO BE PRINTED
        SWPB LSW
        MOVB *TABMIN+,LSW
        SWPB LSW

        C    LSW,MAXNUM         STOP LINE FOUND?
        JLE  GOTMAX             YES

        AI   TABMIN,2
        JMP  GETMAX
GOTMAX
        AI   TABMIN,-2
        JMP  INIT

*--------------------------------------

LIST
        LWPI LISTWS
        LI   SPTR,STACK

        MOV  @START,TINDEX      GET START ADDRESS IN LINE-# TABLE
        AI   TINDEX,-3
        MOV  @STOP,TABMIN

*--------------------------------------

INIT
        MOV  @SCRFLG,@SCRFLG    OUTPUT ON SCREEN?
        JNE  DSROUT             NO

        LI   LENGTH,BUFFER      STORE POINTER TO LAST CHARACTER IN LINE
        AI   LENGTH,30
        MOV  LENGTH,@ROW

        CLR  @KDEV              KSCAN CHECKS ENTIRE KEYBOARD

        JMP  LOOP
DSROUT
        LI   VDPADD,PAB         SAVE VDP
        LI   CPUADD,VDPSAV
        LI   SIZE,120
        BLWP @VMBR
```

```
        LI   CPUADD,PABDAT      WRITE PAB TO VDP
        MOVB @PABDAT+9,SIZE
        SRL  SIZE,8
        AI   SIZE,10
        BLWP @VMBW

        LI   CHAR,PAB+9         POINTER TO NAME-LENGTH
        MOV  CHAR,@PABPTR
        MOV  CHAR,@TEMPPT

        BLWP @DSRLNK            OPEN DEVICE
        DATA 8

        BLWP @DERROR

        LI   VDPADD,PAB         PREPARE PAB FOR WRITING
        LI   VALUE,>0300
        BLWP @VSBW

        LI   LENGTH,BUFFER      CALCULATES POINTER TO LAST CHARACTER IN LINE
        MOV  @ROWUSR,@ROW
        A    LENGTH,@ROW
LOOP
        LI   BUFIND,BUFFER      INITIALIZES BUFFER POINTER

        C    TINDEX,TABMIN      CHECK IF ALL TABLE DONE
        JL   OUT                YES

        MOVB *TINDEX+,LSW       GET LINE-#
        SWPB LSW
        MOVB *TINDEX+,LSW
        SWPB LSW

        CI   LSW,>7FFF          RANGE CHECKING
        JH   OUT

        MOV  LSW,@LINNUM
        BL   @NUMSTR            CONVERT LINE-# TO STRING

        MOVB *TINDEX+,INDEX     GET LINE ADDRESS
        SWPB INDEX
        MOVB *TINDEX+,INDEX
        SWPB INDEX

        BL   @CHECK             CALCULATE AND PRINT CHECK CHARACTER

        MOVB @SPACE,*BUFIND+

        LI   REP,NUM            PRINT LINE-#
        LI   COUNT,6
LNPR
        MOVB *REP+,*BUFIND+
        DEC  COUNT              ALL DIGITS PRINTED?
        JNE  LNPR               NO
LINE
        CLR  CHAR               PRINTS ONE LINE
        MOVB *INDEX+,CHAR
        CI   CHAR,0
        JEQ  NEXT               END OF LINE, GET NEXT

        CB   CHAR,@LIMIT        CONTROLL-CODES?
        JL   LINE               YES
```

```
        CB    CHAR,@LIMIT+1      TOKEN-VALUE?
        JH    TOKEN              YES

        MOVB  CHAR,*BUFIND+      PRINTABLE ASCII CHARACTER
        C     BUFIND,@ROW
        JNE   CON1
        BL    @NEWL
CON1
        JMP   LINE

TOKEN
        CB    CHAR,@SPECIA       PRINTS RESERVED WORDS
        JL    TESTDC             TOKEN VALUE LESS THAN 199
        CB    CHAR,@SPECIA+1
        JH    TABPR              TOKEN VALUE GREATER THAN 201

        AI    CHAR,->C700        199 IN MSBy
        SRL   CHAR,7
        MOV   @TOKTAB(CHAR),ADDR     GET ADDRESS TO ROUTINE
        BL    *ADDR

        JMP   LINE
TESTDC
        CI    CHAR,>8200         STATEMENT SEPARATOR?
        JNE   TESTEL             NO

        BL    @PRINT             SEND ROW TO DESTINATION
        LI    BUFIND,BUFFER
        JMP   TABPR
TESTEL
        CI    CHAR,>8100         ELSE STATEMENT?
        JNE   TABPR              NO

        DEC   BUFIND
        CB    *BUFIND,@SPACE     LAST CHARACTER A SPACE?
        JNE   ISPACE             NO
        INC   BUFIND
        JMP   TABPR
ISPACE
        INC   BUFIND             INSERT EXTRA SPACE
        MOVB  @SPACE,*BUFIND+

        C     BUFIND,@ROW        SPLIT ROW?
        JNE   TABPR
        BL    @NEWL
TABPR
        AI    CHAR,->8000        128 IN MSBy
        SRL   CHAR,7
        MOV   @TABLE(CHAR),ADDR      GET ADDRESS OF DATA

        MOVB  *ADDR+,COUNT       GET LENGTH OF WORD
        SRL   COUNT,8
        JEQ   LINE               NO DATA, GET NEXT CHARACTER
TABPRL
        MOVB  *ADDR+,*BUFIND+    PRINTS THE DATA
        C     BUFIND,@ROW
        JNE   CON2
        BL    @NEWL
CON2
        DEC   COUNT              END OF WORD?
        JNE   TABPRL             NO

        JMP   LINE               GET NEXT CHARACTER IN LINE
```



```
NEXT
        BL    @PRINT             SEND ROW

        AI    TINDEX,-8
        JMP   LOOP
OUT
        MOV   @SCRFLG,@SCRFLG    SCREEN OUTPUT?
        JNE   DSREND             NO

        JMP   COMEND
DSREND
        LI    VALUE,>0100        CLOSE DESTINATION FILE
        LI    VDPADD,PAB
        BLWP  @VSBW
        INC   VDPADD
        LI    VALUE,PAB1
        MOV   @TEMPPT,@PABPTR
        BLWP  @DSRLNK
        DATA  8
        BLWP  @DERROR

        LI    R2,120
        LI    R1,VDPSAV          RESTORE VDP
        BLWP  @VMBW
COMEND
        LI    R0,80
        MOV   R0,@ROW
        LWPI  GPLWS              RETURN TO CALLING PROGRAM
        B     *RTN

*---------------------------------------

NUMSTR
        DECT  SPTR               CONVERTS INTEGER IN LSW TO ASCII STRING
        MOV   RTN,*SPTR          IN NUM

        LI    REP,NUM
        AI    REP,4
CONV
        CLR   MSW
        DIV   @TEN,MSW
        SWPB  LSW
        MOVB  LSW,*REP
        MOV   MSW,LSW
        DEC   REP
        CI    REP,NUM
        JHE   CONV               NUMBER NOT DONE

        LI    REP,NUM            INSERT SPACES AND CONVERT TO ASCII CODES
        LI    COUNT,5
BLANK
        MOVB  *REP,*REP          INSERT SPACE?
        JNE   ASCI               NO

        MOVB  @SPACE,*REP+
        DEC   COUNT              NUMBER DONE?
        JEQ   ASCO               YES
        JMP   BLANK
ASCI
        AB    @ASCII,*REP+
        DEC   COUNT              NUMBER DONE?
        JNE   ASCI               NO
```

```
ASCO
        MOVB @SPACE,*REP
        MOV  *SPTR+,RTN
        B    *RTN

*--------------------------------------

CHECK
        DECT SPTR               CALCULATES CONTROL CHARACTER
        MOV  RTN,*SPTR

        MOV  INDEX,@LINADD
        DEC  INDEX              GET LINE LENGTH
        MOVB *INDEX+,LENGTH
        SRL  LENGTH,8

        CLR  CHSUM
CSL
        MOVB *INDEX+,CHAR       CHARACTER FOR LINE
        SRL  CHAR,8
        BL   @SUM
        DEC  LENGTH             LINE DONE?
        JNE  CSL                NO

        MOVB @LINNUM,CHAR       CHARACTER FOR LINE-#, MSBy
        SRL  CHAR,8
        BL   @SUM

        MOVB @LINNUM+1,CHAR     LSBy
        SRL  CHAR,8
        BL   @SUM

        MOV  CHSUM,LSW          MSBy IN SEND := ( CHSUM MOD @MOD ) + @OFFSET
        CLR  MSW
        DIV  @MOD,MSW
        A    @OFFSET,LSW
        MOV  LSW,SEND
        SLA  SEND,8

        MOVB SEND,*BUFIND+      SENDS CONTROL CHARACTER TO DESTINATION

        MOV  @LINADD,INDEX

        MOV  *SPTR+,RTN
        B    *RTN

*--------------------------------------

SUM
        DECT SPTR               CALCULATES CHECKSUM FOR ONE CHARACTER, OR
        MOV  RTN,*SPTR          TOKEN VALUE, WICH IS FOUND IN LSBy OF CHAR
*                               USING HORNER'S METHOD
        LI   MSW,1
        MOV  @SUMTAB,COUNT      GET NUMBER OF EXPONENTS
CSL1
        MOV  @SUMTAB(COUNT),REP     GET EXPONENT
CSL2
        MPY  CHAR,MSW
        MOV  LSW,MSW
        DEC  REP                EXPONENTIATION DONE?
        JNE  CSL2               NO
        INC  MSW
        DECT COUNT              EXPRESSION MADE?
        JNE  CSL1               NO

        DEC  MSW
        A    MSW,CHSUM

        MOV  *SPTR+,RTN
        B    *RTN

*--------------------------------------

NEWL
        DECT SPTR               SPLITS LONG LINE INTO TWO
        MOV  RTN,*SPTR

        BL   @PRINT             SENDS FIRST PART TO DESTINATION

        LI   BUFIND,BUFFER
        LI   R1,8
NLLOOP
        MOVB @SPACE,*BUFIND+    STARTS NEW LINE WITH SPACES
        DEC  R1
        JNE  NLLOOP

        MOV  *SPTR+,RTN
        B    *RTN

*--------------------------------------

PRINT
        DECT SPTR               SENDS ROW IN BUFFER TO DESTINATION FILE
        MOV  RTN,*SPTR

        CI   BUFIND,BUFFER+8    LINE LENGTH = 8? (=NO STATEMENT ON THAT ROW)
        JNE  SPTEST
        B    @PRINTO            YES
SPTEST
        CI   BUFIND,BUFFER+9    LINE ONLY CONTAINS A SPACE?
        JNE  CONPRI             MORE THAN ONE CHARACTER
        DEC  BUFIND
        CB   *BUFIND,@SPACE     ONLY CHARACTER ON LINE A SPACE?
        JEQ  PRINTO             YES
        INC  BUFIND
CONPRI
        MOV  INDEX,@LINADD
        MOV  ADDR,@LINNUM

        MOV  BUFIND,R1          CALCULATES ROW LENGTH
        LI   BUFIND,BUFFER
        S    BUFIND,R1

        MOV  @SCRFLG,@SCRFLG    OUTPUT ON SCREEN?
        JNE  DSRPR              NO

        LI   R0,31              CALCULATES NUMBER OF EXTRA SPACES AT END OF LINE
        S    R1,R0

        LI   R2,BUFFER          ADD X-BASIC SCREEN OFFSET TO ASCII VALUES
XOFFSE
        AB   @SCROFF,*R2+
        DEC  R1
        JNE  XOFFSE
```

```
        MOV   R0,R0             INSERT EXTRA SPACES?
        JEQ   NOXSP             NO
XBSPAC
        MOVB  @XSPACE,*R2+      INSERT SPACES
        DEC   R0
        JNE   XBSPAC
NOXSP
        LI    R0,32             SCROLL SCREEN
        LI    R1,SCROLL
        LI    R2,736
        BLWP  @VMBR
        CLR   R0
        BLWP  @VMBW

        MOV   R2,R0             WRITE NEW LINE
        LI    R1,BUFFER-1
        LI    R2,32
        BLWP  @VMBW

        BLWP  @KSCAN            STOP PRINTING IF KEY PRESSED
        CB    @KEY,@NOKEY
        JEQ   NOERR2            NO KEY PRESSED
WAIT1
        BLWP  @KSCAN            KEY RELEASED?
        CB    @KEY,@NOKEY
        JNE   WAIT1             NO
WAIT2
        BLWP  @KSCAN            START PRINTING KEY PRESSED
        CB    @KEY,@NOKEY
        JEQ   WAIT2             YES
WAIT3
        BLWP  @KSCAN            RESUME PRINTING WHEN KEY RELEASED
        CB    @KEY,@NOKEY
        JNE   WAIT3

        JMP   NOERR2
DSRPR
        LI    R0,PAB+5          WRITES ROW LENGTH TO PAB
        SLA   R1,8
        BLWP  @VSBW

        MOV   R1,R2             WRITES ROW TO VDP
        SWPB  R2
        LI    R0,VDPBUF
        LI    R1,BUFFER
        BLWP  @VMBW

        MOV   @TEMPPT,@PABPTR   RESTORE POINTER TO PAB NAME-LENGTH
        LI    R0,PAB+1          RESTORE PAB FLAG/STATUS BYTE
        LI    R1,PAB1

        BLWP  @DSRLNK           SEND ROW TO DESTINATION
        DATA  8
        BLWP  @DERROR
NOERR2
        MOV   @LINADD,INDEX
        MOV   @LINNUM,ADDR
PRINTO
        MOV   *SPTR+,RTN
        B     *RTN

*---------------------------------------
```

```
T199
        DECT  SPTR              TOKEN VALUE 199
        MOV   RTN,*SPTR

        LI    SEND,'" '         PRINT QUOTE SIGN
        MOVB  SEND,*BUFIND+
        C     BUFIND,@ROW
        JNE   CON3
        BL    @NEWL
CON3
        BL    @T200             PRINT STRING

        LI    SEND,'" '         PRINT QUOTE SIGN
        MOVB  SEND,*BUFIND+
        C     BUFIND,@ROW
        JNE   CON4
        BL    @NEWL
CON4
        MOV   *SPTR+,RTN
        B     *RTN

*---------------------------------------

T200
        DECT  SPTR              TOKEN VALUE 200
        MOV   RTN,*SPTR

        MOVB  *INDEX+,COUNT     GET LENGTH OF STRING
        SRL   COUNT,8           LENGTH = 0
        JEQ   T2000             YES
T200L
        MOVB  *INDEX+,*BUFIND+  PRINT STRING
        C     BUFIND,@ROW
        JNE   CON5
        BL    @NEWL
CON5
        DEC   COUNT
        JNE   T200L
T2000
        MOV   *SPTR+,RTN
        B     *RTN

*---------------------------------------

T201
        DECT  SPTR              TOKEN VALUE 201
        MOV   RTN,*SPTR

        MOVB  *INDEX+,LSW       GET INTEGER
        SWPB  LSW
        MOVB  *INDEX+,LSW
        SWPB  LSW

        BL    @NUMSTR           CONVERT TO STRING

        LI    COUNT,6
        LI    REP,NUM
T201L1
        MOVB  *REP+,SEND        STARTS WITH SPACE
        CB    SEND,@SPACE
        JNE   T201L3            NO

        DEC   COUNT             NUMBER DONE?
        JNE   T201L1            NO
        JMP   T2010             YES
```

```
T201L2
        MOVB *REP+,SEND
T201L3
        MOVB SEND,*BUFIND+      PRINTS DIGITS
        C    BUFIND,@ROW
        JNE  CON6
        BL   @NEWL
CON6
        DEC  COUNT              NUMBER DONE?
        JNE  T201L2             NO
T2010
        MOV  *SPTR+,RTN
        B    *RTN

*--------------------------------------

DSRERR
        COC  @MASK,R15          CHECKS IF ANY ERROR HAS OCCURED UNDER A DSRLNK
        JEQ  IOERR              IF SO AN I/O ERROR IS RETURNED

        MOV  @GPLST,R2          ERROR IN GPL STATUS BYTE?
        COC  @MASK,R2
        JEQ  IOERR

        LI   R0,PAB+1           ERROR IN PAB FLAG/STATUS BYTE
        BLWP @VSBR
        SRL  R1,13
        JNE  IOERR

        RTWP                    NO ERROR, CONTINUE
IOERR
        LI   R0,PAB-4           TELL ERROR ROUTINE WHERE PAB IS
        MOV  R0,@>8304
        LI   R0,ERRIO           RETURNS 'I/O ERROR'
        BLWP @ERR

*--------------------------------------

**************************
*
* DSRLNK ROUTINE TO BE USED IN ASSEMBLY PROGRAMS CALLED BY EXTENDED BASIC.
* APPEARS EXACTLY LIKE THE ED/AS DSRLNK TO THE CALLING PROGRAM.
* BLWP @DSRLNK
* DATA >8          (FOR NORMAL DSR CALL)
* IN CASE OF AN ERROR, THE EQUAL BIT IS SET UPON RETURN.
*
* NOTE: THIS ROUTINE AUTOMATICALLY CHECKS IF THE FILENAME IS CS1 OR CS2,
* WHICH MEANS A GPL DSR, OR ANY OTHER, WHICH MEANS ML DSR.
* THIS IS AN IMPROVEMENT OVER THE ED/AS DSRLNK, AND RELIEVES THE CALLING
* PROGRAM FROM THAT TASK.
*
* THE NAME LENGTH POINTER MUST BE STORED @>8356 BY THE CALLING PROGRAM.
* ALSO, IN CASE OF AN ERROR THAT IS SUPPOSED TO BE REPORTED BY THE ERR
* ROUTINE, THE CALLING PROGRAM MUST STORE THE PAB POINTER @>8304.
*
* A-DATA 841011
*
**************************

PAD     EQU  >8300
NMLPNT  EQU  >8356
DSRMOD  EQU  >836D
SP      EQU  >8373
STATUS  EQU  >837C
GPL11   EQU  >83F6
GRMWA   EQU  >9C02
GRMRA   EQU  >9802
GRMRD   EQU  >9800
*                       DSRLNK HAS NO DEF, SINCE IT IS SUPPOSED TO BE
VSBR    EQU  >2028      CALLED BY AN ASSEMBLY PROGRAM ONLY.


DSRLNK  DATA DSRWS,DSRPGM
SAVGRM  DATA 0          TO SAVE GROM ADDRESS IN
DSRWS   BSS  >20
HIGROM  BYTE >00
LOGROM  BYTE >10
EQUMSK  DATA >2000      POSITION OF EQUAL BIT IN ST AND COND BIT IN GPLST
IOERMS  DATA >E000      MASK FOR I/O ERROR CODE IN PAB STATUS BYTE

**********

DSRPGM  MOVB @GRMRA,@SAVGRM     SAVE GROM ADDRESS
        MOV  *R14+,R0           GET DSR MODE
        MOVB @GRMRA,@SAVGRM+1
        DEC  @SAVGRM            CORRECT GROM ADDRESS
        MOVB @HIGROM,@GRMWA     SET UP NEW GROM ADDRESS
        MOV  @GPL11,R10      SAVE GPL RETURN ADDRESS
        MOVB @LOGROM,@GRMWA

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

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

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

        MOVB R0,@GRMWA          SET UP NEW GROM ADDRESS
        SWPB R0
        MOVB R0,@GRMWA

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

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

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

```
      MOVB @STATUS,R2   ZERO. MUST CHECK STATUS BYTE
      CZC  @EQUMSK,R2
      JEQ  NOERR        IF ZERO THEN NOERROR
ERROR SRL  R1,5         STORE ERROR CODE IN MSB OF R0 (CALLING WS)
      CLR  *R13
      MOVB R1,*R13
      SOC  @EQUMSK,R15  SET EQUAL BIT IN ST
NOERR MOVB @SAVGRM,@GRMWA    RESTORE GROM ADDRESS
      MOV  R10,@GPL11        RESTORE GPL RETURN ADDRESS
      MOVB @SAVGRM+1,@GRMWA
      RTWP
**********
*--------------------------------
PAB1  BYTE >12
SPACE BYTE >20
ASCII BYTE 48
SCROFF BYTE >60
NOKEY BYTE >FF
XSPACE BYTE >80
BUFFER BSS  80
LISTWS BSS  32
ERRWS BSS  32
STACK BES  20
NUM   BSS  6
SCROLL BSS  736
VDPSAV BSS  120
TEN   DATA 10
LIMIT DATA >207F
SPECIA DATA >C7C9
MASK  DATA >2000
ROWUSR DATA 80
ROW   DATA 0
TEMPPT DATA 0
LINNUM DATA 0
LINADD DATA 0
SCRFLG DATA >FFFF
MOD   DATA 26
OFFSET DATA 65
MINLEN DATA 10
MAXLEN DATA 80
SUMTAB DATA 8           # OF EXPONENTS * 2
      DATA 1,4,7,4      EXPONENTS FROM OUTSIDE IN HORNER'S METHOD
DERROR DATA ERRWS,DSRERR
PABDAT DATA >0012,VDPBUF,>5000,>0000,>0003
      TEXT 'PID'
      BSS  26
TOKTAB
      DATA T199,T200,T201
```

Apropå Alfaslang

av Niels Hovmöller

Jag blev lite ledsen när jag såg listningen av mitt program "Alfaslang" i förra numret av Programbiten. En hel massa tomma DISPLAY AT-satser! Satserna innehåller – som Du kanske redan listat ut – kontrolltecken som omdefinierats. De syns hur bra som helst på min bildskärm och jag har ingen printer som kan skriva ut mina Texas-program. Därav också de tokiga programraderna 1850 och 1900. Gör så här för att skriva in rad 790 t o m 980 korrekt:

1. Skriv in (utan radnummer eller med radnummer 90):

```
FOR I=120 TO 143 :: CALL CHARPAT(I,T$) :: CALL CHAR
(I-63,T$) :: NEXT I
```

2. Skriv in följande rad:

```
655 GOTO 790
```

3. Håll CONTROL-tangenten nere när du skriver in detta mellan citationstecken i raderna 790 t o m 980:

```
790 DISPLAY AT(2,1):"AAACBEAABBAAAABBBBBACBEAAAAA"
800 DISPLAY AT(3,1):"AACFDBEABBAAAABBAAACFDBEAAAA"
810 DISPLAY AT(4,1):"AABAABBABBAAAABBAAABAABBAAAA"
820 DISPLAY AT(5,1):"AABAABBABBAAAABBAAABAABBAAAA"
830 DISPLAY AT(6,1):"AABBBBBABBAAAABBAAABAABBAAAA"
840 DISPLAY AT(7,1):"AABAABBABBAAAABBAAABAABBAAAA"
850 DISPLAY AT(8,1):"AABAABBABBAAAABBAAABAABBAAAA"
860 DISPLAY AT(9,1):"AABAABBABBAAAABBAAABAABBAAAA"
870 REM
880 DISPLAY AT(10,1):"AABAABBABBBBBABBAAABAABBAAAAA"
890 CALL HCHAR (som i tidningen)
900 DISPLAY AT(11,1):"BBBEABBAAAAACBEAAEAABBACBBBE"
910 DISPLAY AT(12,1):"BABBABBAAAACFDBEABEABBABB BB"
920 DISPLAY AT(13,1):"BAAAABBAAAABAABBABDEBDABBAAA"
930 DISPLAY AT(14,1):"BAAABBAAAABAABBABADBDABBAAA"
940 DISPLAY AT(15,1):"BBBEABBAAAABBBBBABAABBABBABB"
950 DISPLAY AT(16,1):"AABBABBAAAABAABBABAABBABBABB"
960 DISPLAY AT(17,1):"AABBABBAAAABAABBABAABBABBABB"
970 DISPLAY AT(18,1):"AABBABBAAAABAABBABAABBABBABB"
980 DISPLAY AT(19,1):"BBBFABBBBBABAABBABAABBADBBBF"
```

4. Kolla att allt fungerar, slopa sedan raderna 90 och 655. Nu skall du kunna läsa "ALFASLANG" med jättebokstäver på skärmen. Hoppas Du gillar spelet!

Bokföring på 99:an

av Patric Ericsson

Programmet är skrivet i Extended Basic och vad man mer behöver är minnesexpansion, två diskdrivar och en skrivare med 80 teckens bredd.

Jag visade upp det här programmet på medlemsmötet den 26 september i år och jag har sedan dess utvecklat det lite.

Om någon undrar vad bokföring är så ska jag försöka förklara det.

De flesta företag är tvungna enligt lag att föra bokföring över sin verksamhet. Det går ut på att man skriver ner alla sina ekonomiska händelser på ett visst sätt. Till exempel när man betalar ut löner, telefonräkningen eller om man sålt eller köpt någonting. På speciella konton håller man ordning på sina tillgångar, skulder, inkomster och utgifter. Varje gång man bokför något så gör man det på minst två konton. Som att betala telefonräkningen till exempel. Då tar man ett belopp (krediterar) från sina tillgångar, ex. postgirots konto, och för in det (debiterar) på kontot för telefonavgifter.

Postgiro	Teleavgifter
debet : kredit	debet : kredit
----------	----------
: 890.20	890.20:

Summan i debet ska vara lika stor som summan i kredit. Man kan naturligtvis använda flera konton än två. Man kanske har betalt fler räkningar på samma gång. Det viktiga är att summorna blir lika stora i debet som i kredit. I mitt program kontrolleras hela tiden att det inte blir obalans. När man sedan vill veta hur det har gått så låter man datorn skriva ut en rapport. Då ser man exakt hur man ligger till och hur läget har förändrats på varje konto. Kontoplanen gör man precis som man vill ha den.

I princip så fungerar mitt program som ett riktigt bokföringsprogram men kapaciteten går ju inte att jämföra med en PC:s. Jag tycker ändå att jag kommit ganska nära när det gäller funktion och finesser.

Till saken ...

Det går att lägga upp en kontoplan med upp till 99 olika konton fördelade i fem kontogrupper. Tillgångar, Skulder, Kapital, Intäkter och Kostnader. Det finns två färdiga kontoplaner i programmet som man kan utgå ifrån och sedan göra de ändringar som behövs för att det ska passa ens eget behov. Den ena kontoplanen är anpassad till ett mindre företag i försäljningsbranshen och den andra är anpassad till en privat hushållsekonomi.

När så kontoplanen är klar kan man, om man vill, lägga upp en budget. Det går att lägga upp ett belopp per månad på alla konton. Det beloppet jämförs sedan med det resultat som blir, per månad eller över en längre period.

Man kan när som helst gå in och ändra i kontoplanen eller budgeten om man så önskar. För en företagare gäller visserligen vissa regler om hur man får ändra i kontoplanen men programmet sätter inga begränsningar i det avseendet.

Sedan är det bara att knappa in sina affärshändelser. Programmet numrerar själv verifikaten och vill man veta hur resultatet påverkas under tiden man knappar så finns den möjligheten. Dels så visas resultatet från början av året och dels från början av den månad man befinner sig i. Maximalt går det att registrera 12 transaktioner per verifikat och samma konto får användas flera gånger. I programmet finns även momsautomatik.

Det går att få ut fem olika rapporter plus att man kan få ett saldobesked på valfritt konto månadsvis. I det saldobeskedet får man reda på årets ingående balans, periodens saldo, budgetbelopp, skillnaden mellan budget och verkligt saldo, periodens utgående balans och den aktuella utgående balansen. Saldobeskedet kan man få på skrivaren och/eller skärmen.

De övriga rapporterna är de som följer.

Huvudbok

En lista över transaktionerna sorterade kontovis. Ingående balans, förändring och utgående balans redovisas för varje konto.

Grundbok

Kallas även verifikationslista. Det är en lista över affärshändelserna sorterade efter verifikationsnumren. Omslutningen redovisas för debet och kredit.

Balansrapport

Balansrapporten är en sammanställning över tillgångar, skulder, och eget kapital. Konton kan grupperas och summeras efter eget huvud. Dessutom kan resultatet presenteras var som helst i rapporten. Resultatet börjar på noll och ändrar sig sedan vartefter kontonens ingående balanser, periodsaldon och utgående balanser summeras.

Resultatrapport

Resultatrapporten fungerar på samma sätt som balansrapporten. Skillnaden är den att resultatrapporten är en sammanställning över intäkter och utgifter. De uppgifter som presenteras är periodens saldo, saldot från årets början tillochmed den period som angetts och skillnaden mellan budgetbelopp och verkligt saldo dels i kronor och dels i procent.

Momsrapport

Momsrapporten är en sammanställning över ingående och utgående moms, hur mycket skatt som ska betalas. Raderna i momsrapporten stämmer överens med den blankett som används vid momsinbetalningen så det är bara att föra över siffrorna rad för rad. För en privatperson som inte behöver redovisa momsen kan momsrapporten användas till vad man vill eftersom det går att använda vilka konton som helst i den.

Alla rapporter utom momsrapporten kan tas ut för minst en månad eller mellan vilka månader man vill. Dessa rapporter går dessutom att ställa i en rapportkö då man talar om vilka rapporter man vill ha och så skrivs de sedan ut i en följd.

Momsrapporten, som inte kan ingå i rapportkön, kan man ta ut mellan vilka datum som helst.

Upptäcker man att en affärshändelse har blivit fel bokförd finns det en möjlighet att makulera det verifikatet. Trots att man makulerar ett verifikat så fortsätter det att visa sig i utskrifterna. Det står då att verifikatet är makulerat och det räknas inte med i resultaten.

När ett år är slut så kör man en rutin som logiskt nog kallas för årsslut. Den rutinen avslutar alla konton och lägger upp de utgående balanserna på en ny diskett som används för det nya året. Intäkts och kostnadskonton nollställs. Ett företag gör bokslutet på den gamla disketten och när det är klart går det att föra över de definitiva utgående balanserna till det nya året.

Man kan i programmet ändra på färgerna på skärmen. Välja i vilken disk som dataskivan ska sitta. När det gäller skrivaren så talar man om adress och formfeedkoden. Om skrivaren inte har den funktionen får man i stället tala om antal rader per sida. Dessutom kan man ange styrkoder som skickas till skrivaren varje gång programmet startas upp. Det finns även möjlighet att manuellt skicka koder direkt till skrivaren.

Alla uppgifter om det företag eller vad det nu är man bokför lagras på en enda diskett så man kan med andra ord använda samma program till fler bokföringsobjekt. Men, det går inte att ha fler företag på samma diskett. Som en säkerhet måste man ange ett lösenord innan man börjar. Lösenordet är knutet till företaget, inte till programmet och denna spärr gör att man inte ens kommer in i programmet om lösenordet är fel. När man väl har kommit in i programmet går det dock att ändra på lösenordet.

Med det här programmet får man en perfekt kontroll över sin ekonomi. Är man bara tillräckligt noga med att mata in allt man gör med sina penningar så får man en väldigt klar

bild över situationen. Jag använder naturligtvis programmet själv och jag tror att skulle få det svårt att vänja mig av med det. Det skulle vara väldigt trevligt om det fanns intresse för mitt program bland medlemmarna så hör av er till mig i så fall.

Patric Ericsson
S:a Bangårdsgatan 34 lgnr 17
633 55 Eskilstuna
Tel. 016/11 99 14
även 0150/300 30

Annonser

Säljes:
TI-99/4A kompl grundenhet
Bandspelare med kablar
Modul: Extended BASIC
Modul: Schack
Modul: Adventure
Pirate adventure
Mission impossible
Teach yourself Extended BASIC
släktregister, spel m m
Fullständig bruksanvisning
Pris 1 600 kr för allt.

Åke Fahlén
Ängsgatan 4
777 00 Smedjebacken
☎ 0240/754 81

Säljes:
Oanvända spelmoduler:
Blasto, The Attack, 95:— st
Defender, Picnic Paranoia, Munch Man, Hunt the Wumpus, Chisholm Trail, Car Wars. 100:— st
Shamus, Alpiner, Protector II, Hopper. 110:— st.
Parsec. 120:—
Moon Mine. 125:—
Sneggit, Moon Patrol. 130:— st
Pole position, Munch Mobile, Jungle Hunt. 140:— st.
Slymoids, Congo Bongo. 160:— st
Fathom. 180:—

Speech Synthesizer. 350:—
Microsoft Multiplan (oanvänt) 32 K och diskdrive krävs. 420:—

Mattias Andersson
☎ 0753/336 70

Säljes:
TI-99/4A, expansionsbox, 32 k memory expansion, RS232 card, två DS/DD drivar, kassett-kabel, joysticks.
TI Extended BASIC, Editor/assembler med en extra manual, Forth, Small-C, BASIC Compiler.
QS Database (disk), Companion (ordbehandling), TI-Writer/Funnelweb (disk), Disk Manager 2 (modul), Funnelweb, Diskmanager 1000, Diskmanager 99, Neatlist, Terminalprogram: B01, QZ. Ett tiotal assemblerspel. Sammanlagt över 30 disketter.
Nittinian 4,5 årgångar. Lär dig använda TI-99/4A. The Texas program book.
Pris: 4 000:—

Anders Virving
Torpkällevägen 5
741 00 Knivsta
☎ 018/38 52 01

Pressgrannar

I PB 87-2 skrev vi om den tyska tidningen *TI-Revue*. Sen dess har det skett en del förändringar kring den. Den ges ut av ett kommersiellt förlag som även ger ut andra datatidningar. TI-Revue har nu slagits ihop med en av de andra tidningarna, MSX-Revue. Den hopslagna tidningen heter *Home Computer Aktiv*. Den utkommer regelbundet och delar innehållet mellan 99:an, Atari (8-bitsmodellerna) och MSX-datorer. I nummer 8/87, som kom ut för ett par månader sedan, fanns bland annat en GPL-disassembler, skriven i c99. Dessutom fanns en byggbeskrivning för en A/D-omvandlare för modulporten. Till denna fanns programlistningar i assembler, avsedda för att anropas från Extended BASIC eller Turbo Pascal 99.

Adressen är:

HCA
Postfach 1161
D-8044 Unterschleissheim
Västtyskland

Priset är 80 DM för 12 nummer till utlandet.

Beträffande en del andra tidningar som vi skrivit om under rubriken "Pressgrannar" är väl läget sämre. *Smart Programmer* som vi skrev om i nummer 86-4 kommer ut väldigt oregelbundet. Det senaste numret är daterat december 1986, men gavs ut i våras. Vi kan nog i dagens läge inte rekommendera någon att prenumerera på den.

R/D Computing som vi beskrev i förra numret är också lite osäker. Det senaste numret fick vi för ett par månader sedan.

Ett säkert kort är dock *MICROpendium* som nu är inne på sin fjärde årgång och kommer regelbundet varje månad. Det senaste numret, november 1987, var på 56 sidor. En prenumeration för 12 nummer per flyg kostar 37 dollar.

Disk manager 1000 version 3.8

av Jan Alexandersson

Version 3.8 av DM 1000 har kommit till föreningen från Ralph Romans vid *Ottawa Users Group*. Ordföranden i deras förening har dött i en hjärtattack och Ralph beklagar att våra brev blivit liggande obesvarade. DM 1000 finns nu på två skivor. Du kan beställa programmet och källkoden separat.

Följande buggar är nu borta (jämfört med version 3.5):
— TYPE av DIS/VAR 80-fil med tom första rad fungerar
— Kontrollkoder större än 99 kan sändas mot printer
— Input av skivnamn kan ej krascha med 10—11 tecken och DELETE
— RAM-Disk på CRU >1200 och större kan anropas
— Anrop av ej existerande disk som t ex DSK5 ger omedelbart felmeddelande
— Sektor 0 med Corcomps diskkontrollkort blir riktig även med DS/DD

Följande förbättringar har införts:
— Printerinställningen kan sparas till valfritt disknummer och filnamnet kan väljas valfritt (tidigare krävdes MGR1 och MGR2)
— Speciell version för Myarc Geneve 9640 finns.

Den nya versionen av FUNNELWEB 4.0 har fortfarande DM 1000 version 3.5 även om vissa ändringar har gjorts jämfört med tidigare versioner.

Mer om Horizon RAM-disk

av Jan Alexandersson

Det har kommit en ny FREEWARE till föreningen kallad MENU av John Johnson och Michael Ballmann vid Miami TI Users Group. Jag har den i version 6.0 och version 6.4. Det tråkiga är bara att båda har ett par allvarliga buggar.

MENU version 6.4 fungerar så att när datorn startas eller du gör reset kommer TI-99/4A att köra igenom alla powerup-rutiner på kort i expansionsboxen och då även Horizon RAM-disk. Denna rutin kommer att ladda in en fil som heter MENU från RAM-disken eller om denna inte finns starta modulen i GROM 3, t ex Extended Basic. Detta är en mycket trevlig funktion eftersom du slipper startbilden med färgbalkarna och modulens normala menybild med BASIC osv. Det är endast den sista saken på menyn i GROM 3 som kan startas på detta sätt vilket inte är så trevligt med t ex TI-Writer om du ej vill köra den spanska versionen som råkar ligga sist. Tyvärr tillåter inte MENU att powerup-rutiner i modulen körs igenom. Detta gör att Terminal Emulator II ej fungerar. Samma problem finns med Corcomps diskkontrollkort som också tar kontroll över datorn innan färgbalkarna kommer upp. En annan modul som ej fungerar bra med MENU vid felmeddelande är då Editor/Assembler. Det finns dock möjlighet att komma till den normala TI-menyn om SHIFT-tangenten trycks ned under reset (strömbrytare, reset-knapp, QUIT eller BYE). Observera att BACK från MENU ej ordnar power up av modulen.

MENU har följande CALL-rutiner som kan anropas från BASIC eller Extended Basic:

— CALL DM laddar DM 1000
— CALL DN ändrar RAM-diskens nummer (1—9)
— CALL WO skrivskyddar
— CALL WF tar bort skrivskydd
— CALL AO aktiverar power up-rutinen
— CALL AF stänger av power up-rutinen
— CALL MENU laddar och startar MENU-programmet
— CALL U1—U9 här finns nio användardefinierade CALL som laddar in och kör önskat program

Av de CALL som fanns med originalversionen av Horizon har följande tagits bort: NF, MS, CO, CF, EX. CALL CO och CALL CF är ändå bara till besvär eftersom CALL CF förutsätter att RAM-disken är på CRU >1000. Alla andra adresser gör att två kort samtidigt kommer att vara aktiverade vilket gör att datorn låser sig eftersom den letar i diskkontrollkortet (CRU >1100) efter CALL innan den kommer till CRU >1200.

I själva MENU-programmet kan du lägga upp ett stort antal egna program i menyn utöver de som finns från början:

1 Show directory
2 View a file
3 Run a program
4 Option 1
5 Option 2
6 Option 3
7 Option 4
8 Option 5
9 Option 6
C EXTENDED BASIC (eller modul i GROM 3)

Du kan även välja B (syns ej på menyn) som startar BASIC eller det som finns i GROM 1, t ex GRAM KRACKER.

Nr 3 Run a program fungerar både med Extended Basic-program och Assembler-program i PROGRAM-format men klarar ej långa XB-program som lagras i INT/VAR 254-format. Laddaren tittar själv efter vilken typ av program det är så du behöver själv inte hålla reda på det. Samma sak gäller det du lägger in under Option. Du editerar menyn samtidigt som du har den på skärmen och sparar enkelt det editerade MENU-programmet till RAM-disken.

Genom att trycka på SPACE kan ytterligare två menyer fås upp på skärmen:

1 Option 7
2 Option 8
3 Option 9
4 Option 10
5 Option 11
6 Option 12
7 Option 13
8 Option 14
9 Option 15
C EXTENDED BASIC

Tryck på SPACE ytterligare en gång och följande kommer upp:

1 U1TIL
2 U2TIL
3 U3TIL
4 U4TIL
5 U5TIL
6 U6TIL
7 U7TIL
8 U8TIL
9 U9TIL
C EXTENDED BASIC

Dessa program kan även anropas med CALL U1—U9. Det är möjligt att ändra menyn och även CALL-rutinens namn.

Det finns en allvarlig bug i operativsystemet som följer med MENU. Om den fil som ligger först i ASCII-ordning på skivan lagras tillbaka flera gånger så kommer den att finnas lika många gånger i katalogen som den har blivit sparad. En lösning på detta är att alltid spara en kort fil med namnet "!" som ju alltid kommer först. Om RAM-disken är tom kan inget program sparas från BASIC eller XB utan man måste först kopiera ett program med t ex DM 1000.

Jag har inte heller fått följande att fungera:

— DELETE "DM"
— CALL DN.3(4)

Båda dessa mycket smarta rutiner fungerar inte utan man får ett felmeddelande.

Den rutin som startar Extended Basic-program kommer även att ladda in en interupt-rutin som gör skärmen blå med vit text. Du kan då i kommandomod editera XB-program med dessa skärmfärger. Detta är en smart sak som jag skulle vilja vara utan eftersom många program förutsätter de normala skärmfärgerna svart text på ljusblå skärm. En ny freeware TEXTLOADER kan ej laddas om denna interupt-rutin är aktiverad utan datorn kraschar.

Den rutin som laddar in assemblerprogram efterliknar inte Editor/Assembler-modulens laddare utan följande program laddas med problem:

— BEYOND PARSEC kommer att sätta fart på en massa sprites som går diagonalt och kraschar rymdskepp på löpande band.
— CUBIT får ett band av kontrolltecken över klossarna.
— TI INVADERS saknar copywright-tecken
— MUNCHMAN ger konstiga svarta tecken som hoppar fram och tillbaka på titelskärmen. Svarta tecken kommer även att finnas mitt på spelplanen.
— TENNIS ger blå bakgrund på titelskärmen.
— PB FORTH kan ej laddas men detta beror på den konstiga UTIL4-filen som har en längdangivelse som går utanför det låga minnet >2000 till >3FFF. Finns det någon som kan förklara varför Björn Gustavsson har gjort på detta sätt.
— RAGCTLG från R. A. Green kan ej laddas av version 6.4 medan version 6.0 klarar detta.

Version 7.1 av Menu

I början av december anlände en ny version av programmet. Den verkar vara kraftigt omgjord, varför en del buggar som nämnts ovan kan vara borta. Mer om detta i kommande PB.

Ombyggd Horizon med 392 kb

Elektronik-Service i Västyskland säljer en ombyggd version av Horizon. Den består av ett originalfoliekort från Horizon i USA. På detta har monterats ett litet dotterkort med några nya kretsar vars nummer är bortfilade. I stället för Horizons normala 8 kbytes RAM-chip används 32 kbytes kretsar.

Detta gör att alla kretsar monteras i egna hållare och inga RAM-chip behöver monteras ovanpå varandra. På kortet finns 12 st 32 kbytes kretsar 58256LP-12L samt 1 st 8 kbytes 6264LP-15. Detta ger totalt 392 kbytes.

Kortet levereras med 5 flexskivor och ett något modifierat operativsystem jämfört med originalversion. RAM-disken kan användas för 1440 sektorer motsvarande 360 kbytes DS/DD. Det finns hela 26 kbytes över som ej används. Det finns ingen manual som täcker de tyska modifieringarna eller någon tysk källkod. Det som finns på flexskivorna är förutom det nödvändiga operativsystemet endast originalkällkoden och originalmanualen (på disk).

Kortet har fungerat mycket bra när det gäller att lagra filer. Även CALL-kommandona fungerar bra medan DELETE "XBCALL" ej fungerar. Eftersom 392 kb RAM-disken måste anropa sektorerna på ett annat sätt än originalversionen kan inte John Johnsons MENU-program användas.

Grand RAM från Databiotics

Det har funnits annonser i Micropendium om en helt ny RAM-disk från DataBiotics kallad GRAND RAM. Denna finns i storlekarna 64, 128, 256 och 512 kbytes RAM med batteri back-up. Den har inget samband med Horizon även om systemlösningen verkar vara snarlik. Priset är USD 230 för 512 kbytes. Kortet har 16 st 32 kbytes statiska RAM-chip. Kortet har en egen TMS 9901 och I/O-kontakt i bakkanten. En realtidsklocka kan pluggas in på kortet och kostar USD 20 extra.

Det finns möjlighet till print spooler och man kan även dela upp RAM-disken i fyra olika disk-nummer. Det är något osäkert om serieleveranserna har börjat.

Nya priser på RAM-disk

Horizon har sänkt priserna på sin RAM-disk:

192 kb RAM-disk inkl manual och skivor	USD 195
Foliekort inkl manual och skivor	USD 45
Endast foliekort	USD 38

Observera att priser på foliekort även gäller vid köp av enstaka kort. Priserna inkluderar frakt inom USA. Adressen är:

Horizon Computer Ltd
P. O. Box 554
Walbridge, OH 43465
USA

PR-BASE

Recension av Anders Josefsson, Tranås

PR-BASE är ett databasprogram skrivet av William Warren.

"Ni har just fått tag i ett diskbaserat databas-program med snabb behandling." Så står det i inledningen till manualen till PR-BASE. Ett program där du själv bestämmer hur Din databas skall se ut. Från första siffran! När menyn vid laddning kommer fram, kan Du välja:

1. Jobba med din databas
2. Skapa en ny bas.

Väljer Du då att skapa en ny bas, tar Du fram manualen (täcker en hel enkelsidig disk), slår upp stycket ang skapande. Där får Du se exempel på hur en "skärm" kan se ut. Det finns vissa likheter med Forth-skärmar. Varje register består alltså av en hel skärm. Hur skärmen skall se ut bestämmer Du själv helt från scratch. Du bestämmer också hur utskrifter på printer skall se ut. Där finns 5 olika utskriftsval, t ex namn på en och adresser på en annan. Tel nr osv. När Ditt skapande är klart, sparas allt detta på de 10 första sektorerna på den diskett som Du ska använda. Resten av disketten används till Dina data. En enkelsidig diskett sparar 350 och en dubbelsidig 710 sådana data. De laddas alltså in på Din skapade skärm.

Manualen är mycket utförlig, tycker jag. Tyvärr var min Doc-disk inte komplett, så sista delen av disken innehöll bara skräp. Men jag fick med det viktigaste.

Så är Du alltså klar att mata in i Din bas. Trycker du på "H", får Du fram alla kommandon på skärmen. De ser ut så här:

A	Add Record	Mata in nya data på tom ruta
B	Boot Data Base	Ladda in en annan färdig bas
C	Copy Data Disk	Kopiera Din skärm för ny bas
D	Delete Record	Ta bort ett registerblad
E	Edit Record	Ändra i data
F	Find String	Sökning efter bestämda data
G	Global Search	Söker hela disken
H	Display Commands	Kommandomeny på skärmen
I	Build new Index	Bestäm söknings-index
L	Print labels	Skriv ut labels
N	Go to screen #	Gå till nästa skärm nr
O	Drive Output	Data till DSKn
P	Print Screen	Skriv ut skärm
Q	Quit PRBASE	Avsluta PRBASE
R	Print Reports	Utskrift rapport
S	Sort Index	Sortera index
U	Use Index to Find	Sök efter index
V	View Index	Visa index

FCTN X	Gå till nästa skärm
FCTN E	Gå till föregående
FCTN D	Nästa alfabetiska skärm
FCTN S	Föregående alfabetiska skärm
CTRL X	Snabb rullning framåt
CTRL E	Snabb rullning bakåt
CTRL D	Första alfabetiska skärm

De flesta kommandon förklarar sig själva. Men något kan kanske förklaras, t ex Index. Där bestämmer Du i vilket fält som Du vill ska sökas efter. Visa Index, så rullar dom fram på sista raden. Enligt manualen kan vilket "blad" som helst tas fram på mindre än 1 sekund. Mycket bra!

Det finns mycket mer att förklara, men då får jag skriva om hela manualen. Det finns en del saker som är svåra att komma underfund med, t ex utskrifter som kräver en del tankeverksamhet. Annars skrivs det ut helt galet.

Sortering av registret fungerar inte i själva programmet, men hav tröst! På systemdisken finns ett XBasic-program som sköter den biten. Ett krux är att 2 st drivar krävs.

Slutkommentaren blir alltså klart *bra*. Här får man en chans att bringa ordning på sina adresser, kalender, ja bara fantasin finns kan detta program användas till i stort sett vad som helst.

Redaktionens kommentar: Köper man programmet från föreningens Programbank får man en utprintad manual som beskriver programmet på köpet.

Annons

Säljes:

RS232-interface TI original Pris 500:—

Göran Åhsberg
Polhemsskolan
Box 6067
800 06 Gävle
☎ 026/717 83 hem
026/17 89 62 arb

Säljes:

Texas TI-99/4A, utbyggnadsbox med minneskort 32K, Pertek diskdrive, diskdrivekort, RS232-kort, Personal Record Keeping, Mailing List och diverse andra program. Skrivare finns.
Pris ca 4 000:—

Anders Mattsson
Timotejvägen 14
191 77 Sollentuna
☎ hem 08/754 83 55
arb 08/757 20 23

Program från Amnion Helpline

Åke Olsson i Tyresö har fått tag på en del program från USA och beskriver här innehållet.

Jag har gått igenom varje program på alla skivorna och tittat efter vad de innehåller. Jag har ej haft tillräckligt med tid att köra varje program, så därför vet jag ej om alla fungerar. Jag har tittat igenom alla elektronikprogram för att se vad jag själv kan använda av dom. Det finns en hel del bra saker inom detta gebit, har t o m hittat något som jag har nytta av i jobbet. Planerar att lägga över alla sådana program på särskilda diskar så att de är samlade. Jag tror att jag har listat alla elektronikprogram också. För den kommande sammanställningen av samtliga program så framgår titeln på innehållet, men det finns i regel inga instruktioner med till programmen.

I listan nedan kallas varje skiva för "Volume" och de enskilda programmen heter F001—F134. Jag har försökt ange ämnesområdet på varje program under "Subject". Jag föredrar att behålla de engelska namnen. Det är nog upplagt för missförstånd annars. Som Du ser finns det mycket statistikprogram. De begriper jag mig inte på!

Sammanställningen gjord av Åke Olsson, Tyresö

om inte annat anges är programmen skrivna i Extended BASIC

VOL 1	SIZE	SUBJECT	PROGRAM
F001	4	MATH.	SIGMA X, SIGMA Y.
F002	13	ELEC.	256 FFT PROGRAM.
F003	16	ELEC.	POLAR PLOT MODELING PROGRAM N6APX (ANTENNAS).
F004	3	MATH.?	DATA FORECASTING DIVERGENCE
F005	6	PHYS.	TEMPERATURE CONVERSION (F,C,R,K,Rankine)
F006	4	ELEC.	BLACKBODY TEMPERATURE (Kelvin)
F007	4	ELEC.	TOTAL RESISTANCE IN SERIES AND PARALLEL RES. CIRC.
F008	14	STATH.	CALCULATING BASIC STAT.SAMPLE "SIZE,MEAN,ST.DEV., POPULATION"/PLOT HISTOGRAM
F009	22	ELEC.	ELECTRONICS:1.R IN PARALLELL, 2.C IN SERIES, 3.RESONANT FREQ. 4.FREQ/WAVEL.,5.OHM'S LAW, 6.ANTENNA DESIGN (FEET).
F010	20	ELEC.	CLOSED LOOP GAIN/MIN DES.FREQ FOR A CIRCUIT USING "INTEGRATED CIRCUIT 301A"
F011	8	GRAPH.	3D FUNCTION PLOT-GRAPHICS DEMONSTRATION
F012	6	PHYS.	CALCULATES BORE DIAM FOR A PRESS. DROP IN A PIPE FLOW.
F013	6	MATH.	CROSS CORRELATION BETWEEN TWO CURVES THROUGH USE OF LEAST-SQUARE METHODE.
F014	11	MATH.	FOURIER ANALYSIS OF CYCLING CURVES."Harmonic curve fitting".
F015	25	ELEC.	CROSSOVER NETWORK FOR 2-WAY SPEAKER SYSTEMS.Calculate H and C value (6 dBor 12dB/octave.)
F016	11	ELEC.	ALTERNATOR AND TRANSFORMER REGULATION (3-PHASE)
F017	14	MATH.?	MATRIX MANIPULATION
F018	23	MATH.	2-D CURVE OR 3-D CURVE
F019	6	ELEC.	CONVERSION FOR VOLTAGE:PEAK,EFFECTIVE,AVERAGE
F021	9	ELEC.	CALCULATION PROGRAM FOR:-POWBR SUPPLY FILTER -ZENER POWER SUPPLY
F022	10	ELEC.	REACTANCE CHART:C ENTER; Xc ENTER; F ENTER; H ENTER
F023	13	ELEC.	ELECTRONICS: C TIME CONSTANT;I PWR DISSIPATION; INDUCTANCE; H TIME CONSTANT; JOULE'S LAW; OHM'S LAW PARALLEL RES; SERIE CAP.; V PWR DISSIPATION
F024	10	PHYS.	COMPOUND REL. EMPIRICAL FORMULA
F025	14	PHYS.?	STATISTICAL ANALYSIS FOR RANDOM BLOCK DESIGN
F026	7	PHYS.?	CALCULATE MOVING AVE. FROM TIME-SERIES DATA
F027	34	MATH.	TRIGONOMETRIC FUNCTIONS
F028	5	MATH.	INTERPOLATION NEWTON'S FWD DIFF FORMULA
F029	5	ELEC.	ANTENNA POSITIONING FOR GEO'S SATELLITE
F030	18	MATH.	REGRESSION ANALYSIS
LOAD	5		LOAD FOR ANY PROGRAM

VOL 2	SIZE	SUBJECT	PROGRAM
F031	16	ELEC.	IC 555 TIMER PROGRAM
F032	21	ELEC.	CONVERSION "DECIBEL"-VOLTAGE RATIO;POWER RATIO
F033	26	MATH.	GEOMETRIC FIGURES:AREA-VOLUMES
F034	15	STATH.	STATISTICS (POPULATION)
F035	24	PHYSI.	OIL WELL CASING DESIGN
F036	14	ELEC.	ELECTRONICS 1: RESISTIVE ATTENUTATOR DESIGN
F037	3	MATH.	PRINT-PROGRAM FACTOR-INTEGER
F039	3	PHYSI.	HYPNOSIS PATTERN
F040	7	PHYSI.	WIND CHILL (Skin temp.at wind)
F041	14	ELEC.	CLOSED LOOP SYSTEM ANALYSIS (Basic).

F042	9	MATH.	PRIME NUMBER
F043	17	STATH.	STATISTICAL PARAMETERS; MENU:1.Mean st dev.2.Moment skewness,kurtusis 3.Chi-square evaluation 4.Permutations and combinations 5.No-rep.probabili. 6.Linear interpolation.
F044	16	PHYSI.	COMET POSITION-DAY
F045	17	PHYSI.	PLANET-WATCH
F046	3	MATH.	FIBONACCI SEQ.?
F047	13	PHYSI.	PIPING SYSTEMS ANALYSIS
F048	7	PHYSI.	PERSONAL TELESCOPE SPECIFICATION
F049	10	MATH.	CONVERSION- UNIVERSAL TIME TO LOCAL TIME (USA)
F050	9	MATH.	POLYNOMIAL CURVE
F051	28	PHYSI.	CONVERSION: TEMPERATURE,WEIGHT,LENGTH,.....
F052	10	MATH.	MATRIX DETERMINANT EVALUATOR
F053	16	MATH.	VECTOR ADD/SUB
F054	5	ELEC.	PARABOLIC REFLECTOR PROGRAM
F055	7	PHYSI.	CALCULATION FOR AEROPLANE?
F056	14	ELEC.	MUF- MAX USABLE FREQUNCIES
F057	11	PHYSI.	MOON-WATCH
F058	14	ELEC.	LOG-PROGRAM (QNI)
LOAD	5		LOAD FOR ANY PROGRAM

VOL 3	SIZE	SUBJECT	PROGRAM
F059	26	MATH.	REGRESSION ANALYSIS:1.Linear regression 2.Multiple reg. 3.NTH-order reg. 4.Geometric reg.5.Expot.reg
F060	19	STATH.	STATISTICS POPULATION
F061	18	BUILD.	BUILDING TECHNICS?
F062	13	MATH.	HIGH RESOLUTION PLOTTER (X-Y COORD)
F063	22	ELEC.	OHM'S LAW DC(AC-RMS)
F0641	6	ELEC.	POWER SUPPLY DESIGN (FULL WAVE RECTIFIER)
F0642	21	ELEC.	" " " (ZENER DIOD REGULATED)
F0643	24	ELEC.	" " " (FULL WAVE RECTIFIER)
F065	5	ELEC.	IC 555 ASTABLE
F066	7	MATH.	LANGRANGIAN INTERPOLATION
F067	55	I/V254	REACTOR KINETICS
F068	26	ELEC.	ANT #1:(10 DIFFERENT ANTENNA TYPES)
F069	8	ELEC.	DX CONTEST DUPEX FINDER (KV4Y)
F070	5	MATH.	MATRIX INVERSION
F071	9	PHYSI.	SOLAR INTENSITY; SUNRISE,SUNSET.
F072	9	PHYSI.	CONVERTS SURVEY TIE-LINES TO VECTOR AND PRINT-PROG.
F073	18	ELEC.	WAVE CALCULATIONS (HF)
F074	8	MATH.	GRAPH MAKER III
F075	26	MATH.	DISCUSSION OF A CURVE
LOAD	5		LOAD FOR ANY PROGRAM

VOL 4	SIZE	SUBJECT	PROGRAM
30	2	I/F10	
6	2	I/F10	
F076	24	PHYSI.	PNEUMATIC CONTROL
F077	7	ELEC.	ANTENNA POSITION FOR GEO'S SATELLITES
F078	12	PHYSI.	CONVERSION PROGRAM: ANGLES TEMP
F079*	20	PHYSI.	ASTRONOMY ONE
F0791	2		I/F192
F0792	30		I/F250
F080	21	PHYSI.	OBLIQUELY CROSSED CYLINDER LENSES
F081	27	MATH.	LIN. PROGRAMM. PROBLEMS:PHILLIPS,RAVINDRAN,SOLBERG
F082	30	PHYSI.	CHEMISTRY
F083	9	DATA	COLOR BAR PLOTTING
F084	15	DATA	PLOTTING ON SCREEN
F085	38	MATH.	GRAPHS OF MATH.FUNCTIONS (Basic) (Memory full)
F086	38	MATH.	" " " " " " "
F087	32	MATH.	EQUATION PLOTTER (Basic) Program stops.
F088*	21	STATH.	HISTOGRAM ACCUMULATOR
F089	18	DATA	DATA IN; CREATE DATA FILE.
LOAD	5		LOAD FOR ANY PROGRAM

VOL 5	SIZE	SUBJECT	PROGRAM
F090	44	PHYSI.	OVERLAND FLOW
F091	34	PHYSI.	PLANET POSITION/SUNRISE/SET TIMES
F092	35	MATH.	X-Y-DATA PLOT
F093	26	MEDIC.	CALCULATIONS FOR THE MEDICAL LAB
F094	11	MATH.	TRUTH TABLE GENERATOR
F095	10	ELEC.	ELECTRONICS: AIR CORE COILS
F096	22	ELEC.	ELECTRONICS: SHORTWAVE STATION STORE PROGRAM (100)
F097	14	BUILD.	BUILDING: SCHEMATIC CONCRETE BEAM CROSS SECTION
F098	30	BUILD.	BUILDING: SIMPLY SUPPORTED BEAM MOMENTS AND REACTIONS
F099	32	MATH.	MATH: INTEGRATION "THE DEFINITE INTEGRAL"
F100	17	ELEC.	ELECTRONICS: ELECTRICAL FAULT CURRENT CALCULATIONS
F101	26	ELEC.	ELECTRONICS: VOLTAGE DROP CALCULATIONS
F102	22	STATH.	STATISTICS: STATSPAK STATISTICAL UTILITIES
F103	21	BUILD.	CENTER OF GRAVITY MODEL
LOAD	5		LOAD FOR ANY PROGRAM

VOL 6	SIZE	SUBJECT	PROGRAM
F104	23	ELEC.	ELECTRONICS: DIGITAL LOGIC SIMULATOR
F105	10	MATH.	ROOTS OF POLYNOMIALS
F106	20	PHYSI.	FLUID POWER FORMULAE
F107	22	BUILD.	BUILDING: CANTILEVER
F108	9	MATH.	XTRKTR: SIMPLIFIED POLYNOMIAL
F109	11	DATA	PLOTTER
F110	14	DATA	LINEAR PROGRAMMING
F111	11	STATH.	OPTIMUM ASSIGNMENT: HUNGARIAN ALGORITHM
F112	26	ELEC.	ELECTRONICS: POLAR ORBITING SATELLITE TRACKING
F113	39	ELEC.	ELECTRONICS: ACTIVE FILTERS BUTTERWORTHS 3RD ORDER
F114	7	ELEC.	ELECTRONICS: INTERMODULATION PROGRAM
F115	21	CHEM.	CHEMESTRY: TWO SAMPLE TESTS BASED ON BIONOMIAL
F116	22	STATH.	WILCOXON SIGNED RANKS TESTS
F117	29	STATH.	GRUBBS TEST OF OUTLYING OBSERVATIONS
F118	42	MEDIC.	EKG STUDY (QUIZ PROGRAM)
F119	13	STATH.	THE MANN-WHITNEY TESTS
F120	12	STATH.	KRUSKAL-WALLIS TEST
F121	15	STATH.	LILLIEFORS-TEST FOR UNSPEC.EXPOTENTIAL DISTRIBUTION
LOAD	5		LOAD FOR ANY PROGRAM

VOL 7	SIZE	SUBJECT	PROGRAM
F122	22	ELEC.	ELECTRONICS: MUF-MAX USABLE FREQUENCY,DISTANCE, BEARING.
F123	23	STATH.	WALD/WOLFOWITZ RUNS TEST
F124*	31	ELEC.	ELECTRONICS: SYMBOLS (Basic)
F124D	3		I/V192
F125	11	STATH.	THE RAVIV TEST
F126*	4	STATH.	STATHISTICS PACKAGE
F126A	7	STATH.	1. 2 Way Anova
F126B	10	STATH.	2. 3 Way Anova
F126C	7	STATH.	3. Randomized Blocks Anova
F126D	10	STATH.	4. Randomized TIK Fact'l
F126E	9	STATH.	5. Split Plot Fact'l SPF-P.Q
F126F	12	STATH.	6. Split Plot Anova
F126G	15	STATH.	7. Split Plot Fact'l SPF-P.QR
F127	34	PHYSI.	TABLE OF VISIBLE SKY OBJECTS
F128	34	MATH.	MEASUREMENT FOR THE CLASSROOM TEACHER
F130	30	DATA	CALC'U'AID MILD'S CALCULATOR
F131	39	DATA	MITI CRYPT-O-GRRR: Message into Cyphered text with/ without Codeword.
LOAD	5		LOAD FOR ANY PROGRAM

VOL 8	SIZE	SUBJECT	PROGRAM
F129*	37	DATA	MITI DATA'LYZER 1983,1985 MILD TSU.
F129DOC1	65		D/V80
F129DOC2	141		D/V80
F132	36	DATA	BAR GRAPHS ON THE EPSON RX80 20/7-86
F133	45	DATA	TEXT AND/OR " " 8/10-85
F134	25	DATA	CONSOLE BASIC PROGRAM BY HUIXIA JIANG (CONSOLE,MINIMEM,CS1)

Hunt the plusses

Ett spel av Jonas Fjellstedt. Det går ut på att med joystickens hjälp jaga så många plustecken som möjligt, utan att kollidera med de hinder som finns på skärmen.

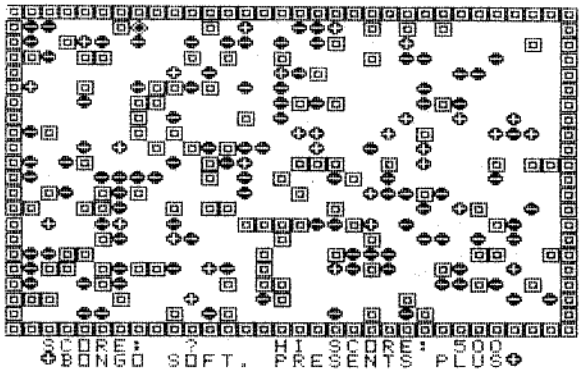
Detta program har listats med en NEC P6-printer med amerikanskt IBM-teckenset men med utbytta ÅÄÖ.

```
Å = ] = FCTN T (tecken nr 93)
Ä = [ = FCTN R (tecken nr 91)
Ö = \ = FCTN Z (tecken nr 92)
```

På rad 360—410 förekommer C med en krok. Detta skall vara tecken 128 som knappas in med CTRL , (tryck alltså på CTRL och kommatecken samtidigt). Det lämpliga är att Du först knappar in rad 1630—1720 och rad 300 och gör RUN på det. Omdefinierade tecken större än 127 finns sedan kvar i minnet under editeringen.

```
100 ! ********************
110 ! *                  *
120 ! * HUNT THE PLUSSES *
130 ! *        By        *
140 ! * JONAS FJELLSTEDT *
150 ! *                  *
160 ! ********************
170 ! TI EXTENDED BASIC
180 ! VERSION 2.E
190 ! 860623-861102
200 !
210 ! TI-99/4A ÖNSKAR
220 ! LYCKA TILL!!!
230 !
240 RANDOMIZE
250 ! *** HI-SCORE SET ***
260 FOR X=1 TO 4 :: HS(X)=100+100*X :: NEXT X
270 ! *** TITLE SCREEN ***
280 CALL CLEAR :: CALL SCREEN(8):: FOR SET=1
    TO 8 :: CALL COLOR(SET,16,12):: NEXT SET
290 CALL COLOR(9,14,12,10,3,12,11,9,12,12,5,1
    2,13,16,8)
300 FOR X=1 TO 54 :: READ A,C$ :: CALL CHAR(A
    ,C$):: NEXT X
310 CALL HCHAR(1,1,104,32):: CALL HCHAR(24,1,
    104,32):: CALL VCHAR(2,1,104,22):: CALL V
    CHAR(2,32,104,22):: CALL HCHAR(2,2,112,30
    )
320 CALL HCHAR(23,2,112,30):: CALL VCHAR(2,2,
    112,22):: CALL VCHAR(2,31,112,22):: CALL
    HCHAR(3,3,120,28):: CALL HCHAR(22,3,120,2
    8)
330 CALL VCHAR(3,3,120,20):: CALL VCHAR(3,30,
    120,20)
340 DISPLAY AT(6,6)SIZE(18):"!""#$%&'()* +,Z.
    /-0" :: DISPLAY AT(7,6)SIZE(18):"12345678
    9:;<=>?@ÄÅB"
350 DISPLAY AT(8,6)SIZE(18):"CDFGDFIJKM OQVWJ
    XY" :: DISPLAY AT(10,11)SIZE(11):"PRESENT
    S"
360 DISPLAY AT(12,7)SIZE(4):"ÇÇÇÇ"
370 DISPLAY AT(13,7)SIZE(16):"Ç  Ç Ç   Ç Ç ÇÇ
    Ç"
380 DISPLAY AT(14,7)SIZE(14):"Ç  Ç Ç   Ç Ç Ç"
390 DISPLAY AT(15,7)SIZE(16):"ÇÇÇÇ Ç   Ç Ç ÇÇ
    Ç"
400 DISPLAY AT(16,7)SIZE(16):"Ç    Ç   Ç Ç
    Ç"
410 DISPLAY AT(17,7)SIZE(16):"Ç    ÇÇÇ ÇÇÇ ÇÇ
    Ç"
420 GOSUB 1430 ! TO TITLE SOUND
430 DISPLAY AT(23,9)SIZE(11):"PRESS ENTER"
440 ! *** KSCAN ***
450 CALL KEY(1,K,S):: IF S=0 THEN 460 ELSE 49
    0
460 CALL KEY(0,K,S):: IF X=300 THEN 490
470 IF S=0 THEN X=X+1 :: GOTO 460
480 ! *** MENY SCREEN ***
490 CALL CLEAR :: CALL CHARSET :: CALL SCREEN
    (9):: FOR SET=1 TO 8 :: CALL COLOR(SET,2,
    12):: NEXT SET
500 DISPLAY AT(5,6):"CHOOSE LEVEL (1-4)"
510 DISPLAY AT(8,6):"1. EASY"
520 DISPLAY AT(10,6):"2. MEDIUM"
530 DISPLAY AT(12,6):"3. HARD"
540 DISPLAY AT(14,6):"4. INSANE"
550 DISPLAY AT(16,6):"E. END PROGRAM"
560 CALL KEY(0,K,S):: IF S=0 THEN 560
570 IF K<>ASC("1")THEN 590
580 O=10 :: LEVEL=1 :: SCREEN=16 :: GOTO 710
590 IF K<>ASC("2")THEN 610
600 O=30 :: LEVEL=2 :: SCREEN=8 :: GOTO 710
610 IF K<>ASC("3")THEN 630
620 O=60 :: LEVEL=3 :: SCREEN=10 :: GOTO 710
630 IF K<>ASC("4")THEN 650
640 O=120 :: LEVEL=4 :: SCREEN=14 :: GOTO 710
650 IF K<>ASC("E")OR K<>ASC("e")THEN CALL CLE
    AR :: STOP
660 DISPLAY AT(12,9)ERASE ALL:"GOOD BYE"
670 FOR DELAY=1 TO 500 :: NEXT DELAY
680 CALL CLEAR :: STOP
690 GOTO 490
700 ! *** PLAY SCREEN ***
710 CALL SCREEN(SCREEN):: CALL COLOR(9,14,12,
    10,3,12,11,9,12,12,5,12)
720 CALL CLEAR
730 PO=0
740 CALL HCHAR(24,1,32,32)
750 DISPLAY AT(23,1)SIZE(12):"SCORE:  ?   "
760 DISPLAY AT(23,14)SIZE(15):"HI SCORE:";HS(
    LEVEL)
770 FOR SET=1 TO 8 :: CALL COLOR(SET,2,12)::
    NEXT SET
780 CALL HCHAR(1,1,104,32):: CALL HCHAR(22,1,
    104,32):: CALL VCHAR(2,1,104,21):: CALL V
    CHAR(2,32,104,21)
790 GGR=0
800 ! *** OBSTACLES & - ***
810 SCR=SCR+10
820 FOR I=1 TO O+SCR
830 CALL HCHAR(INT(RND*20)+2,INT(RND*30)+2,10
    4):: CALL SOUND(1,500,1):: CALL HCHAR(INT
    (RND*20)+2,INT(RND*30)+2,120):: CALL SOUN
    D(1,1000,1)
840 NEXT I
850 ! *** PLUSSES ***
860 FOR U=1 TO 30
870 Y=INT(RND*20)+2 :: X=INT(RND*29)+2 :: CAL
    L GCHAR(Y,X,I):: IF I=112 THEN 870
880 CALL GCHAR(Y+1,X,Q1):: CALL GCHAR(Y,X+1,Q
    2):: IF (Q1=104)*(Q2=104)THEN 870
890 CALL HCHAR(Y,X,112):: CALL SOUND(1,600,1)
900 NEXT U
910 RA=INT(RND*19)+2 :: KO=INT(RND*30)+2 :: C
    ALL GCHAR(RA,KO,STPOS):: IF STPOS=112 THE
    N 910
920 DISPLAY AT(24,1):"pBONGO SOFT. PRESENTS P
    LUSp " :: CALL HCHAR(RA,KO,96):: CALL SOU
    ND(10,900,1)
930 CALL KEY(0,K,S):: CALL JOYST(1,X1,Y1)
940 IF K=69 OR K=83 OR K=68 OR K=88 OR K=101
    OR K=115 OR K=100 OR K=120 THEN JOY=0 ::
    GOTO 1000
950 IF X1<>0 AND Y1<>0 THEN 32767
960 IF X1<>0 OR Y1<>0 THEN JOY=1 :: GOTO 1050
970 GOTO 930
980 ! *** KEY CONTROL ***
990 CALL KEY(0,K,S):: IF GGR=30 THEN 1370 ELS
    E IF S=0 THEN 1090
1000 IF K=69 OR K=101 THEN A=-1 :: B=0 :: GOTO
     1090
1010 IF K=88 OR K=120 THEN A=1 :: B=0 :: GOTO
     1090
1020 IF K=83 OR K=115 THEN A=0 :: B=-1 :: GOTO
     1090
1030 IF K=68 OR K=100 THEN A=0 :: B=1 :: GOTO
     1090
1040 ! *** JOYST CONTROL ***
1050 CALL JOYST(1,X1,Y1):: IF GGR=30 THEN 1370
1060 IF X1=0 AND Y1=0 THEN 1090
1070 IF X1<>0 AND Y1<>0 THEN 1090
1080 A=-Y1/4 :: B=X1/4
1090 RA=RA+A :: KO=KO+B :: CALL GCHAR(RA,KO,X)
     :: CALL HCHAR(RA,KO,96)
1100 IF X=104 OR X=96 THEN 1150
1110 IF X<>112 THEN 1130
1120 CALL SOUND(1,200,1):: CALL SOUND(1,400,1)
     :: CALL SOUND(1,600,1):: PO=PO+(LEVEL*25)
     :: GGR=GGR+1 :: GOTO 1140
1130 IF X<>120 THEN 1140 :: CALL SOUND(1,200,1
     ):: CALL SOUND(1,150,1):: CALL SOUND(1,11
     0,1):: PO=PO-LEVEL*50
1140 CALL SOUND(1,1000,1):: IF JOY=0 THEN 990
     ELSE 1050
1150 ! WHOOPS YOU CRASHED
1160 FOR X=0 TO 30 STEP 2 :: CALL SOUND(75,-5,
```

```
     INT(X)):: NEXT X :: CALL HCHAR(23,10,32,5
     )):: DISPLAY AT(23,7)SIZE(5):PO
 1170 SCR=0
 1180 M$(1)="xx YOU HAVE TO LOOK OUT!! pp" :: M
     $(2)="pxp WHOOPS YOU CRASHED!! pxp" :: M$
     (3)="px BETTER LUCK NEXT TIME! xp"
 1190 M$(4)="xpxp I'M REALLY SORRY!! pxpx" :: M
     $(5)=" YOU HAVE TO WORK HARDER!!! " :: M$
     (6)="pxpx JUST AN ACCIDENT!! xpxp"
 1200 M$(7)="pxpx  DON'T GIVE UP!!!  xpxp" :: M
     $(8)="pp DON'T MAKE A WRY FACE! xx"
 1210 DISPLAY AT(24,1):M$(INT(RND*8)+1)
 1220 FOR D=1 TO 300 :: NEXT D
 1230 ! *** HI-SCORE ? ***
 1240 IF PO>HS(LEVEL)THEN 1250 ELSE 1280
 1250 DISPLAY AT(24,1):"p VERY GOOD!!!  HIGH SC
     ORE p"
 1260 HS(LEVEL)=PO
 1270 GOSUB 1550 ! TO HI-SCORE SOUND
 1280 FOR D=1 TO 30 :: NEXT D
 1290 DISPLAY AT(23,14):"HI SCORE:";HS(LEVEL)
 1300 DISPLAY AT(24,1):"WANT TO PLAY THE SAME (
     Y/N)?"
 1310 CALL KEY(0,K,S)
 1320 IF K=ASC("Y")OR K=ASC("y")THEN 1340
 1330 IF K=ASC("N")OR K=ASC("n")THEN 1360 ELSE
     1310
 1340 GOSUB 1610 ! TO CLS
 1350 GOTO 730 ! BACK TO START
 1360 GOTO 490 ! BACK TO MENY
 1370 REM ** YOU MADE IT **
 1380 DISPLAY AT(24,1):"pxpxp  YOU MADE IT!!!
     pxpxp"
 1390 DISPLAY AT(23,7)SIZE(5):PO
 1400 FOR X=1 TO 30 :: CALL SOUND(100,INT(RND*1
     10)+300,1):: NEXT X
 1410 GOSUB 1610 ! TO CLS
 1420 GOTO 790 ! TO NEW SCREEN
 1430 ! *** TITLE SOUND ***
 1440 TON=131
 1450 FOR A=12 TO 1 STEP -0.5 :: CALL SOUND(45,
     TON,A):: TON=TON+22 :: NEXT A
 1460 CALL SOUND(600,523,0,659,0,784,0)
 1470 FOR X=1 TO 2
 1480 CALL SOUND(300,523,0,880,0,1047,0)
 1490 CALL SOUND(300,523,0,698,0,880,0)
 1500 CALL SOUND(600,523,0,659,0,784,0)
 1510 NEXT X
 1520 CALL SOUND(999,262,X,196,X,165,X)
 1530 RETURN
 1540 ! ** HI-SCORE SOUND **
 1550 TON=262
 1560 FOR A=12 TO 1 STEP -0.5 :: CALL SOUND(50,
     TON,A):: TON=TON+33 :: NEXT A
 1570 CALL SOUND(900,1047,0,1319,0,1568,0)
 1580 CALL SOUND(1200,2098,0,1568,0,659,0)
 1590 CALL SOUND(100,32000,1)
 1600 RETURN
 1610 ! *** CLEAR SCREEN ***
 1620 FOR X=2 TO 21 :: CALL HCHAR(X,2,32,30)::
     NEXT X :: RETURN
 1630 ! *** CHARACTERS ***
 1640 DATA 33,00000000FFFFC3C3,34,000000000F0F0
     C0C,35,00000000FCFC0C0C,36,0000000038383C
     34
 1650 DATA 37,00000000CFCFCCCC,38,00000000FCFC3
     C3C,39,00000000FFFFC0C,40,00000000C0C0C0C
 1660 DATA 41,0000000000060909,42,00003C42424242
     24,43,0000000003030303,44,00000000FEFE1E1
     E,45,000000003F3F3333,46,00000000F3F33333
 1670 DATA 47,00000000FFFF,48,00000000F0F0303,4
     9,C3C3C3FFFFC3C3C3,50,0C0C0CCFCFCFCFCF,51
     ,0C0C0C0C0C0C0C0C,52,363233F1F1F0F0F
 1680 DATA 53,CCCCCCCFCFCFCFCF,54,0000003C3C3C3
     C3C,55,C0C0C0F0F0F0F0F,56,C0C0C0C0C0C0C0C
     ,57,9F807C020101,58,C3000000000008181,59,F
     E0139498986
 1690 DATA 60,0303030303,61,000000FFFF0F0F0F,62
     ,3030303C3C3C3C3C,63,3333333333333333,64,
     000000FFFFC0C0C,91,0303030303030303
 1700 DATA 66,000000C0C0C0C0C0,67,C3C3FFFF,68,C
     FCFCFCF,70,0C0CFCFC,71,F0F0F0F,73,F0F0FFF
     F,74,C0C0C0C
 1710 DATA 75,0F10100F,77,811929C9090906,79,000
     00303,81,0F0FFFFF,86,3C3C3F3F,87,3333F3F3
     ,88,03030303,89,C0CECECE
 1720 DATA 90,000000003F3F3030,96,18245ABDBD5A2
     418,104,FF81BDA5A5BD81FF,112,386CEE82EE6C
     3800,120,387CFE82FE7C3800,128,FF81BDA5A5B
     D81FF
```



Screendumparna visar titelbilden på programmet och skärmen under den svåraste versionen (nummer 4).

Det kom ett brev från Australien

20 oktober 1987

Bäste Jan!

En vinter av programmering är nu över (inte för att en svensk skulle känna igen en vinter i Newcastle, NSW) och version 4.0 av Funnelweb är färdig. Jag har sänt ut några kopior för testning och rapporterna bör komma snart. Vem vet vilka buggar som kan dyka upp, men jag tror att den är ganska väl avlusad före frisläppandet. Jag har hittat en liten (uppenbar när den inträffar, men ej allvarlig) och jag är ej säker att jag bryr mig om att rätta den. Jag kommer i fortsättningen att ägna mig åt högnivåspråk ett tag.

Vi har nu en andra fungerande expansionsbox. Det är en 110 V-modell, så den körs via transformator. Detta är ingen nackdel eftersom flerspänningsmodellerna (med flera primärlindningar) är för 220 V och ibland upp till 250 V. Detta betyder att spänningen på kraftbussen i boxen är mycket större än den är konstruerad för och mer effektförluster i spänningsstabiliseringen på varje kort. Vi hade tillräckligt med extrakort för att få igång boxen (RS232 är det enda viktiga kort som vi inte har i reserv) och köpte en begagnad DS diskdrive (MPI 52 A - en tidig IBM PC-typ) så att DS-diskar kan användas från den andra maskinen.

Alla detaljer om den nya versionen finns på disken. Option 6 på TI-Writer sidan av huvudmenyn är avsedd som laddare till en fil för att styra den europeiska TI-Writer versionen med flera språk. Jag är inte säker på om Euro-Writer, den nya SD och den fullständiga Funnelweb kan fungera samtidigt i maskinen.

Hur som helst, ha det trevligt med den nya utgåvan.

Bästa hälsningar

Tony McGovern

Filöverföring via modem

av Peter Odelryd

När vi inom föreningen nu börjat diskutera en egen databas med meddelanden och program blir det också intressant att diskutera hur överföringen ska gå till. När man skriver eller läser meddelanden i ett sånt system gör det ofta inte så mycket om ett eller annat tecken försvinner eller förvrängs under överföringen.

Överföringen mellan två 99:or eller mellan en 99:a och en annan dator sker via det vanliga telefonnätet — med de begränsningar som det medför. Ibland får man en "dålig linje" med störningar. I professionella sammanhang utnyttjas ofta samma telenät. Televerket kallar denna trafik för *Datel*. I andra sammanhang används ett speciellt datanät, exempelvis för Bankomaterna. Det finns olika sätt att sända text via Datel. Det ena kallas asynkront, dvs tecken för tecken. Det andra är synkront i block av tecken av en viss storlek, t ex 128 eller 256 tecken.

Protokollet kollar överföringen

För att kontrollera filöverföring mellan två 99:or eller mellan en 99:a och en annan dator tog TI fram ett kommunikationsprogram som låg i modulen *Terminal Emulator II*. I manualen är detta kortfattat beskrivet, men fullt tillräckligt för att kunna användas praktiskt. När man angivit vilken fil som skall sändas iväg avgör programmet själv vilken typ filen är av. På skärmen kan man sedan följa överföringen. Efter varje block som överförts gör programmet en kontroll. Upptäcks fel sänds blocket om igen. Om ett block inte kunnat sändas felfritt inom 8 försök avbryts sändningen. För att kunna lägga in detta protokoll på andra datorer skrev TI ett dokument om hur kommunikationen fungerade. Detta dokument nämns också i TE II-manualen. Det heter *Texas Instruments Terminal Emulator Protocol Manual* och är daterat den 18 maj 1981. Det är på ett 50-tal A4-sidor. Dokumentet innehåller en hel del information om filformat och annat som inte finns tillgängligt från TI på annat håll. Den felkontroll som programmet använder beskrivs också. Den heter Longitudinal Redundancy Check (LRC).

Andra terminalprogram

Förutom TI:s terminalprogram har det kommit en rad program som är avsedda att fylla samma funktion. Det finns både kommersiella och freeware-program. Ett par av de senare har vi i *Programbanken*, t ex *Fast-Term*. De flesta av dessa program har lagt in filöverföring enligt TE II som en option i sina program. Många har dessutom lagt in annat filöverföringsprogram, *Xmodem*. Det är ett program som används på många andra datorer, t ex inom PC-världen. Xmodem använder en annan och snabbare metod för felkontroll, CRC. Xmodem sänder text i block om 128 tecken. Därför kan filer till MAX-RLE (som beskrevs i förra PB) lagras i form av D/F 128-filer.

Ordföranden ...

Det är snart dax för ett nytt årsmöte. Om du kan tänka dig att vara med i styrelsen eller har förslag till namn tag då kontakt med valberedningen. Valberedningens ordförande är Åke Olsson, telefon 08/712 76 51.

Du som kan göra något inom föreningen utan att vara med i styrelsen skriv gärna och berätta vad du tycker.

Tidigare förfrågan om aktiviteter har gett mycket få svar. Ingen vill skriva om TI-74 och TI-75. Du som har en sådan kalkylator/dator måste höra av dig snarast. I annat fall kan vi inte skriva något om dessa eftersom ingen i styrelsen är ägare av någon av dessa.

Endast fyra medlemmar har hört av sig med önskan om att testa freeware. Du som kan göra något hör av dig eller sänd in dina egna program eller frågor som du vill ha besvarade.

Privatlektioner i Extended BASIC från Funnelweb farm

Med början i detta nummer kommer vi att publicera en serie om Extended BASIC. Den är skriven av Tony McGovern från Australien (känd för sitt Funnelweb-program). Materialet har tidigare publicerats på olika håll. Vi har fått originaltexten från Stephen Shaw i England.

Översättningen har gjorts av Niels Hovmöller.

I. Inledning

I denna serie om TI Extended Basic (XB) för TI-99/4A kommer vi att koncentrera oss på de egenskaper som inte har fått tillräcklig uppmärksamhet i medlemstidningar och kommersiella tidskrifter. Faktum är att de flesta av de program som publicerats där knappast utnyttjar XB:s förnämsta egenskap, de användardefinierade underprogrammen och vissa andra finesser hos XB. Vad värre är, det säljs ofta program som är typiska exempel på krånglig och svårtolkad programmering. Det som gav inspiration till dessa "privatlektioner" var en totalt felaktig kommentar i TI.S.H.U.G. Newsdigest i juni 1983. En del böcker jag sett om TI Basic tar inte ens upp detta enklare språk korrekt, och jag känner inte till några systematiska försök att tränga in i hur XB fungerar. Det bästa hjälpmedlet är TI:s egen instruktionskassett eller -diskett. De program som ingår där är öppna och kan studeras som exempel på hur underprogram kan ge en lättbegriplig struktur åt ett program.

Vad ska detta då handla om? Så här tänker jag lägga upp kursen:

1 Användardefinierade underprogram
2 Pre-scan (in- och urkoppling)
3 Programmering för snabb exekvering
4 Buggar i Extended Basic
5 Hur man skriver korta program
6 Extended Basic och expansionsboxen
7 Sammanlänkning med Assembler-rutiner

Till en början håller vi oss till sådant som kan göras med bara grundenheten och Extended Basic-modulen. I de flesta fall går spelprogram inte så värst mycket snabbare med minnesexpansionen, eftersom hastigheten tycks begränsas av de inbyggda underprogrammen (CALL COINC etc) som utförs från GROM genom GPL-tolken. Den stora fördelen med minnesexpansionen för programmering av spel är, förutom att längre program kan skrivas, att GPL kan makas undan till förmån för maskinkodsrutiner på de ställen i programmet där snabbheten är avgörande, vanligen bara en mycket liten del av programmet. Dessutom kan man ofta uppnå tillräcklig hastighet genom omsorgsfull programmering. Ett exempel: puckens fart i TEX-BOUNCE är 10 gånger högre i den senaste versionen än när programmet först skrevs.

— — —

II. Underprogram — en översikt

Varje Basic-dialekt, inklusive Extended Basic, tillåter användandet av subrutiner. En subrutin är ett antal programrader vars slut markeras av en RETURN-sats, och som utförs när datorn kommer till en GOSUB-sats på annan plats i programmet. Väl framme vid RETURN återgår datorn alltså till den sats som följer närmast efter GOSUB. Studera följande utdrag ur ett program:

```
290 ....
300 GOSUB 2000
310 ....
...
...
2000 CALL KEY (Q, X, Y) :: IF Y=1 THEN RETURN ELSE 2000
```

Detta enkla exempel inväntar och rapporterar ASCII-koden för en ny tangenttryckning och kan anropas från olika ställen i programmet. Mycket användbart, men inte utan problem. Om subrutinens radnummer ändras (på annat sätt än

genom RESequence — och många mikrodatorers Basic har inte ens den finessen) går gosub-satserna vilse. En annan svårighet som man ofta stöter på, om man återupptar arbetet med ett program efter att det legat nere en tid, är att man inte har en aning om vad GOSUB 2000 innebär om man inte ser efter eller har använt REM-satser. Vad värre är, det jämna radnumret 2000 ändras också vid en omnumrering, och så har man inte längre ens detta stöd för minnet. Men det finns ett ännu försåtligare problem: man struntar i vad variabeln Y kallas eftersom den bara skall användas där — men om man råkar använda Y någon annanstans i programmet, kan dess värde påverkas.

Variablerna i underprogrammen hålls inte isär från variablerna i resten av programmet. I Extended Basic däremot finns det fyra sätt att isolera delar av ett program:

1 Inbyggda underprogram
2 DEFinierade funktioner
3 CALL LINK till maskinkodsrutiner
4 Användardefinierade underprogram i Basic

Det första sättet, de inbyggda underprogrammen, är redan välbekanta från grundbasic-en. Det viktigaste med dem är att CALL-satserna har begripliga namn, och att informationen överförs till och från underprogrammen genom en fastställd parameterlista och returvariabler. Inga dunkla PEEK och POKE behövs. Priset som man får betala för kraftfullheten och tydligheten i TI Basic och Extended Basic är långsamheten i implementeringen av GROM/GPL.

DEF-funktionen är en primitiv form av användardefinierat underprogram som finns i nästan alla Basic-dialekter. Ofta begränsas dess användning till en speciell uppsättning variabelnamn (FNA, FNB, FNC...), men TI Basic ger full frihet när det gäller att benämna DEFinierade funktioner (så länge de inte sammanfaller med variabelnamn).

```
100 DEF CUBE(X)=X*X*X
```

Den "tomma" variabeln X används som i en matematisk formel, inte som ett fältindex. Den berör inte en variabel med samma namn på annat håll i programmet. "CUBE" kan alltså inte vara en variabel som tilldelas ett värde på annat håll i programmet, medan X kan användas på andra ställen i samma program. Fastän DEF underlättar författandet av lättlästa program, utförs det mycket långsamt i TI Basic, och långsammare än användardefinierade underprogram med CALL i Extended Basic.

CALL LINK för att anropa maskinkodsrutiner heter olika saker i olika Basic-dialekter om det ingår (t ex USR( ) i några fall). Det är bara tillgängligt i Extended Basic om minnesexpansionen är inkopplad, eftersom TI-99/4A:s grundenhet bara har 256 bytes CPU RAM för den TMS9900 som döljs under skalet. Vi återkommer till detta senare.

Du bör ha din Extended Basic-manual till hands och ögna igenom avsnittet om SUB-program. Texten är i stort sett korrekt men alldeles för kortfattad, och lämnar mycket osagt. Genom egna experiment och erfarenheter har jag kommit fram till att det fungerar som man rimligen skulle väntat sig (vilket inte är fallet med andra delar av Extended Basic). Det gäller att lära sig att vänta sig rätt, vilket går lättare om man kommer underfund med — åtminstone i stora drag — hur datorn gör det den gör. Tyvärr undviker de flesta handböcker för TI-99/4A djupare förklaringar, antagligen för att fabrikanten vill att den skall framstå som en "hemdator". Förklaringarna räcker inte alltid till, och vi skall nu försöka komma underfund med det som TI inte vågat ge sig i kast med.

Användardefinierade underprogram i Extended Basic tillåter att man skriver egna underprogram i BASIC som kan anropas med CALL och ett namn man själv har valt, på samma sätt som de inbyggda underprogrammen. Till skillnad från de rutiner som nås via GOSUB-satser berörs inte huvudprogrammet av vad som sker i underprogrammet utom i den mån det anges i parametrar som ingår i CALL-satsen. Till skillnad från de inbyggda underprogrammen som överför information endast i ena riktningen, antingen till eller från underprogrammet enligt parameterlistan, kan ett användardefinierat underprogram med CALL i Extended Basic använda vilken variabel som helst i variabellistan för att flytta information i önskad riktning. Dessa underprogram möjliggör det som i programmeringssammanhang kallas *procedurer* på andra dataspråk, t ex Pascal, Logo och Fortran. Bristen på riktiga procedurer har alltid varit den svåraste begränsningen för Basic som programmeringsspråk. TI:s Extended Basic är ett språk som ger den möjligheten. Inte ens alla nyare BASIC-dialekter har den finessen. Tidningen Australian Personal Computer hade t ex en recension av IBM:s PCJunior när den nyss hade lanserats i USA (i mars 1984). Det förefaller som om denna dators Cartridge Basic inte klarar procedurer.

Kanske IBM inte önskar eller väntar sig att någon på allvar skriver egna program i Basic. Men med tillgång till riktiga användardefinierade underprogram kan man inte ens föreställa sig längre hur någon skulle stå ut med att skriva ordentliga program utan dem, ens inom 14 kB-gränsen hos en outbyggd TI-99/4A, och än mindre på en maskin med större minne.

Hur procedurer och underprogram fungerar i detalj är olika i olika programspråk. Det vanligaste är att variablerna inom en procedur är begränsade till den proceduren. Hur de samverkar med resten av programmet och vad som händer med dem sedan underprogrammet genomlöpts varierar från programspråk till programspråk. Extended Basic gör det på sitt eget väldefinierade sätt, men är inte alls flexibelt härvidlag.

Låt oss nu se på hur Extended Basic hanterar underprogram. När ett Extended Basic-program körs (med kommandot RUN), sker det i två steg. Det första är "prescan", den stund som förflyter sedan man skrivit RUN och tryckt på <ENTER> och innan något synbarligen sker. Under denna tid går Extended Basic-tolken igenom programmet och kollar att några få saker stämmer som det inte hade varit möjligt att kontrollera alltftersom de enskilda raderna skrevs in, t ex att det finns ett NEXT till varje FOR. TI:s Basicdialekter gör bara den allra elementäraste granskning sedan en rad skrivits in, och lämnar den egentliga kontrollen rad för rad tills programmet körs. Detta är inte det bästa sättet, men det är vad vi har och det har en fördel. Extended Basic letar samtidigt upp alla variabelnamn, reserverar utrymme för variablerna och sätter upp de procedurer varigenom variabelnamnen förknippas med minnesutrymme under programkörningen.

Precis hur Extended Basic utför detta är inte helt klart, men det måste innebära att hela variabellistan genomsöks varje gång en variabel påträffas, varvid lagerekonomi vinnes på bekostnad av snabbhet.

Extended Basic känner också igen vilka SUB-program som faktiskt anropas med CALL. Hur kan det känna skillnad mellan ett underprograms namn och ett variabelnamn? Det är lätt, för inbyggda underprograms namn föregås alltid av CALL. Det är därför som underprogrammens namn inte är reserverade ord och även kan användas som variabelnamn. Detta innebär att det tidsödande genomletandet av GROM-biblioteket bara sker vid Pre-scan, varefter Basic har en egen lista för varje program där det står vart det skall gå i GROM för att hämta GPL-rutinen utan att behöva leta igenom GROM varje gång det stöter på ett underprogram under en programkörning. I direktmod har datorn ingen möjlighet att hitta namnen på användardefinierade underprogram i ett Extended Basic-program i minnet ens sedan programmet avbrutits med BREAK. Extended Basic förbereder också processen för att "slå upp" DATA- och IMAGE-satser i programmet.

Jaha, vad gör då Extended Basic med användardefinierade underprogram? Först letar Extended Basic reda på de underprogram-namn som inte är inbyggda i Basicen. Detta sker genom att datorn letar upp alla namn efter en CALL- eller SUB-sats, och därefter slå upp det i GROM-bibliotekets register över inbyggda underprogram-namn. Man kan lätt kolla detta genom att skriva detta program på en rad

```
100 CALL NOTHING
```

Fortsättning i PB 88-1